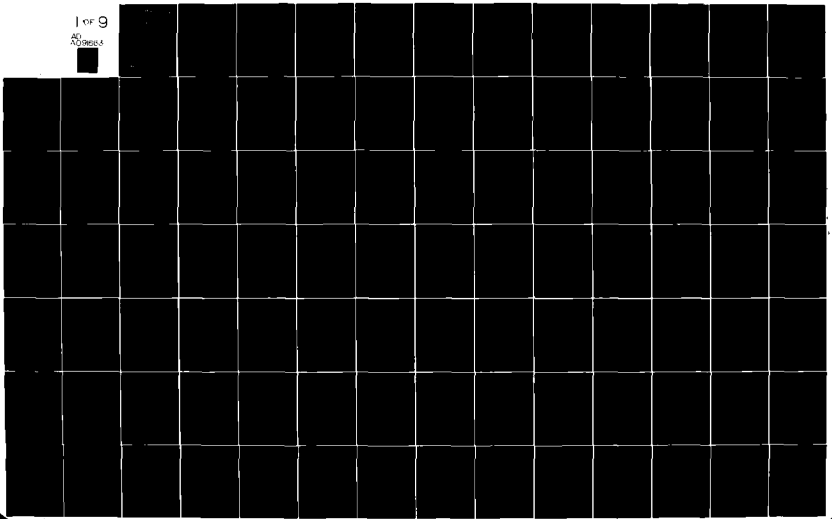
AD-A091 663   GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80   A J GOLDBERG, L COSELL, S KWON   DCA100-78-C-0064

UNCLASSIFIED   NL

1 OF 9
AD
A091663



LEVEL III A091662

12

AD A091663

FINAL REPORT

SPEECH OPTIMIZATION AT 9600 BITS/SECOND

VOLUME 2

REAL-TIME SOFTWARE AND HARDWARE

SUBMITTED TO
DEFENSE COMMUNICATIONS AGENCY

SEPTEMBER 30, 1980

Sylvania Systems Group
Communication Systems Division
GTE Products Corporation
77 A Street
Needham Heights, Mass. 02194 U.S.A.
Area Code 617 449-2000
TELEX: 92-2497

GTE Systems

FINAL REPORT

Speech Optimization at 9600 Bits/Second

Volume 2

Real-Time Software and Hardware

Submitted to

Defense Communications Agency

September 30, 1980

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A091 664
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) 9600 BPS Speech Optimization Study Volume 2	5. TYPE OF REPORT & PERIOD COVERED Final Report October 1978 - September 1980
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) A. J. Goldberg, L. Cosell, S. Kwon, L. Bergeron, and R. Cheung	8. CONTRACT OR GRANT NUMBER(s) DCA 100-78-C-0064
9. PERFORMING ORGANIZATION NAME AND ADDRESS GTE Sylvania 77 "A" Street Needham Heights, MA 02194	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305	12. REPORT DATE September 30, 1980
	13. NUMBER OF PAGES Volume 2 - 821 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

speech coding, 9600 bps speech transmission, adaptive transform coder, adaptive bit allocation, discrete cosine transform, digital voice terminal, real-time speech coder

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design and development of a real-time adaptive transform coder that transmits high-quality speech over a 9600 bps channel with bit-error rates of up to 1% without significant loss of speech fidelity. The report presents the results of our FORTRAN simulations on the adaptive transform coder which maximized the quality of the transmitted speech. Important aspects of the ATC algorithm which are optimized

(cont'd)

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (cont'd)

were specification and transmission of the side-band information, accuracy of the pitch and voicing decisions, and error-protection of the important transmission parameters. Also included is the system design, detailed documentation, and program listings of the MAP-300 real-time implementation of the optimized ATC speech coder. Finally, the report includes a description of analog equipment GTE built to interface the MAP-300 to telephone handsets and tape recorders and a description of digital circuits (RS 423 compatible) to interface the MAP-300 to a modem.

This report is bound in two volumes. Volume I contains a description of the ATC system and the results of the FORTRAN simulations. Volume II contains all the information on the real-time system including documentation for implementing the ATC system on the MAP, listing of the MAP software, and documentation for the hardware built by GTE.

TABLE OF CONTENTS

for

Volume 2

Real-Time Software

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 3

Real-Time Implementation

The GTE Sylvania Speech Processing System functions as a real-time, full duplex terminal of a digital voice communications system. As such, the GTE system accepts and digitizes analog voice input, processes it using an Adaptive Transform Coding (ATC) algorithm, and transmits the resulting serial bit stream at 9600 bits/second across a digital channel to a similar terminal. Because of the full duplex requirement, the system must simultaneously accept a similar bit stream, decode it to produce synthetic speech, and finally convert this synthetic speech to an analog waveform which it then plays out.

This chapter describes the design and operation of the GTE system and the hardware and software components used to construct it.

3.1 System Components

The GTE ATC system is implemented on a MAP-300 processor manufactured by CSP, Inc., Billerica, Mass. The MAP architecture consists of several independent processors, capable of parallel operation. Some of these processors are the Arithmetic Processor, used for high speed floating point operations; the Input/Output Scroll processors, of which there are three in the system, used for the Analog In, Analog Out, and Digtial In/Out functions; and the CSPU, which is used to control the other processors and also to perform general computation chores.

Attached to the MAP is a GTE-manufactured Speech Processing Interface (SPI). The SPI includes a handset with earphone and high-quality microphone, amplifiers, and band-limiting filters. The SPI interfaces to the ADAM and AOM scrolls. The Full Duplex Interface (FDI) is also attached to the MAP. The FDI resides physically on the I/O scroll board and provides 9600 bps digital ports and their interfaces to the I/O scroll. Two additional SPI's and FDI's to be used on other MAP systems have been supplied by GTE.

The MAP is attached to a host PDP-11 which is used to load the MAP programs and to initialize and start them. Once the MAP has been started, it runs independently of the host. A block diagram of the system is shown in Figure 3-1.

The software components of the ATC system include modules supplied by CSP, Inc., as well as those written by GTE. Three categories of software are used in the ATC system. The first, the Fortran Control and Support programs, resides in the PDP-11 and is used for communication between the user and the MAP. The second category, Executive software, consists of the body of code that runs in the CSPU. This software is used to control and schedule all the processes in the MAP. The third category, MAP Functions, consists of the program modules that are loaded into the Arithmetic Processor and I/O Scroll processors. The scrolls require only one program each, but the Arithmetic Processor, because of limited memory, must be loaded with successive program modules and started for each. The CSPU Executive software performs this task.

3.2 System Design and Operation

The ATC system is composed of seven distinct processes. These are the Analog In process, the Analog Out process, the Digital In process, the Digital Out process, the Analysis process, the Synthesis process, and the Executive process. Figure 3-2 shows the function of and the relationship between the various processes. Conceptually, all of these

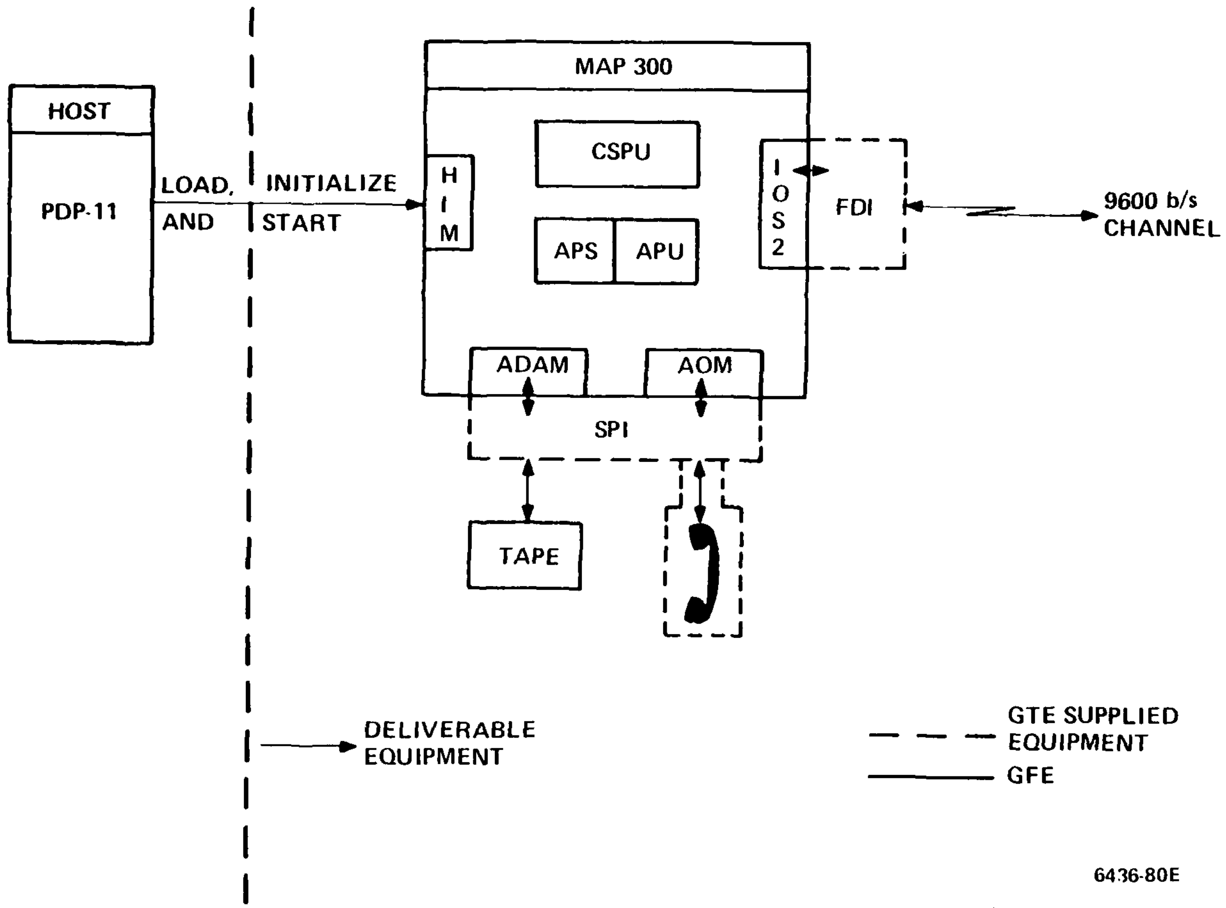


FIGURE 3-1: ATC SYSTEM COMPONENTS

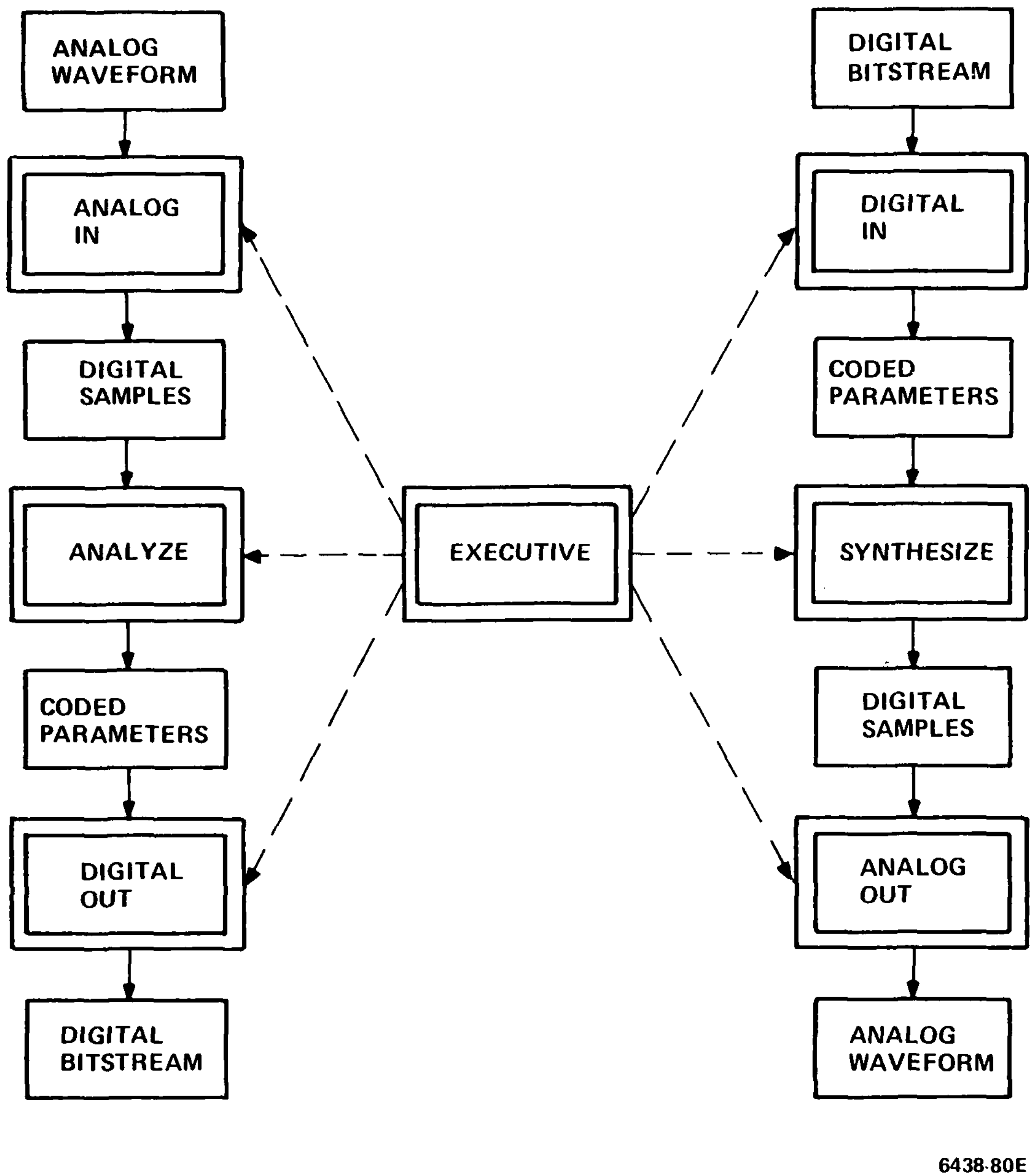


FIGURE 3-2: BASIC PROCESSES OF ATC SYSTEM

processes operate simultaneously, however, since at least portions of all of them must be executed by the CSPU or the Arithmetic Processor, both of which are sequential machines, the scheduling and synchronization of these processes is at the heart of the system design. Since the four I/O processes are essentially similar, they will be discussed first. The Analyze and Synthesize processes, which depend on their inputs on the Analog and Digital In processes, will be discussed next. Finally, the Executive process and the scheduling of the other processes will be presented.

3.2.1 Analog In Process

The Analog In process controls the A/D converter and makes the digitized samples available to the Analysis process. The Analog In process consists of three separate elements: the scroll program, the interrupt routine, and the background program. Figure 3-3 shows the components of the Analog In process.

3.2.1.1 A/D Scroll Program

This program runs in the ADAM scroll. It causes the A/D converter to operate at a pre-selectable sampling rate, which is 6400 samples/second for the purposes of the ATC system. As each sample is digitized, it is placed in the next (empty) location in the current buffer. Two buffers are used so that one can be filling while the data already in the other is used. Each buffer is 246 samples, or 37.5 milliseconds long. When a buffer is filled, an interrupt to the CSPU is generated by the scroll program, and the other buffer becomes the current buffer.

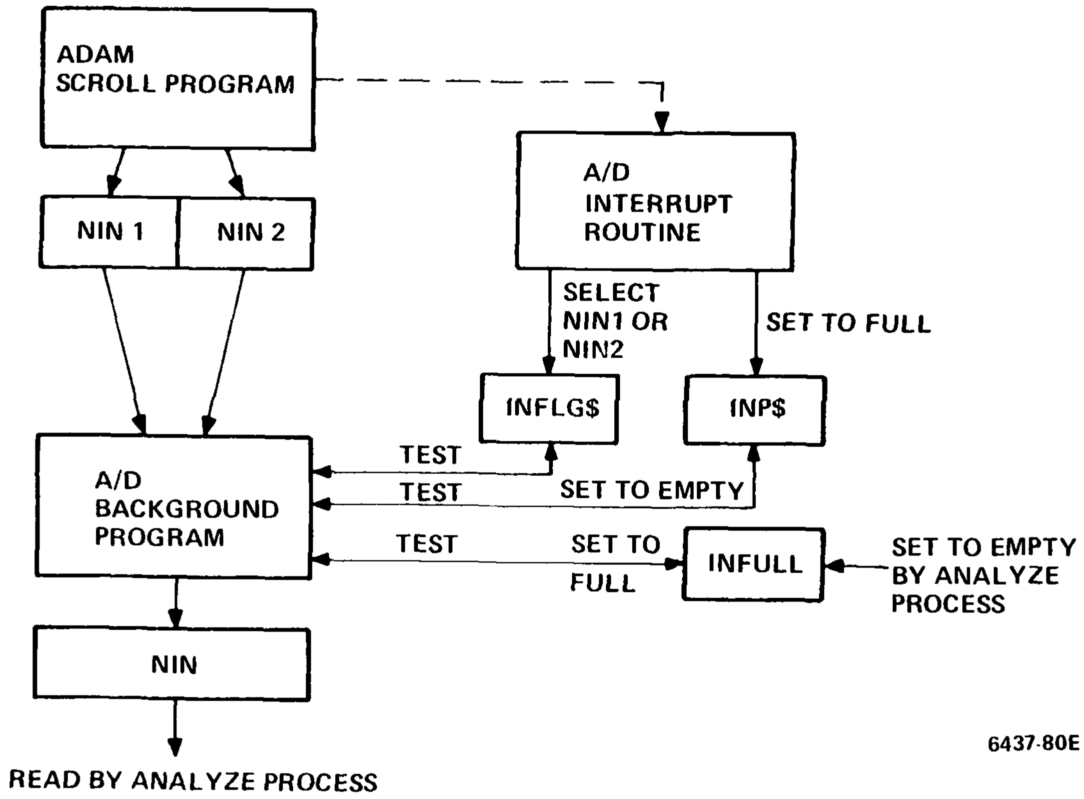


FIGURE 3-3: ANALOG IN PROCESS

This program is used as supplied by CSP, Inc., and as described in their documentation. The program used is ADMSD. The parameters supplied to it are ADAMPM (=63), the BID of the buffer containing the ADAM program; two identical sampling rates, each 6400 Hz; a control word which selects the internal clock, fixed point sampling, internal triggering, and no slow sampling; channel 1 select; and the double buffering pair where the input will be stored, NIN1 and NIN2.

3.2.1.2 A/D Interrupt Routine

The A/D interrupt routine runs in the CSPU at interrupt level 10. It clears INPS (integer scalar 106) to indicate that a new frame of samples is available, and it also clears or sets INFLGS (integer scalar 107) to indiciate that buffer NIN1 or buffer NIN2, respectively, contains the newest data.

3.2.1.3 A/D Background Program

The A/D background program runs in the CSPU. It is enabled by the combination of INPS clear and INFULL (integer scalar 110) clear, which indicates that the previous data has been used by the Analysis process. The background process copies the new data from NIN1 or NIN2, depending on the state of INFLGS, into the analysis buffer, NIN. Finally, it sets INFULL and exists.

3.2.2 Analog Out Process

The Analog Out process is similar to the Analog In process except that the data flow is reversed. The Analog Out process consists of the D/A Scroll program, the D/A interrupt routine, and the D/A background program. Figure 3-4 shows the components of the Analog Out process.

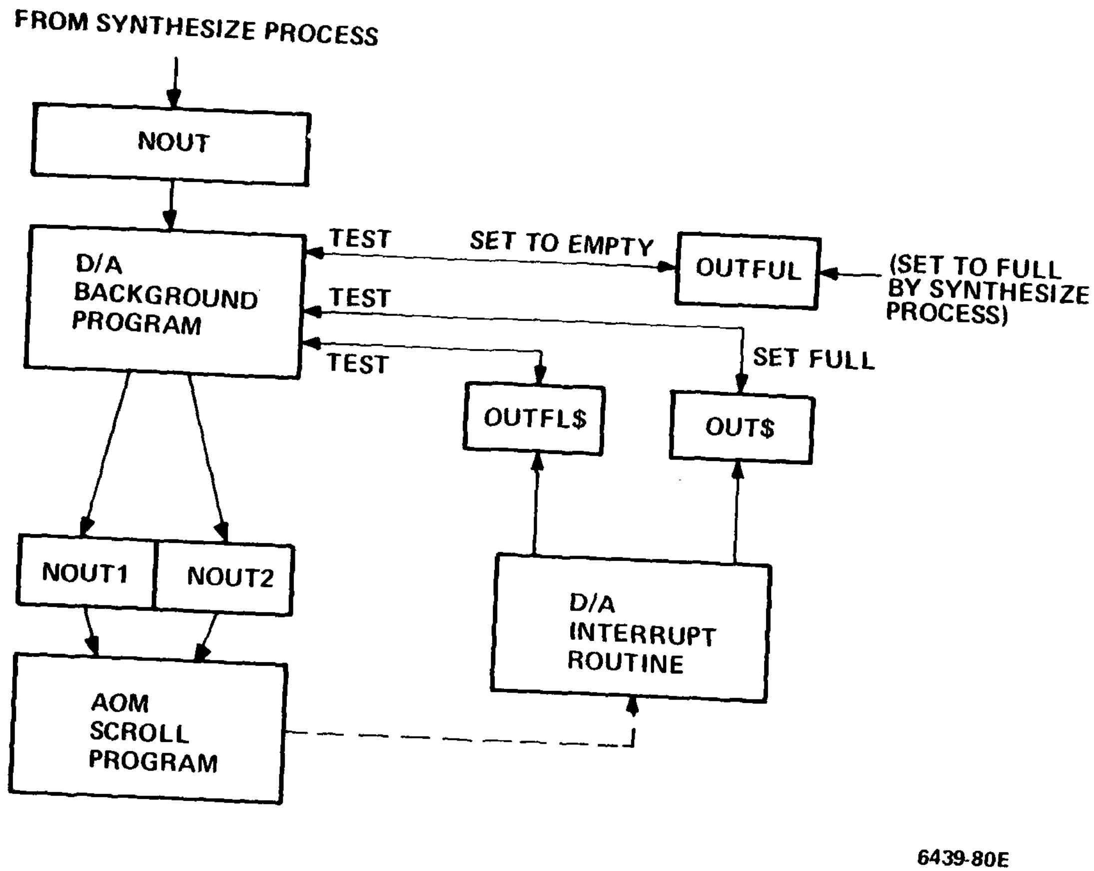


FIGURE 3-4: ANALOG OUT PROCESS

3.2.2.1 D/A Scroll Program

This program runs in the AOM scroll. It controls the D/A converter and uses a double buffering scheme similar to that used by the A/D scroll program. As each of the two buffers is emptied, the program generates a Device 22 interrupt to the CSPU. This program is used as supplied by CSP, Inc., and as described in their documentation. The program module used is AOMID. The parameters supplied to it are AOMPM, (=62), the BID where the AOM program is stored; a sampling frequency divider which is 1; a control word which selects fixed point output, single channel mode, and external triggering; the double buffering pair which stores the output, NOUT1 and NOUT2; and an offset, which is 2. The AOM program produces two channels of output, one of which is the data from the MAP and one of which is a ramp from 0 to 491.

3.2.2.2 D/A Interrupt Routine

The D/A interrupt routine runs in the CSPU at level 9. It clears OUTS (integer scalar 104) to indicate that a buffer has been emptied. It also clears and sets the flag OUTFLS (integer scalar 105) to indicate that buffer NOUT1 or buffer NOUT2, respectively, is the buffer most recently emptied and, hence, available.

3.2.2.3 D/A Background Program

The D/A background program module, called LOBUF, runs in the CSPU and is enabled by the combination of OUTS clear and OUTFUL (integer scalar 111) set, indicating that the Synthesis process has made new data available for output. If synchronization has been acquired, the background program copies the new data from buffer NOUT into either buffer NOUT1 or buffer NOUT2, depending on the state of OUTFLS. If the system is not in synchronization, the relevant output buffer is set to zero. Finally, the background program clears OUTFUL, indicating that NOUT is available for filling by the Synthesis process.

3.2.3 Digital In Process

The Digital In process controls the digital input from the I/O scroll, examines the digital data to determine or confirm synchronization, and makes a frame of digital data available to the Synthesis process. The Digital In process consists of three components: the I/O scroll program, the Receiver interrupt routine, and the Receiver background program. Figure 3-5 shows the components of the Digital In process.

3.2.3.1 I/O Scroll Program

This program, which runs in the IOS2 Scroll, both transmits and receives digital data, and, thus, is a component of both the Digital In and the Digital Out process. This GTE written program is called LINE. It uses two sets of double buffers. One set, RECV1 and RECV2, stores the incoming bits from the modem. The other set, SEN1 and SEN2, store the bits to be output to the modem. As either of the receive buffers is filled, an interrupt from Device 16, line 2 to the CSPU is generated, and the other receive buffer begins to fill. Similarly, as either of the send buffers is emptied, an interrupt from Device 16, line 1 to the CSPU is generated, and the next bit to be output is taken from the other send buffer. All four of these buffers are 369 bits long and 16 bits wide. However, only the low order bit from each 16-bit word contains

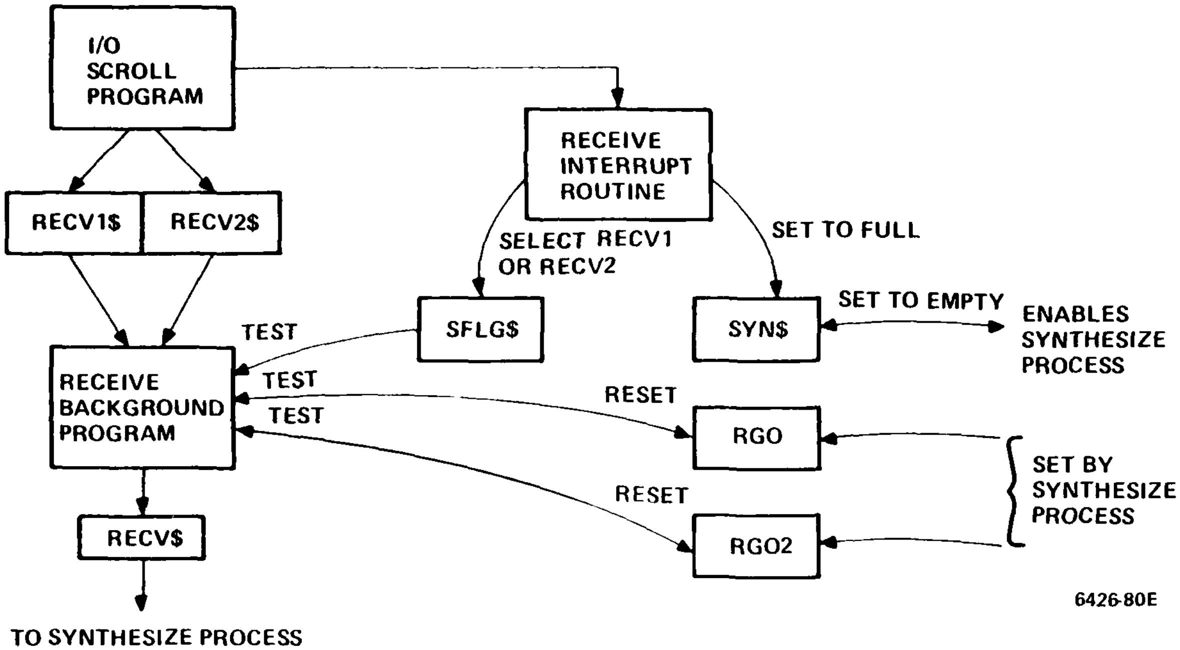


FIGURE 3-5: DIGITAL IN PROCESS

the digital data. The reader is referred to section 3.4 for the meanings of the other bits.

3.2.3.2 Receive Interrupt Routine

The Receive Interrupt routine runs in the CSPU at level 7. It has two functions, first, to keep track of which receive buffer has the newest data and to transmit this information to other processes, and, second, to confirm that synchronization is still valid.

When an interrupt from the receive portion of the I/O Scroll program occurs, the Receive interrupt routine clears SYNS (integer scalar 102) and clears or sets flag SFLGS (integer scalar 103) to indicate that buffer RECV1 or buffer RECV2, respectively, has been filled.

When the ATC system has achieved synchronization (integer scalar 118, RSYN, greater than zero), the bit assumed to be the sync bit for the new data is compared to the sync bit from the previous buffer. If these bits are opposite, synchronization is assumed to be still good. It they are the same, and it is the fourth non-alternating sync bit without two consecutive correctly alternating sync bits intervening, synchronization is declared lost (RSYN is cleared). The location of the sync bit in the data buffer is stored in SYNCS (integer scalar 117).

3.2.3.3 Receive Background Program

The Receive background program runs in the CSPU and is enabled by flag RGO (integer scalar 114) being set. This flag is set by the Synthesis process which is in turn enabled by SYNS being clear. The Receive background program sets SYNS to one on entry.

The Receive background program performs several functions. First, it must collect a new frame of data. This consists of finding the newest sync bit location, and using the 369 previous received bits as the frame. This is implemented by having the background program copy the new data into another set of contiguous buffers, BF1, BF2, and BF3. If RECV2 contains the new data, it is copied into BF2. If RECV1 contains the new data, it is copied into both BF1 and BF3. Then, the 369 data bits prior to the sync bit location are copied into buffer RECV. This double copying of RECV1 assures that, regardless of the position of the synchronization bit in the new frame, the 369 data bits previous to it are contained (contiguously) in the triple length buffer.

Second, the background program must acquire synchronization. That is, it must determine the position of the sync bit in the frame of data. The background program builds a histogram containing 369 bins, one for each bit in the frame. When it receives a new frame of data, the new frame is compared bit for bit against the previous frame. If a given bit alternates in the two frames, the corresponding bin in the histogram is incremented. If it does not alternate, the bin is cleared. This process continues until one bin both exceeds the acquired threshold (10.) and is a unique maximum in the histogram. When this occurs, the bit position corresponding to the maximum bin is declared the Sync position and is stored in SYNCS (integer scalar 117) and RSYN (integer scalar 118) is set to one to indicate that synchronization has been acquired. Until synchronization is acquired, the Receive background process terminates at this point.

Once synchronization has been acquired, the background process must present an actual frame of data, one that has the sync bit at the beginning of the frame, for further processing. To do this, it copies the 369 bits previous to the sync bit of the newest frame from the triple length buffer (BF1, BF2, and BF3) into the receive processing buffer RECVS. This procedure is shown in Figure 3-6. Reasons of efficiency require that the new actual frame be present as a contiguous block within the triple length buffer, in order that a "block move" instruction can be used for the copying and no "corner turning" test is needed.

Once a new frame of data has been copied into RECVS, transmission bit errors are detected and corrected if possible. A (63, 45) BCH code is used to protect 45 bits of the sideband parameters with 18 protection bits. These 63 bits are decoded using a table-look-up procedure for efficiency. For each of the 63 transmitted bits, there are three associated values (Table 3-1). Three residues S_1, S_3, and S_5 are computed by exclusive-or'ing the appropriate value for each bit that is set into each sum. If each of these residues is zero, no transmission errors are detected, and the 45 data bits are used as received.

If the residues are not all zero, the number of errors is determined by computing three syndromes σ_1, σ_2, and σ_3, and a determinant, which is

$$S_1^3 + S_3$$

If this determinant is zero, one error has occurred. σ_1 is set to one and σ_2 and σ_3 are zeroed. Otherwise,

$$\sigma_1 = S_1$$

$$\sigma_2 = S_1^2 S_3 + S_5/(S_1^3 + S_3)$$

$$\sigma_3 = (S_1^3 + S_3) + S_1\sigma_2$$

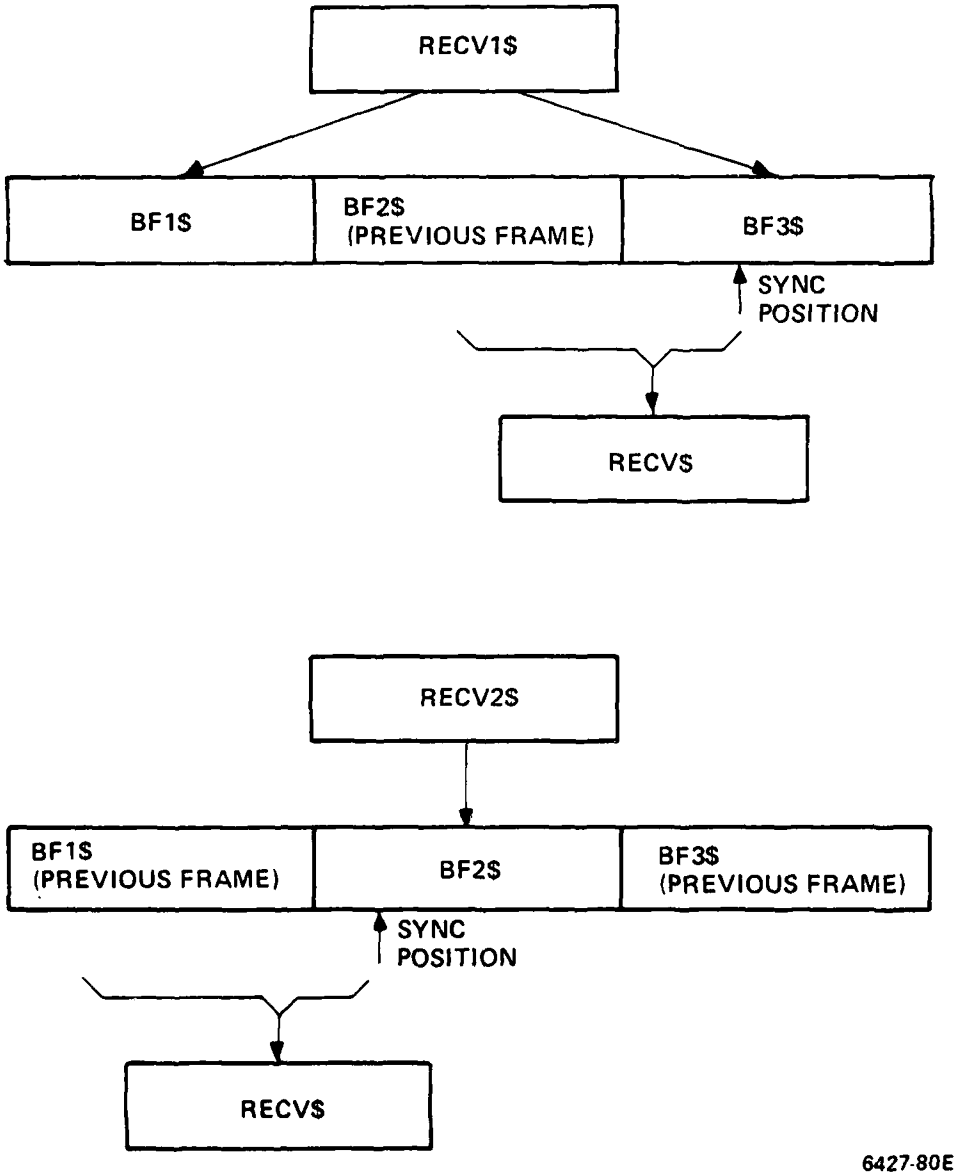


FIGURE3-6: FRAME COLLECTION

Bit Position	S_1 (decimal)	S_3	S_5	Bit Position	S_1 (decimal)	S_3	S_5
0	33	57	63	33	56	17	30
1	49	62	42	34	28	59	20
2	57	23	13	35	14	15	24
3	61	43	55	36	7	40	16
4	63	13	27	37	34	5	33
5	62	25	18	38	17	24	62
6	31	58	28	39	41	3	21
7	46	54	41	40	53	8	39
8	23	22	15	41	59	1	58
9	42	18	10	42	60	57	44
10	21	51	12	43	30	62	9
11	43	14	8	44	15	23	14
12	52	17	49	45	38	43	53
13	26	59	31	46	19	13	38
14	13	15	43	47	40	25	5
15	39	40	50	48	20	58	6
16	50	5	29	49	10	54	4
17	25	24	22	50	5	22	57
18	45	3	37	51	35	18	46
19	55	8	7	52	48	51	52
20	58	1	59	53	24	14	25
21	29	57	19	54	12	17	47
22	47	62	35	55	6	59	11
23	54	23	3	56	3	15	51
24	27	43	2	57	32	40	34
25	44	13	61	58	16	5	60
26	22	25	23	59	8	24	40
27	11	58	26	60	4	3	48
28	36	54	45	61	2	8	32
29	18	22	54	62	1	1	1
30	9	18	36				
31	37	51	56				
32	51	14	17				

TABLE 3-1: RESIDUE TABLE TBPSM

The powers are formed using two tables TBRPC and TBPRC, shown in Table 3-2. These are essentially logarithm and anti-logarithm tables for Galois field arithmetic. For example, S_1^2 is found by using S_1 as an index to TBRPC, doubling the value found in the table (modulo 63) and using the result as an index into TBPRC. The division is performed similarly.

The Chien search procedure (refer to Section 2.4) is used to determine the position of the errors. This procedure uses three additional tables, TBAL1, TBAL2, and TBAL3, shown in Table 3-3, to look up the products $\sigma_1\alpha$, $\sigma_2\alpha^2$, and $\sigma_3\alpha^3$ which replace σ_1, σ_2, and σ_3. For each bit position, σ_1, σ_2, and σ_3 are exclusive or'ed together. If the result is unity, that bit position is in error and is complemented.

When transmission errors have been corrected, the background process must deserialize the bitstream into code words of various lengths. This deserialization procedure is performed in two parts. The first part deserializes the sideband parameters and makes them available to the Synthesis process. Figure 3-7 shows a map of the data frame and the bits allocated to each of the sideband parameters. The sideband code words are stored in buffer RQPBS, which is later read by the Synthesize process.

Before the second part of the deserialization procedure, deserialization of the Discrete Cosine Transform (DCT) coefficients can be executed, the word lengths of the DCT's must be obtained from the Synthesize process, which has determined these word lengths from the sideband parameters. To avoid delays and the consequent buffering, the transmitted data consists of the current frame of sideband information and the

TBRPC				TBPRC			
S	Index	S	Index	Index	S	Index	S
0	0	32	5	0	1	32	9
1	0	33	62	1	2	33	18
2	1	34	25	2	4	34	36
3	6	35	11	3	8	35	11
4	2	36	34	4	16	36	22
5	12	37	31	5	32	37	44
6	7	38	17	6	3	38	27
7	26	39	47	7	6	39	54
8	3	40	15	8	12	40	47
9	32	41	23	9	24	41	29
10	13	42	53	10	48	42	58
11	35	43	51	11	35	43	55
12	8	44	37	12	5	44	45
13	48	45	44	13	10	45	25
14	27	46	55	14	20	46	50
15	18	47	40	15	40	47	39
16	4	48	10	16	19	48	13
17	24	49	61	17	38	49	26
18	33	50	46	18	15	50	52
19	16	51	30	19	30	51	43
20	14	52	50	20	60	52	21
21	52	53	22	21	59	53	42
22	36	54	39	22	53	54	23
23	54	55	43	23	41	55	46
24	9	56	29	24	17	56	31
25	45	57	60	25	34	57	62
26	49	58	42	26	7	58	63
27	38	59	21	27	14	59	61
28	28	60	20	28	28	60	57
29	41	61	59	29	56	61	49
30	19	62	57	30	51	62	33
31	56			31	37		

TABLE 3-2: TBRPC AND TBPRC TABLES

σ	$\sigma_1\alpha$ TBAL1	$\sigma_2\alpha^2$ TBAL2	$\sigma_3\alpha^3$ TBAL3	σ	$\sigma_1\alpha$ TBAL1	$\sigma_2\alpha^2$ TBAL2	$\sigma_3\alpha^3$ TBAL3
0	0	0	0	32	3	6	12
1	2	4	8	33	1	2	4
2	4	8	16	34	7	14	28
3	6	12	24	35	5	10	20
4	8	16	32	36	11	22	44
5	10	20	40	37	9	18	36
6	12	24	48	38	15	30	60
7	14	28	56	39	13	26	52
8	16	32	3	40	19	38	15
9	18	36	11	41	17	34	7
10	20	40	19	42	23	46	31
11	22	44	27	43	21	42	23
12	24	48	35	44	27	54	47
13	26	52	43	45	25	50	39
14	28	56	51	46	31	62	63
15	30	60	59	47	39	58	55
16	32	3	6	48	35	5	10
17	34	7	14	49	33	1	2
18	36	11	22	50	39	13	26
19	38	15	30	51	37	9	18
20	40	19	38	52	43	21	42
21	42	23	46	53	41	17	34
22	44	27	54	54	47	29	58
23	46	31	62	55	45	25	50
24	48	35	5	56	51	37	9
25	50	39	13	57	49	33	1
26	52	43	21	58	55	45	25
27	54	47	29	59	53	41	17
28	56	51	37	60	59	53	41
29	58	55	45	61	57	49	33
30	60	59	53	62	63	61	57
31	62	63	61	63	61	57	49

TABLE 3-3: CHIEN SEARCH TABLES

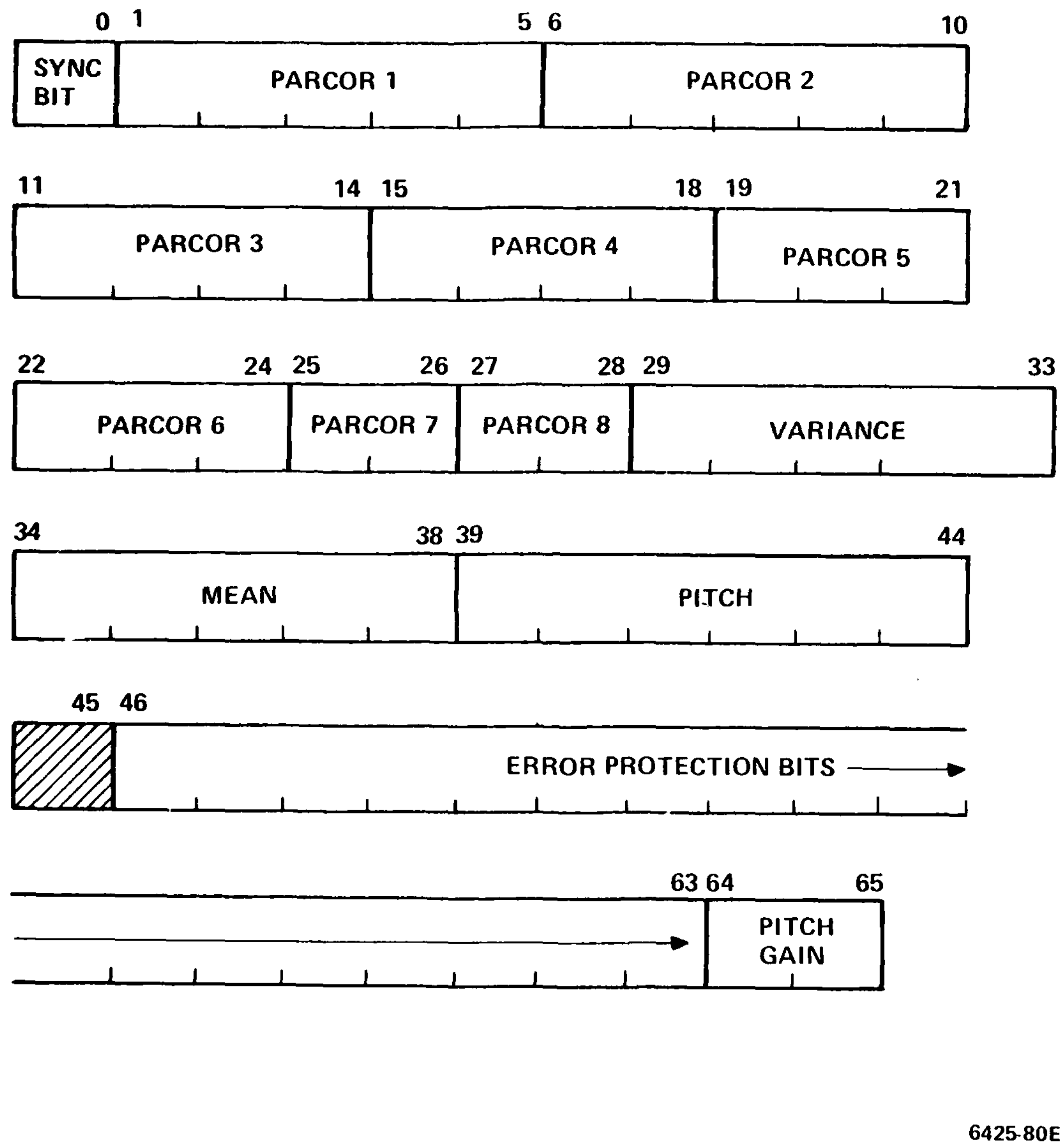


FIGURE 3-7: PARAMETER FRAME MAP

previous frame of DCT coefficients. Thus, the Synthesize process determines the DCT code word lengths for a given frame based on sideband information received in the previous frame. It passes this information to the Receive background program in buffer RIBITS and sets RG02 (integer scalar 115) to 1 to indicate that new data is present. The Receive background program waits for RG02 to be set before it proceeds with the DCT deserialization and clears RG02. The deserialized DCT coefficients are stored in buffer RQTDCT. The Receive background program then terminates. Figure 3-8 shows the organization of the complete Receive background program.

3.2.4 Digital Out Process

The Digital Out process serializes and protects the data to be transmitted and controls the digital output to the I/O scroll. The Digital Out process consists of three components: the I/O scroll program, the Transmit interrupt routine, and the Transmit background program. Figure 3-9 shows the components of the Digital Out process.

3.2.4.1 I/O Scroll Program

The transmitter function of the I/O scroll program is described in subsection 3.2.3.1. This program puts out the low order bit of each half-word in the double buffer pair SEN1 and SEN2 and generates an interrupt when either buffer is emptied.

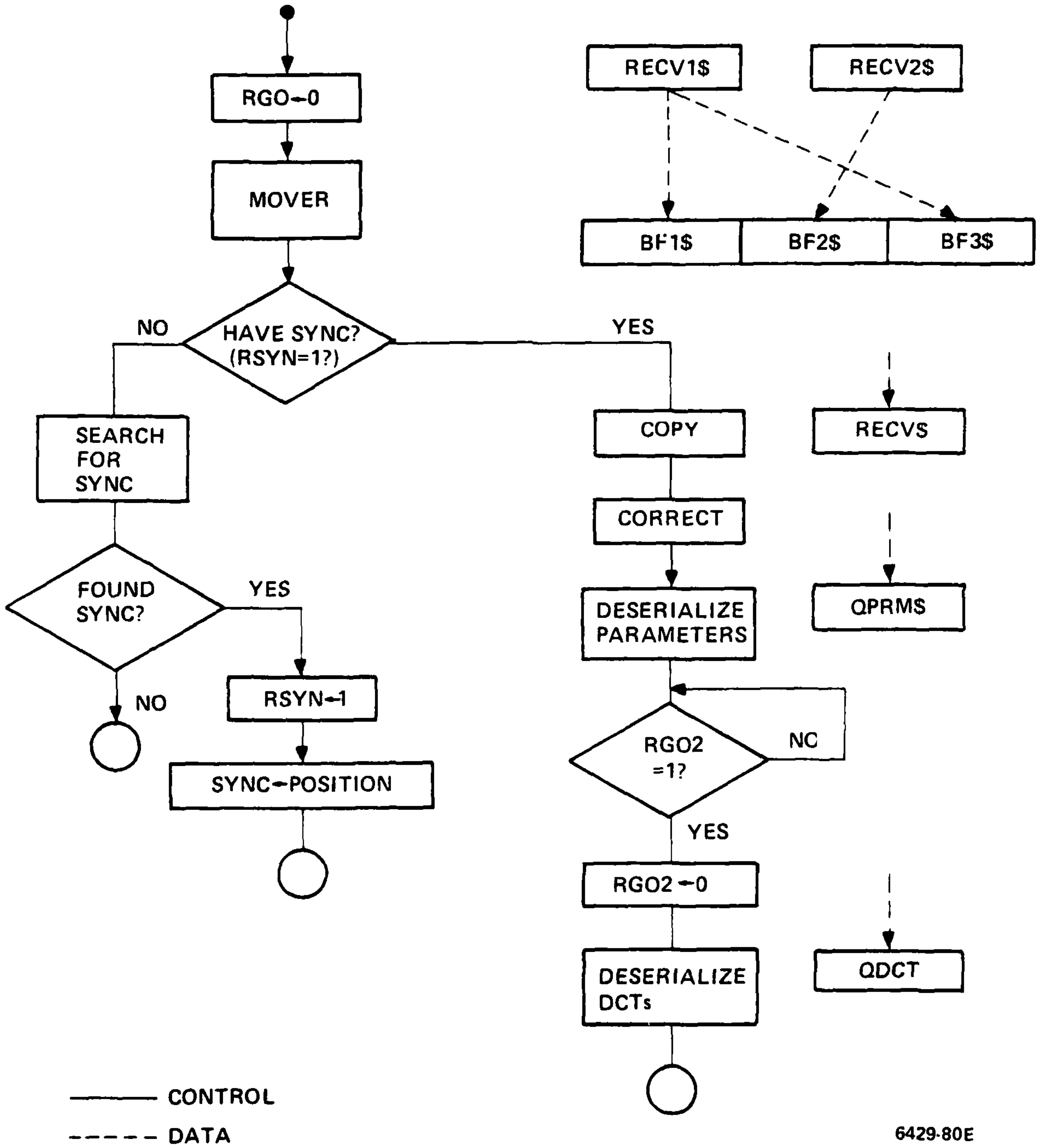


FIGURE 3-8: RECEIVE BACKGROUND PROCESS

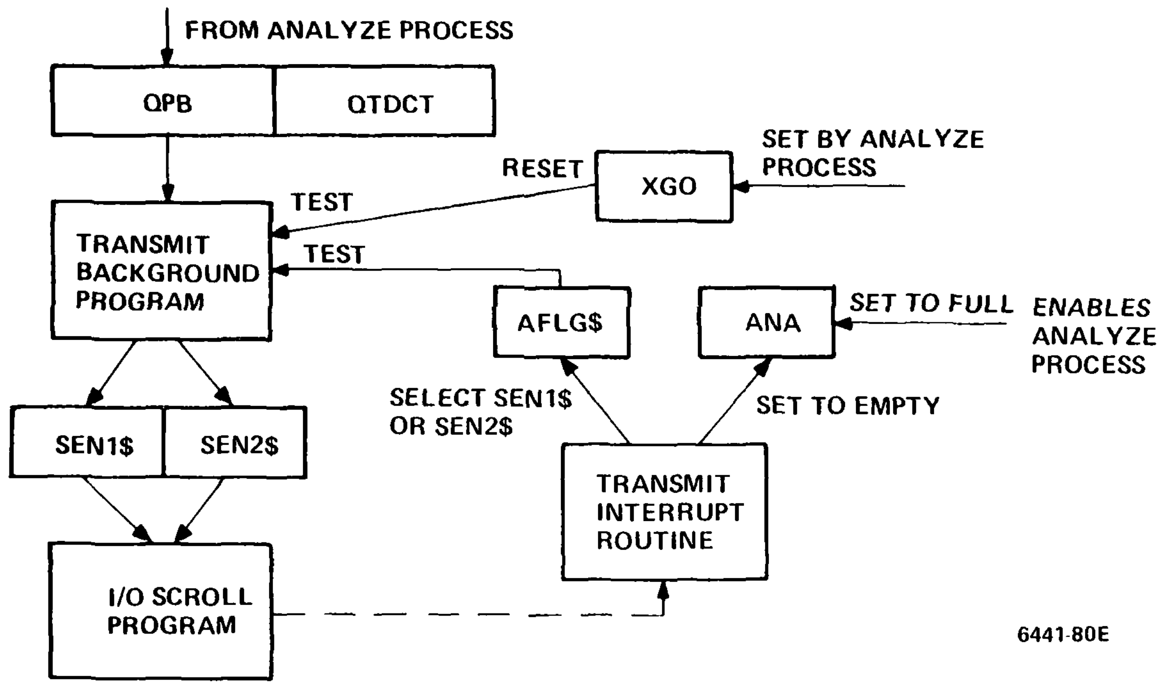


FIGURE 3-9: DIGITAL OUT PROCESS

3.2.4.2 Transmit Interrupt Routine

The Transmit interrupt routine runs in the CSPU at level 7. Its function is to keep track of which transmit buffer, SEN1 or SEN2, has been most recently emptied, and to transmit this information to the Transmit background program.

When an interrupt from the transmit portion of the I/O scroll program occurs, the Transmit interrupt routine clears ANAS (integer scalar 100) to indicate that a transmit buffer has become empty and clears or sets AFLG (integer scalar 101) to indicate that buffer SEN1 or SEN2, respectively, has been most recently emptied.

3.2.4.3 Transmit Background Program

The Transmit background program runs in the CSPU and is enabled by flag XGO (integer scalar 115) being set. This flag is set by the Analysis process, which is in turn enabled by ANAS being clear. The Transmit background process sets ANAS to one on entry.

The Transmit background program first transforms the quantized sideband parameters from buffer QPBS and the quantized DCT coefficients from buffer QTDCTS into a bitstream which is built up in buffer SENS. The SENS buffer contains 369 half-words, the low order bits of which form the bitstream. The function of the higher order bits is described in Section 3.4.

The serialization procedure is performed in two parts. The first part serializes the sideband information according to the frame map in Figure 3-7. This part also skips 18 words in SENS to allow for later insertion of protection bits. The second part of the serialization procedure serializes the DCT coefficients. These coefficients are those obtained from the previous frame. That is, the DCT coefficients are delayed in the Analysis process by one frame before being serialized, while the sideband parameters are those from the current frame and experience no additional delay. The quantized DCT coefficients are to be transmitted as code words of variable lengths. The background program obtains the length for each coefficient from buffer IBITS. The DCT coefficients in buffer QTDCTS are ordered by word length. These word lengths are constrained to be monotonically decreasing, so that the word length of each DCT coefficient must be equal to or less than that of the previous coefficient.

When all parameters and coefficients have been serialized, the background program generates BCH error protection bits for the resulting bit stream. The protection bits are found using a table look-up procedure. Table 3-4 shows the protection table, TBENC. For each of the 45 bits that is set, the bit position is used as an index into this table. The 18-bit values found from the table are exclusive or'ed together to form the protection bits. Each of the 18-bit values occupies two half words, the 15 least significant bits in one half word and the remaining 3 bits in an adjacent half word.

When all error correction bits have been inserted into buffer SENS, the background program copies this buffer into either SEN1S or SEN2S, depending on AFLG being clear or set. The Transmit background program then terminates.

Bit Position	Value Low Order	Value High Order	Bit Position	Value Low Order	Value High Order
0	30764	7	23	21534	7
1	17466	4	24	21027	4
2	8733	2	25	10513	6
3	4336	5	26	5256	7
4	28843	5	27	29288	4
5	16505	1	28	14644	2
6	22544	3	29	7322	1
7	21540	6	30	30305	7
8	10770	3	31	17180	0
9	27941	6	32	8590	0
10	13970	7	33	4295	0
11	25445	4	34	2147	4
12	12722	6	35	1073	6
13	6361	3	36	536	7
14	29760	2	37	31008	4
15	14880	1	38	15504	2
16	25916	7	39	7752	1
17	19122	4	40	30472	7
18	9561	2	41	17320	4
19	4780	5	42	8660	2
20	29050	5	43	4330	1
21	16529	5	44	28761	7
22	22628	1			

TABLE 3-4: ERROR PROTECTION ENCODING TABLE TBENC

3.2.5 Analyze Process

The Analyze process performs the Adaptive Transform Coding algorithm on the digital speech waveforms supplied to it. The result of each instance of this process is a frame of quantized and coded sideband parameters and Discrete Cosine Transform coefficients.

The Analyze process is composed of six functional procedures, each of which consists of one or more array functions. The entire Analyze process runs in the Arithmetic Processor portion of the MAP. The six functional procedures are: Cycle Delay Lines, Compute DCT Coefficients, Compute and Quantize Sideband Parameters, Determine Basis Spectrum, Bit Allocation, and DCT Coefficient Quantization. The relationships of these procedures are shown in Figure 3-10.

3.2.5.1 Cycle Delay Lines Procedure (FCB 237, PBFCB 172)

The Cycle Delay Lines procedure partially implements the buffering between processes required by the ATC system. It consists of the single array function VMOV1, as shown in Figure 3-11. The buffer delays implemented in VMOV1 are shown in Figure 3-12.

VMOV1 first moves the quantized parameters (QPRM) computed in the previous instance of the Analyze process to the buffer (AQPB) used as input by the Transmit background program. Then, it cycles two buffer delay lines, one for the DCT coefficients (QTDCT1) and one for the corresponding bit allocations (MIBIT4). The outputs of these delay lines are moved to input buffers (AQTDCT and AIBIT4) used by the Transmit background program. The delay of one frame in the DCT coefficients and bit assignments is required for the proper operation of the Synthesize process. VMOV1 also sets flag XGO (integer scalar 115) which enables the Transmit background program to run.

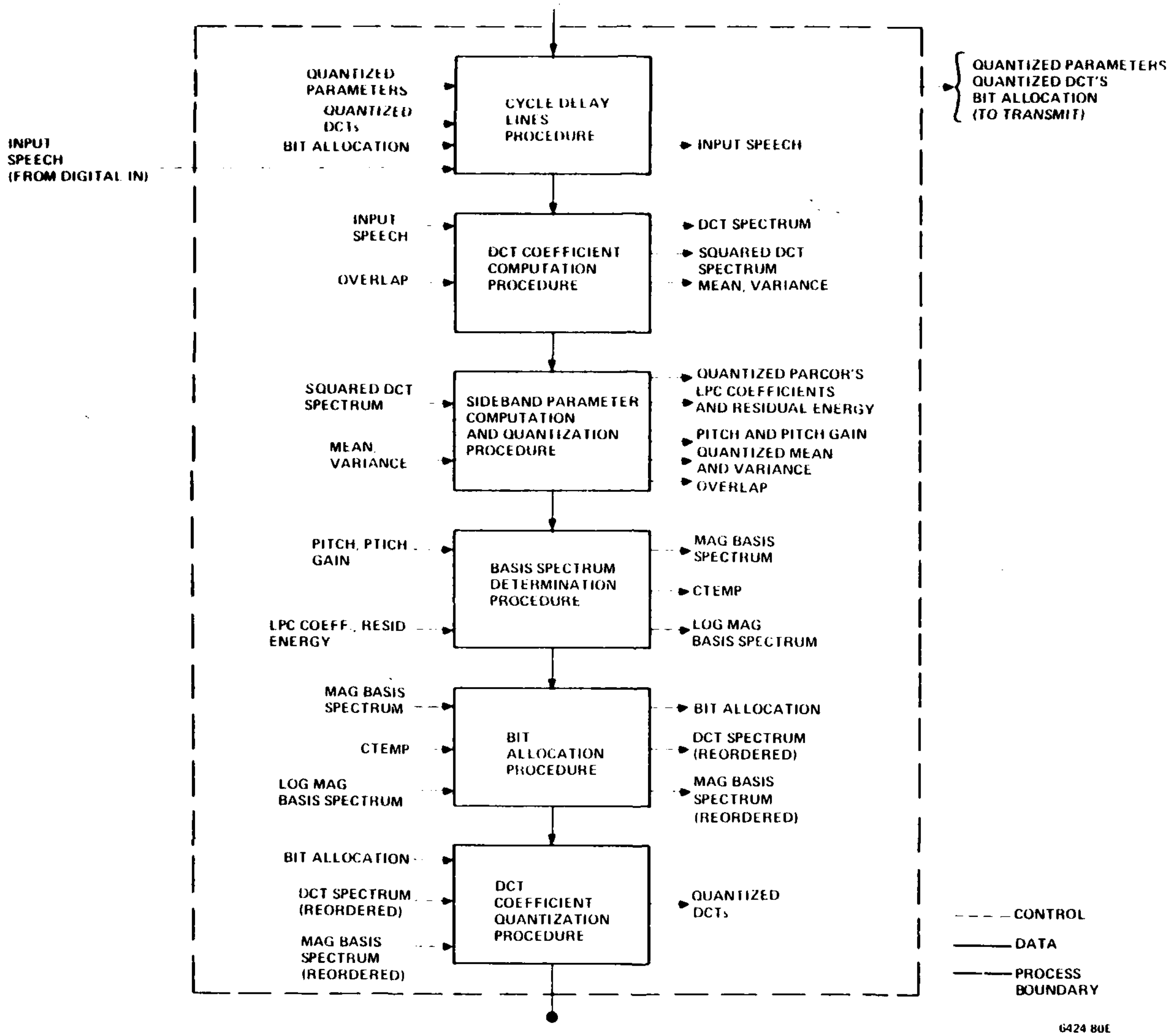


FIGURE 3-10: ANALYZE PROCESS

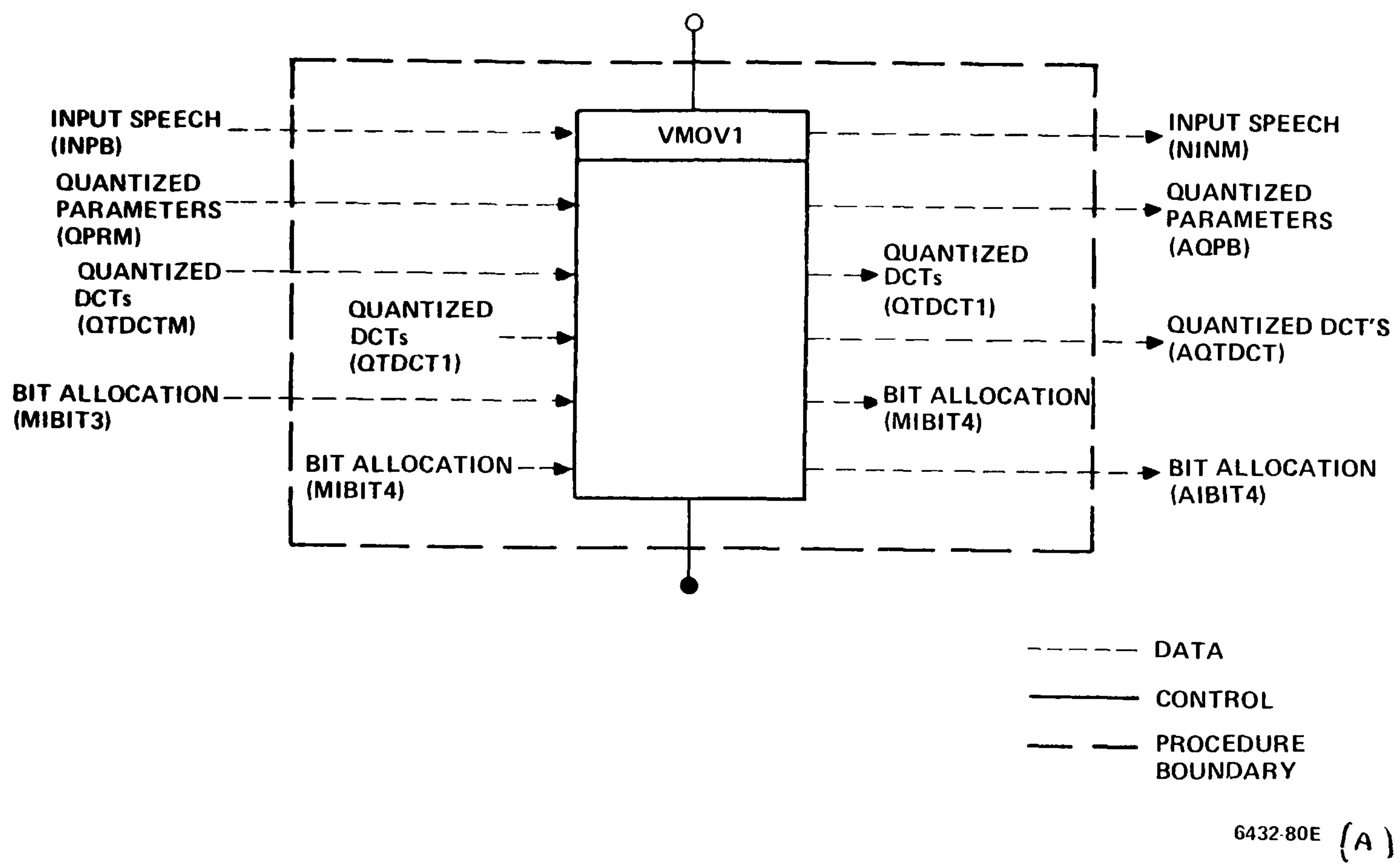


FIGURE 3-11: CYCLE DELAY LINES (ANALYZE)

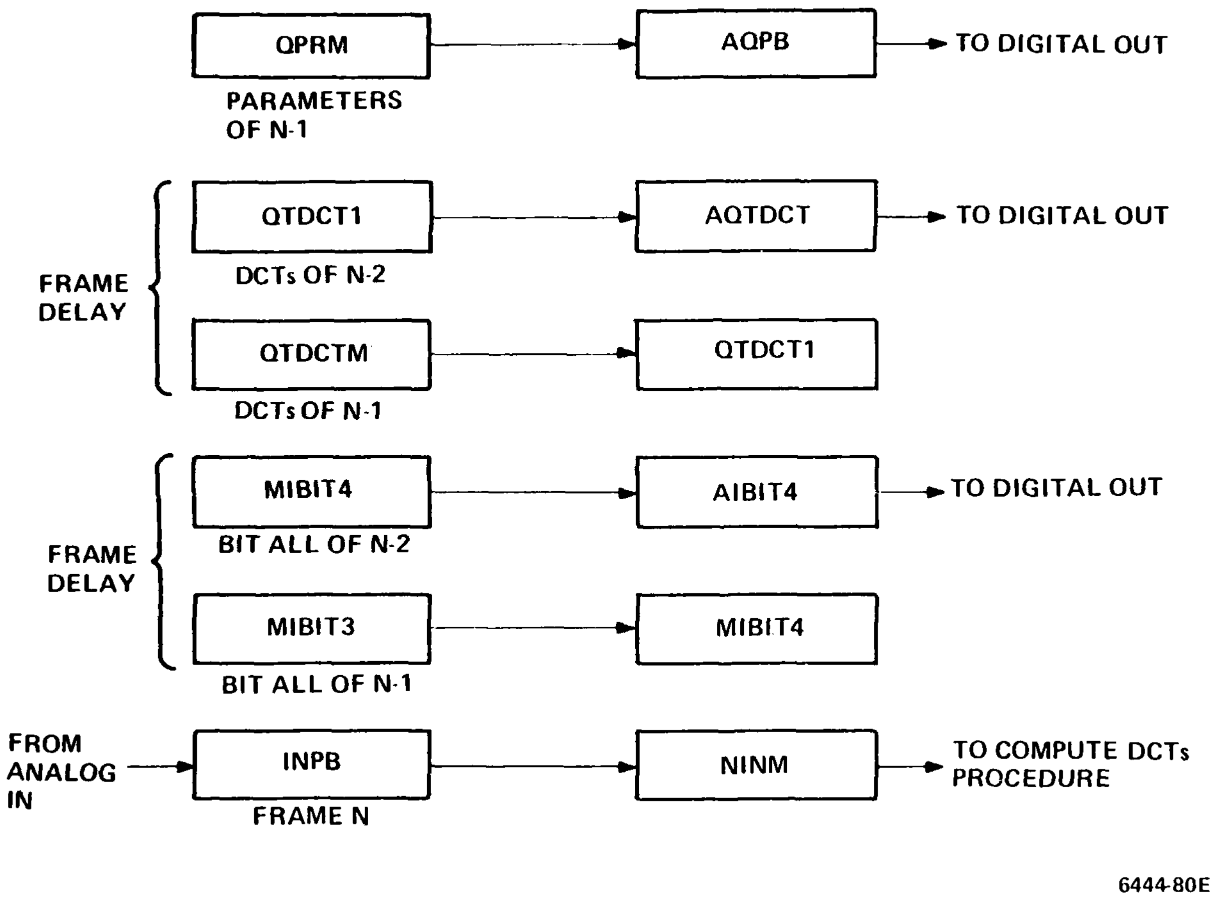


FIGURE 3-12: VMOV1 OPERATION FOR FRAME N

3.2.5.2 Compute DCT Coefficients Procedure

The Compute DCT Coefficients procedure is used to find the Discrete Cosine Transform representation of the current input frame. This procedure performs this transformation by using the appropriate pre-and post-processing around a Fast Fourier Transform. In addition, it performs the reformatting of the input data and the input frame overlapping.

This procedure consists of three array functions: MPFDVM, FFTN (supplied by CSPI), and MPDCTM. The relationships between these array functions are shown in Figure 3-13.

3.2.5.2.1 MPFDVM (FCB240, PBFCB 153)

MPFDVM converts the data in NINM from fixed point to floating point, concatenates these 246 samples with the last 10 input samples of the previous frame, computes and removes the mean and variance of the frame of 256 samples, and reorders the frame in preparation for the following FFT. The reordering algorithm, given by Makhoul[3], consists of generating a 256 point complex buffer, of which the imaginary points are all zero, the first half of the real points are the even numbered points of the original, and the second half are the reversed odd-numbered points of the original buffer. This array function stores the variance (in dB) in scalar 55 and stores the mean divided by the variance in scalar 52. The re-ordered input is stored in buffer X2.

3.2.5.2.2 FFTN (FCB 204, PBFCB 168)

The FFTN array function is supplied by CSPI. As used in the Compute DCT Coefficients procedure, it performs a 256 complex-to-complex FFT not in place. To do so, it makes use of a cosine table (VSHRT) generated during initialization. The output is stored in buffer X2.

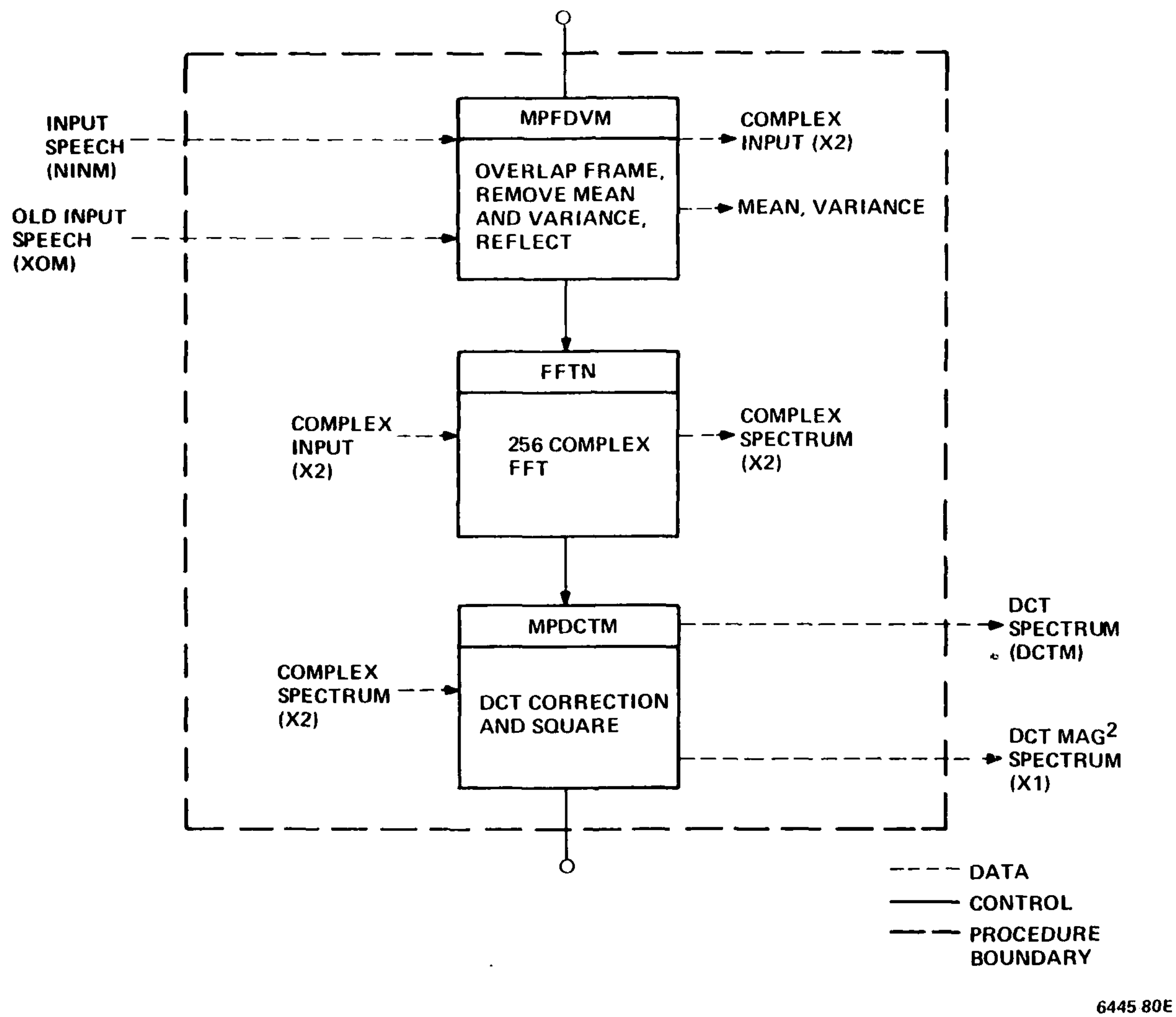


FIGURE 3-13: COMPUTE DCT COEFFICIENTS PROCEDURE

3.2.5.2.3 MPDCTM (FCB 241, PBFCB 154)

MPDCTM forms the 256 unique, real Discrete Cosine Transform coefficients from the 256 complex frequency samples generated by FFTN. It also forms the squared magnitudes of the DCT coefficients. These will be used later by the Compute and Quantize Sideband Parameters procedure for the calculation of an autocorrelation function.

The transformation from Complex Fourier coefficients is given by Makhoul[9]. Since the Fourier coefficients result from the FFT of a real sequence, they are conjugate symmetric and only the first half (128) of them need be used. These 128 Fourier coefficients are multiplied by one eighth of a cycle of a complex exponential. The real parts of the result form the first 128 DCT coefficients and the imaginary parts reversed form the negative of the remaining 128 DCT's. These 256 DCT coefficients are stored in buffer DCTM. The squares of the DCT's are also formed and are used to generate the real parts of 256 complex samples, which are used later. The imaginary parts of these 256 samples are zeroed. These 256 complex samples are then reflected to form a 512 point conjugate symmetric spectrum. The magnitude-squared spectrum is stored in buffer X2.

3.2.5.3 Compute and Quantize Sideband Parameters Procedure

This procedure consists of three array functions, as shown in Figure 3-14: FFTIN, MPMWGX, and MPQDPP. Its function is to compute the LPC spectrum and the pitch which are later used to form the bit allocation basis spectrum.

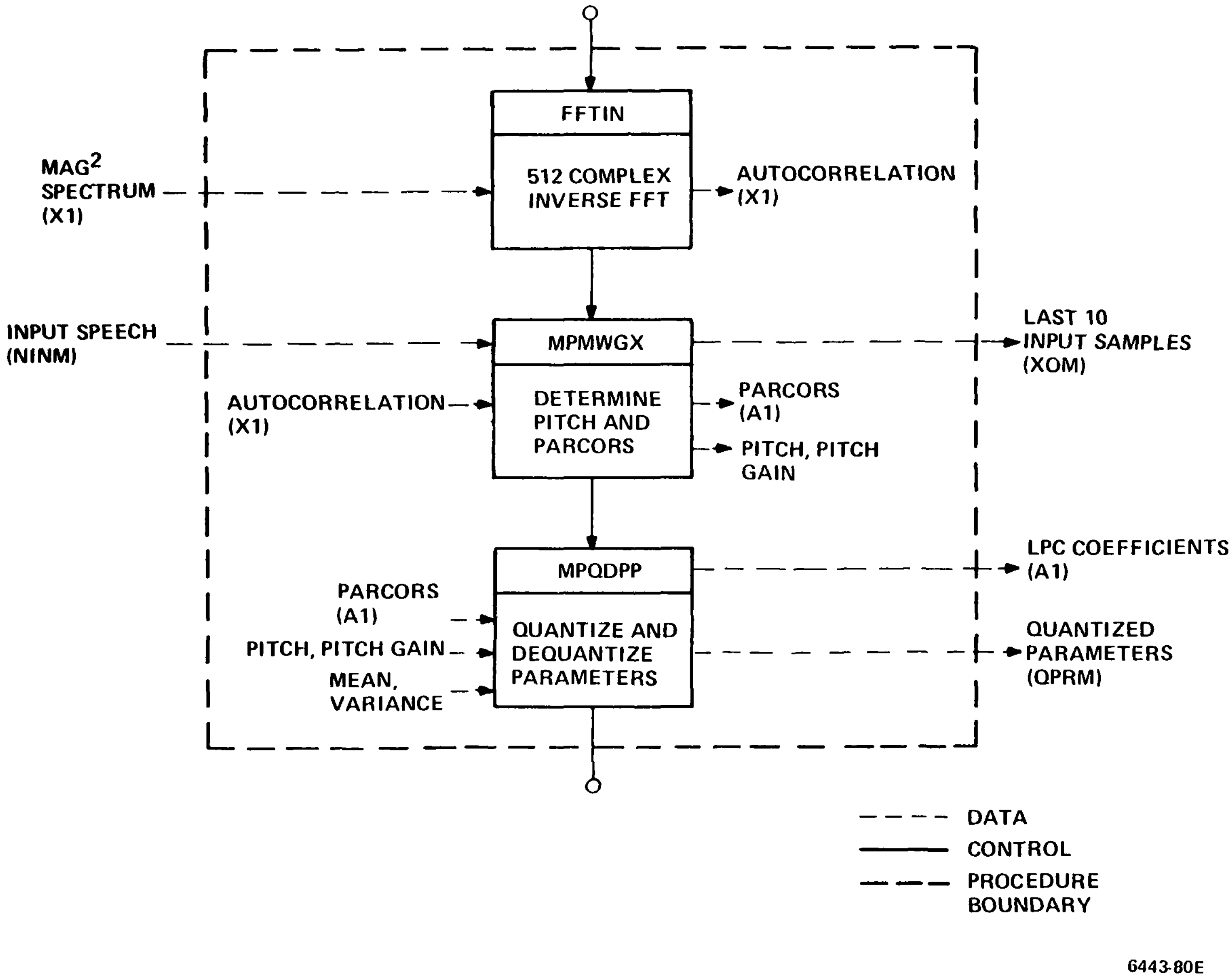


FIGURE 3-14: COMPUTE AND QUANTIZE SIDEBAND PARAMETERS PROCEDURE

3.2.5.3.1 FFTIN (FCB 206, PBFCB 169)

FFTIN is a CSPI supplied array function that performs an inverse Fast Fourier Transform, not in place. As used in Sideband Parameter procedure, it performs a 512 point complex transformation. Since the input to this function is a symmetric magnitude spectrum, the result is an autocorrelation function consisting of 256 unique real points.

3.2.5.3.2 MPMWGX (FCB 244, PBFCB 155)

The MPMWGX function determines 8 parcor coefficients and an estimate of the pitch and pitch gain from the autocorrelation input. In addition, it also copies the last 10 samples from the current input buffer to an area in the front of the input buffer. These samples will form the first 10 samples of the next frame.

The parcor coefficients are found using the Weiner-Levinson-Durbin algorithm as implemented by CSPI in the MWLD array function.

The pitch is estimated by finding the maximum autocorrelation value that lies between sample indices 15 and 94. The position of the maximum is used as the pitch period M, and the ratio of the maximum value to the zeroth autocorrelation is used as the pitch gain. The pitch is stored as a floating point number in scalar 61, and the pitch gain is stored in scalar 62.

3.2.5.3.3 MPQDPP (FCB 243, PBFCB 156)

This array function quantizes the 8 parcors determined earlier, as well as the mean, variance, pitch, and pitch gain. It also computes the associated dequantized values, in order that the basis spectrum generated from them later will be the same as that found by the receiver, assuming no uncorrectable channel errors. The quantization/dequantization tables used are shown in Table 3-5.

Once the parcors have been quantized and dequantized, they are used to find the corresponding Linear Prediction (LPC) coefficients. The following recursion is used to find the first through eighth LPC coefficients:

$$A_m(m) = -P(m)$$

$$A_m(L) = A_{m-1}(L) - P_m * A_{m-1}(M-L) \qquad 1 \leq m \leq 8$$
$$1 \leq L < m$$

where $A_i(j)$ is the jth LPC coefficient during the ith iteration and P(j) is the jth parcor coefficient. These LPC coefficients are used later in the Compute Basis Spectrum procedure. The residual energy of the LPC coefficients is also computed and stored as ENG in scalar 60.

The quantized, coded parameters are collected in buffer QPRM with the parcor coefficients first, followed by the mean, variance, pitch, and pitch gain.

3.2.5.4 Determine Basis Spectrum Procedure

This procedure, shown in Figure 3-15 computes the basis spectrum from the quantized pitch value and the LPC coefficients. It consists of three array functions: MPFSTV, FF2R, and MPBASP.

3.2.5.4.1 MPFSTV (FCB 245, PBFCB 157)

This array function forms two time-domain functions, one from the quantized pitch and one from the LPC coefficients, and stores them as real and imaginary parts of a complex buffer. The time domain function

Parameter	Threshold	Code	Dequantized Value	Parameter	Threshold	Code	Dequantized Value
PARCOR 1		31	1.997	PARCOR 2		31	0.087
	1.933				1.962		
		30	1.869			30	0.263
	1.789				1.926		
		29	1.703			29	0.369
	1.644				1.896		
		28	1.581			28	0.462
	1.541				1.867		
		27	1.502			27	0.556
	1.444				1.836		
		26	1.385			26	0.624
	1.343				10804		
		25	1.300			25	0.681
	1.240				1.768		
		24	1.181			24	0.736
	1.122				1.732		
		23	1.063			23	1.714
	1.019				1.695		
		22	0.975			22	1.676
	0.930				1.653		
		21	0.885			21	1.630
	0.843				1.602		
		20	0.802			20	1.574
	0.771				1.547		
		19	0.740			19	1.520
	0.696				1.492		
		18	0.652			18	1.464
	0.615				1.436		
		17	0.577			17	1.407
	0.543				1.373		
		16	0.509			16	1.338
	0.477				1.306		
		15	0.445			15	1.274
	0.418				1.238		
		14	0.392			14	1.203
	0.368				1.162		
		13	0.344			13	1.121
	0.324				1.085		
		12	0.304			12	1.048
	0.286				1.012		
		11	0.268			11	0.977
	0.251				0.942		
		10	0.235			10	0.907
	0.219				0.876		
		9	0.204			9	0.846
	0.190				0.819		
		8	0.176			8	0.792
	0.162				0.764		
		7	0.150			7	0.736
	0.137				0.709		
		6	0.124			6	0.681
	0.112				0.653		
		5	0.100			5	0.624
	0.090				0.590		
		4	0.080			4	0.556
	0.070				0.509		
		3	0.060			3	0.462
	0.051				0.415		
		2	0.042			2	0.369
	0.033				0.316		
		1	0.025			1	0.263
	0.014				0.175		
		0	0.008			0	0.087

TABLE 3-5: PARAMETER QUANTIZATION AND DEQUANTIZATION TABLE

Parameter	Threshold	Code	Dequantized Value	Parameter	Threshold	Code	Dequantized Value
PARCOR 3		15	1.844	PARCOR 4		15	1.950
	1.750				1.882		
		14	1.655			14	1.183
	1.585				1.754		
		13	1.516			13	1.695
	1.462				1.642		
		12	1.408			12	1.589
	1.358				1.539		
		11	1.309			11	1.489
	1.261				1.441		
		10	1.213			10	1.393
	1.168				1.346		
		9	1.123			9	1.300
	1.080				1.255		
		8	1.037			8	1.209
	0.995				1.165		
		7	0.953			7	1.121
	0.911				1.076		
		6	0.870			6	1.032
	0.829				0.988		
		5	0.787			5	0.943
	0.745				0.897		
		4	0.703			4	0.851
	0.658				0.801		
		3	0.613			3	0.752
	0.562				0.696		
		2	0.512			2	0.640
	0.452				0.572		
		1	0.392			1	0.504
	0.315				0.409		
		0	0.238			0	0.314

TABLE 3-5: PARAMETER QUANTIZATION AND DEQUANTIZATION TABLE (CONT'D)

Parameter	Threshold	Code	Dequantized Value	Parameter	Threshold	Code	Dequantized Value
PARCOR 5		7	1.591	PARCOR 6		7	1.621
	1.482				1.509		
		6	1.374			6	1.398
	1.297				1.319		
		5	1.219			5	1.240
	1.153				1.173		
		4	1.086			4	1.105
	1.022				1.041		
		3	0.958			3	0.976
	0.891				0.909.		
		2	0.825			2	0.842
	0.747				0.763		
		1	0.669			1	0.685
	0.557				0.573		
		0	0.445			0	0.462
PARCOR 7		3	1.320	PARCOR 8		3	1.382
	1.167				1.243		
		2	1.013			2	1.105
	0.875				0.985		
		1	0.736			1	0.865
	0.568				0.722		
		0	0.397			0	0.879

Mean		15	3.132
	2.854		
		14	2.576
	2.394		
		13	2.212
	2.072		
		12	1.932
	1.816		
		11	1.699
	1.598		
		10	1.496
	1.403		
		9	1.314
	1.231		
		8	1.148
	1.070		
		7	0.992
		6	0.846
		5	0.706
		4	0.572
		3	0.441
		2	0.313
		1	0.187
		0	0.062
Pitcngain		3	0.898
	0.824		
		2	0.750
	0.667		
		1	0.584
	0.488		
		0	0.392

Variance		31	60.709
	59.760		
		30	58.810
	58.016		
		29	57.223
	56.491		
		28	55.758
	55.072		
		27	54.385
	53.709		
		26	53.034
	52.334		
		25	51.633
	50.900		
		24	50.167
	49.431		
		23	48.695
	47.960		
		22	47.224
	46.483		
		21	45.741
	44.982		
		20	44.222
	43.443		
		19	42.664
	41.864		
		18	41.065
	40.204		
		17	39.343
	38.386		
		16	37.430
	36.477		
		15	35.525
	34.633		
		14	33.741
	32.888		
		13	32.035
	31.179		
		12	30.322
	29.409		
		11	28.495
	27.476		
		10	26.458
	25.415		
		9	24.372
	23.405		
		8	22.439
	21.426		
		7	20.412
	19.276		
		6	18.140
	16.994		
		5	15.849
	14.548		
		4	13.242
	12.05[illegible]		
		3	10.868
	9.831		
		2	8.794
	7.798		
		1	6.802
	5.631		
		0	4.460

Pitch (M)	Code	Decoded Pitch	Pitch	Code	Decoded Pitch	Pitch	Code	Decoded Pitch
0	0	15	32	17	32	64	48	63
1	0	15	33	18	33	65	49	65
2	0	15	34	19	34	66	49	65
3	0	15	35	20	35	67	50	67
4	0	15	36	21	36	68	50	67
5	0	15	37	22	37	69	51	69
6	0	15	38	23	38	70	51	69
7	0	15	39	24	39	71	52	71
8	0	15	40	25	40	72	52	71
9	0	15	41	26	41	73	53	73
10	0	15	42	27	42	74	53	73
11	0	15	43	28	43	75	54	75
12	0	15	44	29	44	76	54	75
13	0	15	45	30	45	77	55	77
14	0	15	46	31	46	78	55	77
15	0	15	47	32	47	79	56	79
16	1	16	48	33	48	80	56	79
17	2	17	49	34	49	81	57	81
18	3	18	50	35	50	82	57	81
19	4	19	51	36	51	38	58	83
20	5	20	52	37	52	48	58	83
21	6	21	53	38	53	85	59	85
22	7	22	54	39	54	86	59	85
23	8	23	55	40	55	87	60	87
24	9	24	56	41	56	88	60	87
25	10	25	57	42	57	89	61	89
26	11	26	58	43	58	90	16	89
27	12	27	59	44	59	91	62	91
28	13	28	60	45	60	92	62	91
29	14	29	61	46	61	93	63	93
30	15	30	62	47	62	94	63	93
31	16	31	63	48	63			

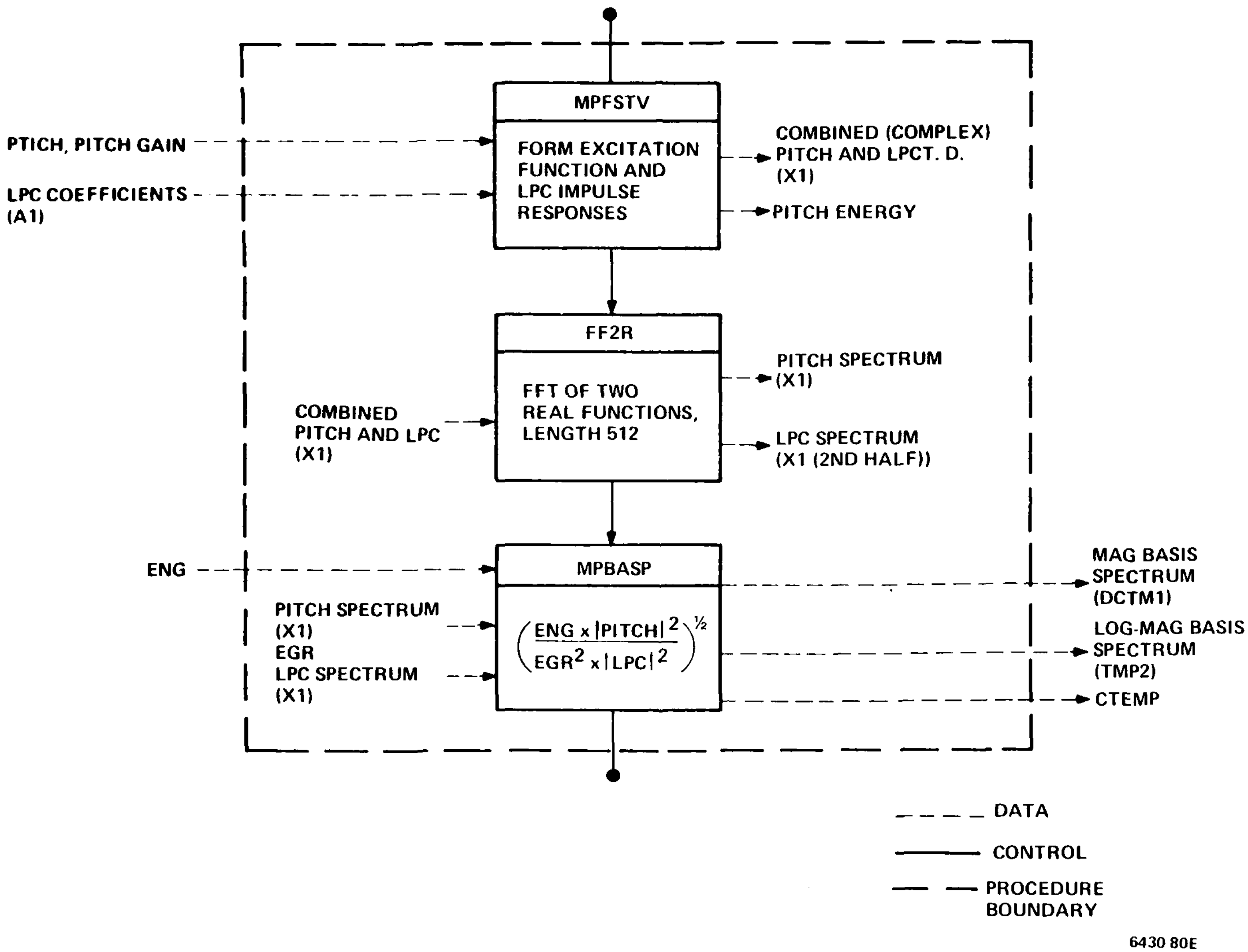


FIGURE 3-15: BASIS SPECTRUM DETERMINATION PROCEDURE

formed from the pitch represents the excitation function. It is an exponentially decaying pulse train with a period equal to the pitch and contains zeroes except at multiples of the pitch period. The values at the pulse points are the positive powers of the pitch gain. That is:

$$XR(K*M) = PG^{K+1} \qquad \text{for } 0 \leq K*M < 256$$

where M is the pitch period, PG the pitch gain, and XR is the output. The excitation is stored in the real part of the 512 point complex output buffer. The sum of the pitch pulses is accumulated and stored in scalar 73.

The other time domain function is the vocal tract filter function. It is formed from the impulse response of the LPC filter, which is simply 1, -A(1), -A(2), ... -A(8). The remaining 503 imaginary values are zeroed.

The two time domain functions, though both real, are stored in a complex buffer to facilitate the use of an efficient FFT algorithm, FF2R.

3.2.5.4.2 FF2R (FCB 214, PBFCB 170)

FF2R is a CSPI supplied array function used to transform two real signals simultaneously. One input function is stored in the real part of a complex buffer, and its transform appears in the first half of the complex buffer output. Similarly, the second function is stored in the imaginary parts of the input buffer, and its transform appears in the second half of the output buffer. As used in the ATC system, FF2R uses 512 point complex buffers as input and output. The output buffer is divided in half into two 256 point complex buffers. The first contains the frequency spectrum of the pitch excitation function. The second contains the frequency response of the vocal tract inverse filter.

3.2.5.4.3 MPBASP (FCB 253, PBFCB 158)

MPBASP forms the magnitude basis spectrum from the frequency spectrum and the inverse filter response. It also forms the base 2 logarithm of this basis spectrum to be used later in bit allocation.

This routine first calculates the magnitude squared of the basis spectrum by the following formula:

$$|\mathrm{BASIS}|^2 = \frac{\mathrm{ENG} * |\mathrm{Pitch}|^2}{\mathrm{EGR}^2 * |\mathrm{Inverse\ Filter}|^2}$$

where ENG is the residual energy of the LPC spectrum as calculated in MPQDPP and stored in scalar 60, and EGR is the sum of the excitation pulses as determined by MPFSTV and stored in scalar 73. Then one half the base 2 logarithm of the result is calculated. This is equivalent to the base 2 log-magnitude basis spectrum. For reasons of efficiency, the single MAP instructions LGS and RCP, which approximate the base 16 log and the reciprocal, are used. The DC component of the log magnitude basis spectrum is forced to be very small.

The magnitude basis spectrum is found as follows:

$$|\mathrm{BASIS}| = 2^{\left(\frac{\log|\mathrm{Basis}|^2}{2}\right)}$$

This exponentiation is performed using the same technique as the CSPI supplied array function VEXP. This approach was found to be faster than finding the square root directly.

The log base 2 magnitude basis spectrum output is stored in buffer TMP2 (256 real points). The magnitude basis spectrum output is stored in buffer DCTM1 (also 256 real points). Also, scalar 73, EGR, is modified to become 1.0/EGR. Finally, the sum of the log magnitudes is subtracted from the number of bits to be allocated, the difference is divided by 256, and the resultant average bit allocation is stored in CTEMP, scalar 80.

3.2.5.5 Bit Allocation Procedure

The bit allocation procedure determines how many bits are to be used to code each DCT coefficient for transmission under four constraints. The four constraints are: 1) each DCT must be coded with an integral number of bits; 2) that the sum of the bits allocated to the DCT coefficients must equal 267; 3) that the maximum number of bits allocated to any DCT coefficient is three; and 4) that the numbers of bits allocated to the DCT coefficients approximate the log-magnitude basis spectrum.

Satisfying these constraints exactly is a difficult, time consuming task. An approximate solution is found by reordering to log magnitude basis spectrum coefficients (and the corresponding DCT coefficients and magnitude spectrum coefficients) by size. The bit allocation for the reordered DCT coefficients will then be monotonically decreasing. Monotonically decreasing bit allocations are useful for horizontal encoding and protection.

The bit allocation procedure consists of three array functions, as shown in Figure 3-16. They are MPASRT, which performs the reordering, MPSCAN, which determines the preliminary bit allocation, and MPCDBA, which performs a correction on the preliminary bit allocation to produce a final bit allocation.

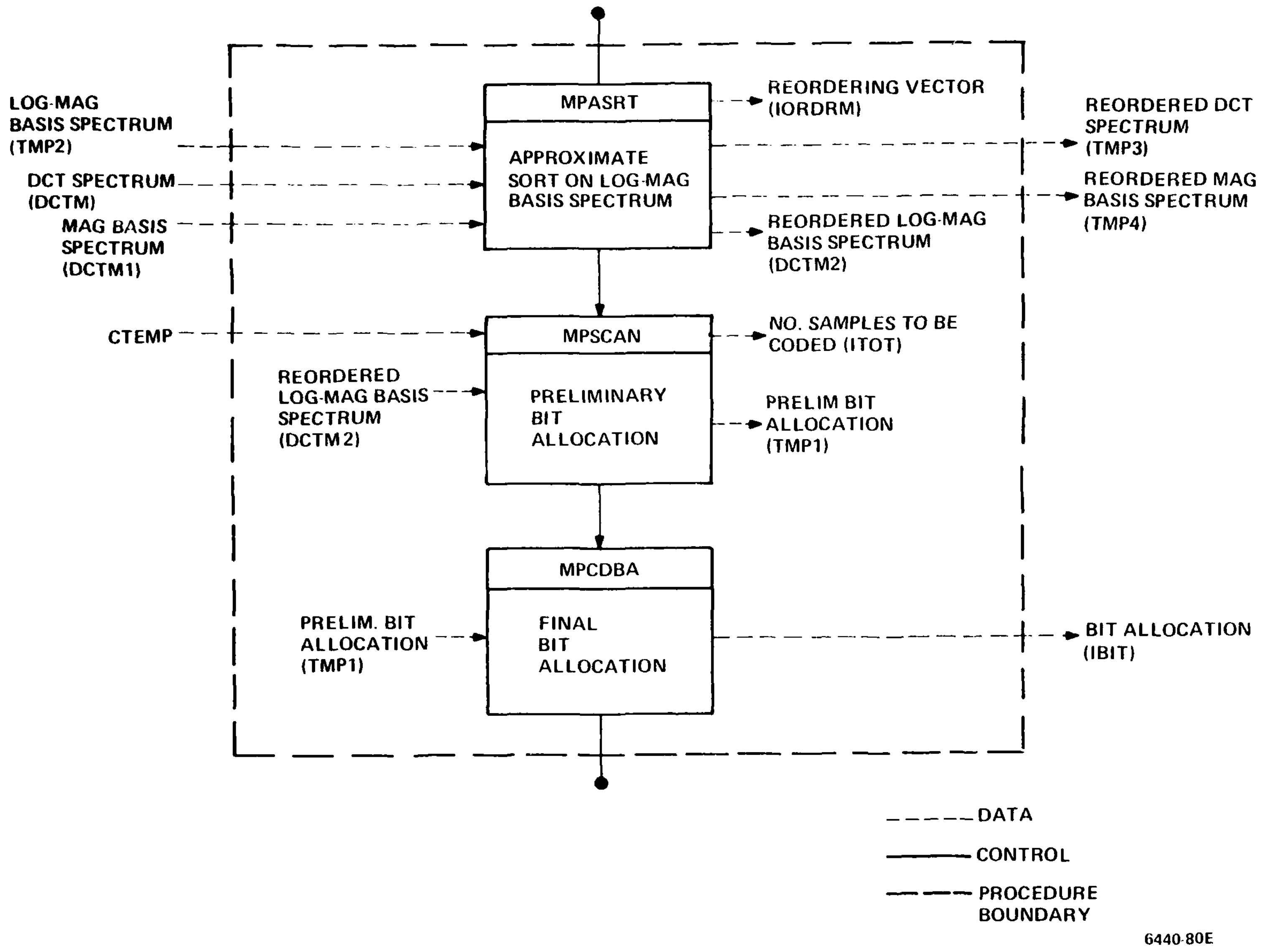


FIGURE 3-16: BIT ALLOCATION PROCEDURE

3.2.5.5.1 MPASRT (FCB 249, PBFCB 159)

This array function performs an approximate sort on the log-magnitude basis spectrum coefficients in buffer TMP2. It determines a vector of indices expressing the sorted order, and reorders the DCT coefficients (in buffer DCTM) and the magnitude basis spectrum coefficients (in DCTM1) accordingly.

The sort is performed by assigning each log-magnitude basis spectrum coefficient to one of four bins, depending on size. The index, the position in the buffer of each coefficient, is added to an ordering list corresponding to the bin. The thresholds for the four bins are shown in Table 3-6. When all indices have been added to one of the four lists, the lists are concatenated to form a complete ordering vector in buffer IORDR.

Once IORDR has been obtained, the three input buffers DCTM, (DCT coefficients), DCTM1 (magnitude basis spectrum), and TMP2 (log-magnitude basis spectrum) are reordered by using IORDR as indirect addresses into each of the input buffers and forming sequential outputs. That is,

$$\text{OUTPUT}(I) = \text{INPUT}\big(\text{IORDRM}(I)\big)$$

The three output buffers are TMP3, TMP4, and DCTM2, respectively.

The required indirect addressing is achieved by performing a "scatter write" into APS program memory to modify the addresses used in the input instructions.

3.2.5.5.2 MPSCAN (FCB 254, PBFCB 160)

This array function proceeds in two passes to form a preliminary bit allocation. The first calculates a new log magnitude spectrum whose

Bin #	Threshold
3	
	2.907
2	
	1.322
1	
	-0.488
0	

TABLE 3-6: BIT ALLOCATION THRESHOLDS

mean is equal to the average number of bits per sample to be allocated. It also determines the last sample of this new spectrum to be coded. That is, the samples which follow will be allocated zero bits. It also keeps a running sum of the samples of the new spectrum to be coded.

The second pass generates another log magnitude spectrum whose mean is the average number of bits available per sample to be coded and whose trailing samples are zero. The coefficients of this spectrum are then rounded and fixed to form the preliminary bit allocation.

Inputs to MPSCAN are buffer DCTM2, the log-magnitude basis spectrum, and CTEMP, the average bit offset to be applied in pass 1, computed in MPBASP and stored in scalar 80. The output is stored as floating point numbers in buffer TMP1. In addition, ITOT, the number of samples to be coded, is stored in scalar 83.

3.2.5.5.3 MPCDBA (FCB 255, PBFCB 161)

This array function consists of at least four passes. The first pass forces the preliminary bit allocations to be monotonically decreasing and sums them. If this sum ever exceeds the available number of bits, BTLTH = 267, the remaining bit allocations in the input buffer are zeroed. The second pass clips the bit allocations at 3, and updates the total bits allocated sum accordingly. Pass 3 allocates the leftover bits, beginning at the front of the buffer, but never raising the bit allocation of any coefficient over 3.0. If some bits remain to be allocated at the end of this pass, the pass is repeated. Finally, the bit allocations are changed from floating point integers to fixed point numbers and stored.

The input to MPCDBA is buffer TMP1, the preliminary bit assignment. The output is stored in IBIT.

3.2.5.6 DCT Coefficient Quantization Procedure (FCB 250, RBFCB 162)

This procedure, shown in Figure 3-17, consists of one array function, MPFSTQ. From the bit allocations in IBIT, this array function determines the number of quantization levels and the associated thresholds for each (reordered) DCT coefficient. The thresholds are multiplied by the corresponding magnitude basis spectrum value and then used to quantize and code the DCT coefficients into code words of various lengths. The thresholds, quantization levels, and resultant code words are shown in Table 3-7.

The inputs to MPFSTQ are IBIT; TMP3, the reordered DCT coefficients; and TMP4, the reordered magnitude basis spectrum. The output of MPFSTQ is stored as fixed point numbers in buffer QTDCTM.

3.2.6 Synthesize Process

The Synthesize process performs the Adaptive Transform Decoding algorithm to obtain digital speech waveforms. The input to this process is coded, quantized sideband parameters and coded, quantized DCT coefficients.

The Synthesize process is composed of seven functional procedures: Cycle Delay Lines, Dequantize Sideband Parameters, Basis Spectrum Determination, Bit Allocation, Copy Temporary Buffers, DCT Coefficients Dequantization, and Inverse DCT Computation. The relationships between these procedures are shown in Figure 3-18. Each procedure consists of one or more array functions. The entire Synthesize process runs in the Arithmetic Processor portion of the MAP.

Code	3 Bit Threshold	2 Bit Threshold	1 Bit Threshold
3			
	2.3796	-	-
2			
	1.2327	-	-
1			
	0.5332	1.1269	-
0			
	0	0	0

Note: Negative values are quantized similarly, sign bit of code is set

TABLE 3-7: DCT COEFFICIENT QUANTIZATION

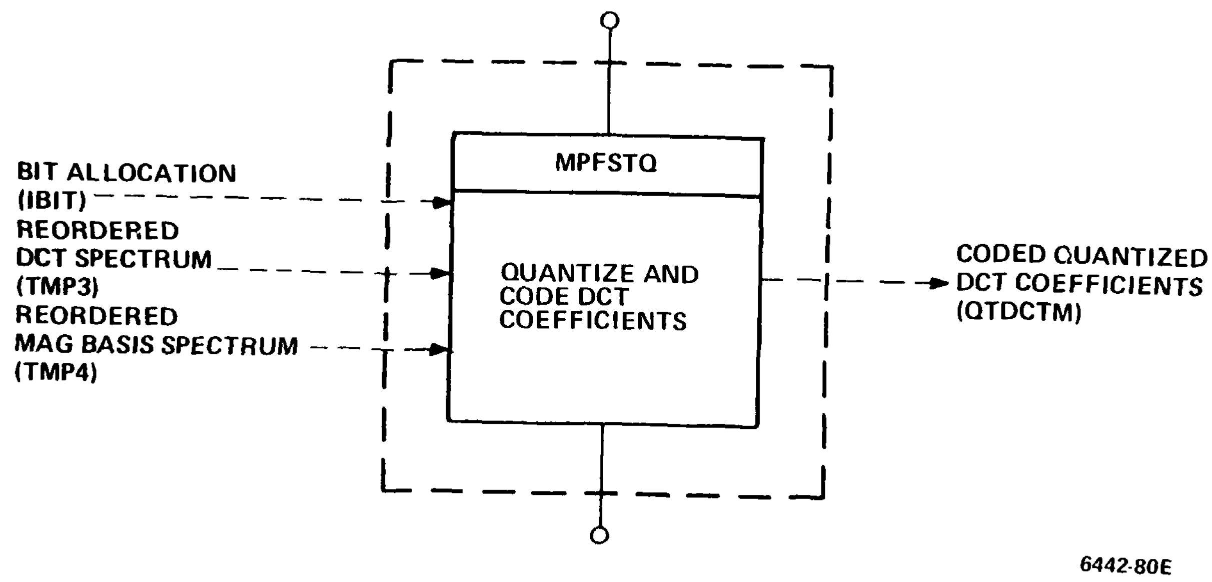


FIGURE 3-17: DCT QUANTIZATION PROCEDURE

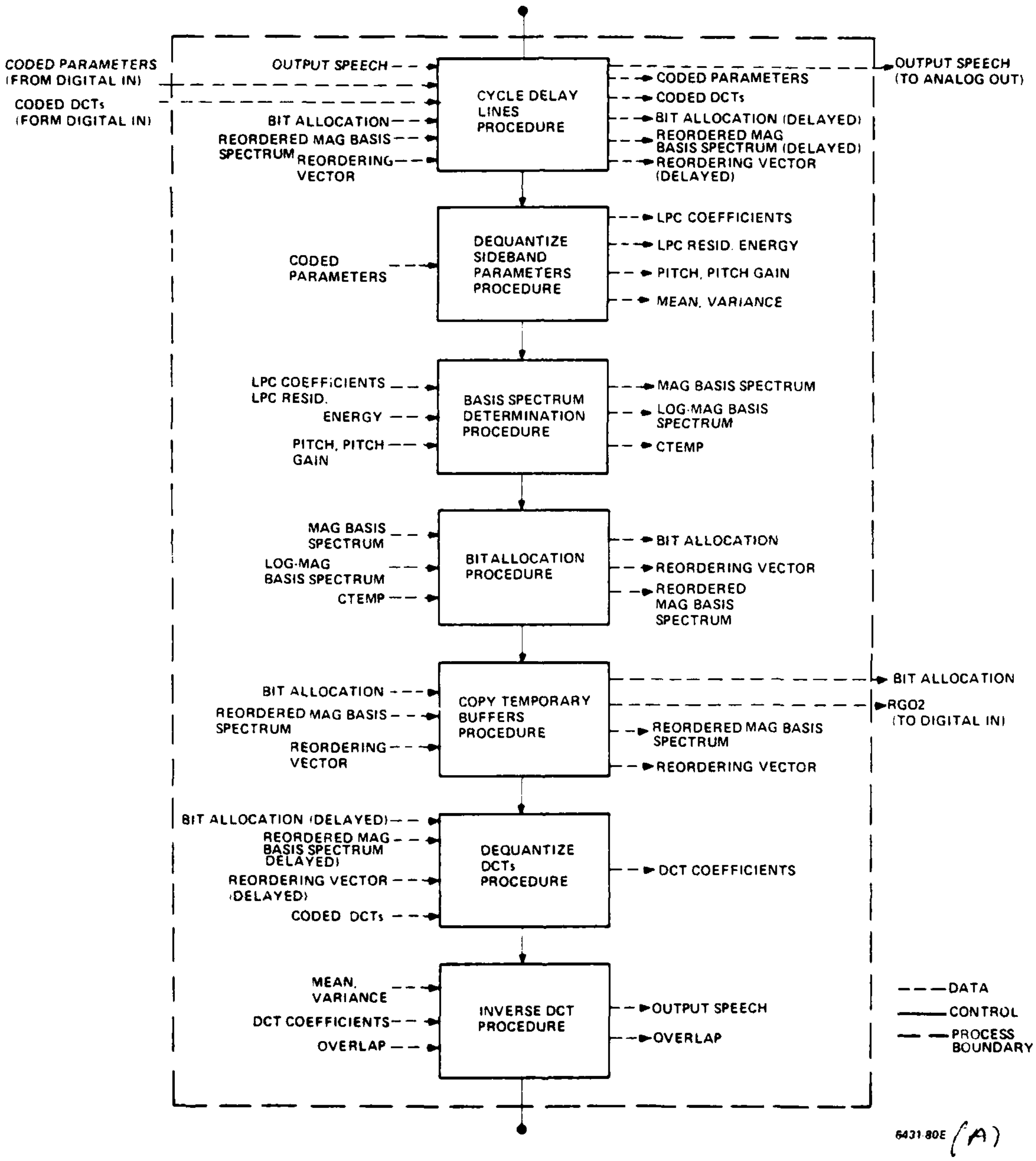


FIGURE 3-18: SYNTHESIZE PROCESS

3.2.6.1 Cycle Delay Lines Procedure (FCB 238, PBFCB 173)

The Cycle Delay Lines procedure consists of a single array function, VMOV2. It provides data to the Analog Out process and accepts data from the Digital In process, enables the Receive background program, and provides a single frame delay. This procedure is shown in Figure 3-19.

Output speech data from the previous instance of the Synthesize process is stored in buffer NOUT. VMOV2 copies this data to buffer OUTB, which will be used by the Analog Out process. VMOV2 sets the flag OUTFUL (integer scalar 111) to indicate that new data is present, thus enabling the A/D background program.

Similarly, VMOV2 copies data from two buffers, ARQPB and ARQDCT. These buffers have been filled by the Receive background program and contain the de-serialized sideband parameters and DCT coefficients. VMOV2 copies these buffers into QPRM and QRDCTM and sets flag RGO to indicate that the input buffers are empty, enabling the Receive background program to execute and fill them again.

The input sideband parameters in QPRM do not represent the same frame of speech as the DCT coefficients in QRDCTM. Because the bit allocation, which is derived from the sideband parameters, must be obtained before the DCT coefficients can be deserialized, the transmitted DCT coefficients lag the transmitted sideband parameters by one frame. This means that the bit allocation, basis spectrum, and reordering vector determined in one instance of the Synthesize processs must be saved to be used in the next instance of the process. VMOV2 implements this delay, as shown in Figure 3-20. The bit allocation, basis spectrum, and reordering vector have been stored in buffers MIBIT1, DCTIT1, and IORDR1, by the

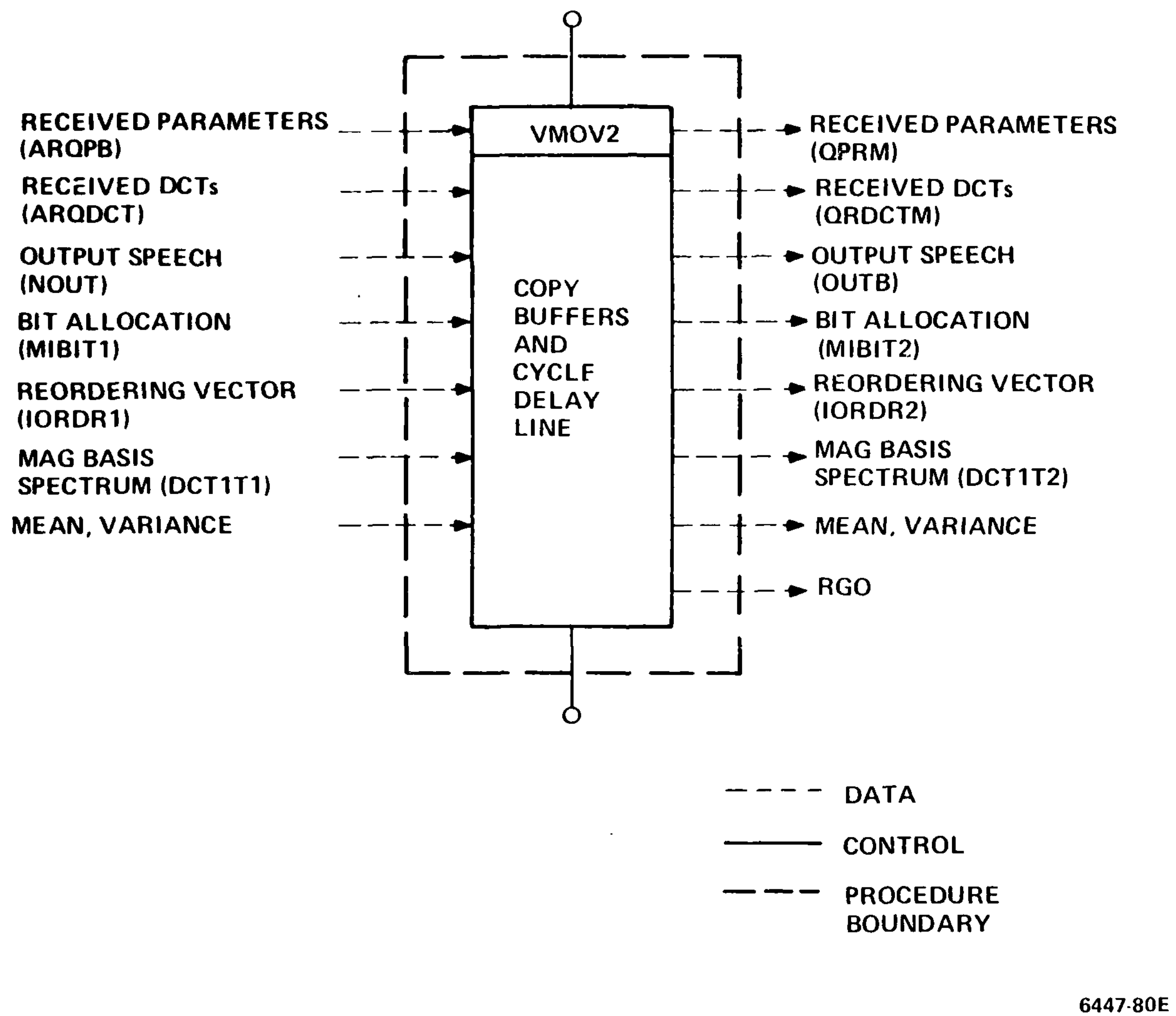


FIGURE 3-19: CYCLE DELAY LINES PROCEDURE (SYNTHESIZE)

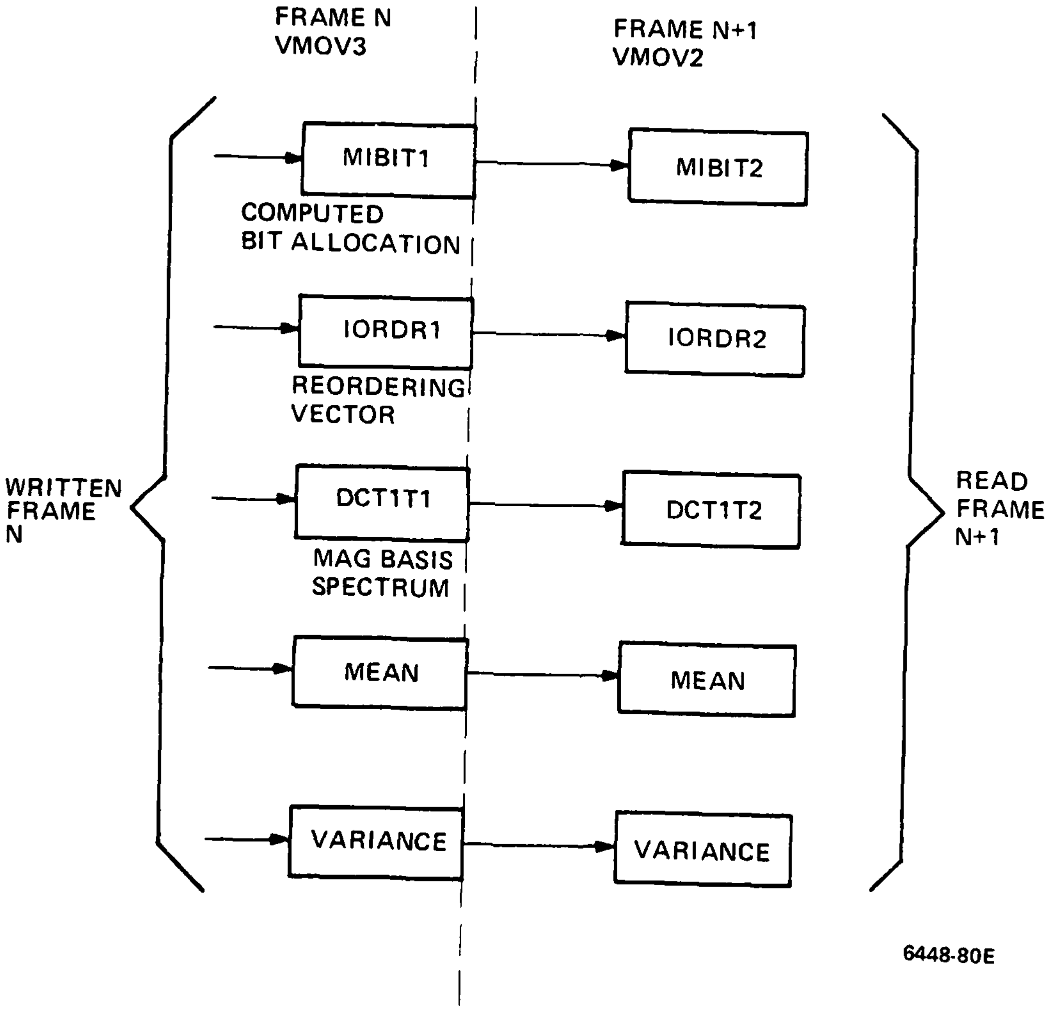


FIGURE 3-20: SYNTHESIZE DELAY LINES

previous instance of the process, and are moved by VMOV2 to MIBIT2, DCT1T2, and IORDR2, from which they will be used in the DCT dequantization procedure. The mean and variance have been stored in scalars 102 and 103 and are moved to scalars 100 and 101 to implement a similar frame delay on these parameters.

3.2.6.2 Sideband Parameter Dequantization Procedure (FCB 244, PBFCB 163)

This procedure, shown in Figure 3-21, consists of a single array function, MPDQPP. MPDQPP decodes 12 sideband parameters: 8 parcor coefficients, pitch, pitch gain, mean, and variance using the same tables used by MPQDPP (subsection 3.2.5.3.3). The input parameters are fixed point integers and are contained in buffer QPRM. These integers are used as indices into the appropriate decoding tables. The dequantized parcor coefficients are transformed to LPC coefficients as in MPQDPP, which are then output to buffer A1. The residual energy of the LPC coefficients (ENG) is stored in scalar 60. The dequantized pitch is output to scalar 91, the pitch gain to scalar 90, the variance to scalar 89, and the mean to scalar 88.

3.2.6.3 Basis Spectrum Determination Procedure

This procedure is identical to the Basis Spectrum Determination performed in the Analyze process (subsection 3.2.5.4, Figure 3-15). This procedure uses MPFSTV, FF2R, and MPBASP to determine the log-magnitude basis spectrum, stored in TMP2, and the magnitude basis spectrum, stored in DCTM1. If there are no uncorrectable transmission errors, these spectra will be identical to those determined in the Analyze process.

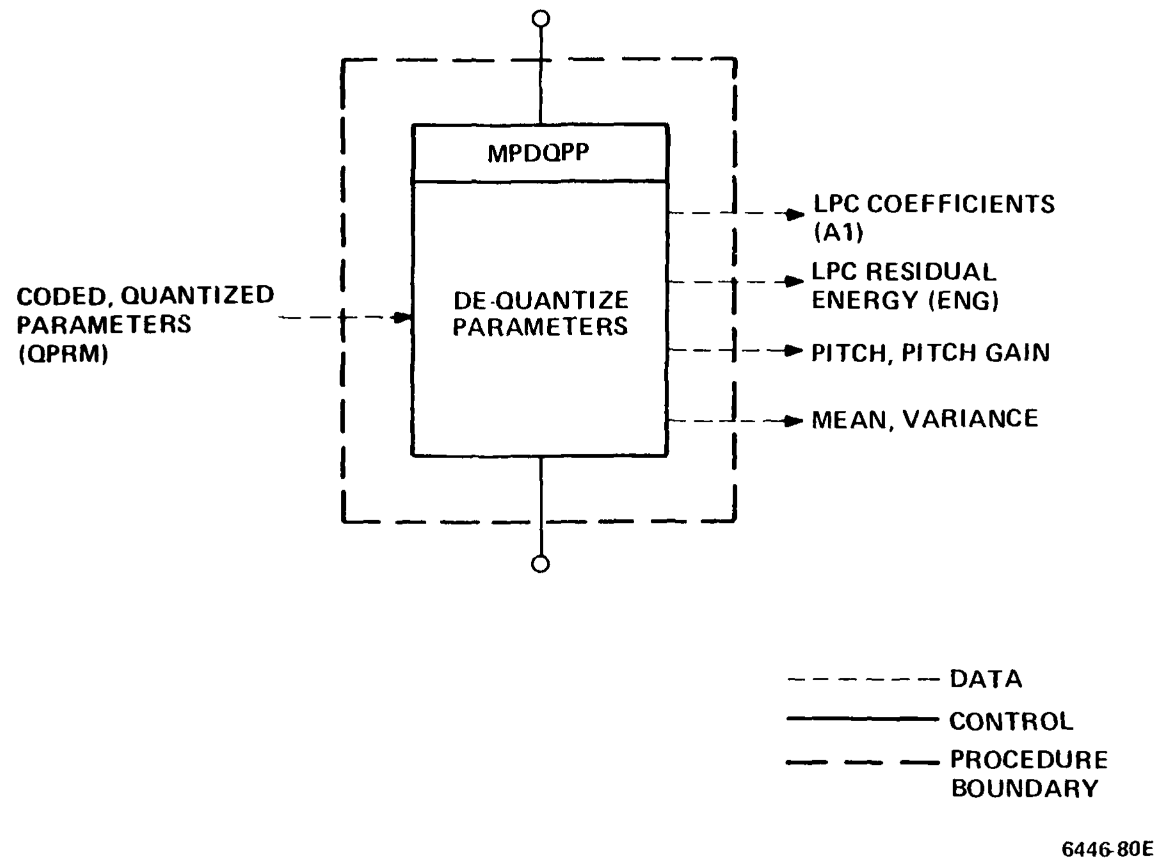


FIGURE 3-21: PARAMETER DEQUANTIZATION PROCEDURE

3.2.6.4 Bit Allocation Procedure

This procedure determines the number of bits used to code each of the DCT coefficients. It is very similar to the bit allocation procedure used in the Analyze process (subsection 3.2.5.5) except that the sorting function is slightly changed.

The bit allocation procedure in the Synthesize process also consists of three array functions: MPSSRT, MPSCAN, and MPCDBA. This procedure is shown in Figure 3-22.

3.2.6.4.1 MPSSRT (FCB 248, PBFCB 167)

This array function is similar to MPASRT (subsection 3.2.5.5.1) in that it reorders the magnitude and log-magnitude basis spectra contained in buffers DCT1 and TMP2, respectively, and stores the reordering information in a vector of indices, IORDRM. However, MPSSRT performs no reordering on the (coded) DCT coefficients. The coded DCT coefficients are transmitted in sorted order. They will be reordered to normal order in the dequantization procedure (subsection 3.2.6.6) later.

3.2.6.4.2 MPSCAN (FCB 254, PBFCB 160)

This is the same array function used in the Analyze process and described in subsection 3.2.5.5.2. It uses the same input and output buffers and is invoked using the identical prebound FCB.

3.2.6.4.3 MPCDBA (FCB 255, PBFCB 175)

This is the same array function used in the Analyze process and described in subsection 3.2.5.5.3. It is the identical FCB, but prebound differently. As used in the Synthesize process, it is prebound FCB

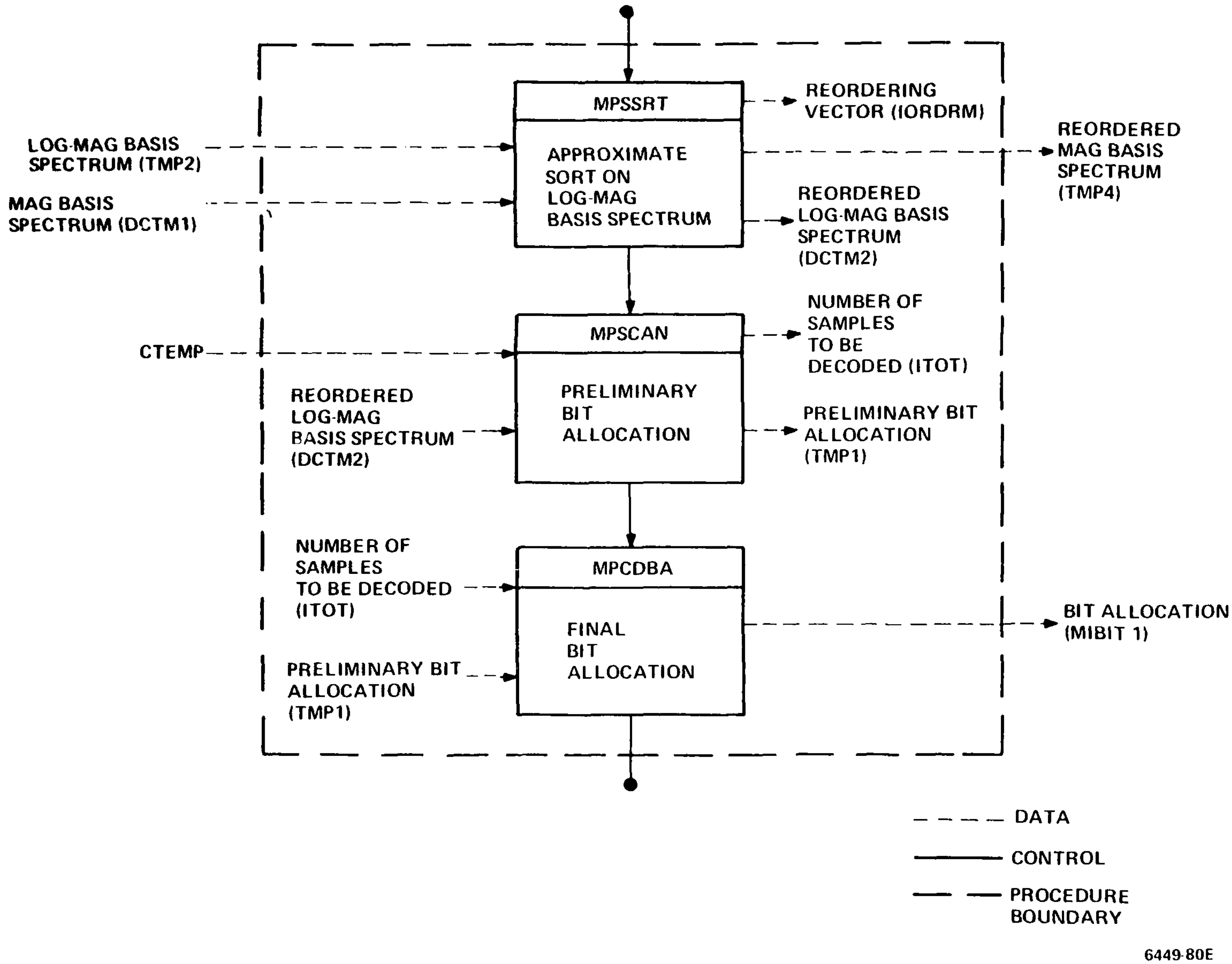


FIGURE 3-22: BIT ALLOCATION PROCEDURE (SYNTHESIZE)

175. The only difference made by this alternate prebinding is that the output buffer, which stores the bit allocations, is MIBIT1 instead of MIBIT2, as is used in the Analyze process.

3.2.6.5 Copy Temporary Buffers Procedure (FCB 240, PBFCB 174)

This procedure, shown in Figure 3-23, consists of a single array function, VMOV3. Its purpose is to copy data from temporary buffers, used to speed up computation, to permanent buffers, which will remain unmodified until the next instance of the Synthesize process. It also makes the DCT bit allocation information available to the Receive process and enables the Receive process to proceed by setting the flag RG02 (integer scalar 116).

VMOV3 copies buffer IORDRM, the reordering indices, into buffer IORDR1, which is the first stage of the Synthesize ordering delay line described in subsection 3.2.6.1. It also copies TMP4, the reordered magnitude basis spectrum, into DCT1T1, the first stage of the Synthesize basis spectrum delay line. Buffer MIBIT1, the first stage of the Synthesize bit allocation delay line, already contains the bit allocation information, as a result of MPCDBA. VMOV3 copies this information into buffer RIBIT, which will be used by the Receive process to deserialize the received DCT coefficients, and sets flag RG02 to enable the Receive process to proceed with this deserialization. VMOV3 also copies the dequantized mean and variance from scalars 88 and 89 to scalars 102 and 103.

3.2.6.6 DCT Coefficient Dequantization Procedure (FCB 252, PBFCB 164)

This procedure, shown in Figure 3-24, consists of a single array function, MPFSTD. MPFSTD uses each coded DCT coefficient (sorted order)

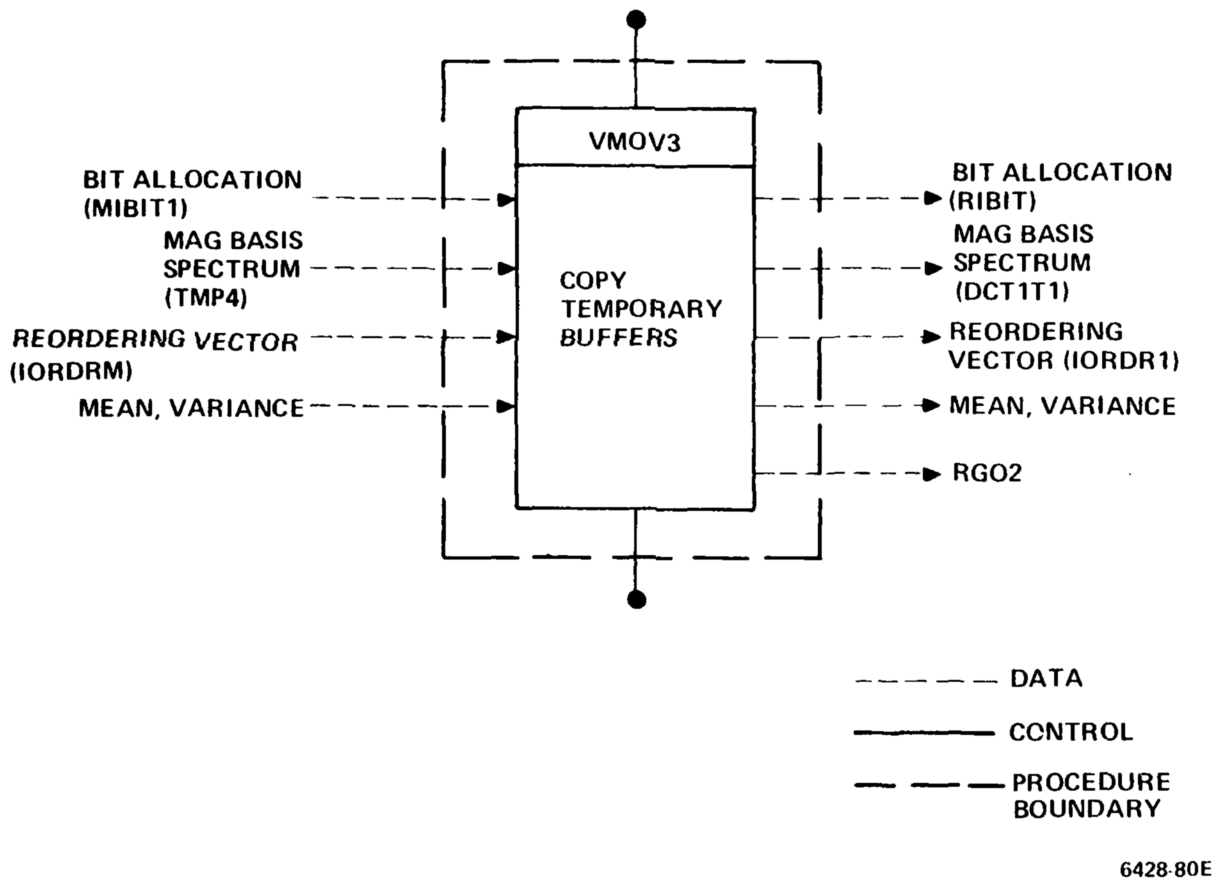


FIGURE 3-23: COPY TEMPORARY BUFFERS PROCEDURE

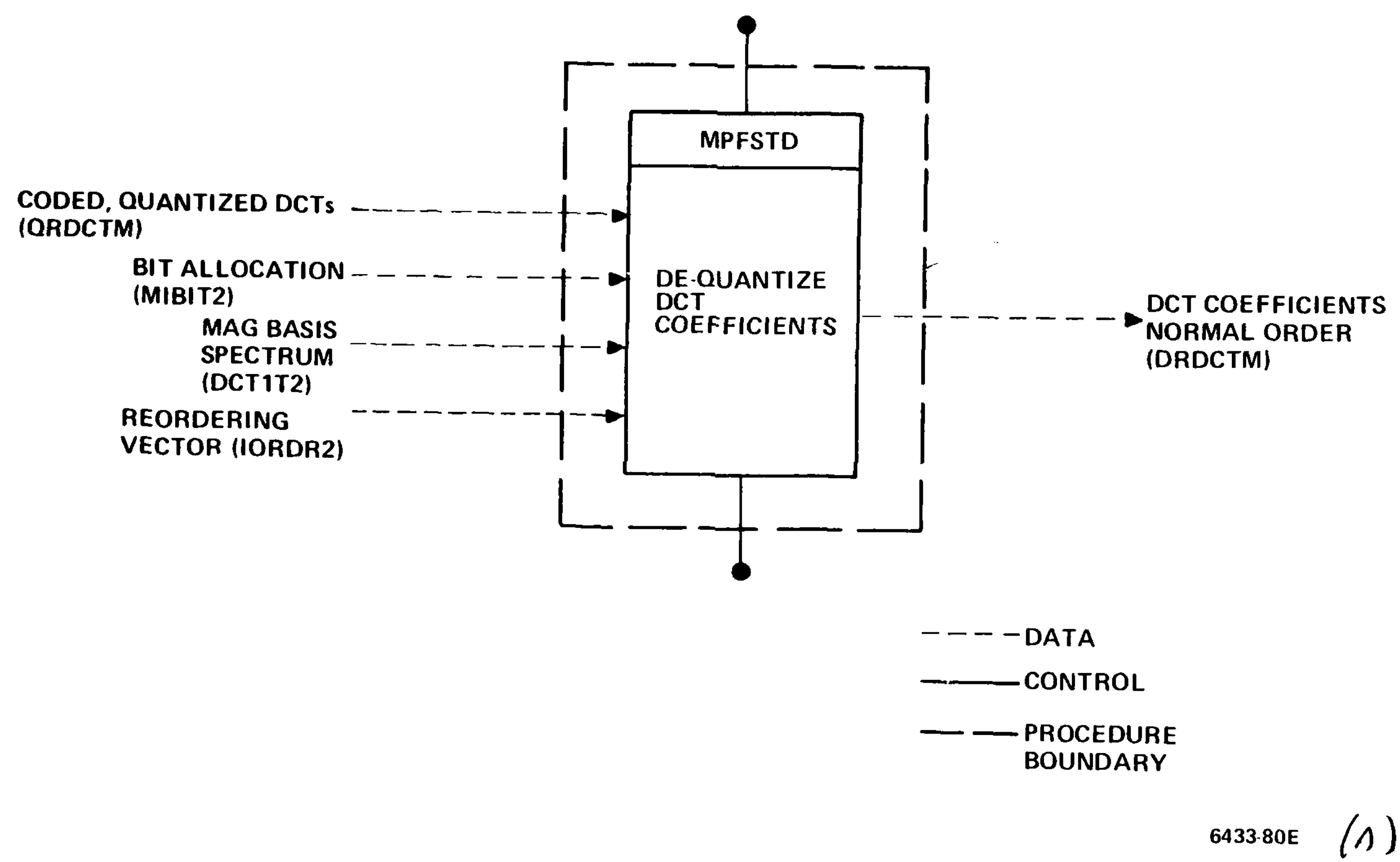


FIGURE 3-24: DCT DEQUANTIZATION PROCEDURE

from buffer QRDCTM as an index into a decoding table. The particular table used is determined from the bit allocation vector, MIBIT2. The value found from the decoding table is multiplied by the corresponding magnitude basis spectrum coefficient (DCT1T2). The resulting product is the dequantized DCT coefficient and is stored in buffer DRDCTM, using the reordering vector (ICRDR2) as the index into this output buffer so that the results will be in normal order. That is,

$$DRDCTM\left(IORDR2\ (I)\right) = Decode\left(QRDCTM(I)\right) \times DCT1T2(I)$$

The decoding tables are given in Table 3-8.

3.2.6.7 Inverse DCT Procedure

This procedure, shown in Figure 3-25, consists of three array functions, MPIDCM, FFTIN, and MPVDNM. It transforms the dequantized DCT coefficients back to a time domain waveform, restores the variance and mean to this waveform, and overlaps it with the waveform obtained in the previous frame to generate the output speech waveform.

3.2.6.7.1 MPIDCM (FCB 249, PBFCB 165)

This array function performs the inverse of the correction applied by the Analyze process array function MPDCTM (subsection 3.2.5.2.3). It forms a conjugate symmetric spectrum to which an inverse FFT will later be applied from the dequantized DCT coefficients in buffer DRDCTM. The real and complex parts of the output buffer X2 $\left(XR(K)\text{ and }XI(K)\right)$, respectively, are formed as follows:

Code	3 Bit Value	2 Bit Value	1 Bit Value
7	-3.0867	-	-
6	-1.6725	-	-
5	-0.8330	-	-
4	-0.2334	-	-
3	3.0867	-1.8340	-
2	1.6725	-0.4196	-
1	0.8330	1.8340	-0.7071
0	0.2334	0.4196	0.7071

TABLE 3-8: DCT COEFFICIENT DEQUANTIZATION

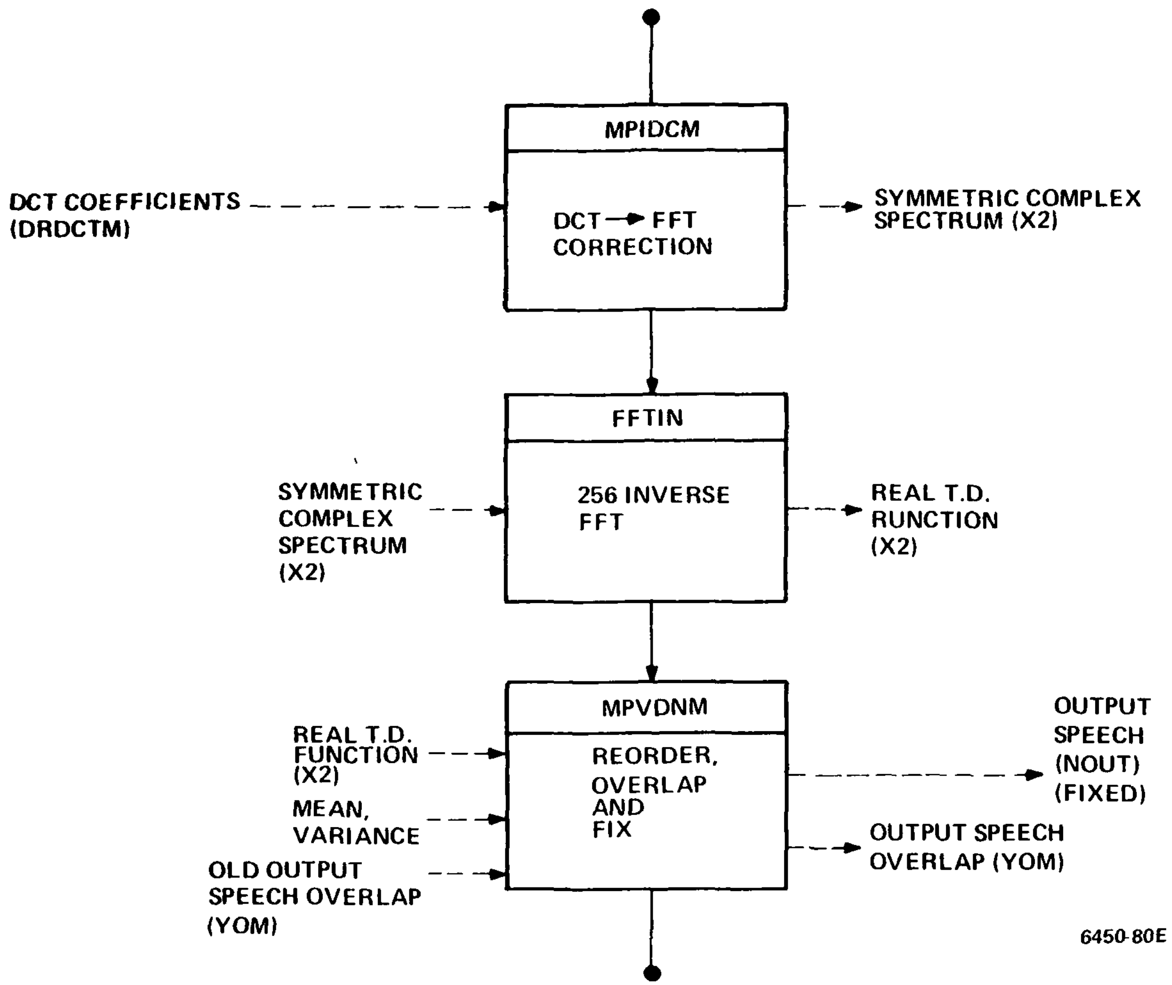


FIGURE 3-25: INVERSE DCT PROCEDURE

$$XR(K) = XR(N-K) = \cos(\pi K/2N)*DCT(K) + \sin(\pi K/2N)*DCT(N-K)$$

$$XI(K) = -XI(N-K) = \sin(\pi K/2N)*DCT(K) - \cos(\pi K/2N)*DCT(N-k)$$

$$XR(0) = DCT(0),\ XI(0) = \emptyset$$

where N is 256 and K varies between 1 and 255.

This array function also computes a corrected mean and variance from the (delayed) dequantized mean and variance. The dequantized variance (scalar 101) is in dB and is changed to be linear. The dequantized mean (scalar 100) is multiplied by this linear variance to form an adjusted mean. Finally, the linear variance is divided by the frame length to form the per sample variance. The resulting mean and variance are stored in scalars 100 and 101, respectively.

3.2.6.7.2 FFTIN (FCB 206, PBFCB 171)

This array function, supplied by CSPI, performs a 256 complex inverse FFT (not in place) on the corrected DCT coefficients in buffer X2. The resulting time domain function is also stored in X2.

3.2.6.7.3 MPVDNM (FCB 253, PBFCB 166)

This array function restores natural order to the 256 point time domain function in X2, multiplies it by the variance (scalar 101), adds the mean (scalar 100), and overlaps the result with the last 10 samples of the output from the previous frame to form 246 output speech samples which are stored in buffer NOVT.

The overlapping consists of multiplying the first 10 samples by an increasing ramp and the last 10 samples of the previous frame by a descending ramp and adding the results as follows:

OUTPUT(I) = NEW(I)*.1*I + OLD (236) * (1-.1*I)

where I ranges from 0 through 9.

The reordering of the time domain function forms the even numbered DCT coefficients from the first half of the real parts of buffer X2. The odd numbered coefficients are obtained by accessing the second half of the buffer in reverse order.

3.2.7 Executive Process

The Executive process is used to schedule the tasks performed by the Arithmetic Processor and the CSPU. The Executive, as supplied by CSPI, runs in the CSPU and assigns the AP sequential tasks from function lists, loading the AP with and starting it on a new array function as soon as it has finished with the previous one. Responding to interrupts and this AP scheduling are the only functions required of the CSPU in the standard SNAP software system. The ATC system, however, also uses the CSPU to perform computations required by the algorithm and unsuited to the AP, such as serialization and error correction. The standard SNAP executive has been modified to include this additional use of the CSPU.

3.2.7.1 Executive Modifications

The new CSPU program, which will be referred to as the Background program, actually includes the four background processes and a dispatch routine to schedule them. It is interfaced to the existing executive as a co-routine. However, in order to minimize modifications to the executive, it is implemented as a single interrupt routine containing multiple exits.

The standard Executive loads and starts the AP, prepares the next function to be loaded into the AP, and then waits for the AP to finish. The ATC modification changes this wait to a programmed interrupt (Device 24, level 8) which causes the Background program to continue from its last exit. The Background program periodically checks flag APDNFL (integer scalar 126) to determine whether the AP has finished. If it has, the Background program saves its state and exits, returning to the point in the Executive program following the original wait. The Executive then continues normally, restarting the AP and preparing the next array function, until it again reaches the state where it would wait for the AP.

The APDONE interrupt service routine, also part of the original Executive, has been modified for this new program structure. It now sets APDNFL before returning.

3.2.7.2 Process Scheduling

The process scheduling required in the ATC system includes the scheduling of the Analyze and Synthesize processes as well as the scheduling of the four I/O process background programs. The internal scheduling of array functions in the Analyze and Synthesize processes is accomplished by function lists, as described in subsection 3.3. The scheduling of the processes themselves is determined by flags ANA (integer scalar 100) and SYN (integer scalar 102) as shown in Figure 3-26. The Analyze process is enabled by flag ANA being clear. ANA is cleared by the Transmit interrupt routine to indicate that the last output has been used and the output buffer is available. ANA is set by the Transmit background program, which will fill the output buffer. The Synthesize process is enabled by flag SYN being clear. SYN is cleared by the Re-

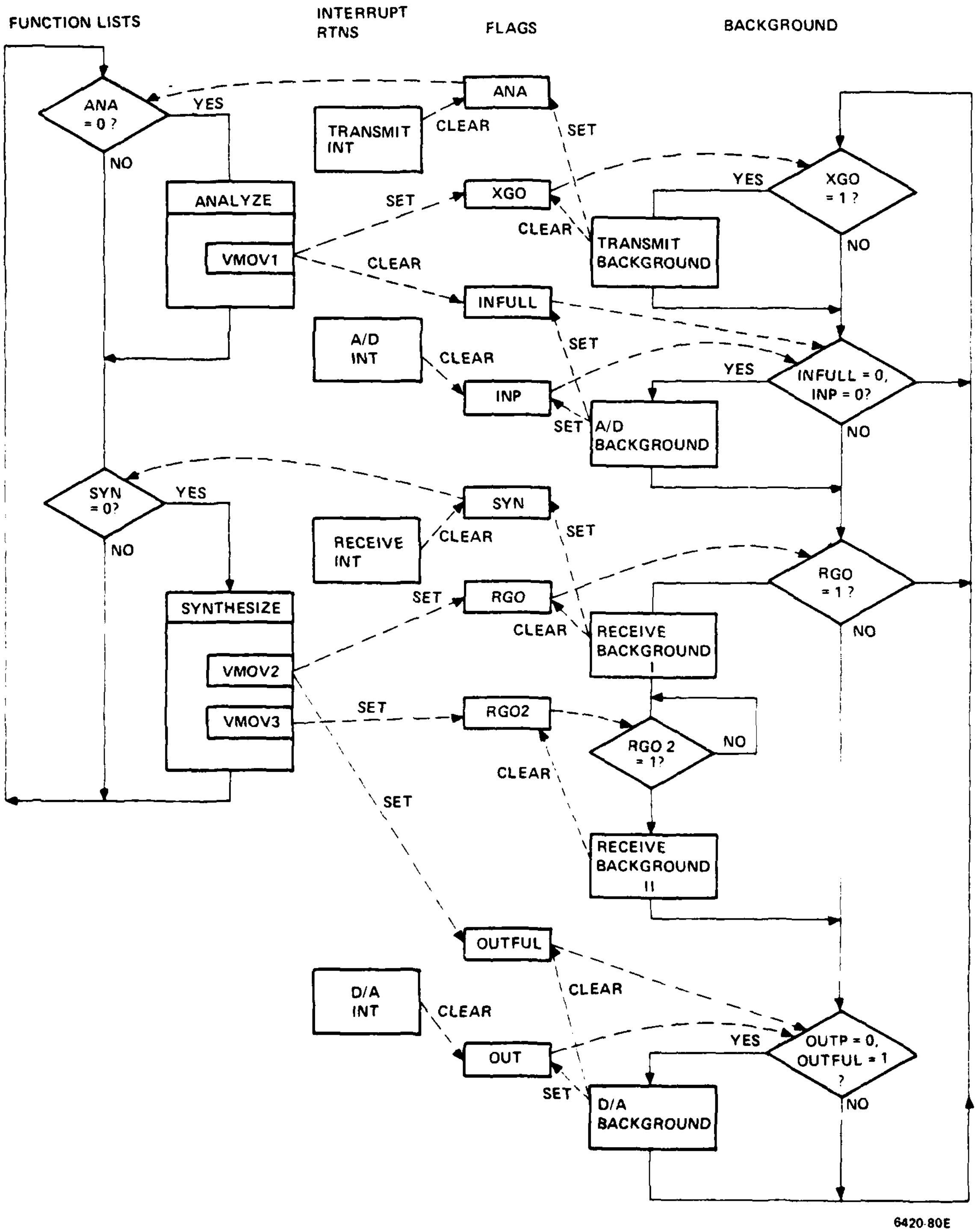


FIGURE 3-26: PROCESS SYNCHRONIZATION

ceive interrupt routine to indicate that the input buffer has been filled and is, therefore, available. SYN is set by the Receive background program which will read this buffer.

The Transmit background program is enabled by flag XGO being set. It is set by VMOV1 in the Analyze process when the output data has been copied into buffers AQPB and ACTDCT and is thus available to the Transmit background process. The background process clears the flag.

The Receive background program is enabled by flag RGO being set. It is set by VMOV2 in the Synthesize process when the data in ARQPB has been copied into a processing buffer, thus making ARQPB available for writing by the Receive background program, which clears the flag. When the Receive background program has deserialized the sideband parameters, it waits until the bit allocation determined by the Synthesize process is made available to it. VMOV3 in the Synthesize process sets flag RGO2 when it has written new data into buffer RIBIT. RGO2 being set enables the Receive background program to proceed to deserialize the DCT coefficients and to clear RGO2.

T: A/D background program is enabled by a combination of two flags. It runs when new input is available to it and its previous output has been used. Flag INP is cleared by the A/D interrupt routine to indicate that new input data is available. Flag INFULL is cleared by VMOV1 in the Analyze process to indicate that it has read the previous output of the background program (INPB) and that buffer is available for refilling. The A/D background program sets both of these flags.

The D/A background program is also enabled by a combination of two flags. It runs when its output buffer is empty and its input buffer is full. Flag OUTP is cleared by the D/A interrupt routine to indicate that a buffer has been emptied and is available for more output. Flag OUTFUL is set by VMOV2 in the Synthesize process when new output speech has been copied to buffer OUTB and is available to the background program. The background program clears OUTFUL and sets OUTP.

3.3 System Software

The ATC system software consists of MAP-300 software and host PDP-11 software.

3.3.1 MAP-300 Software

The MAP-300 software includes the array functions described individually in section 3.2 and the Executive program. It also includes the IOS-2SM program, LINE. Areas of MAP memory used by the MAP-300 software are shown in Table 3-9.

3.3.1.1 Array Functions

The array functions contained in the ATC system are shown in Table 3-10. The starred functions have been written by GTE. All other array functions normally included in the CSPI SNAP II system have been deleted. This means that before a program using any of these deleted functions can be run after running the ATC system, the standard SNAP II system must be reloaded into the MAP.

3.3.1.2 Executive Program

The Executive program that is included in the ATC system has been modified from the standard SNAP II release. The modifications consist of a revised APDONE routine which has had buffer checking eliminated for reasons of efficiency and which has added flag control and additional software to provide computational support for the AP. The additions to the Executive are described in subsection 3.2.7.

Bus 1

0-21FE	EXEC
2260-38A4	Prebinding Buffers
4000-4678	SNAP Array Functions
4690-49C4	FF2R
49DØ-515A	IOS
5714-5BC4	Scroll Program Buffers
639C-6EC4	GTE Array Functions
7150-89F8	GTE Array Functions
8AØØ-8AA5	Data Buffers
8AA6-AC6Ø	CSPUIS (background programs)
AC62-ACA8	Scroll Interrupt Additions
BFFF	Top of Memory

Bus 2

400-17CØ	Data Buffers
1D30-23ØD	Table Storage
231E-2F43	Data Buffers
3FFF	Top of Memory

Bus 3

0-1FFF	Data Buffers
1FFF	Top of Memory

TABLE 3-9: MEMORY MAP

Function	FCB Number	GTE-Supplied
VFLT (Y, A, U, B)	136	
VMOV (Y, U)	143	
VSMA1 (Y, A, U, B)	144	
VCOS (Y, A, U, B, V, C)	186	
FFTN (Y, U, V, W)	204	
FFTNB	205	
FFTIN (Y, U, V, W)	206	
FFTINB	207	
FF2R (Y, U, V, W)	214	
FF2RB	215	
VMOV1 (Y)	237	*
VMOV2 (Y)	238	*
VMOV3 (Y)	239	*
MPFDVM (Y, U, V)	240	*
MPDCTM (Y, U, V, W)	241	*
MPQDPP (Y, U, V)	242	*
MPDQPP (Y, U, V)	243	*
MPMWGX (Y, U, V, W)	244	*
MPFSTV (U)	245	*
MPSSRT (Y)	247	*
MPIDCM (Y, U, V)	248	*
MPASRT (Y)	249	*
MPFSTQ (Y, U, V, W)	250	*
MPFSTD (Y, U, V, W)	251	*
MPVDNM (Y, U, V)	252	*
MPBASP (Y, U, V, W)	253	*
MPSCAN (Y)	254	*
MPCDBA (Y)	255	*

TABLE 3-10: ARRAY FUNCTIONS

3.3.1.3 LINE Program

This program runs in the IOS-2SM scroll. It is loaded, initialized, and started by the host Fortran program. The function and operation of this program is described in subsection 3.2.3.1.

3.3.2 PDP-11 Software

The PDP-11 software consists of a Fortran control program, used to configure and initialize MAP buffers and to define function lists; software which provides the interface between this control program and the MAP driver; and utility software which provides an improved user interface to the MAP assembler.

3.3.2.1 Fortran Control Program

The Fortran control program consists of five major segments. These are: Configure and Initialize Buffers and Scalars (MAPON), Prebind Functions (CREATE), Define Function Lists (DEFINE), Configure Scroll Programs (SETUP), and Execute Function List (System).

These five segments are each subroutines, contained in separate overlays. The main program, MASTER, calls these subroutines in sequence. Data is passed between the main program and the subroutines through named common blocks. Other subroutines, mainly debugging aids, are included in the Fortran program which was designed to operate in any of three modes - Real-Time, Timing, and File-to-File. Only the Real-Time mode is operative in the delivered system.

3.3.2.1.1 Configure and Initialize Buffers

The buffer locations, sizes, types, and identifiers are shown in Table 3-11. The buffer initialization defines three cosine vectors.

M A P - 3 0 0
B U F F E R   A R E A S

BID	BID #	BUS	BA	BA+NS-1	NS	ST	SI	WS
AOPB	43	1	35328.	35341.	14	I	E	L
AOTDCT	44	1	35342.	35597.	256	I	E	L
AIBIT	45	1	35598.	35853.	256	I	E	L
SEN	49	1	35854.	36223.	370	I	E	L
INPB	50	1	36224.	36479.	256	I	E	L
BF1	63	1	36480.	36848.	369	I	E	L
BF2		1	36849.	37217.	369	I	E	L
BF3		1	37218.	37586.	369	I	E	L
ZRO	1	1	37587.	37955.	369	I	E	L
OUTB	59	1	37956.	36211.	256	I	E	L
RECV	56	1	38212.	38581.	370	I	E	L
ARGPB	57	1	38582.	38595.	14	I	E	L
AROPCT	58	1	38596.	38851.	256	I	E	L
SYNB1	62	1	38852.	39220.	369	I	E	L
SYNB2		1	39221.	39589.	369	I	E	L
RIBIT	2	1	39590.	36245.	256	I	E	L
BIDBF1	63	1	10000.	14499.	4500	I	E	L
BIDBF2		1	8800.	9999.	1200	I	E	L
COSZ	24	2	1024.0	1535.0	256	R	E	L
YOM	26	2	2048.0	2067.0	10	R	E	L
XOM	27	2	2068.0	2087.0	10	R	E	L
MAPX#	28	2	2068.0	2579.0	256	R	E	L
MIBIT	29	2	2088.0	2343.0	256	I	E	L

TABLE 3-11: ATC SYSTEM BUFFER CONFIGURATION

DCTM	30	2	2344.0	2855.0	256	R	E	L
		2	2856.0	2867.0	12	I	E	L
QTDCTM	32	2	2868.0	3123.0	256	I	E	L
NINM	33	2	3124.0	3369.0	246	I	E	L
NIN1	34	2	3370.0	3615.0	246	I	E	L
NIN2	35	2	3616.0	3861.0	246	I	E	L
NOUTM	36	2	3862.0	4107.0	246	I	E	L
NOUT1	37	2	4108.0	4353.0	246	I	E	L
NOUT2	38	2	4354.0	4599.0	246	I	E	L
RECV1	39	2	4600.0	4968.0	369	I	E	L
RECV2	40	2	4969.0	5337.0	369	I	E	L
SEN1	41	2	5340.0	5709.0	370	I	E	L
SEN2	42	2	5710.0	6079.0	370	I	E	L
DCT1T1	51	2	9000.0	9511.0	256	R	E	L
DCT1T2	52	2	9512.0	10023.0	256	R	E	L
IORDR1	53	2	10024.0	10279.0	256	I	E	L
IORDR2	54	2	10280.0	10535.0	256	I	E	L
MIBIT1	55	2	10536.0	10791.0	256	I	E	L
MIBIT2	60	2	10792.0	11047.0	256	I	E	L
QTDCT1	7	2	11048.0	11303.0	256	I	E	L
MIBIT3	5	2	11304.0	11559.0	256	I	E	L
MIBIT4	6	2	11560.0	11815.0	256	I	E	L
QPRM	31	2	11816.0	11829.0	14	I	E	L
QRPRM	8	2	11830.0	11843.0	14	I	E	L
QRDCTM	9	2	11844.0	12099.0	256	I	E	L
WORK	10	3	0.0	2047.0	512	C	E	L

TABLE 3-11: ATC SYSTEM BUFFER CONFIGURATION (CONT'D)

X1	11	3	2048.0	4095.0	512	C	E	L
X1R*		3			512	R	E	L
X1I*		3			512	R	E	L
X2	48	3	2048.0	3071.0	256	C	E	L
IORDRM#	12	3	0.0	255.0	256	I	E	L
PWEITM#	13	3	0.0	255.0	256	I	E	L
DCTM1#	14	3	512.0	1023.0	256	R	E	L
DCTM2#	15	3	1024.0	1535.0	256	R	E	L
R1#	16	3	0.0	17.0	9	R	E	L
A1#	17	3	18.0	33.0	8	R	E	L
KF#	18	3	35.0	66.0	16	R	E	L
PARC*	3	3			8	R	E	L
TMP1#	19	3	2048.0	2559.0	256	R	E	L
TMP2#	20	3	2560.0	3071.0	256	R	E	L
TMP3#	46	3	3584.0	4095.0	256	R	E	L
TMP4#	47	3	3072.0	3583.0	256	R	E	L
ORDCTM#	21	3	1024.0	1535.0	256	R	E	L
YNM#		3	512.0	1023.0	256	R	E	L
VLONG	23	2	4096.0	5119.0	512	R	E	L
VSHRT	25	2	5120.0	5631.0	256	R	E	L

TABLE 3-11: ATC SYSTEM BUFFER CONFIGURATION (CONT'D)

Two of these, VSHRT and VLONG, are used for FFT's and contain a full cycle of the cosine each, consisting of 256 and 512 points, respectively. The third, COSZ, is used in the DCT correction and contains one quarter cycle of the cosine in 256 points. The initialization also clears the overlap buffers XOM and YOM, the I/O buffers NIN1, NIN2, NOUT1, NOUT2, RECV1, RECV2, SEN1, and SEN2. It also clears the data delay line buffers AQPB, AQTDCT, DCT1T1, DCT1T2, QTDCT1, QPRM, and QRPRM, and the sync buffers BF1, BF2, SYNB1, and SYNB2. It stores a buffer of zeros, ZRO, for subsequent use by the ATC system. It initializes the reordering vectors IORDR1 and IORDR2 to be the integers from 0 to 255, and sets the bit allocation buffers, MIBIT1, MIBIT2, MIBIT3, and MIBIT3 to contain 134 values of two and the remainder of the values in these buffers to be zero.

The scalars used and their initial values are shown in Table 3-12.

3.3.2.1.2 Prebound FCB's

To reduce AP loading overhead, all array functions used in the real-time loop of the ATC system are prebound. That is, a separate copy of the APS portion of each array function is maintained for each set of calling parameters given the function. The values of these parameters are stored in the copy at pre-binding and the time copy is block-loaded into the APS at execution time. Table 3-13 lists the prebound functions and the associated calling parameters.

3.3.2.1.3 Function Lists

The prebound array functions are collected into function lists. The use of function lists allows the MAP to run independently of the host which, by eliminating host overhead, increases the speed of the system.

SID	NAME	DESCRIPTION	INITIAL VALUE	CONSTANT
50		constant	1/512.	*
51		constant	1/256.	*
52	unused			
53	unused			
54	LTHINV	LTH Inverse	1/256.	*
55	unused			
56	DARG	constant	1/1024.	*
57	MDARG	constant	1/1024.	*
58	unused			
.	.			
.	.			
.	.			
73	unused			
74	LPCNP1	LPC order +1	9	*
75	LPCN	LPC order	8	*
76		constant	4.0	*
77		constant	10E-10	*
78	unused			
79	BTLTH	Bits avail. for DCTs	268	*
80	CTEMP	Bit all offset	-	
81	unused			
82	BITSMN	Temp for CTEMP calc.	-	
83	ITOT	# of DCTs to be coded	-	
84	BITMAX	Max bits/DCT	3.0	*
85		CTEMP +.5	-	
86		LPC Threshold	.995	*
87	LTH	Frame Length	256	*
88	DQDC	Dequantized Mean	-	
89	DQVAR	Dequantized Variance	-	
90	DQPG	Dequantized Pitch Gain	-	
91	DQM	Dequantized Pitch	-	
92	LTH2	2*LTH	512	*
93		Inverse Variance	-	
94		constant	20.0	*
95		Log 16 (10)		*
96	LTH4	4*LTH	1024.	*
97	LTHM1	LTH-1	255	*
98		constant	.05	*
99	unused			
100	DC^{-2}	Mean from 2nd prev. frame	-	
101	VAR^{-2}	Variance from 2nd prev. frame	-	
102	DC^{-1}	Mean from prev. frame	-	
103	VAR^{-1}	Variance from prev. frame	-	

TABLE 3-12: SCALARS

```
STATUS=MPCBF(BIDBF1,153)
INSTR=153
STATUS=MPFDVM(MAPX,MINM,X2)
INSTR=154
STATUS=MPDCTM(DCTM,X1,X2,COSZ)
INSTR=155
STATUS=MPWWGX(P1,NIUM,KF,A1)
INSTR=156
STATUS=MPQDPP(QPRM,PARC,A1)
INSTR=157
STATUS=MPFSTV(A1)
INSTR=158
STATUS=MPBASP(DCTM1,PWEITM,TMP2,X1)
INSTR=159
STATUS=MPASRT(IORDRM)
INSTR=160
STATUS=MPSCAN(DCTM2)
INSTR=161
STATUS=MPCDHA(MIBIT3)
INSTR=162
STATUS=MPFSTQ(QTOCTM,MIBIT3,TMP1,TMP2)
INSTR=163
STATUS=MPDQPP(PARC,QRPRM,A1)
INSTR=164
STATUS=MPFSTD(DRDCTM,QRDCTM,TMP1,IORDR2)
INSTR=165
STATUS=MPIDCM(X2,DRDCTM,COSZ)
INSTR=166
STATUS=MPVDNM(NOUTM,YOM,X2)
INSTR=167
STATUS=MPSSRT(IORDRM)
STATUS=MPTBF(0)
INSTR=3
STATUS=MPCBF(BIDBF2,168)
INSTR=168
STATUS=FFTIN(X2,1,X2,VSHRT,WORK)
INSTR=169
STATUS=FFTIN(X1,1,X1,VLONG,WORK)
INSTR=170
STATUS=FF2R(X1,4,X1,VLONG,WORK)
INSTR=171
STATUS=FFTIN(X2,1,X2,VSHRT,WORK)
INSTR=4
STATUS=MPTBF(0)
STATUS=MPCBF(BIDBF3,172)
INSTR=172
STATUS=VMOV1(AQPB)
INSTR=173
STATUS=VMOV2(QRPRM)
INSTR=174
STATUS=VMOV3(RIBIT)
INSTR=175
STATUS=MPCDHA(MIBIT1)
STATUS=MPTBF(0)
```

TABLE 3-13: PREBOUND FCB'S

The top level function list (WAIT) is very simple. It consists of three subordinate function lists, STRTUP, ATCSYN, and ATCANA, which are conditionally executed or not based on the state of three associated scalars, INIT, SYN, and ANA. STRTUP is executed only once. Its execution causes INIT to be set to unity, which denies the execution condition for all succeeding passes through the main function list. STRTUP initializes the process synchronization flags and double buffer pointers as shown in Table 3-14. It also loads and starts the three I/O scrolls.

Both ATCSYN and ATCANA are unconditional, sequential lists of prebound array functions. They are shown in Table 3-15. ATCANA is the control structure of the Analyze process, as discussed in section 3.2.5, and ATCSYN is the control structure of the Synthesize process, as discussed in section 3.2.6.

3.3.2.2 Host Support Software

The host support software consists of a function definition module for each array function, an additional function, FCBGN, called by each of these definitions, error reporting routines, and the MAP driver interface routine. This software is contained in the SNAPHS library. This library has been modified for the ATC system, as detailed in section 3.5. The changes are to delete unused array function definitions, to make separate libraries for each of the three MAP drivers in the host operating system, and to change FCBGN to support newly released array functions. This last modification increases the size of table ISARG from 18 to 22. ISARG contains parameter usage configurations. The ATC system requires these additional configurations.

Flag	Integer Scalar	Value	Meaning	Description
OUT	104	1	false	A/D buffer full
INP	106	1	false	D/A buffer empty
AFLG	101	0	buffer 1	current send buffer
SFLG	103	0	buffer 1	current receive buffer
IFLG	107	0	buffer 1	current A/D buffer
OFLG	105	0	buffer 1	current D/A buffer
APDNFL	126	0	false	APDONE has occurred
RGO	114	0	false	enable Receive background
RGO2	116	0	false	enable Receive background part 2
XGO	115	0	false	enable Transmit background
INFULL	110	0	false	Analyze input full
OUTFUL	111	0	false	Synthesize output full
SYNC	117	0	-	sync position
RSYN	118	0	sync lost	sync state
NEWSYN	119	0	-	frames since sync acquired
OLSYN	120	0	-	previous frame sync bit
LSTER	121	0	false	error previous frame
ERSYN	122	0	-	number of sync bit errors

TABLE 3-14: SYSTEM FLAG INITIALIZATION

```
C         LIST "WAIT": THE WAIT LOOP
          STATUS=MPBFL(WAIT)
          STATUS=MPIIF(INIT,EQ,0,STRTUP,FL0)
          STATUS=MPIIF(SYN,EQ,0,ATCSYN,FL0)
          STATUS=MPIIF(ANA,EQ,0,ATCANA,FL0)
          STATUS=MPEFL(FL3)

C         LIST "STRTUP":INITIALIZATION LOOP
          STATUS=MPBFL(STRTUP)
          STATUS=MPIST(INIT,1)
          STATUS=MPIST(OUT,1)
          STATUS=MPIST(INP,1)
          STATUS=MPIST(AFLG,0)
          STATUS=MPIST(SFLG,0)
          STATUS=MPIST(IFLG,0)
          STATUS=MPIST(OFLG,0)
          STATUS=MPIST(INFULL,-1)
          STATUS=MPIST(OUTFUL,-1)
          STATUS=MPIST(APDNFL,0)
          STATUS=MPIST(RGO,0)
          STATUS=MPIST(XGO,0)
          STATUS=MPIST(RGO2,0)
          STATUS=MPIST(SYNC,0)
          STATUS=MPIST(RSYN,0)
          STATUS=MPIST(MFNSYN,0)
          STATUS=MPIST(OLSYN,0)
          STATUS=MPIST(LSTER,0)
          STATUS=MPIST(FPSYN,0)
          STATUS=MPLDS(AOM,IOS,AOMPM)
          STATUS=MPRNS(AOM,IOS,AOMSA)
          STATUS=MPLDS(ADAM,IOS,ADAMPM)
          STATUS=MPRNS(ADAM,IOS,ADAMSA)
          STATUS=MPLDS(IOSID,IOS,IOSPM)
          STATUS=MPRNS(IOSID,IOS,IOSSA)
          STATUS=MPEFL(FL4)
```

TABLE 3-15: FUNCTION LISTS

```
C         LIST "ATCANA": ATC ANALYZER W/ERROR CODING AND SERIALIZATION
          INSTR=1043
          STATUS=MPFFL(ATCANA)
C         BF(172)=VMOVI(AQPB)
          STATUS=MPXBF(172)
C         BF(153)=MPFDVM(MAPX,HINM,X2)
          STATUS=MPXBF(153)
C         BF(168)=FFTN(X2,1,X2,VSHRT,WORK)
          STATUS=MPXBF(168)
C         BF(154)=MPDCTM(DCTM,X1,X2,COSZ)
          STATUS=MPXBF(154)
C         BF(169)=FFTIN(X1,1,X1,VLONG,WORK)
          STATUS=MPXBF(169)
C         BF(155)=MPMWGX(R1,HINM,KE,A1)
          STATUS=MPXBF(155)
C         BF(156)=MPQDPP(QPRM,PARC,A1)
          STATUS=MPXBF(156)
C         BF(157)=MPFSIV(A1)
          STATUS=MPXBF(157)
C         BF(170)=FF2P(X1,4,X1,VLONG,WORK)
          STATUS=MPXBF(170)
C         BF(158)=MPRASP(DCTM1,PWEITM,TMP2,X1)
          STATUS=MPXBF(158)
C         BF(159)=MPASRT(IORDRM)
          STATUS=MPXBF(159)
C         BF(160)=MPSCAN(DCTM2)
          STATUS=MPXBF(160)
C         BF(161)=MPCDRA(WINIT3)
          STATUS=MPXBF(161)
C         BF(162)=MPFSTQ(QTDCTM,WINIT,TMP1,TMP2)
          STATUS=MPXBF(162)
C         IF(TIMING.EQ.YES)STATUS=MPXFL(ATCSYN)
          INSTR=1050
          STATUS=MPFFL(FL1)
C
```

TABLE 3-15: FUNCTION LISTS (CONT'D)

```
C         LIST "ATCSYN": ATC SYNTHESIZER W/ERROR DECODER & DESERIALIZATION
          INSTR=1055
          STATUS=MPREL(ATCSYN)
C         BF(173)=VMOV2(QRPRM)
          STATUS=MPXBF(173)
C         BF(163)=MPPWER(PARC,QRPRM,A1)
          STATUS=MPXBF(163)
C         BF(157)=MPESTV(A1)
          STATUS=MPXBF(157)
C         BF(170)=FF2R(X1,4,X1,VLONG,WORK)
          STATUS=MPXBF(170)
C         BF(158)=MPBASP(DCTM1,PREITM,TMP2,X1)
          STATUS=MPXBF(158)
C         BF(167)=MPSSRI(IORDRM)
          STATUS=MPXBF(167)
C         BF(160)=MPSCAN(DCTM2)
          STATUS=MPXBF(160)
C         BF(175)=MPCDBA(MINIT1)
          STATUS=MPXBF(175)
C         BF(174)=VMOV3(RIBIT)
          STATUS=MPXBF(174)
C         BF(164)=MPFSTD(DRDCTM,QRDCTM,TMP1,IORDR2)
          STATUS=MPXBF(164)
C         BF(165)=MPIDCM(X2,DRDCTM,COSZ)
          STATUS=MPXBF(165)
C         BF(171)=FFTIN(X2,1,X2,VSHRT,WORK)
          STATUS=MPXBF(171)
C         BF(166)=MPVDAM(NOUTM,YOM,X2)
          STATUS=MPXBF(166)
          STATUS=MPREL(FL2)
```

TABLE 3-15: FUNCTION LISTS (CONT'D)

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

2 OF 9

AD
A091663

3.3.2.3 Utility Software

A utility program, PREMAP, is included in the delivered ATC system. This program reformats text file to be acceptable to the MAP assembler, MAPASM. In particular, it changes tabs in the input text to sequences of spaces in the output text and supplies continuation lines for comments.

3.4 System Hardware

The ATC system includes equipment manufactured by CSP Inc., Billerica, Mass., and equipment designed and built by GTE Sylvania.

3.4.1 CSPI-Supplied Hardware

The CSPI-supplied hardware consists of a MAP-300 Array Processor, Model 1030, with attached options. These options are:

8K x 32 MOS Master Memory, 500 nsec, Bus 1, model 2030
16K x 32 MOS Slave Memory, 500 nsec, Bus 1, model 2050
8K x 32 MOS Master Memory, 300 nsec, Bus 2, model 2203
4K x 32 MOS Master Memory, 170 nsec, Bus 3, model 2410
PDP-11 Interface, model 3110
I/O Scroll type 2SM, model 4020
Bus Switch (2), model 4040
Analog Data Acquisition Module (ADAM), model 5120
Analog Output Module (AOM), model 5130
Expansion Chassis, model 6100
Auxiliary Power Supply, model 6200

The MAP includes preprogrammed micro-code in read-only memory. The ATC system requires Revision 18 of this micro-code.

3.4.2 GTE-Supplied Hardware

The GTE-supplied hardware consists of the Speech Processing Interface (SPI), which provides an analog interface to the MAP, and the Full Duplex Interface (FDI), which provides a digital interface to the MAP. These interfaces were supplied to two other contractors as well as being used in the GTE ATC system.

As the ATC project progressed, it became apparent that modifications to the GTE-supplied hardware were required. Subsection 3.4.2.3 describes these modifications.

3.4.2.1 GTE Speech Processing Interface (SPI)

The SPI assembly provides the amplification, equalization, and filtering necessary to interface an audio handset to the MAP-300 A/D and D/A converters. A block diagram of the SPI is shown in Figure 3-27. The SPI also serves as the common junction point for all digital signals between the MAP-300 and an external modem. The interconnections between the MAP, the SPI, and the external devices are shown in Figure 3-28.

A switch on the SPI front panel, which is shown in Figure 3-29, permits the selection of either of two sets of filters, thereby permitting a choice of cutoff frequency. Filters having cutoff frequencies of 3200 Hz and 3800 Hz are provided. The filters are of the plug-in type, thereby enabling the user to install other filters with different cutoff frequencies of his choice if so desired.

The handset provided with the equipment uses a dynamic microphone which has been designed to GTE Sylvania specifications and nas been optimized for use in speech processing applications. The handset connects directly to the front panel of the audio interface assembly and may be stored on the hookswitch, which is also located on the front panel. When "on hook," the audio circuits (both receiving and transmitting) for the handset are disabled. A 25-foot extension cable for use with the handset is also provided.

A pair of telephone jacks located on the rear panel of the audio interface unit (shown in Figure 3-30) may be used to connect a tape recorder or test equipment for test and measurement purposes. The audio circuits for the tape recorder are always active and are unaffected by the operation of the hookswitch. A single connector, also located on the rear panel, is used to connect the assembly to the A/D-D/A converters of the MAP-300. Table 3-16gives the specifications of the SPI.

Handset Interface

Input impedance......................150 ohms, balanced

Output impedance.....................150 ohms, balanced

Gain, input to A/D converter.........42 dB

Gain, D/A converter to output........-24 dB

Tape Recorder Interface

Input impedance......................600 ohms, balanced

Output impedance.....................600 ohms, balanced

Gain, input to A/D converter.........0 dB

Gain, D/A converter to output........0 dB

MAP-300 Interface

Output impedance - A/D converter.....3000 ohms

Signal level - A/D converter ....±5 V pk-pk

Input impedance - D/A converter......3000 ohms

Signal level - D/A converter.........±5 V pk-pk

Filter Characteristics

Cutoff frequency.....................3.2 KHz } switch selectable
3.8 KHz

Rolloff..............................130 dB per octave

TABLE 3-16: SPI SPECIFICATIONS

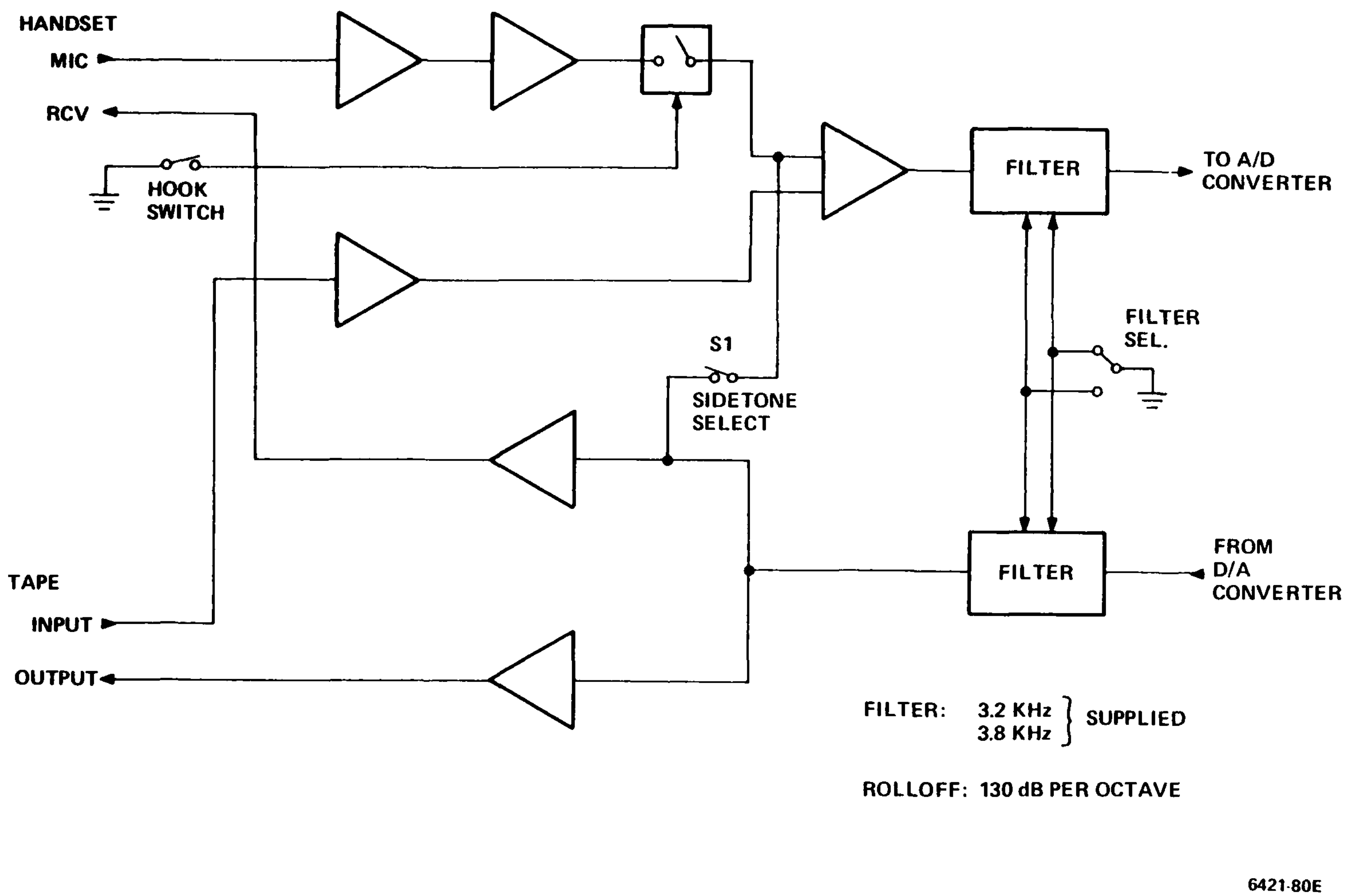
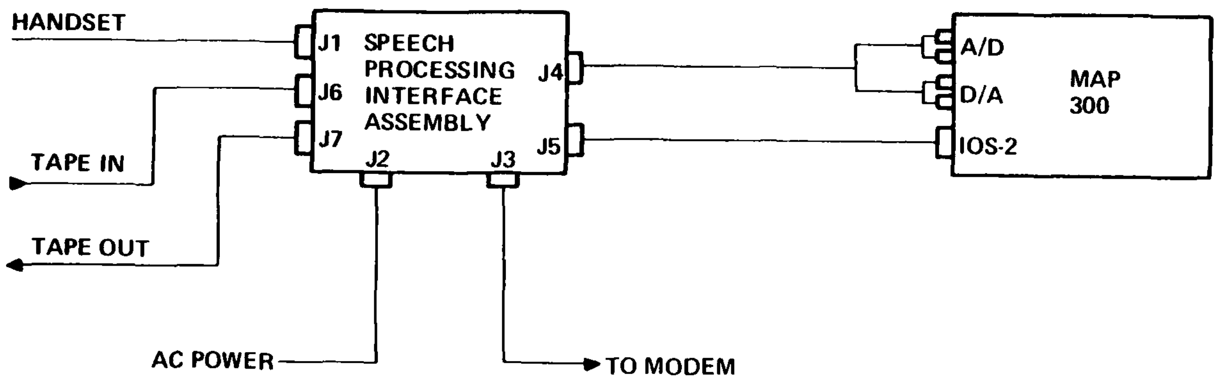


FIGURE 3-27: SPI BLOCK DIAGRAM



6435-80E

FIGURE 3-28: INTERCONNECTION DIAGRAM

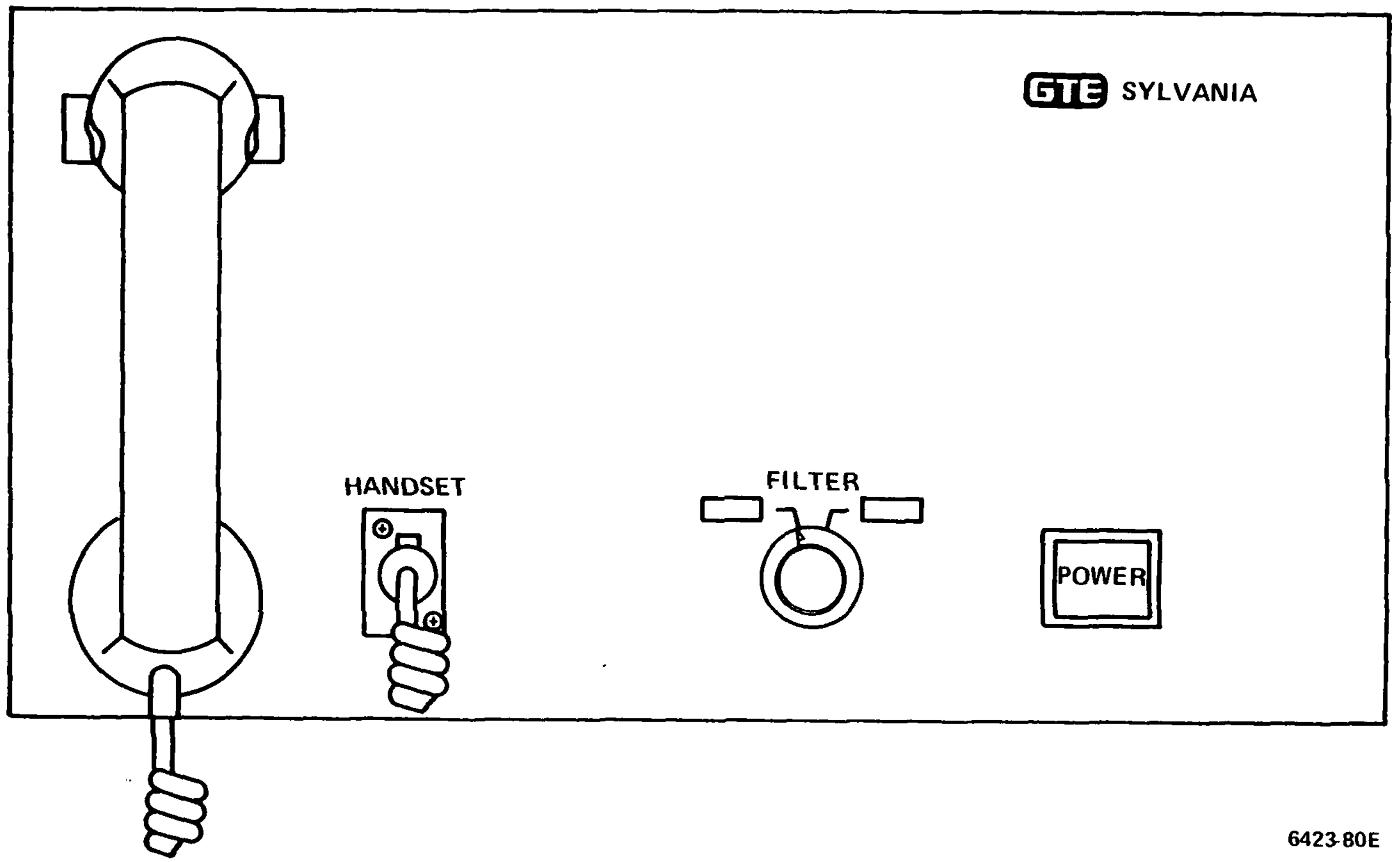
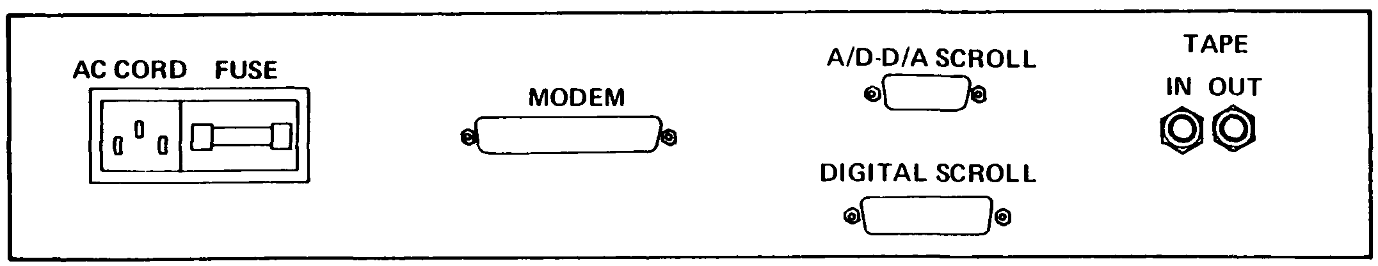


FIGURE 3-29: FRONT PANEL



6422-80E

FIGURE 3-30: REAR PANEL

3.4.2.2 Full Duplex Interface (FDI)

The CSPI IOS-2SM scroll has been modified by GTE Sylvania to provide a means for interfacing to any modem employing an EIA RS-449/RS-432 interface. The interface connections are given in Table 3-17. The interface design is such that an EIA RS-232-C interface may also be used, and the line drivers and receivers have been selected such that the protective networks for RS-449/RS-232 interoperation (refer to EIA Industrial Bulletin No. 12) are not required. Table 3-18 is a comparison between RS-449 and RS-232-C conventions. In addition to the modem interface, the modified I/O scroll includes a programmable real-time clock which generates the timing signals for speech sampling and the modem data. The I/O scroll is connected to the audio interface assembly by means of a single cable.

The data and speech sampling rates are set by issuing a single 16-bit control word from the IOS-2SM. The format of this control word and the FDT data formats are shown in Figure 3-31. The most significant byte of the control word determines the data rate, whereas the least significant byte determines the speech sampling rate. These control bytes and the associated data and sampling rates are given in Table 3-19. Each control byte controls a separate frequency divider. The basic clock frequency, 1.536 MHz, is first divided by four and then further divided by the selected divide ratio. The modem clock is then divided by an additional factor of two to produce a square wave with a 50% duty cycle. Once the real-time clock has been so programmed, it is not necessary to issue any other control words unless it is desired to change either of the clock rates, or unless power is removed from the MAP-300. It should be noted that the entire control word must be issued whenever changing rates, even if only one rate is to be changed.

PIN	SIGNAL
2	SI - Signal Rate Indicator
4	SD - Send Data
6	RD - Receive Data
7	RS - Request to Send
8	RT - Receive Timing
9	CS - Clear to Send
11	DM - Data Mode
12	TR - Terminal Ready
13	RR - Receiver Ready
15	IC - Incoming Call
16	SR - Signal Rate Selector
17	TT - Terminal Timing
19	SG - Signal Ground
20	RC - Receive Common
33	SQ - Signal Quality
37	SC - Send Common

TABLE 3-17: MODEM INTERFACE CONNECTIONS (RS-449)

RS-449		RS-232C	
SG SC RC	SIGNAL GROUND SEND COMMON RECEIVE COMMON	AB	SIGNAL GROUND
IS IC TR DM	TERMINAL IN SERVICE INCOMING CALL TERMINAL READY DATA MODE	 CE CD CC	 RING INDICATOR DATA TERMINAL READY DATA SET READY
SD RD	SEND DATA RECEIVE DATA	BA BB	TRANSMITTED DATA RECEIVED DATA
TT	TERMINAL TIMING	DA	TRANSMITTER SIGNAL ELEMENT TIMING (DTE SOURCE)
ST	SEND TIMING	DB	TRANSMITTER SIGNAL ELEMENT TIMING (DCE SOURCE)
RT	RECEIVE TIMING	DD	RECEIVER SIGNAL ELEMENT TIMING
RS CS RR	REQUEST TO SEND CLEAR TO SEND RECEIVER READY	CA CB CF	REQUEST TO SEND CLEAR TO SEND RECEIVED LINE SIGNAL DETECTOR
SQ	SIGNAL QUALITY	CG	SIGNAL QUALITY DETECTOR
NS SF	NEW SIGNAL SELECT FREQUENCY		
SR	SIGNALING RATE SELECTOR	CH	DATA SIGNAL RATE SELECTOR (DTE SOURCE)
SI	SIGNALING RATE INDICATOR	CI	DATA SIGNAL RATE SELECTOR (DCE SOURCE)
SSD SRD	SECONDARY SEND DATA SECONDARY RECEIVE DATA	SBA SBB	SECONDARY TRANSMITTED DATA SECONDARY RECEIVED DATA
SRS	SECONDARY REQUEST TO SEND	SCA	SECONDARY REQUEST TO SEND
SCS	SECONDARY CLEAR TO SEND	SCB	SECONDARY CLEAR TO SEND
SRR	SECONDARY RECEIVER READY	SCF	SECONDARY RECEIVED LINE SIGNAL DETECTOR
LL RL TM	LOCAL LOOPBACK REMOTE LOOPBACK TEST MODE		
SS SB	SELECT STANDBY STANDBY INDICATOR		

TABLE 3-18: RS-449/RS-232C COMPARISON

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
0	00	00000000	256	0.750	1.500
1	01	00000001	255	0.753	1.506
2	02	00000010	254	0.756	1.512
3	03	00000011	253	0.759	1.518
4	04	00000100	252	0.762	1.524
5	05	00000101	251	0.765	1.530
6	06	00000110	250	0.768	1.536
7	07	00000111	249	0.771	1.542
8	08	00001000	248	0.774	1.548
9	09	00001001	247	0.777	1.555
10	0A	00001010	246	0.780	1.561
11	0B	00001011	245	0.784	1.567
12	0C	00001100	244	0.787	1.574
13	0D	00001101	243	0.790	1.580
14	0E	00001110	242	0.793	1.587
15	0F	00001111	241	0.797	1.593
16	10	00010000	240	0.800	1.600
17	11	00010001	239	0.803	1.607
18	12	00010010	238	0.807	1.613
19	13	00010011	237	0.810	1.620
20	14	00010100	236	0.814	1.627
21	15	00010101	235	0.817	1.634
22	16	00010110	234	0.821	1.641
23	17	00010111	233	0.824	1.648
24	18	00011000	232	0.828	1.655
25	19	00011001	231	0.831	1.662
26	1A	00011010	230	0.835	1.670
27	1B	00011011	229	0.838	1.677
28	1C	00011100	228	0.842	1.684
29	1D	00011101	227	0.846	1.692
30	1E	00011110	226	0.850	1.699
31	1F	00011111	225	0.853	1.707
32	20	00100000	224	0.857	1.714
33	21	00100001	223	0.861	1.722
34	22	00100010	222	0.865	1.730
35	23	00100011	221	0.869	1.738
36	24	00100100	220	0.873	1.745
37	25	00100101	219	0.877	1.753
38	26	00100110	218	0.881	1.761
39	27	00100111	217	0.885	1.770
40	28	00101000	216	0.889	1.778
41	29	00101001	215	0.893	1.786

TABLE 3-19: REAL TIME CLOCK CONTROL

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

DECIMAL	COUNTER SETTING HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
42	2A	00101010	214	0.897	1.794
43	2B	00101011	213	0.901	1.803
44	2C	00101100	212	0.906	1.811
45	2D	00101101	211	0.910	1.820
46	2E	00101110	210	0.914	1.829
47	2F	00101111	209	0.919	1.837
48	30	00110000	208	0.923	1.846
49	31	00110001	207	0.928	1.855
50	32	00110010	206	0.932	1.864
51	33	00110011	205	0.937	1.873
52	34	00110100	204	0.941	1.882
53	35	00110101	203	0.946	1.892
54	36	00110110	202	0.950	1.901
55	37	00110111	201	0.955	1.910
56	38	00111000	200	0.960	1.920
57	39	00111001	199	0.965	1.930
58	3A	00111010	198	0.970	1.939
59	3B	00111011	197	0.975	1.949
60	3C	00111100	196	0.980	1.959
61	3D	00111101	195	0.985	1.969
62	3E	00111110	194	0.990	1.979
63	3F	00111111	193	0.995	1.990
64	40	01000000	192	1.000	2.000
65	41	01000001	191	1.005	2.010
66	42	01000010	190	1.011	2.021
67	43	01000011	189	1.016	2.032
68	44	01000100	188	1.021	2.043
69	45	01000101	187	1.027	2.053
70	46	01000110	186	1.032	2.065
71	47	01000111	185	1.038	2.076
72	48	01001000	184	1.043	2.087
73	49	01001001	183	1.049	2.098
74	4A	01001010	182	1.055	2.110
75	4B	01001011	181	1.061	2.122
76	4C	01001100	180	1.067	2.133
77	4D	01001101	179	1.073	2.145
78	4E	01001110	178	1.079	2.157
79	4F	01001111	177	1.085	2.169
80	50	01010000	176	1.091	2.182
81	51	01010001	175	1.097	2.194
82	52	01010010	174	1.103	2.207
83	53	01010011	173	1.110	2.220

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

	COUNTER SETTING		DIVIDE	OUTPUT RATE - KHZ	
DECIMAL	HEXADECIMAL	BINARY	RATIO	LINE	SPEECH
84	54	01010100	172	1.116	2.233
85	55	01010101	171	1.123	2.246
86	56	01010110	170	1.129	2.259
87	57	01010111	169	1.136	2.272
88	58	01011000	168	1.143	2.286
89	59	01011001	167	1.150	2.299
90	5A	01011010	166	1.157	2.313
91	5B	01011011	165	1.164	2.327
92	5C	01011100	164	1.171	2.341
93	5D	01011101	163	1.178	2.356
94	5E	01011110	162	1.185	2.370
95	5F	01011111	161	1.193	2.385
96	60	01100000	160	1.200	2.400
97	61	01100001	159	1.208	2.415
98	62	01100010	158	1.215	2.430
99	63	01100011	157	1.223	2.446
100	64	01100100	156	1.231	2.462
101	65	01100101	155	1.239	2.477
102	66	01100110	154	1.247	2.494
103	67	01100111	153	1.255	2.510
104	68	01101000	152	1.263	2.526
105	69	01101001	151	1.272	2.543
106	6A	01101010	150	1.280	2.560
107	6B	01101011	149	1.289	2.577
108	6C	01101100	148	1.297	2.595
109	6D	01101101	147	1.306	2.612
110	6E	01101110	146	1.315	2.630
111	6F	01101111	145	1.324	2.648
112	70	01110000	144	1.333	2.667
113	71	01110001	143	1.343	2.685
114	72	01110010	142	1.352	2.704
115	73	01110011	141	1.362	2.723
116	74	01110100	140	1.371	2.743
117	75	01110101	139	1.381	2.763
118	76	01110110	138	1.391	2.783
119	77	01110111	137	1.401	2.803
120	78	01111000	136	1.412	2.824
121	79	01111001	135	1.422	2.844
122	7A	01111010	134	1.433	2.866
123	7B	01111011	133	1.444	2.887
124	7C	01111100	132	1.455	2.909
125	7D	01111101	131	1.466	2.931

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ PAGE 4

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
126	7E	01111110	130	1.477	2.954
127	7F	01111111	129	1.488	2.977
128	80	10000000	128	1.500	3.000
129	81	10000001	127	1.512	3.024
130	82	10000010	126	1.524	3.048
131	83	10000011	125	1.536	3.072
132	84	10000100	124	1.548	3.097
133	85	10000101	123	1.561	3.122
134	86	10000110	122	1.574	3.148
135	87	10000111	121	1.587	3.174
136	88	10001000	120	1.600	3.200
137	89	10001001	119	1.613	3.227
138	8A	10001010	118	1.627	3.254
139	8B	10001011	117	1.641	3.282
140	8C	10001100	116	1.655	3.310
141	8D	10001101	115	1.670	3.339
142	8E	10001110	114	1.684	3.368
143	8F	10001111	113	1.699	3.398
144	90	10010000	112	1.714	3.429
145	91	10010001	111	1.730	3.459
146	92	10010010	110	1.745	3.491
147	93	10010011	109	1.761	3.523
148	94	10010100	108	1.778	3.556
149	95	10010101	107	1.794	3.589
150	96	10010110	106	1.811	3.623
151	97	10010111	105	1.829	3.657
152	98	10011000	104	1.846	3.692
153	99	10011001	103	1.864	3.728
154	9A	10011010	102	1.882	3.765
155	9B	10011011	101	1.901	3.802
156	9C	10011100	100	1.920	3.840
157	9D	10011101	99	1.939	3.879
158	9E	10011110	98	1.959	3.918
159	9F	10011111	97	1.970	3.959
160	A0	10100000	96	2.000	4.000
161	A1	10100001	95	2.021	4.042
162	A2	10100010	94	2.043	4.085
163	A3	10100011	93	2.065	4.129
164	A4	10100100	92	2.087	4.174
165	A5	10100101	91	2.110	4.220
166	A6	10100110	90	2.133	4.267
167	A7	10100111	89	2.157	4.315

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
168	A8	10101000	88	2.182	4.364
169	A9	10101001	87	2.207	4.414
170	AA	10101010	86	2.233	4.465
171	AB	10101011	85	2.259	4.518
172	AC	10101100	84	2.286	4.571
173	AD	10101101	83	2.313	4.627
174	AE	10101110	82	2.341	4.683
175	AF	10101111	81	2.370	4.741
176	B0	10110000	80	2.400	4.800
177	B1	10110001	79	2.430	4.861
178	B2	10110010	78	2.462	4.923
179	B3	10110011	77	2.494	4.987
180	B4	10110100	76	2.526	5.053
181	B5	10110101	75	2.560	5.120
182	B6	10110110	74	2.595	5.189
183	B7	10110111	73	2.630	5.260
184	B8	10111000	72	2.667	5.333
185	B9	10111001	71	2.704	5.408
186	BA	10111010	70	2.743	5.486
187	BB	10111011	69	2.783	5.565
188	BC	10111100	68	2.824	5.647
189	BD	10111101	67	2.866	5.731
190	BE	10111110	66	2.909	5.818
191	BF	10111111	65	2.954	5.908
192	C0	11000000	64	3.000	6.000
193	C1	11000001	63	3.048	6.095
194	C2	11000010	62	3.097	6.194
195	C3	11000011	61	3.148	6.295
196	C4	11000100	60	3.200	6.400
197	C5	11000101	59	3.254	6.508
198	C6	11000110	58	3.310	6.621
199	C7	11000111	57	3.368	6.737
200	C8	11001000	56	3.429	6.857
201	C9	11001001	55	3.491	6.982
202	CA	11001010	54	3.556	7.111
203	CB	11001011	53	3.623	7.245
204	CC	11001100	52	3.692	7.385
205	CD	11001101	51	3.765	7.529
206	CE	11001110	50	3.840	7.680
207	CF	11001111	49	3.918	7.837
208	D0	11010000	48	4.000	8.000
209	D1	11010001	47	4.085	8.170

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
210	D2	11010010	46	4.174	8.348
211	D3	11010011	45	4.267	8.533
212	D4	11010100	44	4.364	8.727
213	D5	11010101	43	4.465	8.930
214	D6	11010110	42	4.571	9.143
215	D7	11010111	41	4.683	9.366
216	D8	11011000	40	4.800	9.600
217	D9	11011001	39	4.923	9.846
218	DA	11011010	38	5.053	10.105
219	DB	11011011	37	5.189	10.378
220	DC	11011100	36	5.333	10.667
221	DD	11011101	35	5.486	10.971
222	DE	11011110	34	5.647	11.294
223	DF	11011111	33	5.818	11.636
224	E0	11100000	32	6.000	12.000
225	E1	11100001	31	6.194	12.387
226	E2	11100010	30	6.400	12.800
227	E3	11100011	29	6.621	13.241
228	E4	11100100	28	6.857	13.714
229	E5	11100101	27	7.111	14.222
230	E6	11100110	26	7.385	14.769
231	E7	11100111	25	7.680	15.360
232	E8	11101000	24	8.000	16.000
233	E9	11101001	23	8.348	16.696
234	EA	11101010	22	8.727	17.455
235	EB	11101011	21	9.143	18.286
236	EC	11101100	20	9.600	19.200
237	ED	11101101	19	10.105	20.211
238	EE	11101110	18	10.667	21.333
239	EF	11101111	17	11.294	22.588
240	F0	11110000	16	12.000	24.000
241	F1	11110001	15	12.800	25.600
242	F2	11110010	14	13.714	27.429
243	F3	11110011	13	14.769	29.538
244	F4	11110100	12	16.000	32.000
245	F5	11110101	11	17.455	34.909
246	F6	11110110	10	19.200	38.400
247	F7	11110111	9	21.333	42.667
248	F8	11111000	8	24.000	48.000
249	F9	11111001	7	27.429	54.857
250	FA	11111010	6	32.000	64.000
251	FB	11111011	5	38.400	76.800

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING			DIVIDE	OUTPUT RATE - KHZ	
DECIMAL	HEXADECIMAL	BINARY	RATIO	LINE	SPEECH
252	FC	11111100	4	48.000	96.000
253	FD	11111101	3	64.000	128.000
254	FE	11111110	2	96.000	192.000
255	FF	11111111	1	192.000	384.000

TABLE 3-19: REAL TIME CLOCK CONTROL (CONT'D)

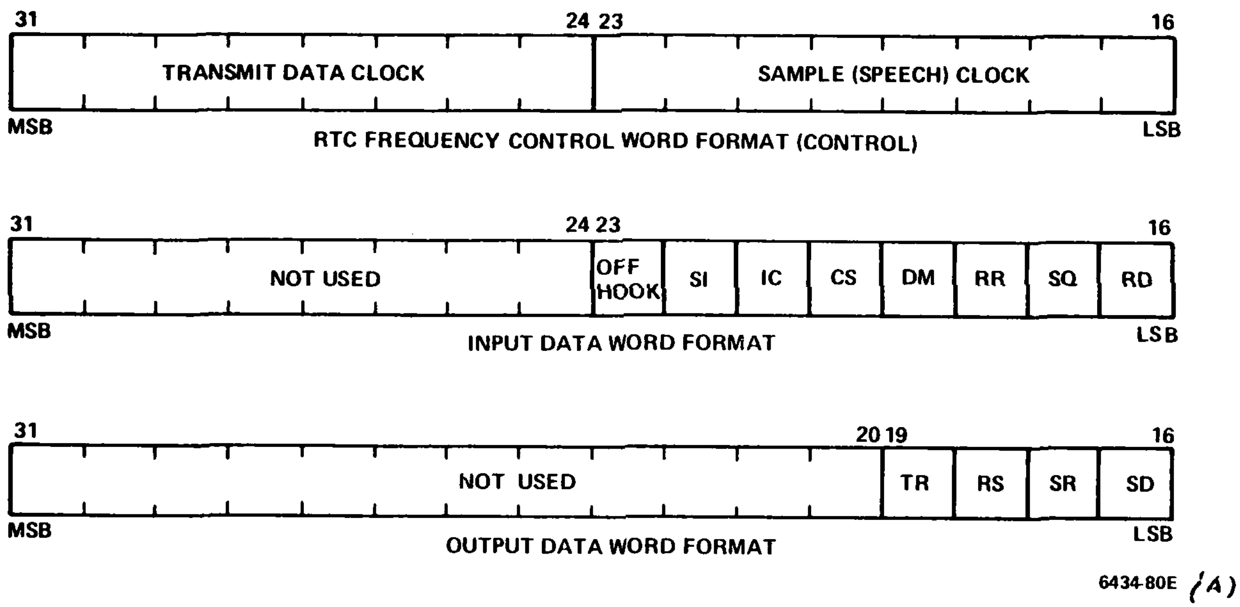


FIGURE 3-31: FDI FORMATS

Data transfers between the MAP-300 and the modem take place via the IOS-2SM data bus. The data transfer process is interrupt driven, with timing based on the transmit data clock, for output transfers, and the modem receiver clock, for input transfers. The interrupts will, therefore, occur at the data rate, thereby allowing sufficient time for the IOS-2SM to acknowledge each interrupt and perform the appropriate data transfer. The interrupts are controlled by a two-phase clock such that simultaneous interrupts can never occur. Furthermore, the interrupts are mutually exclusive so that the IOS-2SM can receive only one interrupt at a time. Interrupt number 1 is used for receive data, and interrupt number 2 is used for transmit data.

Upon receiving interrupt number 1, the processor should perform an input data transfer, acknowledging the interrupt after the transfer is complete. The input data word will contain the current modem data sample (least significant bit), RS-449 interface status data (bits 17-22), and a hookswitch status bit (bit 23). The receive interrupts are timed by the modem receive data clock; hence, the maximum response time to input the data and acknowledge the interrupt is approximately one-half of a period at the modem data rate. (The one-half period results from the fact that the processor must also supply transmit data.)

Upon receiving interrupt number 2, the processor should perform an output data transfer, acknowledging the interrupt after the transfer is complete. The output data word must contain the next data bit in the least significant bit position, and the appropriate RS-449 interface control bits in bit positions 17-19. The transmit interrupts are timed by the transmit data clock; hence, the maximum response time to output the data and acknowledge the interrupt is approximately one-half of a period at the transmit data rate. In order to insure the correct timing of the transmit data at the modem interface, the output data is double buffered. Hence, while the MAP-300 is transferring transmit data bit n, data bit n-1 appears at the modem interface.

The audio interface has been tested according to the attached table. These tests insure that all amplifiers and filters are performing to the designed specifications. The loop tests performed for both the handset and tape audio paths insure that the entire audio interface functions properly with the MAP-300, and that the overall loop gains and frequency response requirements are met.

The real time clock portion of the modified IOS-2SM scroll has been tested by programming several different line and speech sampling rates. A sufficient number of possible rates have been programmed to insure proper functioning of all RTC control bits. The modem interface has been tested by first outputting various control and data bit patterns from the MPA-300, and insuring that the correct bit patterns appear at the modem interface connector. Once the output had been determined to function properly, the modem input was tested by inputting various control and data bit patterns, looping through the MAP-300, and observing the bit patterns at the interface connector. A digital loop test was performed in which the MAP-300 output a digital data stream which was looped back at the modem interface connector and fed back into the processor where the resulting input was compared to the original output.

3.4.2.3 Modifications on Map Analog I/O Hardware

Due to the fact that the Map Analog circuitry is located in a very noisy environment (with the switching power supplies and the digital circuitry nearby), noise reduction techniques are very important in order to reduce the background noise in the presence of a signal. The modifications listed below were found necessary to reduce the background noise to ± 1 LSB of a 12 bit A/D converter operating at 5V full scale. The noise reduction techniques consisted of the reducing ground loops, separating ground and signal ground, and the reconfiguring the sample and hold amplifier with a differential input to reject common mode noise. It was observed that a good air circulation was required to reduce the terminal drift of the components in the analog circuitry.

List of Modifications on Map Analog I/O Hardware

1. ADAM Board

   a. Added decoupling capacitors between ± 15V and ground at the A/D, S/H, and MUX

   b. Cut ground etch to eliminate ground loops on analog circuit (around S/A and Hold capacitor)

   c. Added ground from A/D to S/H directly.

   d. Converted S/H to differential input. As a result, signal inputs are on Pin 28 (signal +) and Pin 29 (signal -) instead of Pin 34 and Pin 36. (Note: Only S/H designated as #4 on schematic has been modified.)

2. AOM Board

   a. Cut ground etch from I/O pin 15 and 16 and added ground wire directly from D/A (pin 21) to I/O pin 15 and 16.

   b. D/A cable modified. Wires to connector P4 pin 7 (purple) and pin 8 (black) have been removed to separate I/O drawer ground from AOM board ground.

3.5 System Usage

This section describes the operating instructions for executing the GTE system and for generating it. The examples in this section will use MAP A. Changes required for MAPs B and C are noted when not apparent.

Three steps are necessary for running the ATC system once it has been generated. First, the MAP must be allocated and loaded with the required programs as follows:

```
>ALL MA:
>

>RUN MALD
OBJECT INPUT?
BINARY INPUT?GTE.BO
      --  STOP
>
```

Second, the task must be installed as follows:

```
>INS ATCA
>
```

Third, the ATC task must be run, as follows:

```
>RUN ATCA
>
```

```
                >>>>> GTE 9600 BPS ATC SYSTEM <<<<<

START-UP(Y/N)? Y
            (1) MAP OPENED AND BUFFERS DECLARED!
            (2) PRE-BOUND FUNCTIONS CREATED!
            (3) FUNCTION LISTS DEFINED!
            (4) IOS-2 SETUP COMPLETE!
            (5) AOM SETUP COMPLETE!
            (6) ADAM SETUP COMPLETE!


PDP <==> MAP LINKAGE HAS BEEN SUSPENDED
   (DATE AND TIME)
TYPE "RES ATCA" TO TERMINATE
```

The file LINE.PBJ must be available for proper execution. The task starts MAP A and then suspends itself. To halt the ATC system, the task must be resumed as follows:

```
RES ATCA
>
ATCA -- STOP
>
```

When resumed, the task closes the MAP and exits.

If a complete, two MAP system is desired, the process above must be repeated for the second MAP.

If the ATC system is to be started again, without additional loading of GTE.BO, the start-up procedure can be avoided by answering "N" to the "Start-up" question, as follows:

```
>RUN ATCA
>



             >>>>> GTE 9600 BPS ATC SYSTEM <<<<<

START-UP(Y/N)? N
            (1) MAP OPENED AND BUFFERS DECLARED!
            (2) PRE-BOUND FUNCTIONS CREATED!
            (3) FUNCTION LISTS DEFINED!
            (4) IOS-2 SETUP COMPLETE!
            (5) AOM SETUP COMPLETE!
            (6) ADAM SETUP COMPLETE!


PDP <==> MAP LINKAGE HAS BEEN SUSPENDED
   (DATE AND TIME)
TYPE "RES ATCA" TO TERMINATE
```

3.5.1 System Generation

The ATC system generation consists of generating the MAP load module and of generating the ATCA, ATCB, and ATCC tasks.

3.5.1.1 MAP Load Module Generation

The process of generating a MAP load module consists of four steps. For purposes of illustration, the array function VMOV1 will be used in the example to follow.

First, the source file, VMOV1.TXT must be reformatted using PREMAP, as follows:

```
>RUN PREMAP
INPUT FILENAME= VMOV1.TXT
OUTPUT FILENAME= VMOV1.MAP
NORMAL EXIT AFTER NNN LINES!

      --  STOP
>
```

VMOV1.MAP is the reformatted source file, acceptable to the MAP assembler. Instructions for building the PREMAP tasks are included in Section 3.5.2.2.

Second, the reformatted source file must be assembled as follows:

```
>RUN MAPASM
SOURCE FILENAME?
VMOV1.MAP
OBJECT FILENAME?
VMOV1.OBJ
LISTING OUTPUT?
VMOV1.LST
 U:
$1,9999

LINES WITH ERRORS:    0   (MAP VERSION 800101.10)    E=    0
>
```

Third, all of the PBJ modules must be combined into a single PBJ file. This can be accomplished by using the command file GTESYS.CMD as follows:

```
@GTESYS
PIP MAP.PBJ;*/DE
PIP GTE.PBJ;*/DE
PIP GTE.BU;*/DE
PIP MAP.PBJ=S300EX.PBJ,APDONE.PBJ,GTE300.PBJ,FF2R.PBJ,IOS.PBJ
PIP GTE.PBJ=MAP.PBJ,MPFDVM.PBJ,MPDCTM.PBJ,MPQDPP.PBJ,MPDQPP.PBJ
PIP GTE.PBJ=GTE.PBJ,MPMWGX.PBJ,MPFSTV.PBJ,MPASRT.PBJ,MPSSRT.PBJ
PIP GTE.PBJ=GTE.PBJ,MPFSTQ.PBJ,MPFSTD.PBJ,MPBASP.PBJ,MPSCAN.PBJ
PIP GTE.PBJ=GTE.PBJ,MPCDBA.PBJ,MPIDCM.PBJ,MPVDNM.PBJ,GPATCH.PBJ
PIP GTE.PBJ=GTE.PBJ,VMOV1.PBJ,VMOV2.PBJ,VMOV3.PBJ,CSPU15.PBJ
PIP GTE.PBJ/PU
;
;        USE MACRO ONCE!
;        1<M(CR)FFFF(CR)S-1LSKSTS>SS
;
;        USE MACRO 23 TIMES!
;        23<MFFFF   SOGS2KTS>SS
;
TEC GTE.PBJ
*1<M
FFFF
S-1LSKSTS>SS
10'      APSDONE     AP PROCESS INTERRUPT HANDLER     MAY 28,1980
23<MFFFF   SOGS2KTS>SS
10*  SNAP-II MAP-300 ARITH. MODULES - PROG #830101.03 MAY 7, 1980
10;FAST FOURIER TRANSFORM AND INVERSE TRANSFORM ALGORITHMS          18 AUG
10'   SNAP-II IOS PACKAGE  ---- JUNE 25, 1979 ----  PROGRAM # 840001.00
10'?FCH 241... "MPDCTM(Y,U,V,W)"?FORM TRUE DCT AND ITS REFLECTED MAG SQU
10'?FCH 242... "MPQDPP(Y,U,V)"?QUANTIZE/DEQUANTIZE SIDEBAND
10'?FCH 243... "MPDQPP(Y,U,V)"?DEQUANTIZE SIDEBAND
10'?FCH 244... "MPMWGX(Y,U,V,W)"?EXTRACT M,PG,MOVE X
10'?FCH 245... "MPFSTV(U)"?CREATE PITCH AND VOCAL TRACT RESPONSE
10'?FCH 249... "MPASRT(U)"?SORT DCT1(.) COEFFICIENTS
10'?FCH 247... "MPSSRT(Y)"?SORT DCT1(.) COEFFICIENTS
10'?FCH 250... "MPFSTQ(Y,U,V,W)"?QUANTIZE DCT PARAMETERS
10'?FCH 251... "MPFSTD(Y,U,V,W)"?DEQUANTIZE DCT PARAMETERS
10'?FCH 253... "MPBASP(Y,U,V,W)"?BASIS SPECTRUMCALCULATION
10'?FCH 254... "MPSCAN(Y,U,V)"?SCAN DCT BASIS SPECTRUM
10'?FCH 255... "MPCDBA(Y,U,V)"?COMPLETE DCT BIT ASSIGNMENT
10'?FCH 248... "MPIDCM(Y,U,V)"?MAKHOUL IDCT CORRECTION
10'?FCH 252... "MPVDNM(Y,U,V)"?VAR.,DC,INTERPOLATE,FIX
10          #L=S7AID
10'?FCH 237... "VMOV1(Y)"?MOVE BUNCH OF BUFFERS TO OTHER BUFFERS
10'?FCH 238... "VMOV2(Y)"?MOVE BUNCH OF BUFFERS TO OTHER BUFFERS
10'?FCH 239."VMOV3(Y)"?MOVE BUNCH OF BUFFERS TO OTHER
10'?PROG. CSPUSP.TXT
*
*EXSS
PIP GTE.PBJ/PU
MPL
MCR -- TASK NOT IN SYSTEM
>
```

Note that once the command file has invoked TECO, the user must give the specified TECO commands. The PREMAP and assembly procedures must be repeated for each of the files used in GTESYS.CMD. If the file 5300EX.PBJ is unavailable, it must be obtained from the delivered tape, as assembly is not possible. In addition, the procedure must be performed on LINE.TXT if LINE.PBJ does not exist.

Fourth, the file GTE.PBJ must be transformed by the MAP Loader into a binary object file, as follows:

```
>RUN MALD
OBJECT INPUT?GTE.PBJ
BINARY OUTPUT?GTE.BO
LOAD MAP? (Y OR N) N
        -- STOP 2
>
```

The file GTE.BO is then ready for use.

3.5.1.2 Task Generation

The command files ATCA.CMD and TBATCA.CMD are used to build the ATC task. When taskbuilding the ATC tasks, it is necessary, for reasons of space, to use the GTE-supplied RSX system library, F4PLIB.OLB. The existing system library, SYSLIB.OLB, should be renamed to something else, e.g., OSYSLIB.OLB, and F4PLIB.OLB should be renamed to SYSLIB.OLB. The file ATCA.ODL is used to specify the overlay structure. The contents of ATCA.CMD are:

```
TIM
TKB @TBATCA
TIM
```

The contents of TBATCA are:

```
ATCA=ATCA/*P
UNITS=10
ASG=TI:7
ASG=NI:6
ASG=SY:4
//
```

The contents of ATCA.ODL are:

```
        .ROOT A-*(B,C,D,K,H,I,N,E,F,G,J,L,M,O,P,R,S,T,T0,U,U0,Y,Z,Z0,Z2,Z3)
A:      .FCTR MASTER-GSNPA
B:      .FCTR INIT
C:      .FCTR USER
D:      .FCTR MAPGN
K:      .FCTR CREATE
H:      .FCTR OPERDR-LO/LB
I:      .FCTR SETUP-(LOAD,IOSM-LO/LB,ADM-LO/LB,ADAM-LO/LB)
N:      .FCTR SYSTMA-LO/LB
E:      .FCTR INPUT-TAPE2-(OPT1,OPT2,OPT3,OPT4,OPT5,OPT6,OPT7,OPTRA)
F:      .FCTR DCVAR
G:      .FCTR DCTF
J:      .FCTR QOPARM
M:      .FCTR PITLPC
L:      .FCTR VPBS
O:      .FCTR BASIS
P:      .FCTR ASRT
R:      .FCTR SPCT
S:      .FCTR BITA
T:      .FCTR QDCT
T0:     .FCTR CHANL
U:      .FCTR LPARM
U0:     .FCTR SSRT
Y:      .FCTR DDCT
Z:      .FCTR OCTR
Z0:     .FCTR VARDC
Z2:     .FCTR OUTPUT-TAPE2-(OPT1,OPT2,OPT3,OPT4,OPT5,OPT6,OPT7,OPTRA)
Z3:     .FCTR SNR
        .END
```

ATCA.ODL requires a particular host support library, NSNPA, which has the unused array functions deleted. The command file to create this library is AGTESN.CMD. Its contents are

```
TIM
;
;       CREATE SEPARATE VERSION OF "SNPLIB.OLB"
;
PIP NSNPA.OLB=SNALB.OLB
;
;       ADD ATC HSP ROUTINES
;
LBR NSNPA/IN=FF2RHSP
LBR NSNPA/IN=MPFDVM,MPDCIM,MPMWGX,MPQDPP,MPFSTV,MPRASP,MPASRT
LBR NSNPA/IN=MPSCAM,MPCDBA,MPFSTG,MPLGPP,MPSSRT,MPFSTD,MPIDCM,MPVDNM
;
;       DELETE UNUSED "COMPLEX FCN" HSP ROUTINES
;
LBR SY:NSNPA/DE:CCVMD:CMINV:CPOL:CRECT:CSDOT:CSMA1:CSMA2:CVADD
LBR SY:NSNPA/DE:CVCMJ:CVMUL:CVRCP:CVSUB:CXMLR:CXMUL
;
;       DELETE UNUSED "ARITHMETIC FCN" HSP ROUTINES
;
LBR SY:NSNPA/DE:SDOT:SMAX:SMIN:SMVAB:SMXAB:SSMAB:SSMSQ:SADD:SSUB
LBR NSNPA/IN=VMOV1,VMOV2,VMOV3
LBR SY:NSNPA/DE:DCVM:DDIFF:DFL22:DINIG:ERRR
LBR NSNPA/DE:MVLD:SDIV:SMUL:SSUM:VSMA1:VTHR
LBR SY:NSNPA/DE:FFIR3:FFT:FFTI:FFTIF:FFTI3:FFIR:FFIR3:FFT3:FFINR:FFTNR
LBR SY:NSNPA/DE:VALGN:VALN:VHIST:VLN:VRAMP:VRAN1:VSMA2:VSMA3
LBR SY:NSNPA/DE:VMAX:VMIN:VMINS:VA:VAB:VABSQ:VAD
LBR SY:NSNPA/DE:VALOG:VAL2:VCLIP:VCOMP:VDIV:VDV:VFIX8:VFLT8
LBR SY:NSNPA/DE:VFRCT:VFTI:VINPT:VITF:VLG16:VLIM:VLOG:VLOGN
LBR SY:NSNPA/DE:VL2:VMAG:VMG:VMGSQ:VML:VMNAB:VMSQ:VMTHR:VMUL
LBR SY:NSNPA/DE:VMXAB:VNEG:VPOLY:VPOW:VRC:VRCP:VS:VSAD:VSB
LBR SY:NSNPA/DE:VSIN1:VSML:VSQ:VSQRT:VSSQ:VTAG:VXP
;
;       DELETE UNUSED "MANAGEMENT FCN" HSP ROUTINES
;
LBR SY:NSNPA/DE:MPCLM:MPCOL:MPNXC:MPNXR:MPRUN:MPRSA:MPRSI:MPABT
LBR NSNPA/DE:MPCLA:MPCRB:MPFTM:MPFOR:MPTLB
LBR NSNPA/DE:MPCSQ:MPAPA:MPCSE:MPTSS:MPWTC:MPRBC:MPLIM:MPMUL
LBR NSNPA/DE:MPIAD:MPICS:MPIDV:MPIF:MPICL:MPIFR:MPIFX
LBR NSNPA/DE:MPIMD:MPIML:MPIRS:MPISR:MPITM:MPITF:MPITS:MPIWL:MPIWS
LBR NSNPA/DE:MPBFI:MPBFO:MPDBR:MPFBA:MPRBS:MPSCR:MPSRB
LBR NSNPA/DE:MPTSR:MPTBS:MPWBA:MPFVN:MPUDD:MPWTS
;
;       EXTRACT AS AN OBJECT FOR ROOT SEGMENT
;
LBR SY:NSNPA=NSNPA/EX
PIP NSNPA.OBJ/LI
PIP NSNPA.*/PU
TIM
```

The library NSNPA is based on the library SNALB, which contains the GTE-delivered version of FCBGN, as well as SNAPHX, EAFHSP, MRTPCK, and MAADVR.

If it is necessary to compile the Fortran files, three command files are provided for this purpose. GTEFTN.CMD compiles the host support modules for the GTE-written array functions. Its contents are:

```
F4P VMOV1,LP/LI:1=VMOV1/DE/NOTR
F4P VMOV2,LP/LI:1=VMOV2/DE/NOTR
F4P VMOV3,LP/LI:1=VMOV3/DE/NOTR
F4P MPFDVM,LP/LI:1=MPFDVM/DE/NOTR
F4P MPDCTM,LP/LI:1=MPDCTM/DE/NOTR
F4P MPGDPP,LP/LI:1=MPGDPP/DE/NOTR
F4P MPDQPP,LP/LI:1=MPDQPP/DE/NOTR
F4P MPMAGX,LP/LI:1=MPMAGX/DE/NOTR
F4P MPFSTV,LP/LI:1=MPFSTV/DE/NOTR
F4P MPSSRT,LP/LI:1=MPSSRT/DE/NOTR
F4P MPIDCM,LP/LI:1=MPIDCM/DE/NOTR
F4P MPASRT,LP/LI:1=MPASRT/DE/NOTR
F4P MPFSTQ,LP/LI:1=MPFSTQ/DE/NOTR
F4P MPFSTD,LP/LI:1=MPFSTD/DE/NOTR
F4P MPVDNM,LP/LI:1=MPVDNM/DE/NOTR
F4P MPBASP,LP/LI:1=MPBASP/DE/NOTR
F4P MPSCAN,LP/LI:1=MPSCAN/DE/NOTR
F4P MPCDRA,LP/LI:1=MPCDRA/DE/NOTR
```

CFTN.CMD compules most of the modules in the Fortran control program. The delivered version is incorrect and must be modified to correspond to the following:

```
F4P MASTER=MASTER.ATC/NOTR
F4P INIT=INIT/DE/NOTR
F4P USER=USER/DE/NOTR
F4P MAPON=MAPON/DE/NOTR
F4P CREATE=CREATE/DE/NOTR
F4P DEFINE=DEFINE/DE/NOTR
F4P SETUP=SETUP/DE/NOTR
F4P LOAD=LOAD/NOTR
F4P IOSW=IOSW/NOTR
F4P AOM=AOM/NOTR
F4P ADAW=ADAW/NOTR
F4P SYSTMB=SYSTMB/DE/NOTR
F4P SYSTMC=SYSTMC/DE/NOTR
F4P SYSTMA=SYSTMA/DE/NOTR
F4P INPUT=INPUT/DE/NOTR
F4P DCVAR=DCVAR/DE/NOTR
F4P DCTF=DCTF/DE/NOTR
F4P PITLPC=PITLPC/DE/NOTR
F4P QDPARM=QDPARM/DE/NOTR
F4P VPFS=VPFS/DE/NOTR
F4P BASIS=BASIS/DE/NOTR
F4P ASRT=ASRT/DE/NOTR
F4P SDCT=SDCT/DE/NOTR
F4P BITA=BITA/DE/NOTR
F4P QDCT=QDCT/DE/NOTR
F4P CHANL=CHANL/DE/NOTR
F4P DPARM=DPARM/DE/NOTR
F4P SSRT=SSRT/DE/NOTR
F4P DDCT=DDCT/DE/NOTR
F4P DCTR=DCTR/DE/NOTR
F4P VARDC=VARDC/DE/NOTR
F4P OUTPUT=OUTPUT/DE/NOTR
F4P SWR=SWR/DE/NOTR
```

The command file TAPFTN.CMD compiles the remaining files needed by the Fortran control program. Its contents must be modified to conform to the following:

```
F4P TAPE2=TAPE2/NOTR
F4P OPT1=OPT1/NOTR
F4P OPT2=OPT2/NOTR
F4P OPT3=OPT3/NOTR
F4P OPT4=OPT4/NOTR
F4P OPT5=OPT5/NOTR
F4P OPT6=OPT6/NOTR
F4P OPT7=OPT7/NOTR
F4P OPT8=OPT8/NOTR
F4P OPT8A=OPT8A/NOTR
F4P OPT8B=OPT8B/NOTR
F4P OPT8C=OPT8C/NOTR
```

A command file for generating PREMAP is also provided. The contents of PREMAP.CMD are:

```
FOR SY:PREMAP,LP/LI:1=SY:PREMAP
TKB SY:PREMAP,LP=SY:PREMAP,FIDEN
PIP PREMAP.*/PU
```

3.6 System Listings

The ATC system listings which follow are divided into three sections: MAP Functions, Executive Programs, and Fortran Host Support, Control, and Support Programs.

3.6.1 MAP Functions

The modules contained in this section are:

VMOV1

VMOV2

VMOV3

MPFDVM

MPDCTM

MPQDPP

MPDQPP

MPMWGX

MPFSTV

MPSSRT

MPIDCM

MPASRT

MPFSTQ

MPFSTD

MPVDNM

GPATCH

MPBASP

MPSCAN

MPCDBA

LINE

```
                    (00001) *  FCH 237..."VMOV1(Y)"    MOVE BUNCH OF BUFFERS TO OTHER BUFFERS
                    (00002) *                                  ORIGINATED: 12-MAY-80
                    (00003) *                                  UPDATED: 20-JUL-80
                    (00004) *
                    (00005) *  MOVE FROM        TO
                    (00006) *  OPRM             AOPR
                    (00007) *  OTDCT1           AOTDCT
                    (00008) *  OTDCTM           OTDCT1
                    (00009) *  MIBI14           AIBIT4
                    (00010) *  MIBIT3           MIBIT4
                    (00011) *  INPR             NINM
                    (00012) *
                    (00013) *  DEFINE GLOBAL SYMBOLS
           000008F8 (00014)         AFDTSORG=$8F8
           000021FC (00015)         CSPUSNOS=$21FC
           00000794 (00016)         DMYS=$794
           00000003 (00017)         #M=3
           00000000 (00018)         MSS=0
           00000502 (00019)         ISVTS=$0502
           00000503 (00020)         ISAS001=ISVTS+D'1'
           0000056C (00021)         ISAS106=ISVTS+D'106'
           00000575 (00022)         ISAS115=ISVTS+D'115'
           00002E28 (00023)         OPRMS=11816
           00002B28 (00024)         OTDCT1S=11048
           00000B34 (00025)         OTDCTMS=2868
           00002D28 (00026)         MIBIT4S=11560
           00002C28 (00027)         MIBIT3S=11304
           00008D80 (00028)         INPBS=36224
           00008A00 (00029)         AOPBS=35328
           00008A0E (00030)         AOTDCTS=35342
           00008B0E (00031)         AIBIT4S=35598
           00000C34 (00032)         NINMS=3124
                    (00033) *
           0000639C (00034)         START=25500
                    (00035) *
                    (00036) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
           00000876 (00037)         #L=AFDTSORG+3*2*(237-128)                      FCH 237
     00876 001E639E (00038)         ADDR VMV1S(R7,1)
     0087B 001E63B2 (00039)         ADDR VMV11S(R7,1)
     0087A 001021FC (00040)         ADDR CSPUSNOS(,1,0)
                    (00041)         EJECT
```

```
               0000639C (00042)          #L=START
                        (00043) #>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<
                        (00044) *
                        (00045)          EVEN
          0639C 0000    (00046)          DATA VMV1SSA
          0639D 0006    (00047)          DATA VMV1SSZ
                        (00048) VMV1S    BEGIN APU(VMV1)
               00000000 (00049)          #A=0
    A00 0639F 20460000 (00050) VMV1SSA SET(G2)\NOP                          START TIMING OF ANALYZER
    A01 063A0 08FC0000 (00051) #1      MOV(IOA,OO)\NOP
    A02 063A2 90100001 (00052)         JUMPC(#1,F1)
                        (00053) *
    A03 063A4 20322032 (00054)         CLEAR(RA)
    A04 063A6 00000000 (00055)         NOP
    A05 063A8 10000000 (00056)         JUMP(0)
               00000006 (00057)         VMV1SSZ=#A-VMV1SSA
          063AA         (00058)         END VMV1SSZ
                        (00059)         EJECT
```

```
                    (00060) *>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<
                    (00061) *
                    (00062)         EVEN
    063AA 0000642A  (00063)         ADDR VMV1S1
    063AC 000063B4  (00064)         ADDR VMV1IS+2*SCLRS
    063AE 0000      (00065)         DATA 0
    063AF 007A      (00066)         DATA VMV1SZ
    063B0 000063FF  (00067)         ADDR VMV1SA
                    (00068)         EVEN
                    (00069) *
                    (00070) VMV1IS  BEGIN APS(VMV1I)
A00 063B2 00201P40  (00071)         JSN(VMV1OS,P2)
A01 063B4 02300030  (00072) SCLRS   SET(RO)                             ENABLE OUTPUT
A02 063B6 04C42F28  (00073)         LOAD(BRO,QPRMS(2),TF)
A03 063B8 06500005  (00074)         LOAD(BR1,5)
A04 063BA 08BA0002  (00075) #1      ADD(BRO,2,TF)
A05 063BC 0A1904B1  (00076)         SUBL(BR1,1),JUMPP(#1)               OPRM IN
A06 063BE 0CC42B28  (00077)         LOAD(BRO,QTDCT1S(2),TF)
A07 063C0 0E50007E  (00078)         LOAD(BR1,126)
A08 063C2 10BA0002  (00079) #2      ADD(BRO,2,TF)
A09 063C4 121908B1  (00080)         SUBL(BR1,1),JUMPP(#2)               QTDCT1 IN
                    (00081)
A0A 063C6 14C42D28  (00082)         LOAD(BRO,MIBIT4S(2),TF)
A0B 063C8 1650007E  (00083)         LOAD(BR1,126)
A0C 063CA 18BA0002  (00084) #4      ADD(BRO,2,TF)
A0D 063CC 1A190CB1  (00085)         SUBL(BR1,1),JUMPP(#4)               MIBIT4 IN
A0E 063CE 1CC2BDB0  (00086)         LOAD(BRO,INPBS(1),TF)
A0F 063D0 1E500079  (00087)         LOAD(BR1,121)                       123*2 HALFWORDS
A10 063D2 20BA0002  (00088) #6      ADD(BRO,2,TF)
A11 063D4 221910B1  (00089)         SUBL(BR1,1),JUMPP(#6)               INPB IN
A12 063D6 24C40B34  (00090)         LOAD(BRO,QTDCTMS(2),TF)
A13 063D8 2650007E  (00091)         LOAD(BR1,126)
A14 063DA 28BA0002  (00092) #3      ADD(BRO,2,TF)
A15 063DC 2A1914B1  (00093)         SUBL(BR1,1),JUMPP(#3)               QTDCTM IN
A16 063DE 2CC42C28  (00094)         LOAD(BRO,MIBIT3S(2),TF)
A17 063E0 2E50007E  (00095)         LOAD(BR1,126)
A18 063E2 30BA0002  (00096) #5      ADD(BRO,2,TF)
A19 063E4 321918B1  (00097)         SUBL(BR1,1),JUMPP(#5)               MIBIT3 IN
A1A 063E6 35420503  (00098) INPST   LOAD(BRO,ISAS001(1),TF)             IO=1
                    (00099) *
A1B 063E8 36200031  (00100) DNEST   CLEAR(RI)
A1C 063EA 38000020  (00101)         NOP(0)
                    (00102) *
A1D 063EC 3A300032  (00103) VMV1OS  SET(RA)
```

```
A1E 063EE 3C500000 (00104)        LOAD(RW1,E01)                    DUMMY FOR EXEC
A1F 063F0 3E500000 (00105)        LOAD(RW1,MSS)                    DUMMY
A20 063F2 40500000 (00106)        LOAD(RW1,MSS)
A21 063F4 42C2BA00 (00107)        LOAD(RW0,AOPHS(1),TF)
A22 063F6 44500005 (00108)        LOAD(RW1,5)
A23 063F8 46BA0002 (00109) #1     ADD(RW0,2,TF)
A24 063FA 48112381 (00110)        SUBL(RW1,1),JUMPP(#1)            AOPH OUT
A25 063FC 4AC2BA0E (00111)        LOAD(RW0,AOTDCTS(1),TF)
A26 063FE 4C50007E (00112)        LOAD(RW1,126)
A27 06400 4EBA0002 (00113) #2     ADD(RW0,2,TF)
A28 06402 50112781 (00114)        SUBL(RW1,1),JUMPP(#2)            AOTDCT OUT
A29 06404 52C2BB0E (00115)        LOAD(RW0,AIBIT4S(1),TF)
A2A 06406 5450007E (00116)        LOAD(RW1,126)
A2B 06408 56BA0002 (00117) #4     ADD(RW0,2,TF)
A2C 0640A 58112B81 (00118)        SUBL(RW1,1),JUMPP(#4)            AIBIT4 OUT
A2D 0640C 5AC40C34 (00119)        LOAD(RW0,NINMS(2),TF)
A2E 0640E 5C500079 (00120)        LOAD(RW1,121)                    123*2 HALFWORDS
A2F 06410 5EBA0002 (00121) #6     ADD(RW0,2,TF)
A30 06412 60112F81 (00122)        SUBL(RW1,1),JUMPP(#6)            NINM OUT
A31 06414 62C42D2B (00123)        LOAD(RW0,OTDCT1S(2),TF)
A32 06416 6450007E (00124)        LOAD(RW1,126)
A33 06418 66BA0002 (00125) #3     ADD(RW0,2,TF)
A34 0641A 68113381 (00126)        SUBL(RW1,1),JUMPP(#3)            OTDCT1 OUT
A35 0641C 6AC42D2B (00127)        LOAD(RW0,MIBIT4S(2),TF)
A36 0641E 6C50007E (00128)        LOAD(RW1,126)
A37 06420 6EBA0002 (00129) #5     ADD(RW0,2,TF)
A38 06422 70113781 (00130)        SUBL(RW1,1),JUMPP(#5)            MIBIT4 OUT
                   (00131) *
                   (00132) *
A39 06424 73420575 (00133) INPSO  LOAD(RW0,ISAS115(1),TF)          SET XGO
                   (00134) *
A3A 06426 74200030 (00135) DNFSO  CLEAR(R0)
A3B 06428 76000020 (00136)        NOP(0)
          000063FE (00137)        VMV1SA=#C
    0642A          (00138)        END #A-1
                   (00139) *STORAGE BLOCK FOR CONSTRUCTED INSTRUCTION
    0642A 00000000 (00140) VMV1SI DATA 1F'0,0'
          0000007A (00141)        VMV1SZ=#L-VMV1IS
    0642C          (00142)        END
```

```
AFDTSORG:   00HFH (00014) (00037)
AIBIT4S:    0BB0F (00031) (00115)
AOPRS:      0BA00 (00029) (00107)
AOTDCTS:    0BA0F (00030) (00111)
CSPUSNDS:   021FC (00015) (00040)
DMYS:       00794 (00016)
DNFSI:      0001B (00100)
DNFSO:      0003A (00135)
INFSI:      0001A (00098)
INPSO:      00039 (00133)
INPRS:      0BD80 (00028) (00086)
ISAS001:    00503 (00020) (00098)
ISAS106:    0056C (00021)
ISAS115:    00575 (00022) (00133)
ISVTS:      00502 (00019) (00020) (00021) (00022)
MSS:        00000 (00018) (00105) (00106)
MIBIT3S:    02C28 (00027) (00094)
MIBIT4S:    02D28 (00026) (00082) (00127)
NINMS:      00C34 (00032) (00119)
OPRMS:      02F28 (00023) (00073)
OTDCTIS:    02B28 (00024) (00077) (00123)
OTDCTMS:    00B34 (00025) (00090)
SCLRS:      00001 (00064) (00072)
START:      0639C (00034) (00042)
VMV1S:      0639F (00038) (00048)
VMV1SA:     063FF (00067) (00137)
VMV1SI:     0642A (00063) (00140)
VMV1SSA:    00000 (00046) (00050) (00057)
VMV1SSZ:    00006 (00047) (00057) (00058)
VMV1SZ:     0007A (00066) (00141)
VMV1IS:     063B2 (00039) (00064) (00070) (00141)
VMV1OS:     0001D (00071) (00103)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E=    0
```

```
              (00001) *  FCB 23B..."VMOV2(Y)"    MOVE BUNCH OF BUFFERS TO OTHER BUFFERS
              (00002) *                                   ORIGINATED:13-MAY-80
              (00003) *                                   UPDATED: 20-JUL-80
              (00004) *
              (00005) *  MOVE FROM        TO
              (00006) *  ARQPR            ORPRM
              (00007) *  ARQDCT           ORDCTM
              (00008) *  MIBIT1           MIBIT2
              (00009) *  IORDR1           IORDR2
              (00010) *  DCTIT1           DCTIT2
              (00011) *  NOUT             OUTB
              (00012) ** AND SCALAR PAIRS:
              (00013) *  VAR,DC           OVAR,ODC
              (00014) *
              (00015) *  DEFINE GLOBAL SYMBOLS
00000BEB      (00016)        AFDTSORG=$BEB
000021FC      (00017)        CSPUSNOS=$21FC
00000794      (00018)        DMYS=$794
00000502      (00019)        ISVTS=$502
00000503      (00020)        ISAS001=ISVTS+D'1'
0000056A      (00021)        ISAS104=ISVTS+D'104'
00000571      (00022)        ISAS111=ISVTS+D'111'
00000574      (00023)        ISAS114=ISVTS+D'114'
00000576      (00024)        ISAS116=ISVTS+D'116'
00000003      (00025)        #M=3
00000000      (00026)        MSS=0
00000382      (00027)        SVTS=$0382
0000044A      (00028)        SAS100=SVTS+2*D'100'
0000044C      (00029)        SAS101=SVTS+2*D'101'
0000044E      (00030)        SAS102=SVTS+2*D'102'
00000450      (00031)        SAS103=SVTS+2*D'103'
000096B6      (00032)        ARQPRS=38582
000096C4      (00033)        ARQDCTS=38596
00002928      (00034)        MIBIT1S=10536
00002728      (00035)        IORDR1S=10024
00002328      (00036)        DCTIT1S=9000
00000F16      (00037)        NOUTS=3862
00002E36      (00038)        ORPRMS=11830
00002E44      (00039)        ORDCTMS=11844
00002A28      (00040)        MIBIT2S=10792
00002828      (00041)        IORDR2S=10280
00002528      (00042)        DCTIT2S=9512
00009444      (00043)        OUTBS=37956
              (00044) *
```

```
          00006450 (00045)        START=$6450
                   (00046) *
                   (00047) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000B7C (00048)        #L=AFDTSORG+3*2*(238-128)                    FCB 238
     00B7C 001F6452 (00049)        ADDR VMV2S(R7,1)
     00B7E 001F6464 (00050)        ADDR VMV21S(R7,1)
     00B80 001021FC (00051)        ADDR CSPUSNOS(,1,0)
                   (00052)        EJECT
```

```
          00006450 (00053)            #L=START
                   (00054) #>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<
                   (00055) #
                   (00056)            EVEN
    06450 0000     (00057)            DATA VMV2$SA
    06451 0005     (00058)            DATA VMV2$SZ
                   (00059) VMV2S      BEGIN APU(VMV2)
          00000000 (00060)            #A=0
A00 06452 081C0000 (00061) VMV2$SA    MOV(ZERO,DQ)\NOP                         CLEAR RG02
A01 06454 08FC0000 (00062) #1         MOV(IOA,DQ)\NOP
A02 06456 901D0001 (00063)            JUMPC(#1,FI)
                   (00064) #
A03 06458 20322032 (00065) DNESA      CLEAR(RA)
A04 0645A 00000000 (00066)            NOP
          00000005 (00067)            VMV2$SZ=#A-VMV2$SA
    0645C          (00068)            END VMV2$SZ
                   (00069)            EJECT
```

```
                            (00070) *>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<
                            (00071) *
                            (00072)          EVEN
      0645C 000064FA (00073)          ADDR VMV2SI
      0645E 00006466 (00074)          ADDR VMV21S+2*SCLRS
      06460 0000     (00075)          DATA 0
      06461 0088     (00076)          DATA VMV2SZ
      06462 000064A6 (00077)          ADDR VMV2SA
                     (00078)          EVEN
                     (00079) *
                     (00080) VMV21S   BEGIN APS(VMV21)
A00 06464 00202040 (00081)          JSN(VMV20S,P2)
A01 06466 02300030 (00082) SCLRS    SET(R0)                                   ENABLE OUTPUT
A02 06468 04C296B6 (00083)          LOAD(BR0,AROPRS(1),TF)
A03 0646A 06500005 (00084)          LOAD(BR1,5)
A04 0646C 088A0002 (00085) #1       ADD(BR0,2,TF)
A05 0646E 0A1904B1 (00086)          SUBL(BR1,1),JUMPP(#1)                     OPRM IN
A06 06470 0CC296C4 (00087)          LOAD(BR0,ARODCTS(1),TF)
A07 06472 0F50007E (00088)          LOAD(BR1,126)
A08 06474 108A0002 (00089) #2       ADD(BR0,2,TF)
A09 06476 121908B1 (00090)          SUBL(BR1,1),JUMPP(#2)                     QTDCT1 IN
A0A 06478 14C4292B (00091)          LOAD(BR0,MIBIT1S(2),TF)
A0B 0647A 1650007E (00092)          LOAD(BR1,126)
A0C 0647C 188A0002 (00093) #3       ADD(BR0,2,TF)
A0D 0647E 1A190CB1 (00094)          SUBL(BR1,1),JUMPP(#3)                     QTDCTM IN
A0E 06480 1CC4272B (00095)          LOAD(BR0,IORDR1S(2),TF)
A0F 06482 1E50007E (00096)          LOAD(BR1,126)
A10 06484 208A0002 (00097) #4       ADD(BR0,2,TF)
A11 06486 221910B1 (00098)          SUBL(BR1,1),JUMPP(#4)                     MIBIT4 IN
A12 06488 24C4232B (00099)          LOAD(BR0,DCT1T1S(2),TF)
A13 0648A 265000FF (00100)          LOAD(BR1,254)                             REAL
A14 0648C 288A0002 (00101) #5       ADD(BR0,2,TF)
A15 0648E 2A1914B1 (00102)          SUBL(BR1,1),JUMPP(#5)                     MIBIT3 IN
A16 06490 2CC40E16 (00103)          LOAD(BR0,NOUTS(2),TF)
A17 06492 2E500079 (00104)          LOAD(BR1,121)                             123*2 HALFWORDS
A18 06494 308A0002 (00105) #6       ADD(BR0,2,TF)
A19 06496 321918B1 (00106)          SUBL(BR1,1),JUMPP(#6)                     INPB IN
A1A 06498 34C2044F (00107)          LOAD(BR0,SAS102(1),TF)
A1B 0649A 36C20450 (00108)          LOAD(BR0,SAS103(1),TF)
A1C 0649C 39420503 (00109)          LOAD(BR0,ISAS001(1),TF)                   IQ=1
A1D 0649E 3B0A0000 (00110)          ADD(BR0,0,TF)                             AND AGAIN
                   (00111) *
A1E 064A0 3C200031 (00112) DNEST    CLEAR(RI)
A1F 064A2 3E000020 (00113)          NOP(0)
```

```
                        (00114) *   OUTPUT PGM
A20 064A4 40300032 (00115) VMV20S  SET(RA)
A21 064A6 42500000 (00116)         LOAD(BW1,101)                   DUMMY OP FOR EXEC
A22 064A8 44500000 (00117)         LOAD(BW1,MSS)
A23 064AA 46500000 (00118)         LOAD(BW1,MSS)
A24 064AC 49420576 (00119)         LOAD(BWO,ISAS116(1),TE)         CLEAR RGO2
A25 064AE 4AC42F36 (00120)         LOAD(BWO,ORPRMS(2),TE)
A26 064B0 4C500005 (00121)         LOAD(BW1,5)
A27 064B2 4EBA0002 (00122) #1      ADD(BWO,2,TE)
A28 064B4 501127B1 (00123)         SUBL(BW1,1),JUMPP(#1)           AQPB OUT
A29 064B6 52C42E44 (00124)         LOAD(BWO,QRDCTMS(2),TE)
A2A 064B8 5450007E (00125)         LOAD(BW1,126)
A2B 064BA 56BA0002 (00126) #2      ADD(BWO,2,TE)
A2C 064BC 581128B1 (00127)         SUBL(BW1,1),JUMPP(#2)           AQTDCT OUT
                   (00128) *
                   (00129) *
A2D 064BE 5AC42A28 (00130)         LOAD(BWO,MIBIT2S(2),TE)
A2E 064C0 5C50007E (00131)         LOAD(BW1,126)
A2F 064C2 5EBA0002 (00132) #3      ADD(BWO,2,TE)
A30 064C4 60112EB1 (00133)         SUBL(BW1,1),JUMPP(#3)           QTDCT1 OUT
A31 064C6 62C42828 (00134)         LOAD(BWO,IORDR2S(2),TE)
A32 064C8 6450007E (00135)         LOAD(BW1,126)
A33 064CA 66BA0002 (00136) #4      ADD(BWO,2,TE)
A34 064CC 681133B1 (00137)         SUBL(BW1,1),JUMPP(#4)           AIBIT4 OUT
A35 064CE 6AC42528 (00138)         LOAD(BWO,DCT1T2S(2),TE)
A36 064D0 6C5000FE (00139)         LOAD(BW1,254)
A37 064D2 6EBA0002 (00140) #5      ADD(BWO,2,TE)
A38 064D4 701137B1 (00141)         SUBL(BW1,1),JUMPP(#5)           MIBIT4 OUT
A39 064D6 72C29444 (00142)         LOAD(BWO,OUTBS(1),TE)
A3A 064D8 74500079 (00143)         LOAD(BW1,121)                   123*2 HALFWORDS
A3B 064DA 76BA0002 (00144) #6      ADD(BWO,2,TE)
A3C 064DC 7B113BB1 (00145)         SUBL(BW1,1),JUMPP(#6)           NINM OUT
A3D 064DE 7AC2044A (00146)         LOAD(BWO,SAS100(1),TE)
A3E 064E0 7CC2044C (00147)         LOAD(BWO,SAS101(1),TE)
A3F 064E2 7E420574 (00148)         LOAD(BWO,ISAS114(1),TE)         SET RGO
A40 064E4 81420571 (00149)         LOAD(BWO,ISAS111(1),TE)         SET OUTFUL
                   (00150) *
A41 064E6 82200030 (00151) DNESD   CLEAR(RO)
A42 064E8 84000020 (00152)         NOP(0)
          000064A6 (00153)         VMV2SA=#C
    064EA          (00154)         END #A-1
                   (00155) *STORAGE BLOCK FOR CONSTRUCTED INSTRUCTION
    064EA 00000000 (00156) VMV2ST  DATA 1F'0,0'
          00000088 (00157)         VMV2SZ=#L-VMV2TS
```

```
064EC        (00158)        END
```

```
PAGE    7:      FCN 238..."VMOV2(Y)"    MOVE BUNCH OF BUFFERS TO OTHER BUFFERS

  AFDTSORG:  008F8 (00016) (00048)
  AROPCTS:   096C4 (00033) (00087)
  AROPHS:    096F6 (00032) (00083)
  CSPHSNOS:  021FC (00017) (00051)
  DCT1T1S:   02328 (00036) (00099)
  DCT1T2S:   02528 (00042) (00138)
  DMYS:      00794 (00018)
  DNFSA:     00003 (00065)
  DNFSI:     0001F (00112)
  DNFSO:     00041 (00151)
  IORDR1S:   02728 (00035) (00095)
  IORDR2S:   02828 (00041) (00134)
  ISAS001:   00503 (00020) (00109)
  ISAS104:   0056A (00021)
  ISAS111:   00571 (00022) (00149)
  ISAS114:   00574 (00023) (00148)
  ISAS116:   00576 (00024) (00119)
  ISVTS:     00502 (00019) (00020) (00021) (00022) (00023) (00024)
  MSS:       00000 (00026) (00117) (00118)
  MIBIT1S:   02928 (00034) (00091)
  MIBIT2S:   02A28 (00040) (00130)
  NOUTS:     00F16 (00037) (00103)
  OUTHS:     09444 (00043) (00142)
  ORPCTMS:   02F44 (00039) (00124)
  ORPHMS:    02F36 (00038) (00120)
  SAS100:    0044A (00028) (00146)
  SAS101:    0044C (00029) (00147)
  SAS102:    0041F (00030) (00107)
  SAS103:    00450 (00031) (00108)
  SCLRS:     00001 (00074) (00082)
  START:     06450 (00045) (00053)
  SVTS:      003B2 (00027) (00028) (00029) (00030) (00031)
  VMV2S:     06452 (00049) (00059)
  VMV2SA:    064A6 (00077) (00153)
  VMV2SI:    064FA (00073) (00156)
  VMV2SSA:   00000 (00057) (00061) (00067)
  VMV2SSZ:   00005 (00058) (00067) (00068)
  VMV2SZ:    0008B (00076) (00157)
  VMV2IS:    06464 (00050) (00074) (00080) (00157)
  VMV2OS:    00020 (00081) (00115)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)    F=    0
```

```
                    (00001) '  FCB 239,"VMOV3(Y)"      MOVE BUNCH OF BUFFERS TO OTHER
                    (00002) *                                   BUFFERS.
                    (00003) *                                   ORIGINATED: 13-MAY-80
                    (00004) *                                   UPDATED: 01-JUL-80
                    (00005) *
                    (00006) *  MOVE FROM       TO
                    (00007) *  MIBIT1          RIBIT1
                    (00008) *  TMP4            DCT1T1
                    (00009) *  IORDRM          IORDR1
                    (00010) ** AND SCALAR PAIRS:
                    (00011) *  NVAR,NDC                VAR,DC
                    (00012) *
                    (00013) *  DEFINE GLOBAL SYMBOLS
           000008EB (00014)        AFDTSORG=$8EB
           000021FC (00015)        CSPUSNOS=$21FC
           00000794 (00016)        DMYS=$794
           00000003 (00017)        #M=3
           00000000 (00018)        MSS=0
           00000382 (00019)        SVTS=$0382
           00000432 (00020)        SAS88=SVTS+2*D'88'
           00000434 (00021)        SAS89=SVTS+2*D'89'
           0000044E (00022)        SAS102=SVTS+2*D'102'
           00000450 (00023)        SAS103=SVTS+2*D'103'
           00000502 (00024)        ISVTS=$0502
           00000503 (00025)        ISAS1=ISVTS+D'1'                      UNITY
           00000576 (00026)        ISAS116=ISVTS+D'116'                  RG02
           00000828 (00027)        MIBITS=2088
           00000C00 (00028)        TMP4S=3072
           00000000 (00029)        IORDRMS=0
           00009AA6 (00030)        RIBITS=39590
           00002328 (00031)        DCT1T1S=9000
           00002728 (00032)        IORDR1S=10024
           00002928 (00033)        MIBIT1S=10536
                    (00034) *
           00006500 (00035)        START=$6500
                    (00036) *
                    (00037) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
           00000BB2 (00038)        *L=AFDTSORG+3*2*(239-128)             FCB 239
00BB2 001E6502 (00039)        ADDR VMV3S(P7,1)
00BB4 001E6516 (00040)        ADDR VMV3IS(P7,1)
00BB6 001021FC (00041)        ADDR CSPUSNOS(,1,0)
                    (00042)        EJECT
```

```
            00006500 (00043)          #L=START
                     (00044) *>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<
                     (00045) *
                     (00046)          EVEN
      06500 0000     (00047)          DATA VMV3$SA
      06501 0006     (00048)          DATA VMV3$SZ
                     (00049) VMV3$    BEGIN APU(VMV3)
            00000000 (00050)          #A=0
  A00 06502 00000000 (00051) VMV3$SA NOP
  A01 06504 08FC0000 (00052) #2       MOV(IQA,DQ)\NOP
  A02 06506 901D0001 (00053)          JUMPC(#2,F1)
  A03 06508 20322032 (00054)          CLEAR(RA)
  A04 0650A 00000000 (00055)          NOP
  A05 0650C 10000000 (00056)          JUMP(0)
            00000006 (00057)          VMV3$SZ=#A-VMV3$SA
      0650F          (00058)          END VMV3$SZ
                     (00059)          EJECT
```

```
                      (00060) *>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<
                      (00061) *
                      (00062)         EVEN
     0650E 0000656B (00063)           ADDR VMV3ST
     06510 00006518 (00064)           ADDR VMV3TS+2*SCLRS
     06512 0000     (00065)           DATA 0
     06513 0054     (00066)           DATA VMV3SZ
     06514 00006540 (00067)           ADDR VMV3SA
                      (00068)         EVEN
                      (00069) *
                      (00070) VMV3TS  BEGIN APS(VMV3T)
A00 06516 00201440 (00071)           JSN(VMV3OS,P2)
A01 06518 02300030 (00072) SCLRS   SET(RO)                          ENABLE OUTPUT
A02 0651A 04C42928 (00073)           LOAD(BRO,MIBIT1S(2),TF).
A03 0651C 0650007D (00074)           LOAD(BR1,125)
A04 0651E 08BA0002 (00075) #1      ADD(BRO,2,TF)
A05 06520 0A1904B1 (00076)           SUBL(BR1,1),JUMPP(#1)            MIBIT IN
A06 06522 0CBA0002 (00077)           ADD(BRO,2,TF)
A07 06524 0FC50C00 (00078)           LOAD(BRO,TMP4S(3),TF)
A08 06526 105000FE (00079)           LOAD(BR1,254)
A09 06528 12BA0002 (00080) #2      ADD(BRO,2,TF)
A0A 0652A 141909B1 (00081)           SUBL(BR1,1),JUMPP(#2)            TMP4 IN
A0B 0652C 16C60000 (00082)           LOAD(BRO,IORDRMS(3),TF)
A0C 0652E 1B50007E (00083)           LOAD(BR1,126)
A0D 06530 1ABA0002 (00084) #3      ADD(BRO,2,TF)
A0E 06532 1C1900B1 (00085)           SUBL(BR1,1),JUMPP(#3)            IORDRM IN
A0F 06534 1EC20432 (00086)           LOAD(BRO,SASB8(1),TF)
A10 06536 20C20434 (00087)           LOAD(BRO,SASB9(1),TF)
A11 06538 23420503 (00088)           LOAD(BRO,ISAS1(1),TF)            UNITY FOR RG02
A12 0653A 24200031 (00089)           CLEAR(RI)
A13 0653C 26000020 (00090)           NOP(0)
                      (00091) *  OUTPUT PGM
A14 0653E 28300032 (00092) VMV3OS  SET(RA)
A15 06540 2A500000 (00093)           LOAD(BW1,(0))                    DUMMY OP FOR EXEC
A16 06542 2C500000 (00094)           LOAD(BW1,MSS)
A17 06544 2E500000 (00095)           LOAD(BW1,MSS)
A18 06546 30C29AA6 (00096)           LOAD(BWO,RIBITS(1),TF)
A19 06548 3250007E (00097)           LOAD(BW1,126)
A1A 0654A 34BA0002 (00098) #1      ADD(BWO,2,TF)
A1B 0654C 36111AB1 (00099)           SUBL(BW1,1),JUMPP(#1)            RIBIT,MIBIT1 OUT
A1C 0654E 38C42328 (00100)           LOAD(BWO,DCT1T1S(2),TF)
A1D 06550 3A5000FE (00101)           LOAD(BW1,254)
A1E 06552 3CBA0002 (00102) #2      ADD(BWO,2,TF)
A1F 06554 3E111EB1 (00103)           SUBL(BW1,1),JUMPP(#2)            DCT1T1 OUT
```

```
A20 06556 40C4272B (00104)          LOAD(RW0,IORDR1S(2),TF)
A21 06558 4250007F (00105)          LOAD(RW1,126)
A22 0655A 448A0002 (00106) #3       ADD(RW0,2,TF)
A23 0655C 461122B1 (00107)          SUBL(RW1,1),JUMPP(#3)                    IORDR1 OUT
A24 0655E 4BC2044F (00108)          LOAD(RW0,SAS102(1),TF)
A25 06560 4AC20450 (00109)          LOAD(RW0,SAS103(1),TF)
A26 06562 4D420576 (00110)          LOAD(RW0,ISAS116(1),TF)                  SET RG02 TO LET CSPU CON
                   (00111) ;                                                 TINUE
A27 06564 4F200030 (00112)          CLEAR(R0)
A28 06566 50000020 (00113)          NOP(0)
          00006540 (00114)          VMV3SA=#C
    06568          (00115)          END #A-1
                   (00116) *STORAGE BLOCK FOR CONSTRUCTED INSTRUCTION
    06568 00000000 (00117) VMV3ST   DATA 1F'0,0'
          00000054 (00118)          VMV3SZ=#L-VMV31S
    0656A          (00119)          END
```

```
AFDT$ORG:   008EE (00014) (00038)
CSPUSNOS:   021FC (00015) (00041)
DCT1T1$:    02328 (00031) (00100)
DMYS:       00794 (00016)
IORDR1$:    02728 (00032) (00104)
IORDRM$:    00000 (00029) (00082)
ISAS1:      00503 (00025) (00088)
ISAS116:    00576 (00026) (00110)
ISVTS:      00502 (00024) (00025) (00026)
MSS:        00000 (00018) (00094) (00095)
MIBITS:     00828 (00027)
MIBIT1$:    02928 (00033) (00073)
RIBITS:     09AA6 (00030) (00096)
SAS102:     0044F (00022) (00108)
SAS103:     00450 (00023) (00109)
SAS88:      00432 (00020) (00086)
SAS89:      00434 (00021) (00087)
SCLR$:      00001 (00064) (00072)
START:      06500 (00035) (00043)
SVTS:       00382 (00019) (00020) (00021) (00022) (00023)
TMP4$:      00C00 (00028) (00078)
VMV3$:      06502 (00039) (00049)
VMV3SA:     06540 (00067) (00114)
VMV3SI:     06568 (00063) (00117)
VMV3SSA:    00000 (00047) (00051) (00057)
VMV3SSZ:    00006 (00048) (00057) (00058)
VMV3SZ:     00054 (00066) (00118)
VMV3I$:     06516 (00040) (00064) (00070) (00118)
VMV3O$:     00014 (00071) (00092)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E=    0
```

```
                   (00001) *  FCH 240... "MPEDVM(Y,U,V)"      FLOAT,DCBIAS,VAR,REFLECT INPUT VIA MAKHOUL ALG.
                   (00002) *                                  ORIGINATED:06-OCT-79
                   (00003) *                                  UPDATED:10-JUL-80
                   (00004) *  DEFINE GLOBAL SYMBOLS
                   (00005)         OPADD EXP,(1 .LS. 14)+(12 .LS. 8)+$1F
                   (00006)         OPADD DOF,(3 .LS. 10) + (12 .LS. 5) + $1B
          00000BEB (00007)         AEDTSORG=$BEB
          000021FC (00008)         CSPUSNOS=$21FC
          00000794 (00009)         DMYS=$794
          00000003 (00010)         #M=3
          00000001 (00011)         US=1
          00000000 (00012)         MSS=0
          00000014 (00013)         KP2S=2*D'10'
          00000502 (00014)         ISVTS=$502
          0000057F (00015)         ISAS125=ISVTS+D'125'
          00000570 (00016)         ISAS110=ISVTS+D'110'                INFULL
          00000382 (00017)         SVTS=$0382
          000003D0 (00018)         SAS39=SVTS+2*D'39'
          000003EA (00019)         SAS52=SVTS+2*D'52'
          000003EE (00020)         SAS54=SVTS+2*D'54'
          000003F0 (00021)         SAS55=SVTS+2*D'55'
          0000043C (00022)         SAS93=SVTS+2*D'93'
          0000043E (00023)         SAS94=SVTS+2*D'94'
          00000440 (00024)         SAS95=SVTS+2*D'95'
          00008600 (00025)         START=$8600
          00000002 (00026)         WS=2
          00000004 (00027)         WWS=4
          00000003 (00028)         ZS=3
                   (00029) *
                   (00030) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000BB8 (00031)         #I=AEDTSORG+3*2*(240-128)           FCH 240
     0BB8 001E8620 (00032)         ADDR EDVS(R7,1)
     0BBA 001E87B6 (00033)         ADDR G200S(R7,1)
     0BBC 001021FC (00034)         ADDR CSPUSNOS(,1,0)
                   (00035)         EJECT
```

```
                      *
                      *>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      *
          00008600             #L=START
                               EVEN
                      *  SPECIAL CONSTANTS
                      *
08600 4000            ILOGS1   DATA $4000                          128
08601 0042                     DATA $0042
08602 2000                     DATA $2000                          64
08603 0042                     DATA $0042
08604 CA77                     DATA $CA77                          C8
08605 68C0                     DATA $68C0
08606 0949            ILOGS2   DATA $0949                          C3
08607 5742                     DATA $5742
08608 D00F                     DATA $D00F                          C2
08609 58C1                     DATA $58C1
0860A 0800                     DATA $0800                          16**4
0860B 0045                     DATA $0045
0860C 1EB5                     DATA $1EB5                          C7
0860D 5541                     DATA $5541
0860E 209F                     DATA $209F                          C1
0860F AB41                     DATA $AB41
08610 DFCC                     DATA $DFCC                          C6
08611 BF41                     DATA $BF41
08612 88D9                     DATA $88D9                          C0
08613 D0C1                     DATA $D0C1
08614 0ABE                     DATA $0ABE                          C5
08615 57C2                     DATA $57C2
08616 8C07                     DATA $8C07                          C4
08617 18C2                     DATA $18C2
                      *
08618 4000            ISORS1   DATA $4000                          0.5
08619 0040                     DATA $0040
0861A 0800                     DATA $0800                          16**7
0861B 0048                     DATA $0048
0861C 1000                     DATA $1000                          2.0
0861D 0041                     DATA $0041
                               EJECT
```

```
                        (00075) *
                        (00076) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        (00077) *
                        (00078)          EVEN
    0861E 0000          (00079)          DATA FDVSSA                                       START ADDRESS
    0861F 00C7          (00080)          DATA FDVSSZ                                       MODULE SIZE
                        (00081) FDVS     BEGIN APU(FDV)
          00000000      (00082)          #A=0
A00 08620 00000000      (00083) FDVSSA   NOP                                               <<<<<FOR TIMING
A01 08622 081C0000      (00084)          MOV(ZERO,OO)\NOP                                  <<<<<CALLING CARD=0
                        (00085) *  INCLUDE UPDATE SAMPLES IN SUM, THEN FLOAT NEW SAMPLES, OUTPUT,
                        (00086) *         AND INCLUDE IN SUM
A02 08624 08100810      (00087)          MOV(ZERO,A0)                                      SUM
A03 08626 02000200      (00088)          R(A0)
A04 08628 08FF08FF      (00089)          MOV(IQA,M7)                                       SA = 2**15
A05 0862A 08FF08FF      (00090)          MOV(IQA,M6)                                       SB = 1/LTH
A06 0862C 08F10890      (00091) FLSM1    MOV(IQA,A1)\MOV(R,A0)
A07 0862E 089008F1      (00092)          MOV(R,A0)\MOV(IQA,A1)
A08 08630 41004100      (00093)          ADD(A0,A1)
A09 08632 90160006      (00094)          JUMPC(FLSM1,FW1)                                  INCLUDE UPDATE SAMPLES I
                        (00095) ;                                                          N SUM
A0A 08634 20372037      (00096)          CLEAR(W1)
A0B 08636 08110811      (00097)          MOV(ZERO,A1)                                      FAKE FOR FIRST ADD
A0C 08638 1000000F      (00098)          JUMP(FLSM2)                                       AVOID FAKE OUTPUTS
A0D 0863A 08810881      (00099) FLTSM    MOV(P,A1)
A0E 0863C 08BC08BC      (00100)          MOV(P,OO)
A0F 0863E 08F20890      (00101) FLSM2    MOV(IQA,A2)\MOV(R,A0)                             X\SUM
A10 08640 089008F2      (00102)          MOV(R,A0)\MOV(IQA,A2)                             SUM\X
A11 08642 32403240      (00103)          NORM(A2)                                          FLOAT X
A12 08644 41084108      (00104)          MOV(M0),ADD(A0,A1)                                FLOAT X,SUM OF X FROM LA
                        (00105) ;                                                          ST TIME THRU
A13 08646 84608460      (00106)          MUL(M0,M7)                                        FLOAT X*2**15
A14 08648 9016000D      (00107)          JUMPC(FLTSM,FW1)
A15 0864A 08810881      (00108)          MOV(P,A1)
A16 0864C 08BC08BC      (00109)          MOV(P,OO)
A17 0864E 41104110      (00110)          MOV(A0),ADD(A0,A1)                                ADD LAST TERM INTO SUM
A18 08650 20372037      (00111)          CLEAR(W1)                                         GIVE APS LOTS OF TIME FO
                        (00112) ;                                                          R NEXT INPUTS
A19 08652 08900898      (00113)          MOV(R,A0)\MOV(R,FX0)                              LEFT SUM\RIGHT SUM
A1A 08654 08510000      (00114)          MOV(FX1,A1)\NOP                                   RIGHT SUM
A1B 08656 41000000      (00115)          ADD(A0,A1)\NOP                                    LEFT SUM + RIGHT SUM
A1C 08658 08880888      (00116)          MOV(R,M0)                                         SUM
A1D 0865A 84408440      (00117)          MUL(M0,M6)                                        SUM*1/LTH
A1E 0865C 08BC0000      (00118)          MOV(P,OO)\NOP                                     DCBIAS
```

```
 A1F 0865E 08B80000 (00119)          MOV(P,FX0)\NOP
 A20 08660 08870857 (00120)          MOV(P,A7)\MOV(FX1,A7)                  DCBIAS
                    (00121) *
                    (00122) *X'=X-DCBIAS,SUM(X')^2 FOR VARIANCE
 A21 08662 08120812 (00123) VSQS     MOV(ZERO,A2)                           ZERO SUM
 A22 08664 02400240 (00124)          R(A2)                                  GET 0 IN R
 A23 08666 10000025 (00125)          JUMP(SQ2)                              SKIP FAKE PRODUCTS
 A24 08668 08820882 (00126) SQ1      MOV(P,A2)
 A25 0866A 08E10890 (00127) SQ2      MOV(IQA,A1)\MOV(R,A0)                  X\SUM,ZERO FIRST TIME
 A26 0866C 08900BE1 (00128)          MOV(R,A0)\MOV(IQA,A1)                  SUM,ZERO FIRST TIME\X
 A27 0866E 4F204F20 (00129)          SUB(A1,A7)                             X-BIAS
 A28 08670 08880888 (00130)          MOV(R,M0)
 A29 08672 088C088C (00131)          MOV(R,M4)                              PREPARE TO SQUARE
 A2A 08674 84008400 (00132)          MUL(M0,M4)                             SQUARE
 A2B 08676 421C421C (00133)          MOV(OQ),ADD(A0,A2)                     OUTPUT X-BIAS,ADD IN SQU
                    (00134) ;                                               ARE FROM LAST TIME
 A2C 08678 90160024 (00135)          JUMPC(SQ1,FWI)
 A2D 0867A 08820882 (00136)          MOV(P,A2)                              LAST SQUARE
 A2E 0867C 08900890 (00137)          MOV(R,A0)                              SUM FROM LOOP
 A2F 0867E 42004200 (00138)          ADD(A0,A2)                             ADD IN LAST SQUARE
 A30 08680 08900898 (00139)          MOV(R,A0)\MOV(R,EX0)                   LEFT SUM\RIGHT SUM
 A31 08682 08520000 (00140)          MOV(EX1,A2)\NOP                        RIGHT SUM
 A32 08684 42000000 (00141)          ADD(A0,A2)\NOP                         LEFT SUM + RIGHT SUM
 A33 08686 08880000 (00142)          MOV(R,M0)\NOP
 A34 08688 84400000 (00143)          MUL(M0,M6)\NOP                         SUM/LTH
 A35 0868A 088C0000 (00144)          MOV(P,GO)\NOP                          VAR^2=SD
 A36 0868C 90180036 (00145) #1       JUMPC(#1,OOE)                          WAIT FOR SD TO BE WRITTE
                    (00146) ;                                               N
 A37 0868E 20372037 (00147)          CLEAR(WI)                              THEN LET INPUT GO
 A38 08690 00000000 (00148)          NOP
                    (00149) *
                    (00150) *  VAR=SQRT(SD)
 A39 08692 08D70000 (00151) SSQRTS   MOV(IQ,A7)\NOP                         A7=$40000040
 A3A 08694 08EF0000 (00152)          MOV(IQA,M7)\NOP                        M7=0.5
 A3B 08696 08E60000 (00153)          MOV(IQA,A6)\NOP                        A6=16**7
 A3C 08698 08E50000 (00154)          MOV(IQA,A5)\NOP                        A5=2.0
 A3D 0869A 08E20000 (00155)          MOV(IQA,A2)\NOP                        A2=X(0,K)
 A3E 0869C 08E00000 (00156)          MOV(IQA,A0)\NOP                        A0=X(1,K)
 A3F 0869E 12400000 (00157)          ABS(A2)\NOP                            ABS(X(0,K))
 A40 086A0 12080000 (00158)          MOV(M0),ABS(A0)\NOP                    M0=ABS(X(0,K)),X(2,K)
 A41 086A2 08900000 (00159)          MOV(R,A0)\NOP                          A0=X(2,K)
 A42 086A4 4F000000 (00160)          SUB(A0,A7)\NOP                         X(3,K)
 A43 086A6 088A0000 (00161)          MOV(R,M2)\NOP                          M2=X(3,K)
 A44 086A8 85600000 (00162)          MUL(M2,M7)\NOP                         X(4,K)
```

```
A45 086AA 02C00000 (00163)          R(A6)\NOP                        X(5)
A46 086AC 08B00000 (00164)          MOV(P,A0)\NOP                    A0=X(4,K)
A47 086AE 47140000 (00165)          MOV(A4),ADD(A0,A7)\NOP           A4=X(5),X(6,K)
A48 086B0 1F100000 (00166)          MOV(A0),EXP(A0)\NOP              A0=X(6,K),X(7,K)
A49 086B2 08BA0000 (00167)          MOV(R,M2)\NOP                    M2=X(7,K)
A4A 086B4 08900000 (00168)          MOV(R,A0)\NOP                    A0=X(7,K)
A4B 086B6 2F000000 (00169) #2       RCP(A0)\NOP                      X(8,K)
A4C 086B8 08BC0000 (00170)          MOV(R,M4)\NOP                    M4=X(8,K)
A4D 086BA 85000000 (00171)          MUL(M2,M4)\NOP                   X(9,K)
A4E 086BC 08B20000 (00172)          MOV(P,A2)\NOP                    A2=X(9,K)
A4F 086BE 4AA00000 (00173)          SUB(A5,A2)\NOP                   X(10,K)
A50 086C0 08BA0000 (00174)          MOV(R,M2)\NOP                    M2=X(10,K)
A51 086C2 85000000 (00175)          MUL(M2,M4)\NOP                   X(11,K)
A52 086C4 02000000 (00176)          R(A0)\NOP                        GET X(7,K)
A53 086C6 08AC0000 (00177)          MOV(P,M4)\NOP                    M4=X(11,K)
A54 086C8 08BA0000 (00178)          MOV(R,M2)\NOP                    M2=X(7,K)
A55 086CA 85000000 (00179)          MUL(M2,M4)\NOP                   X(12,K)
A56 086CC 08B20000 (00180)          MOV(P,A2)\NOP                    A2=X(12,K)
A57 086CE 4AA00000 (00181)          SUB(A5,A2)\NOP                   X(13,K)
A58 086D0 08BA0000 (00182)          MOV(R,M2)\NOP                    M2=X(13,K)
A59 086D2 85000000 (00183)          MUL(M2,M4)\NOP                   X(14,K)
A5A 086D4 08AC0000 (00184)          MOV(P,M4)\NOP                    M4=X(14,K)
A5B 086D6 84000000 (00185)          MUL(M0,M4)\NOP                   X(15,K)
A5C 086D8 08B20000 (00186)          MOV(P,A2)\NOP                    A2=X(15,K)
A5D 086DA 42000000 (00187)          ADD(A0,A2)\NOP                   X(16,K)
A5E 086DC 08BA0000 (00188)          MOV(R,M2)\NOP                    M2=X(16,K)
A5F 086DE 85600000 (00189)          MUL(M2,M3)\NOP                   X(7,K)
A60 086E0 2A800000 (00190)          DECRE(A4)\NOP                    X(5)
A61 086E2 08B00000 (00191)          MOV(P,A0)\NOP                    A0=X(7,K)
A62 086E4 08AA0000 (00192)          MOV(P,M2)\NOP                    M2=X(7,K)
A63 086E6 08940000 (00193)          MOV(R,A4)\NOP                    A4=X(5)
A64 086E8 901F004B (00194)          JUMPC(#2,T1)                     DONE YET?
A65 086EA 02000000 (00195)          R(A0)\NOP                        YES!
A66 086EC 089C0000 (00196)          MOV(R,D0)\NOP                    VAR=SQRT(SD)
A67 086EE 20372037 (00197)          CLEAR(WI)                        RELEASE APS INPUT
                   (00198) *
                   (00199) *  VAR'=1.0/VAR
A68 086F0 08BF0000 (00200) SINVS    MOV(R,M6)\NOP                    M6=SC=VAR
A69 086F2 08900000 (00201)          MOV(R,A0)\NOP                    A0=SC=VAR
A6A 086F4 16A016A0 (00202)          K(2)                             R=2.0;R=2.0
A6B 086F6 08960000 (00203)          MOV(R,A6)\NOP                    A6=2
A6C 086F8 2F000000 (00204)          RCP(A0)\NOP                      R1=1/C+DEL(=F0)
A6D 086FA 08B80000 (00205)          MOV(R,M0)\NOP                    M0=F0
A6E 086FC 84400000 (00206)          MUL(M0,M6)\NOP                   P1=F0*C
```

```
A6F 086FF 08B10000 (00207)         MOV(P,A1)\NOP                 A1=F0*C
A70 08700 49C00000 (00208)         SUB(A6,A1)\NOP                R2=2-F0*C
A71 08702 08BC0000 (00209)         MOV(R,M4)\NOP                 M4=R2
A72 08704 84000000 (00210)         MUL(M0,M4)\NOP                P2=F0*R2(=F1)
A73 08706 08A80000 (00211)         MOV(P,M0)\NOP                 M0=F1
A74 08708 84400000 (00212)         MUL(M0,M6)\NOP                P3=F1*C
A75 0870A 08B10000 (00213)         MOV(P,A1)\NOP                 A1=P3
A76 0870C 49C00000 (00214)         SUB(A6,A1)\NOP                R3=2-F1*C
A77 0870E 08BC0000 (00215)         MOV(R,M4)\NOP                 M4=R3
A78 08710 84000000 (00216)         MUL(M0,M4)\NOP                P4=F1*(2-F1*C)(=F2)
A79 08712 08BC0000 (00217)         MOV(P,00)\NOP                 00=SA=1.0/VAR
A7A 08714 08B80000 (00218)         MOV(P,FX0)\NOP                FX0=SA=1.0/VAR
                   (00219) *
                   (00220) *  MAPX(I)=MAPX(I)*VAR'    I=0,1,2...LTH-1
                   (00221) *  ALSO DO MAKHOUL REFLECTION V(K)=X(2K),V(N-K-1)=X(2K+1)
A7B 08716 08AF0B4F (00222) VMULS   MOV(P,M7)\MOV(FXI,M7)         1.0/VAR
A7C 08718 08F80000 (00223) #1      MOV(IOA,M0)\NOP               X
A7D 0871A 000008F8 (00224)         NOP\MOV(IOA,M0)               X
A7E 0871C 84608460 (00225)         MUL(M0,M7)                    V*VAR'
A7F 0871E 08BC081C (00226)         MOV(P,00)\MOV(ZERO,00)        XR,XI (EVEN)
A80 08720 081C08BC (00227)         MOV(ZERO,00)\MOV(P,00)        XI,XR (ODD)
A81 08722 9016007C (00228)         JUMPC(#1,FWI)
A82 08724 20372037 (00229)         CLEAR(WI)
A83 08726 00000000 (00230)         NOP
                   (00231)
                   (00232) *  VAR''=ALOG10(VAR)
A84 08728 08F00000 (00233) SLOGS   MOV(IOA,M6)\NOP               M6=SA=ALOG16(10.0)
A85 0872A 08FA0000 (00234)         MOV(IOA,M2)\NOP               M2=128
A86 0872C 08F70000 (00235)         MOV(IOA,A7)\NOP               A7=64
A87 0872E 08FF0000 (00236)         MOV(IOA,M7)\NOP               M7=C8
A88 08730 08D20000 (00237)         MOV(IO,A2)\NOP                A2=X(0,K)
A89 08732 08FD0000 (00238)         MOV(IOA,M5)\NOP               M5=X(0,K)
A8A 08734 26400000 (00239)         RCB(A2)\NOP                   X(1,K)
A8B 08736 08F10000 (00240) #1      MOV(IOA,A1)\NOP               A1=X(3,K)
A8C 08738 26290000 (00241)         MOV(M1),RCB(A1)\NOP           M1=X(1,K),X(4,K)
A8D 0873A 84B30000 (00242)         MOV(A3),MUL(M1,M5)\NOP        A3=X(18,K-1),X(2,K)
A8E 0873C 08F20000 (00243)         MOV(IOA,A2)\NOP               A2=X(5,K)
A8F 0873E 45710000 (00244)         MOV(A1),ADD(A3,A5)\NOP        A1=X(4,K),X(19,K-1)
A90 08740 124C0000 (00245)         MOV(M4),ABS(A2)\NOP           M4=X(19,K-1),X(8,K)
A91 08742 08B50000 (00246)         MOV(P,A5)\NOP                 A5=X(2,K)
A92 08744 858B0000 (00247)         MOV(M0),MUL(M3,M4)\NOP        M0=X(2,K),X(20,K-1)
A93 08746 2A2D0000 (00248)         MOV(M5),DECRE(A1)\NOP         M5=X(8,K),X(6,K)
A94 08748 08F20000 (00249)         MOV(IOA,A2)\NOP               A2=C3
A95 0874A 0F310000 (00250)         MOV(A1),LGS(A1)\NOP           A1=X(6,K),X(9,K)
```

```
A96 0874C 8473U000 (00251)         MOV(A3),MUL(M0,M7)\NOP          A3=X(20,K-1),X(7,K)
A97 0874E 43510000 (00252)         MOV(A1),ADD(A2,A3)\NOP          A1=X(9,K),X(21,K-1)
A98 08750 0BF00000 (00253)         MOV(10A,A0)\NOP                 A0=C2
A99 08752 0BF40000 (00254)         MOV(10A,A4)\NOP                 A4=16**4
A9A 08754 442C0000 (00255)         MOV(M4),ADD(A1,A4)\NOP          M4=X21,K-1),CORRECT X(9,
                   (00256) ;                                       K)
A9B 08756 85920000 (00257)         MOV(A2),MUL(M3,M4)\NOP          A2=X(7,K),X(22,K-1)
A9C 08758 4C310000 (00258)         MOV(A1),SUB(A1,A4)\NOP          CORRECT X(9,K)
A9D 0875A 85330000 (00259)         MOV(A3),MUL(M2,M5)\NOP          A3=X(22,K-1),X10,K)
A9E 0875C 43110000 (00260)         MOV(A1),ADD(A0,A3)\NOP          A1=X(9,K),X(23,K-1)
A9F 0875E 0BF40000 (00261)         MOV(10A,A4)\NOP                 A4=C7
AA0 08760 0B8C0000 (00262)         MOV(R,M4)\NOP                   M4=X(23,K-1)
AA1 08762 85900000 (00263)         MOV(A0),MUL(M3,M4)\NOP          A0=X(10,K),X(24,K-1)
AA2 08764 44400000 (00264)         ADD(A2,A4)\NOP                  X(11,K)
AA3 08766 4F0D0000 (00265)         MOV(M5),SUB(A0,A7)\NOP          M5=X(11,K),X(12,K)
AA4 08768 0BF20000 (00266)         MOV(10A,A2)\NOP                 A2=C1
AA5 0876A 84330000 (00267)         MOV(A3),MUL(M0,M5)\NOP          A3=X(24,K-1),X(13,K)
AA6 0876C 43500000 (00268)         MOV(A0),ADD(A2,A3)\NOP          A0=X(12,K),X(25,K-1)
AA7 0876E 0BF40000 (00269)         MOV(10A,A4)\NOP                 A4=C6
AA8 08770 0B8C0000 (00270)         MOV(R,M4)\NOP                   M4=X(25,K-1)
AA9 08772 85920000 (00271)         MOV(A2),MUL(M3,M4)\NOP          A2=X(13,K),X(26,K-1)
AAA 08774 44400000 (00272)         ADD(A2,A4)\NOP                  X(15,K)
AAB 08776 02AD0000 (00273)         MOV(M5),R(A5)\NOP               M5=X(15,K)
AAC 08778 0BF20000 (00274)         MOV(10A,A2)\NOP                 A2=C0
AAD 0877A 84330000 (00275)         MOV(A3),MUL(M0,M5)\NOP          A3=X(26,K-1),X(16,K)
AAE 0877C 434B0000 (00276)         MOV(M3),ADD(A2,A3)\NOP          M3=X(2,K),X(27,K-1)
AAF 0877E 0BF40000 (00277)         MOV(10A,A4)\NOP                 A4=C5
AB0 08780 46850000 (00278)         MOV(A5),ADD(A5,A6)\NOP          A5=X(27,K-1),X(28,K-1)
AB1 08782 49090000 (00279)         MOV(M1),SUB(A0,A1)\NOP          M1=X(28,K-1),X(14,K)
AB2 08784 84030000 (00280)         MOV(A3),MUL(M1,M6)\NOP          A3=X(16,K),X(29,K-1)
AB3 08786 0BF50000 (00281)         MOV(10A,A5)\NOP                 A5=C4
AB4 08788 44760000 (00282)         MOV(A6),ADD(A3,A4)\NOP          A6=X(14,K),X(17,K)
AB5 0878A 0BD20000 (00283)         MOV(10,A2)\NOP                  A2=X(0,K+1)
AB6 0878C 0BFD0000 (00284)         MOV(10A,M5)\NOP                 M5=X(0,K+1)
AB7 0878E 264C0000 (00285)         MOV(M4),RCB(A2)\NOP             M4=X(17,K),X(1,K+1)
AB8 08790 841C0000 (00286)         MOV(00),MUL(M0,M4)\NOP          O0=X(29,K-1),X(18,K)
AB9 08792 901600BB (00287)         JUMPC(#1,FWI)                   FOR I=1,2
ABA 08794 20372037 (00288)         CLEAR(WI)                       RELEASE APS INPUT
                   (00289) *
                   (00290) *  DCBIAS'=DCBIAS*VAR';VAR'''=20.0*VAR''
ABB 08796 0BF80000 (00291) SNRMS   MOV(10A,M0)\NOP                 SA=DCBIAS
ABC 08798 0BFF0000 (00292)         MOV(10A,M7)\NOP                 SB=1.0/VAR
ABD 0879A 00000BF8 (00293)         NOP\MOV(10A,M0)                 SA'=20.0
ABE 0879C 00000BFF (00294)         NOP\MOV(10A,M7)                 SB'=VAR''
```

```
ABF 0879F 84608460 (00295)         MUL(M0,M7)              SA*SB;SA'*SB'
AC0 087A0 08BC08BC (00296)         MOV(P,00)               SC=SA*SB=DC/VAR;SC'=SA'*
                   (00297) ;                               SB'=20.0*ALOG10(VAR)
AC1 087A2 20372037 (00298)         CLEAR(WI)               RELEASE APS INPUT
                   (00299) *
AC2 087A4 081C0000 (00300) DNESA   MOV(ZERO,00)\NOP        CLEAR INFULL
AC3 087A6 20320000 (00301)         CLEAR(RA)\NOP           APU DONE!
AC4 087A8 00000000 (00302)         NOP
AC5 087AA 10000000 (00303)         JUMP(0)
AC6 087AC 20462046 (00304)         SET(G2)                 >>>>> FOR TIMING PURPOSE
                   (00305) ;                               S ONLY!
          000000C7 (00306)         FDV$SZ=#A-FDV$SA
    087AF          (00307)         END FDV$SZ
                   (00308)         EJECT
```

```
                            *
                            *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            *
                                    EVEN
     087AE 00008890                 ADDR G200SI                      POINTER TO CONSTRUCTED I
                            ;                                        NSTRUCTION BLOCK
     087B0 000087BA                 ADDR G200S+2*IFLTS               POINTER TO SCALAR BLOCK
     087B2 0000                     DATA 0                           NUMBER OF SCALARS
     087B3 00F6                     DATA G200SZ                      MODULE SIZE
     087B4 00008866                 ADDR G200SA                      POINTER TO CHAIN ANCHOR
                                    EVEN                             END ON WORD BOUNDARY
                            *
                            G200S   BEGIN APS(G200)
 A00 087B6 00204240                 JSN(G200S0,P2)                   SET OUTPUT PC
 A01 087B8 02300030                 SET(RO)                          ENABLE APS OUTPUT
 A02 087BA 04C203D0         IFLTS   LOAD(BR0,SAS39(1),TF)            SCALAR SA=2**15
 A03 087BC 06C203FE                 LOAD(BR0,SAS54(1),TF)            IQ=1.0/LTH
 A04 087BE 08D00000                 LOAD(BR1,(0),TF)                 MAPX(.) BA
 A05 087C0 0A700000                 LOAD(BR3,MSS)                    MAPX(.) SZ-1
 A06 087C2 0C600000                 LOAD(BR2,MSS)                    DUMMY
 A07 087C4 0E401006                 LOAD(BR0,(1))                    NIN(.) BA
 A08 087C6 10600000                 LOAD(BR2,MSS)                    NIN(.) SIZE-1
 A09 087C8 12020000                 SUB(BR0,MSS)                     NIN(.) BA-1
 A0A 087CA 14390025                 SUBB(BR3,BR2)                    MAPX SZ - NIN SZ = NUMBE
                            ;                                        R OF UPDATE SAMPLES
 A0B 087CC 16390032                 SUBL(BR3,2)                      ALREADY XFERRED ONE
 A0C 087CE 189A0002         IUPS    ADD(BR1,2,TF)
 A0D 087D0 1A390CB1                 SUBL(BR3,1),JUMPP(IUPS)
 A0E 087D2 1C300037                 SET(WI)
 A0F 087D4 1E000020                 NOP(0)
 A10 087D6 210A0001         IFLTSI  ADD(BR0,HS,TF)                   IQ=NIN(I)
 A11 087D8 22291081                 SUBL(BR2,1),JUMPP(IFLTSI)        FOR I=0,1,2...NTUPS-1
 A12 087DA 24300037                 SET(WI)                          STALL APS INPUT
 A13 087DC 26000020                 NOP
                            *
 A14 087DE 2840001A         ISUMSS  LOAD(BR0,(0))                    MAPX(.) BA
 A15 087E0 2A500000                 LOAD(BR1,MSS)                    MAPX(.) SIZE-1
 A16 087E2 2C020000                 SUB(BR0,MSS)                     MAPX(.) BA-2
 A17 087E4 2E290011                 MOVB(BR2,BR0)                    SAVE MAPX(.) BA-2
 A18 087E6 30390013                 MOVB(BR3,BR1)                    SAVE MAPX(.) SIZE-1
 A19 087E8 32BA0002         ISUMSI  ADD(BR0,WS,TF)                   IQ=MAPX(I)
 A1A 087EA 34191981                 SUBL(BR1,1),JUMPP(ISUMSI)        FOR I=0,1,2...LTH-1
 A1B 087EC 36300037                 SET(WI)                          STALL APS INPUT
 A1C 087EE 38000020                 NOP(0)
```

```
                        (00353) *
                        (00354) *
A1D 087F0 3AC28618 (00355) ISORS1  LOAD(BRO,ISORS1(1),TF)        IQ=CONST=0.5
A1E 087F2 3CHA0002 (00356)         ADD(BRO,WS,TF)                IQ=CONST=16**7
A1F 087F4 3EHA0002 (00357)         ADD(BRO,WS,TF)                IQ=CONST=2.0
A20 087F6 40C203F0 (00358)         LOAD(BRO,SAS55(1),TF)         IQ=VAR**2
A21 087F8 424A03F0 (00359)         LOAD(BRO,SAS55(1),S)
A22 087FA 458A0001 (00360)         ADD(BRO,HS,TO)                IQ=EXPONENT
A23 087FC 46300037 (00361)         SET(WI)                       STALL APS INPUT
                        (00362) *
A24 087FE 48090015 (00363)         MOVB(BRO,BR2)                 MAPX(.) BA-2
A25 08800 4A190017 (00364)         MOVB(BR1,BR3)                 MAPX(.) SIZE-1
A26 08802 4CHA0002 (00365) IVDVS1  ADD(BRO,WS,TF)                IQ=MAPX(I)
A27 08804 4E1926B1 (00366)         SUBL(BR1,1),JUMPP(IVDVS1)     FOR I=0,1,2...LTH-1
A28 08806 50300037 (00367)         SET(WI)                       STALL APS INPUT
A29 08808 52000020 (00368)         NOP
                        (00369) *
A2A 0880A 54C20440 (00370) ILOGSS  LOAD(BRO,SAS95(1),L,TF)       IQ=ALOG16(10.0)
A2B 0880C 56C28600 (00371)         LOAD(BRO,ILOGS1(1),L,TF)      CONST=128.
A2C 0880E 58BA0002 (00372)         ADD(BRO,WS,TF)                CONSTANT=64.
A2D 08810 5ABA0002 (00373)         ADD(BRO,WS,TF)                CONSTANT=C8
A2E 08812 5C600001 (00374)         LOAD(BR2,HS)                  2 INPUT LOOPS FOR APU(BA
                        (00375) ;                                D!!)
A2F 08814 5E0203F0 (00376) #2      LOAD(BR1,SAS55(1),TF)         VAR WHOLE WORD
A30 08816 615A03F0 (00377)         LOAD(BR1,SAS55(1),S,TF)       VAR H8
A31 08818 625A03F0 (00378)         LOAD(BR1,SAS55(1),S)          VAR E8
A32 0881A 659A0001 (00379)         ADD(BR1,HS,TO)
A33 0881C 66C28606 (00380)         LOAD(BRO,ILOGS2(1),TF)        CONSTANT=C3
A34 0881E 68700007 (00381)         LOAD(BR3,7)                   8 REMAINING CONSTANTS
A35 08820 6ABA0002 (00382) #1      ADD(BRO,WS,TF)                C2,16**4,C7,C1,C6,C0,C5,
                        (00383) ;                                C4
A36 08822 6C3935B1 (00384)         SUBL(BR3,HS),JUMPP(#1)        FOR CONSTNT=1,2...8
A37 08824 6E292EB1 (00385)         SUBL(BR2,HS),JUMPP(#2)        FOR LOOP=1,2
A38 08826 70C20794 (00386)         LOAD(BRO,DMYS(1),TF)          LAST INPUT THROWN AWAY
A39 08828 72300037 (00387)         SET(WI)                       STALL APS INPUT
A3A 0882A 74000020 (00388)         NOP
                        (00389) *
A3B 0882C 76C203EA (00390) INRMSS  LOAD(BRO,SAS52(1),TF)         IQ=DC
A3C 0882E 78C2043C (00391)         LOAD(BRO,SAS93(1),TF)         IQ=1.0/VAR
A3D 08830 7AC2043E (00392)         LOAD(BRO,SAS94(1),TF)         IQ=20.0
A3E 08832 7CC203F0 (00393)         LOAD(BRO,SAS55(1),TF)         IQ=ALOG10(VAR)
A3F 08834 7E300037 (00394)         SET(WI)                       STALL APS INPUT
                        (00395) *
                        (00396) *
```

```
A40 0HH36 H0200031 (00397) DNES1   CLEAR(R1)                       APS INPUT DONE!
A41 0HH3H H2000020 (00398)         NOP
                   (00399) *
A42 0HH3A H4300032 (00400) G200S0  SET(RA)                         ENABLE APU
A43 0HH3C H742057F (00401)         LOAD(BW0,ISAS125(1),TF)         00=CALLIN CARD=0
A44 0HH3E HH701060 (00402) OFLTS   LOAD(BW3,[1])                   DUMMY OP FOR EXEC
A45 0HH40 HA500000 (00403)         LOAD(BW1,MSS)                   NIN(,) SIZE-1
A46 0HH42 HC320000 (00404)         SUB(BW3,MSS)                    DUMMY OP FOR EXEC
A47 0HH44 HE400006 (00405)         LOAD(BW0,[0])                   MAPX(,) BA
A48 0HH46 90700000 (00406)         LOAD(BW3,MSS)                   MAPX(,) SIZE-1
A49 0HH48 92020000 (00407)         SUB(BW0,MSS)                    MAPX(,) BA-2
A4A 0HH4A 94210010 (00408)         MOVB(BW2,BW0)                   SAVE MAPX(,) BA-2
A4B 0HH4C 960A0014 (00409)         ADD(BW0,NP2S)                   MAPX(,) BA-2+2*NP
A4C 0HH4E 98HA0002 (00410) OFLTSJ  ADD(BW0,WS,TF)                  00=MAPX(J)
A4D 0HH50 9AHA0002 (00411)         ADD(BW0,WS,TF)                  00=MAPX(J+1)
A4E 0HH52 9C114CH2 (00412)         SUBL(BW1,2),JUMPP(OFLTSJ)       FOR J=NP,NP+1,....LTH-1
                   (00413) *
A4F 0HH54 9EC203EA (00414) OSUMSS  LOAD(BW0,SAS52(1),TF)           00=DCBIAS
                   (00415) *
A50 0HH56 A0010014 (00416) OSUBSD  MOVB(BW0,BW2)                   SAVE MAPX(,) BA-2
A51 0HH58 A2110016 (00417)         MOVB(BW1,BW3)                   SAVE MAPX(,) SIZE-1
A52 0HH5A A4HA0002 (00418) OSUBSJ  ADD(BW0,WS,TF)                  00=MAPX(J)
A53 0HH5C A6HA0002 (00419)         ADD(BW0,WS,TF)                  00=MAPX(J+1)
A54 0HH5E AH1152H2 (00420)         SUBL(BW1,2),JUMPP(OSUBSJ)       FOR J=0,1,2,...LTH-1
                   (00421) *
A55 0HH60 AAC203F0 (00422) OVSOSS  LOAD(BW0,SAS55(1),TF)           00=VAR**2
                   (00423) *
A56 0HH62 ACC203F0 (00424) OSORSS  LOAD(BW0,SAS55(1),TF)           00=SQRT(SD)=VAR
                   (00425) *
A57 0HH64 AEC2043C (00426) OINVSS  LOAD(BW0,SAS93(1),TF)           00=1.0/VAR
                   (00427) *
A58 0HH66 H0402022 (00428) ODIVSD  LOAD(BW0,[2]                    REFLECTION BASE
A59 0HH68 H2500000 (00429)         LOAD(BW1,MSS)                   REFLN SIZE-1
A5A 0HH6A H4600000 (00430)         LOAD(BW2,MSS)                   DUMMY
A5B 0HH6C H6210010 (00431)         MOVB(BW2,BW0)
A5C 0HH6E HH11002A (00432)         ADDB(BW1,BW1)                   (N-1)*2
A5D 0HH70 HA21002A (00433)         ADDB(BW2,BW1)
A5E 0HH72 HC21002A (00434)         ADDB(BW2,BW1)                   (N-1)*4 + REFLN BASE =XR
                   (00435) ;                                       (N-1)
A5F 0HH74 HE010032 (00436)         SUBL(BW0,2)                     XI(-1)
A60 0HH76 C021003C (00437)         ADDL(BW2,4)                     XI(N)
A61 0HH78 C2HA0002 (00438) OREFLSJ ADD(BW0,2,TF)                   XR(K)
A62 0HH7A C4HA0002 (00439)         ADD(BW0,2,TF)                   XI(K)
A63 0HH7C C6A20002 (00440)         SUB(BW2,2,TF)                   XI(N-K-1)
```

```
A64 088TE CHA20002 (00441)          SUB(RW2,2,TF)                 XR(N-K-1)
A65 08880 CA116184 (00442)          SUBL(RW1,4),JUMPP(DRFLSJ)     FOR K=0,1,...N/2-1.
                   (00443) *
A66 08882 CCC20794 (00444) DLOGSD   LOAD(RW0,DMYS(1),TF)          00=?
A67 08884 CFC203E0 (00445)          LOAD(RW0,SAS55(1),TF)         00=ALOG10(VAR)
                   (00446) *
A68 08886 D0C203FA (00447) DNRMSS   LOAD(RW0,SAS52(1),TF)         00=DC/VAR
A69 08888 D2C203E0 (00448)          LOAD(RW0,SAS55(1),TF)         00=20.0*ALOG10(VAR)
A6A 0888A D5420570 (00449)          LOAD(RW0,ISAS110(1),TF)       CLEAR INFULL
                   (00450) *
                   (00451) *
A6B 0888C D6200030 (00452) DNESD    CLEAR(R0)                     APS OUTPUT DONE!
A6C 0888E D8000020 (00453)          NOP
                   (00454) *
         00008866 (00455)           G200SA= #C                    ASSIGN VALUE TO CHAIN AN
                   (00456) ;                                      CHOR
    08890          (00457)          END #A-1
                   (00458) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    08890 00000000 (00459) G200SI   DATA 6F'0.0'
     ...
          000000F6 (00460)          G200SZ=#L-G200S
    0889C          (00461)          END
```

```
AFDTS0RG:   008FB (00007) (00031)
CSPUSMUS:   021FC (00008) (00034)
DMYS:       00794 (00009) (00386) (00444)
DNFSA:      000C2 (00300)
DNFSI:      00040 (00397)
DNFSO:      0006B (00452)
FDVS:       08620 (00032) (00081)
FDVSSA:     00000 (00079) (00083) (00306)
FDVSSZ:     000C7 (00080) (00306) (00307)
FLSM1:      00006 (00091) (00094)
FLSM2:      0000F (00098) (00101)
FLTSM:      0000D (00099) (00107)
G200S:      08786 (00033) (00315) (00321) (00460)
G200SA:     08866 (00318) (00455)
G200SI:     08890 (00313) (00459)
G200S0:     00042 (00322) (00400)
G200S7:     000F6 (00317) (00460)
HS:         00001 (00011) (00339) (00360) (00374) (00379) (00384) (00385)
IFLTS:      00002 (00315) (00324)
IFLTSI:     00010 (00339) (00340)
ILOGS1:     08600 (00043) (00371)
ILOGS2:     08606 (00049) (00380)
ILOGSS:     0002A (00370)
INRMSS:     0003B (00390)
ISAS110:    00570 (00016) (00449)
ISAS125:    0057F (00015) (00401)
ISQRS1:     0861B (00068) (00355)
ISQRSI:     0001D (00355)
ISUMS1:     00019 (00349) (00350)
ISUMSS:     00014 (00344)
ISVTS:      00502 (00014) (00015) (00016)
IUPS:       0000C (00335) (00336)
IVDVS1:     00026 (00365) (00366)
MSS:        00000 (00012) (00327) (00328) (00330) (00331) (00345) (00346) (00403) (00404) (00406)
                  (00407) (00429) (00430)
NP2S:       00014 (00013) (00409)
ODIVSD:     0005B (00428)
OFLTS:      00044 (00402)
OFLTSJ:     0004C (00410) (00412)
OINVSS:     00057 (00426)
OLOGSD:     00066 (00444)
ONRMSS:     0006B (00447)
ORFLSJ:     00061 (00438) (00442)
OSORSS:     00056 (00424)
```

```
PAGE    14:       FCB 240... "MPEDVM(Y,U,V)"      FLOAT,DCBIAS,VAR,REFLECT INPUT VIA MAKHOUL ALG.

  DSUBSD:     00050 (00416)
  DSUBSJ:     00052 (00418) (00420)
  DSUMSS:     0004F (00414)
  DVSUSS:     00055 (00422)
  SAS39:      003D0 (00018) (00324)
  SAS52:      003FA (00019) (00390) (00414) (00447)
  SAS54:      003FF (00020) (00325)
  SAS55:      003F0 (00021) (00358) (00359) (00376) (00377) (00378) (00393) (00422) (00424) (00445)
                    (00448)
  SAS93:      0043C (00022) (00391) (00426)
  SAS94:      0043F (00023) (00392)
  SAS95:      00440 (00024) (00370)
  SINVS:      00068 (00200)
  SLDGS:      00084 (00233)
  SNRMS:      00088 (00291)
  SO1:        00024 (00126) (00135)
  SO2:        00025 (00125) (00127)
  SSORTS:     00039 (00151)
  START:      08600 (00025) (00039)
  SVTS:       003B2 (00017) (00018) (00019) (00020) (00021) (00022) (00023) (00024)
  VMULS:      00078 (00222)
  VSOS:       00021 (00123)
  WS:         00002 (00026) (00349) (00356) (00357) (00365) (00372) (00373) (00382) (00410) (00411)
                    (00418) (00419)
  WWS:        00004 (00027)
  ZS:         00003 (00028)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E-    0
```

```
                      * FCB 241... "MPDCTM(Y,U,V,W)"    FORM TRUE DCT AND ITS REFLECTED MAG SQUARED, VIA
                      *                                 ORIGINATED:06-OCT-79
                      *                                 UPDATED:29-FEB-80
                      * DEFINE GLOBAL SYMBOLS
                              OPADD EXP,(1 .LS. 14)+(12 .LS. 8)+$1E
         000008E8             AFDTSORG=$8E8
         000021FC             CSPU$NOS=$21FC
         00000794             DMYS=$794
         00000003             #M=3
         00000001             HS=1
         00000000             MSS=0
         00000014             NP2S=2*D'10'
         00000382             SVTS=$0382
         00008900             START=$8900
         00000002             WS=2
         00000004             WWS=4
         00000003             ZS=3
                      *
                      * EXPAND ARRAY FUNCTION DISPATCH TABLE
         00000B8E             #L=AFDTSORG+3*2*(241-128)              FCB 241
   00B8E 001F8902             ADDR DCTMS(R7,1)
   00B90 001F8964             ADDR DCTMSS(R7,1)
   00B92 001021FC             ADDR CSPU$NOS(,1,0)
                      *
         00008900             #L=START
                              EVEN
                              EJECT
```

```
                      *  MPDCTM(DCT,MDCT,V,MCOS)
                      *  WHERE DCT IS THE DCT OUTPUT, MDCT IS THE MAG SQUARED OF THE
                      *  OUTPUT, REFLECTED TO BE SYMMETRIC, V IS THE COMPLEX INPUT, AND
                      *  VCOS IS A COSINE TABLE FROM 0 TO PI/2, OF THE SAME LENGTH AS DCT.
                      *>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<
                      *JOHN MAKHOUL'S SHORT DCT ALGORITHM
                      *  DCT(K)=2*RE[EXP(-PI*K/2N)*V(K)]    K<=N/2
                      *  DCT(N-K)=-2*IM[... (K) ...]
                      *
                      *  THIS PROGRAM WORKS ONLY FOR N=256!!!<<<<<<<<<<<<<<<<<
                      *  THE APS CODE MUST BE CHANGED FOR OTHER SIZES.
                      *
                                 EVEN
    08900 0000                   DATA DCTMSSA
    08901 0020                   DATA DCTMSSZ
                        DCTMS    BEGIN APU(DCTM)
          00000000               #A=0
                      *
A00 08902 08F80000      DCTMSSA  MOV(IQA,M0)\NOP                          COS(N/2)
A01 08904 08D208D2               MOV(IQ,A2)                               VR(0)
A02 08906 08F308F3               MOV(IQA,A3)                              VR(0)
A03 08908 43404340               ADD(A2,A3)                               2*VR(0)
A04 0890A 089C088B               MOV(R,DQ)\MOV(P,M3)                      D(0)=2VR(0)
A05 0890C 08FD088C               MOV(IQA,M5)\MOV(R,M4)                    VR(N/2)\PREPARE TO SQUAR
                      ;                                                   E
A06 0890E 84208580               MUL(M0,M5)\MUL(M3,M4)                    COS(N/2)*VR(N/2)\D(0)^2
A07 08910 08B20000               MOV(P,A2)\NOP
A08 08912 08B30000               MOV(P,A3)\NOP
A09 08914 43400000               ADD(A2,A3)\NOP                           2*RE[W(N/2)*V(N/2)]
A0A 08916 089C08BC               MOV(R,DQ)\MOV(P,DQ)                      D(N/2)\MR(0)
A0B 08918 088B081C               MOV(R,M3)\MOV(ZERO,DQ)                   D(N/2)\MI(0)
A0C 0891A 088E0000               MOV(R,M6)\NOP                            D(N/2)
A0D 0891C 85C00000               MUL(M3,M6)\NOP                           D(N/2)^2
A0E 0891E 08F808F8               MOV(IQA,M0)                              COS(K)
A0F 08920 08F908F9               MOV(IQA,M1)                              SIN(K)
A10 08922 08FC08FC               MOV(IQA,M4)                              VR(K)
A11 08924 08FD08FD               MOV(IQA,M5)                              VI(K)
A12 08926 08BC081C               MOV(P,DQ)\MOV(ZERO,DQ)                   MR(N/2)\MI(N/2)
A13 08928 081C081C               MOV(ZERO,DQ)                             MR(N)\MI(N)
A14 0892A 08BC081C               MOV(P,DQ)\MOV(ZERO,DQ)                   MR(3N/2)\MI(3N/2)
A15 0892C 84008420      DCTLP    MUL(M0,M4)\MUL(M0,M5)                    COS*VR\COS*VI
A16 0892E 84808490               MOV(A0),MUL(M1,M5)\MOV(A0),MUL(M1,M4)    SIN*VI\SIN*VR
A17 08930 08B108B1               MOV(P,A1)
A18 08932 41004820               ADD(A0,A1)\SUB(A1,A0)                    COS*VR+SIN*VI\SIN*VR-COS
```

```
                         (00072) ;                                           *VI
A19 08934 08F808F8 (00073)         MOV(IOA,M0)
A1A 08936 08F908F9 (00074)         MOV(IOA,M1)                               SIN(K+1)
A1B 08938 08920892 (00075)         MOV(R,A2)
A1C 0893A 43534353 (00076)         MOV(A3),ADD(A2,A3)                        GET FACTOR OF 2 IN THERE
A1D 0893C 08FC08FC (00077)         MOV(IOA,M4)
A1E 0893E 08FD08FD (00078)         MOV(IOA,M5)                               VI(K+1)
A1F 08940 08BA08BA (00079)         MOV(R,M2)                                 PREPARE TO SQUARE
A20 08942 08BE08BE (00080)         MOV(R,M6)
A21 08944 85408540 (00081)         MUL(M2,M6)                                SQUARE REAL IMAG PARTS
A22 08946 089C089C (00082)         MOV(R,OO)                                 D(K)\D(N-K)
A23 08948 08BC081C (00083)         MOV(P,OO)\MOV(ZERO,OO)                    D(K)^2=MR(K)\MI(K)
A24 0894A 000008BC (00084)         NOP\MOV(P,OO)                             MR(N-K)
A25 0894C 081C0000 (00085)         MOV(ZERO,OO)\NOP                          MI(N-K)
A26 0894E 08BC081C (00086)         MOV(P,OO)\MOV(ZERO,OO)                    MR(2N-K)\MI(2N-K)
A27 08950 000008BC (00087)         NOP\MOV(P,OO)                             MR(N+K)
A28 08952 0F1C0000 (00088)         MOV(ZERO,OO)\NOP                          MI(N+K)
A29 08954 901D0015 (00089)         JUMPC(DCTLP,FI)
A2A 08956 20322032 (00090)         CLEAR(RA)
A2B 08958 00000000 (00091)         NOP
A2C 0895A 10000000 (00092)         JUMP(0)
                         (00093) *
          0000002D (00094)         DCTMSSZ=*A-DCTMSSA
    0895C              (00095)         END DCTMSSZ
                         (00096)         EJECT
```

```
                        (00097) *
                        (00098) *>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<
                        (00099) *INPUT STREAM: COS(PI/4),VR(0),VR(N/2)[COS(PI*K/2N),SIN(PI*K/2N)
                        (00100) *          VR(K),VI(K)],FI.  K=[N/2-1,N/2-2,...,1]
                        (00101) *
                        (00102)         EVEN
    0895C 000089E0      (00103)         ADDR DCTMSI
    0895E 00008964      (00104)         ADDR DCTMSS
    08960 0000          (00105)         DATA 0
    08961 0094          (00106)         DATA DCTMSZ
    08962 0000899F      (00107)         ADDR DCTMSA
                        (00108)         EVEN
                        (00109) *
                        (00110) DCTMSS  BEGIN APS(DCTMSS)
A00 08964 00201940      (00111)         JSR(DCTMS0,P2)
A01 08966 02300030      (00112)         SET(R0)
A02 08968 04403000      (00113)         LOAD(BR0,[3])                          COS(0)
A03 0896A 06500000      (00114)         LOAD(BR1,MSS)                          N-1=255
A04 0896C 08600000      (00115)         LOAD(BR2,MSS)                          DUMMY
A05 0896E 0A602006      (00116)         LOAD(BR2,[2])                          VR(0)
A06 08970 0C700000      (00117)         LOAD(BR3,MSS)                          N-1
A07 08972 0E500000      (00118)         LOAD(BR1,MSS)                          DUMMY
A08 08974 1009002F      (00119)         ADDB(BR0,BR3)                          COS(N/2)-1
A09 08976 12890039      (00120)         ADDL(BR0,1,TF)                         COS(N/2)
A0A 08978 14190011      (00121)         MOVB(BR1,BR0)                          SIN(N/2)=COS(N/2)
A0B 0897A 16A90015      (00122)         MOVB(BR2,BR2,TF)                       VR(0)
A0C 0897C 1829002F      (00123)         ADDB(BR2,BR3)                          VR(0)+N-1
A0D 0897E 1A29002F      (00124)         ADDB(BR2,BR3)                          VR+2N-2=VR(N/2)-2
A0E 08980 1CA9003A      (00125)         ADDL(BR2,2,TF)                         VR(N/2)
A0F 08982 1E2A0002      (00126)         ADD(BR2,2)                             VI(N/2)
A10 08984 20390032      (00127)         SUBL(BR3,2)                            COUNT FIRST BUNCH
                        (00128) *  LOOP FOR K=N/2-1 TO 1
A11 08986 22820002      (00129) *1      SUB(BR0,2,TF)                          COS(K)
A12 08988 249A0002      (00130)         ADD(BR1,2,TF)                          SIN(K)
A13 0898A 26A90036      (00131)         SUBL(BR2,6,TF)                         VR(K)
A14 0898C 28A9003A      (00132)         ADDL(BR2,2,TF)                         VI(K)
A15 0898E 2A3911B2      (00133)         SUBL(BR3,2),JUMPP(#1)
A16 08990 2CC20794      (00134)         LOAD(BR0,DMYS(1),TF)                   NEED AT LEAST ONE DUMMY
A17 08992 2E200031      (00135)         CLEAR(RI)
A18 08994 30000020      (00136)         NOP(0)
                        (00137) *
                        (00138) *  OUTPUT PROGRAM
                        (00139) *  OUTPUT STREAM:D(0),D(N/2),MR(0),MI(0),MR(N/2),MI(N/2),MR(N),
                        (00140) *  MI(N),MR(3N/2),MI(3N/2),[D(K),D(N-K),MR(K),MI(K),MR(N-K),
```

```
                    (00141) *  MI(N-K),MR(2N-K),MI(2N-K),MR(N+K),MI(N+K)],EQ.
                    (00142) *        K=[N/2-1,N/2-2,....,1]  N=256
                    (00143) *
A19 08996 32300032 (00144) DCTMSQ  SET(RA)
A1A 08998 3440002A (00145)         LOAD(BW0,[0])                    D(0)
A1B 0899A 36500000 (00146)         LOAD(BW1,M$S)                    N-1
A1C 0899C 38600000 (00147)         LOAD(BW2,M$S)                    DUMMY
A1D 0899E 3A601006 (00148)         LOAD(BW2,[1])                    MR(0)
A1E 089A0 3C700000 (00149)         LOAD(BW3,M$S)                    DUMMY
A1F 089A2 3E700000 (00150)         LOAD(BW3,M$S)                    DUMMY
A20 089A4 40810010 (00151)         MOVB(BW0,BW0,TF)                 D(0)
A21 089A6 42110039 (00152)         ADDL(BW1,1)                      2*N/2
A22 089A8 4481002A (00153)         ADDB(BW0,BW1,TF)                 D(N/2)
A23 089AA 4611002A (00154)         ADDB(BW1,BW1)                    2*N/2*W$ (=2*K)
A24 089AC 48310012 (00155)         MOVB(BW3,BW1)                    2N (=4*N/2)
A25 089AE 4A31002E (00156)         ADDB(BW3,BW3)                    4*2*N/2 (2*K=2*N/2)
A26 089B0 4C010022 (00157)         SUBB(BW0,BW1)                    D(-N/2)
A27 089B2 4E01003A (00158)         ADDL(BW0,2)                      D(-N/2+1)
A28 089B4 50110034 (00159)         SUBL(BW1,4)                      2*(N/2-1)*W$ (=2*K(NEW))
A29 089B6 52A10014 (00160)         MOVB(BW2,BW2,TF)                 MR(0)
A2A 089B8 54AA0002 (00161)         ADD(BW2,2,TF)                    MI(0)
A2B 089BA 56AA01FE (00162)         ADD(BW2,4*128-2,TF)              MR(N/2)
A2C 089BC 58AA0002 (00163)         ADD(BW2,2,TF)                    MI(N/2)
A2D 089BE 5AAA01FE (00164)         ADD(BW2,4*128-2,TF)              MR(N)
A2E 089C0 5CAA0002 (00165)         ADD(BW2,2,TF)                    MI(N)
A2F 089C2 5EAA01FE (00166)         ADD(BW2,4*128-2,TF)              MR(3N/2)
A30 089C4 60AA0002 (00167)         ADD(BW2,2,TF)                    MI(3N/2)
A31 089C6 62320009 (00168)         SUB(BW3,9)                       4*2*(N/2-1)-1 (-1 FOR LO
                    (00169) :                                       OP TEST UPDATE)
A32 089C8 64310039 (00170) #1      ADDL(BW3,1)                      RESTORE FROM LOOP TEST
A33 089CA 6681002A (00171)         ADDB(BW0,BW1,TF)                 D(K)
A34 089CC 680A0200 (00172)         ADD(BW0,2*256)                   D(N+K)
A35 089CE 6A810022 (00173)         SUBB(BW0,BW1,TF)                 D(N-K)
A36 089D0 6C0201FE (00174)         SUB(BW0,2*256-2)                 D(-K+1)
A37 089D2 6EA20406 (00175)         SUB(BW2,4*256+6,TF)              MR(K)
A38 089D4 70AA0002 (00176)         ADD(BW2,2,TF)                    MI(K)
A39 089D6 722A03FE (00177)         ADD(BW2,4*256-2)                 MR(N+K)
A3A 089D8 74A10026 (00178)         SUBB(BW2,BW3,TF)                 MR(N-K)
A3B 089DA 76AA0002 (00179)         ADD(BW2,2,TF)                    MI(N-K)
A3C 089DC 78AA03FE (00180)         ADD(BW2,4*256-2,TF)              MR(2N-K
A3D 089DE 7AAA0002 (00181)         ADD(BW2,2,TF)                    MI(2N-K)
A3E 089E0 7C220402 (00182)         SUB(BW2,4*256+2)                 MR(N-K)
A3F 089E2 7EA1002E (00183)         ADDB(BW2,BW3,TF)                 MR(N+K)
A40 089E4 80AA0002 (00184)         ADD(BW2,2,TF)                    MI(N+K)
```

```
 A41 089E6 82110034 (00185)          SUBL(RW1,4)                        2K GETS 2(K-W$)
 A42 089E8 84320008 (00186)          SUB(RW3,8)                         2K GETS 2(K-C$) (COMPLEX
 A43 089EA 863132B1 (00187)          SUBL(RW3,1),JUMPP(#1)              2K GETS 2K-1 SO TEST FAI
                    (00188) ;                                           LS FOR 0
 A44 089EC 88200030 (00189)          CLEAR(R0)
 A45 089EE 8A000020 (00190)          NOP(0)
                    (00191) *
           0000899E (00192)          DCTMSA=#C
     089F0          (00193)          END #A-1
     089F0 00000000 (00194) DCTMSI   DATA 4F'0,0'
      ...
           00000094 (00195)          DCTMSZ=#L-DCTMSS
     089F8          (00196)          END
```

```
AFDTSORG:   00EF8 (00006) (00020)
CSPUSNOS:   021FC (00007) (00023)
DCTLP:      00015 (00068) (00089)
DCTMS:      08902 (00021) (00043)
DCTMSA:     0899F (00107) (00192)
DCTMSI:     089F0 (00103) (00194)
DCTMSO:     00019 (00111) (00144)
DCTMSS:     08964 (00022) (00104) (00110) (00195)
DCTMSSA:    00000 (00041) (00046) (00094)
DCTMSSZ:    0002D (00042) (00094) (00095)
DCTMSZ:     00094 (00106) (00195)
DMYS:       00794 (00008) (00134)
HS:         00001 (00010)
MSS:        00000 (00011) (00114) (00115) (00117) (00118) (00146) (00147) (00149) (00150)
NP2S:       00014 (00012)
START:      08900 (00014) (00025)
SVTS:       00382 (00013)
WS:         00002 (00015)
WWS:        00004 (00016)
ZS:         00003 (00017)

LINES WITH ERRORS:    0  (MAP VERSION R00101.10)   E-    0
```

```
         (00001) * FCB 242... "MPQDPP(Y,U,V)"      QUANTIZE/DEQUANTIZE SIDEBAND
         (00002) *                                  W/ K->A TRANSFORM
         (00003) *                                  ORIGINATED:08-AUG-79
         (00004) *                                  UPDATED:12-MAY-80
         (00005) * DEFINE GLOBAL SYMBOLS
00000012 (00006)       A1$=D'18'
000008E8 (00007)       AFDTSORG=$8E8
0001FFC0 (00008)       APSSMEM=$1FFC0
000021FC (00009)       CSPU$NOS=$21FC
00000009 (00010)       DCPOS=D'9'
00000003 (00011)       #M=3
00000001 (00012        H$=1
00000000 (0001         M$=0
00000000 (00014)       M$$=0
0000000A (00015)       MPOS=D'10'
0000000C (00016)       PGPOS=D'12'
00002134 (00017)       PDFQ$1=$2134
00002174 (00018)       PDFQ$2=$2174
000021B4 (00019)       PDFQ$3=$21B4
000021D4 (00020)       PDFQ$4=$21D4
000021F4 (00021)       PDFQ$5=$21F4
00002204 (00022)       PDFQ$6=$2204
00002214 (00023)       PDFQ$7=$2214
0000221C (00024)       PDFQ$8=$221C
00002224 (00025)       DC$DEQ=$2224
00002244 (00026)       VAR$DEQ=$2244
00002284 (00027)       PG$DEQ=$2284
0000228C (00028)       M$DEQ=$228C
00000382 (00029)       SVT$=$0382
00000023 (00030)       PARC$=D'35'
00000007 (00031)       PSIZE$=D'8'-D'1'
00002E28 (00032)       QPRM$=D'11816'
00000398 (00033)       SA$11=SVT$+2*D'11'
000003CE (00034)       SA$38=SVT$+2*D'38'
000003D0 (00035)       SA$39=SVT$+2*D'39'
000003EA (00036)       SA$52=SVT$+2*D'52'
000003F0 (00037)       SA$55=SVT$+2*D'55'
000003FA (00038)       SA$60=SVT$+2*D'60'
000003FC (00039)       SA$61=SVT$+2*D'61'
000003FE (00040)       SA$62=SVT$+2*D'62'
00000432 (00041)       SA$88=SVT$+2*D'88'
00000434 (00042)       SA$89=SVT$+2*D'89'
00000436 (00043)       SA$90=SVT$+2*D'90'
00000438 (00044)       SA$91=SVT$+2*D'91'
```

```
         0000043A (00045)         SAS92=SVTS+2*D'92'
         00000452 (00046)         SAS104=SVTS+2*D'104'
         000065F0 (00047)         START=$65F0
         0000000B (00048)         VPOS=D'11'
         00000008 (00049)         VARPOS=D'8'
         00000002 (00050)         WS=2
         00000003 (00051)         ZS=3
                  (00052) *
                  (00053) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
         00000B94 (00054)         *L=AFDTSORG+3*2*(242-128)                     FCB 242
00B94 001E65F2 (00055)         ADDR QUANS(R7,1)
00B96 001E66F2 (00056)         ADDR G105S(R7,1)
00B98 001021FC (00057)         ADDR CSPUSNOS(,1,0)
                  (00058)         EJECT
```

```
                      (00059) *
                      (00060) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      (00061) *
                      (00062)          BUS 2
           00001F40   (00063)          #L=D'8000'
                      (00064)          EVEN
                      (00065) *
                      (00066) *           >>>>> QUANTIZER THRESHOLDS <<<<<
                      (00067) *  PARCOR(1) W/ 5 BITS
01F40 21919ABF (00068) PVECS1  DATA  0.16391E-01,  0.33470E-01,  0.51313E-01,  0.69970E-01
01F42 44BBFB3F
01F44 6916CA3F
01F46 0BE4C6C0
01F48 0E810FC0 (00069)          DATA  0.89676E-01,  0.11219E+00,  0.13700E+00,  0.16288E+00
01F4A 0E5C3DC0
01F4C 11893740
01F4E 14D94040
01F50 184AB640 (00070)          DATA  0.18978E+00,  0.21909E+00,  0.25131E+00,  0.28606E+00
01F52 1C0B2440
01F54 202AFD40
01F56 249D9D40
01F58 297F62C0 (00071)          DATA  0.32420E+00,  0.36803E+00,  0.41824E+00,  0.47689E+00
01F5A 2F189B40
01F5C 35BBF340
01F5E 3D0ABB40
01F60 45B73FC0 (00072)          DATA  0.54319E+00,  0.61470E+00,  0.69616E+00,  0.77133E+00
01F62 4FAF7D40
01F64 591BC540
01F66 628AF0C0
01F68 6BE62B40 (00073)          DATA  0.84345E+00,  0.92970E+00,  0.10190E+01,  0.11218E+01
01F6A 77006BC0
01F6C 0B26F941
01F6E 0BE97241
01F70 09FC2241 (00074)          DATA  0.12403E+01,  0.13426E+01,  0.14437E+01,  0.15414E+01
01F72 0ABDA541
01F74 0B8CB2C1
01F76 0C54C9C1
01F78 0D2786C1 (00075)          DATA  0.16443E+01,  0.17886E+01,  0.19332E+01,  0.10000E+21
01F7A 0E4F0DC1
01F7C 0F7731C1
01F7E 2B5F3AD1
                      (00076) *
                      (00077) *  PARCOR(2) W/ 5 BITS
01F80 166AFCC0 (00078) PVECS2  DATA  0.17514E+00,  0.31624E+00,  0.41542E+00,  0.50895E+00
```

```
01F82 287A8D40
01F84 352C78C0
01F86 41254640
01F88 48885BC0 (00079)          DATA  0.59019E+00,  0.65272E+00,  0.70886E+00,  0.76395E+00
01F8A 538C5440
01F8C 5A88ECC0
01F8E 61C91D40
01F90 68CDC840 (00080)          DATA  0.81878E+00,  0.87625E+00,  0.94152E+00,  0.10123E+01
01F92 7028E5C0
01F94 7883BA40
01F96 081930C1
01F98 08AD7741 (00081)          DATA  0.10847E+01,  0.11620E+01,  0.12381E+01,  0.13059E+01
01F9A 0948CEC1
01F9C 09F7A0C1
01F9E 0A727BC1
01FA0 0AEB4A41 (00082)          DATA  0.13727E+01,  0.14357E+01,  0.14920E+01,  0.15468E+01
01FA2 0B7C5041
01FA4 0BEE9DC1
01FA6 0C5EDBC1
01FA8 0CD04841 (00083)          DATA  0.16017E+01,  0.16527E+01,  0.16951E+01,  0.17323E+01
01FAA 0D38BAC1
01FAC 0D8E9DC1
01FAE 0DDBC041
01FB0 0E257A41 (00084)          DATA  0.17683E+01,  0.18035E+01,  0.18365E+01,  0.18668E+01
01FB2 0E6D9141
01FB4 0EB126C1
01FB6 0EEF34C1
01FB8 0F2B3641 (00085)          DATA  0.18961E+01,  0.19267E+01,  0.19616E+01,  0.10000E+21
01FBA 0F69E1C1
01FBC 0FB15B41
01FBE 2B5F3AD1
               (00086) *
               (00087) *  PARCOR(3) W/ 4 BITS
01FC0 28581840 (00088) PVECS3  DATA  0.31528E+00,  0.45224E+00,  0.56230E+00,  0.65753E+00
01FC2 39F30040
01FC4 47E97240
01FC6 5429F140
01FC8 5F5A31C0 (00089)          DATA  0.74494E+00,  0.82867E+00,  0.91130E+00,  0.99458E+00
01FCA 6A11DBC0
01FCC 74A57A40
01FCE 7E4E65C0
01FD0 08A2D0C1 (00090)          DATA  0.10795E+01,  0.11678E+01,  0.12608E+01,  0.13581E+01
01FD2 0957A7C1
01FD4 0A161E41
```

```
01FD6 0ADD63C1
01FD8 0B81EBC1 (00091)             DATA  0.14619E+01,  0.15854E+01,  0.17491E+01,  0.10000E+21
01FDA 0CAFF641
01FDC 0DFF2841
01FDE 2B5F3AD1
                (00092) *
                (00093) *  PARCOR(4) W/ 4 BITS
01FE0 3462F5C0 (00094) PVFCS4  DATA  0.40927E+00,  0.57217E+00,  0.69580E+00,  0.80120E+00
01FE2 493C0DC0
01FE4 590FF940
01FE6 668DBBC0
01FE8 72CD4AC0 (00095)             DATA  0.89689E+00,  0.98763E+00,  0.10764E+01,  0.11650E+01
01FEA 7F6AABC0
01FEC 089C77C1
01FEE 0951FBC1
01FF0 0A09A041 (00096)             DATA  0.12547E+01,  0.13464E+01,  0.14408E+01,  0.15388E+01
01FF2 0AC56D41
01FF4 0BB6C241
01FF6 0C4F7641
01FF8 0D22D0C1 (00097)             DATA  0.16420E+01,  0.17540E+01,  0.18815E+01,  0.10000E+21
01FFA 0F083141
01FFC 0F0D4FC1
01FFE 2B5F3AD1
                (00098) *
                (00099) *  PARCOR(5) W/ 3 BITS
02000 4747D840 (00100) PVFCS5  DATA  0.55688E+00,  0.74670E+00,  0.89149E+00,  0.10220E+01
02002 5F93DDC0
02004 721C5840
02006 082D0F41
02008 093B8641 (00101)             DATA  0.11526E+01,  0.12967E+01,  0.14823E+01,  0.10000E+21
0200A 0A5FA441
0200C 0BDBC041
0200E 2B5F3AD1
                (00102) *
                (00103) *  PARCOR(6) W/ 3 BITS
02010 49623BC0 (00104) PVFCS6  DATA  0.57331E+00,  0.76317E+00,  0.90902E+00,  0.10407E+01
02012 61AFBDC0
02014 745AC440
02016 0B535AC1
02018 096147C1 (00105)             DATA  0.11725E+01,  0.13190E+01,  0.15093E+01,  0.10000E+21
0201A 0ABD4FC1
0201C 0C1308C1
0201E 2B5E3AD1
                (00106) *
```

```
                  (00107) *  PARCOR(7) W/ 2 BITS
02020 4BA3D740 (00108) PVECS7  DATA  0.56750E+00,  0.87517E+00,  0.11666E+01,  0.10000E+21
02022 300591C0
02024 09553241
02026 2B5F3AD1
                  (00109) *
                  (00110) *  PARCOR(8) W/ 2 BITS
02028 5C6A2B40 (00111) PVECS8  DATA  0.72199E+00,  0.98497E+00,  0.12430E+01,  0.10000E+21
0202A 7E137E40
0202C 09F1A9C1
0202E 2B5F3AD1
                  (00112) *
                  (00113) *  DC BIAS W/ 4 BITS
02030 0FF7CFC0 (00114) DCS     DATA  0.12475E+00,0.25019E+00,0.37706E+00,0.50625E+00
02032 200639C0
02034 30438DC0
02036 40CCCCC0
02038 51C486C0 (00115)         DATA  0.63881E+00,0.77594E+00,0.91903E+00,0.10699E+01
0203A 63520040
0203C 75A2C640
0203E 088E27C1
02040 09D8F241 (00116)         DATA  0.12309E+01,0.14053E+01,0.15977E+01,  0.18155E+01
02042 0B3F0DC1
02044 0CC816C1
02046 0E8624C1
02048 109308C1 (00117)         DATA  0.20718E+01,  0.23938E+01,  0.28541E+01,  0.10000E+21
0204A 13268DC1
0204C 16D53241
0204E 2B5F3AD1
                  (00118) *
                  (00119) *  VARIANCE W/ 5 BITS
02050 2D0CB2C1 (00120) VARS    DATA  0.56312E+01,  0.77983E+01,  0.98311E+01,  0.12057E+02
02052 3E62EB41
02054 4EA617C1
02056 6074BC41
02058 74624DC1 (00121)         DATA  0.14548E+02,  0.16994E+02,  0.19276E+02,  0.21426E+02
0205A 087F3B42
0205C 09A353C2
0205E 0AB68742
02060 0BB3D742 (00122)         DATA  0.23405E+02,  0.25415E+02,  0.27476E+02,  0.29409E+02
02062 0CB51FC2
02064 0DBCFDC2
02066 0EB45A42
02068 0F96F942 (00123)         DATA  0.31179E+02,  0.32888E+02,  0.34633E+02,  0.36477E+02
```

```
0206A 1071A9C2
0206C 11510642
0206E 123D0F42
02070 13316842 (00124)          DATA  0.38386E+02,  0.40204E+02,  0.41864E+02,  0.43443E+02
02072 141A1CC2
02074 14FF97C2
02076 15B8B442
02078 167DB242 (00125)          DATA  0.44982E+02,  0.46483E+02,  0.47960E+02,  0.49431E+02
0207A 173DD2C2
0207C 17FAF142
0207E 18B72B42
02080 19733342 (00126)          DATA  0.50900E+02,  0.52334E+02,  0.53709E+02,  0.55072E+02
02082 1A2AC0C2
02084 1ADAC0C2
02086 1B893742
02088 1C3ED942 (00127)          DATA  0.56491E+02,  0.58016E+02,  0.59760E+02,  0.10000E+21
0208A 1D020C42
0208C 1DF147C2
0208E 2B5E3AD1
               (00128) *
               (00129) *  PITCH GAIN W/ 2 BITS
02090 3F75CD40 (00130) PGS      DATA  0.48797E+00,  0.66711E+00,  0.82399E+00,  0.10000E+21
02092 55630C40
02094 69788140
02096 2B5E3AD1
               (00131) *
               (00132) *  PITCH W/ 6 BITS
02098 0000     (00133) PITCHS  DATA 16D'0'
 ...
020A8 0001     (00134)          DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
020A9 0002
020AA 0003
020AB 0004
020AC 0005
020AD 0006
020AE 0007
020AF 0008
020B0 0009
020B1 000A
020B2 000B
020B3 000C
020B4 000D
020B5 000E
020B6 000F
```

```
020B7 0010
020B8 0011
020B9 0012
020BA 0013
020BB 0014     (00135)     DATA 20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35
020BC 0015
020BD 0016
020BE 0017
020BF 0018
020C0 0019
020C1 001A
020C2 001B
020C3 001C
020C4 001D
020C5 001E
020C6 001F
020C7 0020
020C8 0021
020C9 0022
020CA 0023
020CB 0024     (00136)     DATA 36,37,38,39,40,41,42,43,44,45,46,47
020CC 0025
020CD 0026
020CE 0027
020CF 0028
020D0 0029
020D1 002A
020D2 002B
020D3 002C
020D4 002D
020D5 002E
020D6 002F
020D7 0030     (00137)     DATA 48,48,49,49,50,50,51,51,52,52,53,53,54,54,55,55
020D8 0030
020D9 0031
020DA 0031
020DB 0032
020DC 0032
020DD 0033
020DE 0033
020DF 0034
020E0 0034
020E1 0035
020E2 0035
```

```
020E3 0036
020E4 0036
020E5 0037
020E6 0037
020E7 0038      (00138)         DATA 56,56,57,57,58,58,59,59,60,60,61,61,62,62,63,63
020E8 0038
020E9 0039
020EA 0039
020EB 003A
020EC 003A
020ED 003B
020EE 003B
020EF 003C
020F0 003C
020F1 003D
020F2 003D
020F3 003E
020F4 003E
020F5 003F
020F6 003F
020F7 0800      (00139)         EVEN
020F8 8450      (00140) INSTPS0 DATA INS0*D'512'+X'50',X'0000'
020F9 0000
020FA 9250      (00141) INSTPS1 DATA INS1*D'512'+X'50',X'0000'
020FB 0000
                (00142) *          >>>>> DEQUANTIZER THRESHOLDS...SEE MPDPPS.TXT <<<<<
                (00143)         BUS 1
      000065F0  (00144)         #L=START
                (00145)         EJECT
```

```
                          (00146) *
                          (00147) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00148) *
                          (00149)          EVEN
    065F0 0000            (00150)          DATA QUANSSA                          START ADDRESS
    065F1 0084            (00151)          DATA QUANSSZ                          MODULE SIZE
                          (00152) QUANS    BEGIN APU(QUAN)
                 00000000 (00153)          #A=0
A00 065F2 08FF08FF (00154) QUANSSA MOV(IOA,M7)                            M7=2**(-15)=SA
A01 065F4 16A016A0 (00155)          K(2,0)                                2.0
A02 065F6 16921692 (00156)          MOV(A2),K(1,0)                        A2=LE,LO INCRE: BIAS
A03 065F8 08910891 (00157)          MOV(R,A1)                             A1=1.0
A04 065FA 08980000 (00158)          MOV(R,EXO)\NOP                        EXO=1.0
A05 065FC 08F008F0 (00159) INCSJ    MOV(IOA,A0)                           A0=X(J)
A06 065FE 90090008 (00160)          JUMPC(BIASS,AF1)                      AF1=0 THEN ADD BIAS
A07 065F0 08110811 (00161)          MOV(ZERO,A1)                          END OF PARCORS,BIAS=0!
A08 065F2 41004100 (00162) BIASS    ADD(A0,A1)                            X(J) + BIAS
A09 065F4 08130853 (00163)          MOV(ZERO,A3)\MOV(EXI,A3)              LE"=0;LO"=1
A0A 065F6 02700270 (00164)          MOV(A0),R(A3)                         A0=X(J)+BIAS,LE";LO"
A0B 065F8 08F40000 (00165) LOOPS    MOV(IOA,A4)\NOP                       A4=D(K)
A0C 065FA 000008F4 (00166)          NOP\MOV(IOA,A4)                       A4=D(K+1)
A0D 065FC 08880888 (00167)          MOV(R,M3)                             LE'=LE"+2;LO'=LO"+2
A0E 065FE 4C134C13 (00168)          MOV(A3),SUB(A0,A4)                    LE',T(K);A3=LO',T(K+1)
A0F 06600 85E085E0 (00169)          MUL(M3,M7)                            LE'*SA;LO'*SA
A10 06602 000008F6 (00170)          NOP\MOV(IOA,A6)                       A6=V(K)
A11 06604 08F60000 (00171)          MOV(IOA,A6)\NOP                       A6=V(K+1)
A12 06606 911F0017 (00172)          JUMPS(T1S,T1)                         IF T1=1 THEN K
A13 06608 911F001C (00173)          JUMPS(T2S,T2)                         IF T2=1 THEN K+1
A14 0660A 20370000 (00174)          CLEAR(W1)\NOP                         RELEASE APS INPUT
A15 0660C 42604260 (00175)          ADD(A3,A2)                            LE'+2;LO'+2
A16 0660E 10000008 (00176)          JUMP(LOOPS)                           TRY D(K+2) & D(K+3)
A17 06610 08850000 (00177) T1S      MOV(P,A5)\NOP                         A5=LE'*SA
A18 06612 3AA049C0 (00178)          ALIGN(A5)\SUB(A6,A1)                  FORCE EXP->0;V(K)-BIAS
A19 06614 204A0000 (00179)          SET(AF2)\NOP                          SHUTOFF APS LOOP
A1A 06616 089C089C (00180)          MOV(R,OO)                             OPRM(J);DEQ(OPRM(J))
A1B 06618 10370022 (00181)          JUMP(AF3STST,W1),CLEAR                AF3=?,RELEASE APS INPUT
A1C 0661A 00000885 (00182) T2S      NOP\MOV(P,A5)                         A5=LO'*SA
A1D 0661C 49C03AA0 (00183)          SUB(A6,A1)\ALIGN(A5)                  V(K+1)-BIAS;FORCE EXP->0
A1E 0661E 204A0000 (00184)          SET(AF2)\NOP                          SHUTOFF APS LOOP
A1F 06620 0000089C (00185)          NOP\MOV(R,OO)                         OO=QPRM(J)
A20 06622 089C0000 (00186)          MOV(R,OO)\NOP                         OO=DEQ(QPRM(J))
A21 06624 20370000 (00187)          CLEAR(W1)\NOP                         RELEASE APS INPUT
A22 06626 90080005 (00188) AF3STST  JUMPC(INCSJ,AF3)                      FOR PARCOR,DC,VAR & PG
A23 06628 000008F8 (00189) MQUANS   NOP\MOV(IOA,M0)                       GET PITCH
```

```
A24 0662A 00008460 (00190)              NOP\MUL(M0,M7)                  SA*PITCH
A25 0662C 000008B5 (00191)              NOP\MOV(P,A5)                   NOW "FIX" PITCH
A26 0662E 00003AA0 (00192)              NOP\ALIGN(A5)                   FORCE EXP->0
A27 06630 0000089C (00193)              NOP\MOV(R,OQ)                   M->INSTR$0+1
A28 06632 901C0028 (00194) #1           JUMPC(#1,F0)                    WAIT FOR OUTPUT TO CLEAR
A29 06634 20370000 (00195)              CLEAR(WI)\NOP                   RELEASE APS INPUT
A2A 06636 20500000 (00196)              SET(RO)\NOP                     RELEASE APS OUTPUT
A2B 06638 08FC0000 (00197)              MOV(IQA,OQ)\NOP                 OVERWRITE INS0
A2C 0663A 901C002C (00198) #2           JUMPC(#2,F0)                    WAIT FOR OUTPUT TO CLEAR
A2D 0663C 20370000 (00199)              CLEAR(WI)\NOP                   RELEASE APS INPUT
A2E 0663E 20500000 (00200)              SET(RO)\NOP                     RELEASE APS OUTPUT
A2F 06640 08DC0000 (00201)              MOV(IQ,OQ)\NOP                  OQ=Q(M)=QTM
A30 06642 000008FC (00202)              NOP\MOV(IQA,OQ)                 QTM->INSTR$1+1
A31 06644 901C0031 (00203) #3           JUMPC(#3,F0)                    WAIT FOR OUTPUT TO CLEAR
A32 06646 20370000 (00204)              CLEAR(WI)\NOP                   RELEASE APS INPUT
A33 06648 20500000 (00205)              SET(RO)\NOP                     RELEASE APS OUTPUT
A34 0664A 08FC0000 (00206)              MOV(IQA,OQ)\NOP                 OVERWRITE INS1
A35 0664C 901C0035 (00207) #4           JUMPC(#4,F0)                    WAIT FOR OUTPUT TO CLEAR
A36 0664E 20370000 (00208)              CLEAR(WI)\NOP                   RELEASE APS INPUT
A37 06650 20500000 (00209)              SET(RO)\NOP                     RELEASE APS OUTPUT
A38 06652 08FC0000 (00210)              MOV(IQA,OQ)\NOP                 OQ=D(Q(M))=DTM
                   (00211) *
                   (00212) *  K->A TRANSFORM W/ ENG=SUM(1-K(I))**2
A39 06654 08F10000 (00213) KTOAS        MOV(IQA,A1)\NOP                 P(1)
A3A 06656 0A200000 (00214)              NEG(A1)\NOP                     MAKE IT K(1)
A3B 06658 08880000 (00215)              MOV(R,M0)\NOP
A3C 0665A 088F0000 (00216)              MOV(R,M7)\NOP                   K(1)**2 FOR ENERGY
A3D 0665C 08910000 (00217)              MOV(R,A1)\NOP                   SAVE K(1)
A3E 0665E 84600000 (00218)              MUL(M0,M7)\NOP                  SO SQUARE IT
A3F 06660 16801680 (00219)              K(+1)                           NEED THIS AROUND, TOO
A40 06662 08D008D0 (00220)              MOV(IQ,A0)                      P(2)
A41 06664 08950000 (00221)              MOV(R,A5)\NOP                   1.0
A42 06666 08B608B6 (00222)              MOV(P,A6)                       K(1)^2
A43 06668 4FA04FA0 (00223)              SUB(A5,A6)                      1.0-K(1)^2
A44 0666A 020D020D (00224)              MOV(M5),R(A0)                   PROD(1),GET K(2)
A45 0666C 088F088F (00225)              MOV(R,M7)                       K(2)
A46 0666E 088B088B (00226)              MOV(R,M3)                       K(2)
A47 06670 3000006F (00227)              CALL(X23RD)                     DO 2ND ITERATION (M=2)
A48 06672 3000006F (00228)              CALL(X23RD)                     DO 3RD ITER (M=3)
A49 06674 30000069 (00229)              CALL(X4TH)
A4A 06676 3000006A (00230)              CALL(X5TH)
A4B 06678 00000260 (00231)              NOP\R(A3)                       GET A(5)(3) READY
A4C 0667A 30000062 (00232)              CALL(X6TH)                      M=6
A4D 0667C 30000064 (00233)              CALL(X7TH)                      M=7
```

```
                                (..)          NOP\R(A4)                         GET A[7](4) READY
                                (..)          CALL(X8TH)
                                (..)          MOV(IQ,M3)\NOP                    LTH2 (=SA)
                                (..)          MUL(M3,M5)\NOP                    PROD(8)*LTH2
                                (..238)       R(A1)\NOP
                                (00239)       MOV(P,DQ)\NOP                     OUTPUT ENERGY = LTH2*PRO
                                (00240) ;                                       D(8)
                                (00241)       MOV(DQ),R(A2)\NOP                 A(1) OUT
                                (00242)       MOV(DQ),R(A3)\NOP                 A(2) OUT
                                (00243)       MOV(DQ),R(A4)\R(A4)               A(3) OUT
                     ...0227C   (00244)       MOV(R,DQ)\MOV(DQ),R(A3)           A(4) OUT,A(5) OUT
                     0000025C   (00245)       NOP\MOV(DQ),R(A2)                 A(6) OUT
                     0000023C   (00246)       NOP\MOV(DQ),R(A1)                 A(7) OUT
            .6696    0000089C   (00247)       NOP\MOV(R,DQ)                     A(8) OUT
                                (00248) *
A58  06698  20420000 (00249) DNESA  CLEAR(RA)\NOP                               APU DONE!
A59  0669A  00000000 (00250)        NOP
                     (00251) *
                     (00252) *  K->A APU SUBROUTINES
A5D  0669C  00000898 (00253) X8TH   NOP\MOV(R,FX0)                              \A[7](4)
A5E  0669E  08540854 (00254)        MOV(FX1,A4)                                 A[7](4)
A5F  066A0  85800000 (00255)        MUL(M3,M4)\NOP                              A[M-1](M-4)*K[M] ((M=8))
A60  066A2  08800880 (00256)        MOV(P,A0)
                     (00257) *
A61  066A4  10000064 (00258)        JUMP(X7TH)
A62  066A6  00000898 (00259) X6TH   NOP\MOV(R,FX0)                              A[5](3)
A63  066A8  08530000 (00260)        MOV(FX1,A3)\NOP                             A[5](3)
A64  066AA  85608560 (00261) X7TH   MUL(M2,M7)\MUL(M2,M7)                       A[M-1](M-3)*K[M]\A[M-1](
                     (00262) ;                                                  3)*K[M]
A65  066AC  44000000 (00263)        ADD(A0,A4)\NOP                              SUM FOR A[M](4)
A66  066AE  08940000 (00264)        MOV(R,A4)\NOP                               STORE A[M](4)
A67  066B0  08800880 (00265)        MOV(P,A0)
                     (00266) *
A68  066B2  1000006A (00267)        JUMP(X5TH)
A69  066B4  08520000 (00268) X4TH   MOV(FX1,A2)\NOP                             A[3](2)
A6A  066B6  84F084F0 (00269) X5TH   MUL(M1,M7)\MUL(M1,M7)                       A[M-1](M-2)*K[M]\A[M-1](
                     (00270) ;                                                  2)*K[M]
A6B  066B8  43004300 (00271)        ADD(A0,A3)\ADD(A0,A3)                       SUM FOR A[M](3)\A[M](M-3
                     (00272) ;                                                  )
A6C  066BA  08980898 (00273)        MOV(R,FX0)\MOV(R,FX0)
A6D  066BC  084C084A (00274)        MOV(FX1,M4)\MOV(FX1,M2)                     STORE A[M](M-3)\A[M](3)
A6E  066BE  08800880 (00275)        MOV(P,A0)
                     (00276) *
A6F  066C0  84608460 (00277) X23RD  MUL(M0,M7)\MUL(M0,M7)                       A[M-1](M-1)*K[M]\A[M-1](
```

```
                          (00278) ;                                                        1)*K[M]
A70 066C2 42134214 (00279)        MOV(A3),ADD(A0,A2)\MOV(A4),ADD(A0,A2)        STORE A[M](3),SUM FOR A[
                          (00280) ;                                                        M](2)\A[M](M-3),SUM A[M]
                          (00281) ;                                                        (M-2)
A71 066C4 08980898 (00282)        MOV(R,FX0)\MOV(R,FX0)
A72 066C6 084A0849 (00283)        MOV(FX1,M2)\MOV(FX1,M1)                      ATORE A[M](M-2)\A[M](2)
A73 066C8 08B008B0 (00284)        MOV(P,A0)\MOV(P,A0)
A74 066CA 41124113 (00285)        MOV(A2),ADD(A0,A1)\MOV(A3),ADD(A0,A1)        STORE A[M](2),SUM FOR A[
                          (00286) ;                                                        M](1)\A[M](M-2),SUM A[M]
                          (00287) ;                                                        (M-1)
                          (00288) *
A75 066CC 85F085F0 (00289)        MUL(M3,M7)                                   K[M]^2
A76 066CE 08980898 (00290)        MOV(R,FX0)\MOV(R,FX0)
A77 066D0 08490848 (00291)        MOV(FX1,M1)\MOV(FX1,M0)                      A[M](M-1)\A[M](1)
A78 066D2 08910892 (00292)        MOV(R,A1)\MOV(R,A2)                          A[M](1)\A[M](M-1)
A79 066D4 08F708F7 (00293)        MOV(IOA,A7)                                  -A[M](M) (=P[M])
A7A 066D6 0AF00AF0 (00294)        NEG(A7)                                      MAKE IT REFLN COEF
A7B 066D8 08B608B6 (00295)        MOV(P,A6)                                    K[M]^2
A7C 066DA 4FA80891 (00296)        MOV(M0),SUB(A5,A6)\MOV(R,A1)                 K[M]=A[M](M),1-K[M]^2
A7D 066DC 08D708D7 (00297)        MOV(IO,A7)                                   P[M+1]
A7E 066DE 02F802F8 (00298)        MOV(M3),R(A7)                                1-K[M]^2,GET K[M+1]
A7F 066E0 85A085A0 (00299)        MUL(M3,M5)                                   (1-K[M]^2)*PROD
A80 066E2 08AD08AD (00300)        MOV(P,M5)                                    NEW PROD
A81 066E4 08B808B8 (00301)        MOV(R,M3)                                    K[M+1]
A82 066E6 088F088F (00302)        MOV(R,M7)                                    K[M+1]
A83 066E8 A000A000 (00303)        RETURN
                          (00304) *
          00000084 (00305)        QUANSSZ=*A-QUANSSA
    066EA          (00306)        END QUANSSZ
                   (00307)        EJECT
```

```
                     (00308) *
                     (00309) *>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                     (00310) *
                     (00311)         EVEN
    066EA 000067F2   (00312)         ADDR G105$I                    POINTER TO CONSTRUCTED I
                     (00313) ;                                      NSTRUCTION BLOCK
    066EC 000066F6   (00314)         ADDR G105$+2*SCLR$             POINTER TO SCALAR BLOCK
    066EE 0000       (00315)         DATA 0                         NUMBER OF SCALARS
    066EF 0102       (00316)         DATA G105$Z                    MODULE SIZE
    066F0 00006782   (00317)         ADDR G105$A                    POINTER TO CHAIN ANCHOR
                     (00318)         EVEN                           END ON WORD BOUNDARY
                     (00319) *
                     (00320) G105$   BEGIN APS(G105)
A00 066F2 00105340   (00321)         JSN(G105$I,P1)                 SET INPUT PC(PC1)
A01 066F4 02300030   (00322)         SET(RO)                        ENABLE APS OUTPUT
A02 066F6 04C203CE   (00323) SCLR$   LOAD(BRO,SA$38(1),TF)          IQ=2**(-15)
A03 066F8 06C60023   (00324)         LOAD(BRO,PARC$(3),TF)          IQ=P(1)
A04 066FA 08D41E40   (00325)         LOAD(BR1,PVEC$1(2),TF)         IQ=PTHR(0)
A05 066FC 0A9A0002   (00326)         ADD(BR1,WS,TF)                 IQ=PTHR(1)
A06 066FE 0CE42134   (00327)         LOAD(BR3,PDEQ$1(2),TF)         IQ=PDEQ(0)
A07 06700 0EBE0002   (00328)         ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                     (00329)
A08 06702 10BA0002   (00330)         ADD(BRO,WS,TF)                 IQ=P(2)
A09 06704 12D41E80   (00331)         LOAD(BR1,PVEC$2(2),TF)         IQ=PTHR(0)
A0A 06706 149A0002   (00332)         ADD(BR1,WS,TF)                 IQ=PTHR(1)
A0B 06708 16E42174   (00333)         LOAD(BR3,PDEQ$2(2),TF)         IQ=PDEQ(0)
A0C 0670A 18BE0002   (00334)         ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                     (00335) *
A0D 0670C 1ABA0002   (00336)         ADD(BRO,WS,TF)                 IQ=P(3)
A0E 0670E 1CD41EC0   (00337)         LOAD(BR1,PVEC$3(2),TF)         IQ=PTHR(0)
A0F 06710 1E9A0002   (00338)         ADD(BR1,WS,TF)                 IQ=PTHR(1)
A10 06712 20E421B4   (00339)         LOAD(BR3,PDEQ$3(2),TF)         IQ=PDEQ(0)
A11 06714 22BE0002   (00340)         ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                     (00341) *
A12 06716 24BA0002   (00342)         ADD(BRO,WS,TF)                 IQ=P(4)
A13 06718 26D41EE0   (00343)         LOAD(BR1,PVEC$4(2),TF)         IQ=PTHR(0)
A14 0671A 289A0002   (00344)         ADD(BR1,WS,TF)                 IQ=PTHR(1)
A15 0671C 2AE421D4   (00345)         LOAD(BR3,PDEQ$4(2),TF)         IQ=PDEQ(0)
A16 0671E 2CBE0002   (00346)         ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                     (00347) *
A17 06720 2EBA0002   (00348)         ADD(BRO,WS,TF)                 IQ=P(5)
A18 06722 30D42000   (00349)         LOAD(BR1,PVEC$5(2),TF)         IQ=PTHR(0)
A19 06724 329A0002   (00350)         ADD(BR1,WS,TF)                 IQ=PTHR(1)
A1A 06726 34E421F4   (00351)         LOAD(BR3,PDEQ$5(2),TF)         IQ=PDEQ(0)
```

```
A1B 0672B 36BE0002 (00352)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00353) *
A1C 0672A 38BA0002 (00354)    ADD(BR0,WS,TF)                 IQ=P(6)
A1D 0672C 3AD42010 (00355)    LOAD(BR1,PVECS6(2),TF)         IQ=PTHR(0)
A1E 0672E 3C9A0002 (00356)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A1F 06730 3EF42204 (00357)    LOAD(BR3,PDEQS6(2),TF)         IQ=PDEQ(0)
A20 06732 40BE0002 (00358)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00359) *
A21 06734 42BA0002 (00360)    ADD(BR0,WS,TF)                 IQ=P(7)
A22 06736 44D42020 (00361)    LOAD(BR1,PVECS7(2),TF)         IQ=PTHR(0)
A23 06738 469A0002 (00362)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A24 0673A 48F42214 (00363)    LOAD(BR3,PDEQS7(2),TF)         IQ=PDEQ(0)
A25 0673C 4ABE0002 (00364)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00365) *
A26 0673E 4CBA0002 (00366)    ADD(BR0,WS,TF)                 IQ=P(8)
A27 06740 4ED42028 (00367)    LOAD(BR1,PVECS8(2),TF)         IQ=PTHR(0)
A28 06742 509A0002 (00368)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A29 06744 52F4221C (00369)    LOAD(BR3,PDEQS8(2),TF)         IQ=PDEQ(0)
A2A 06746 54300029 (00370)    SET(AF1)                       SET "END OF PARCOR" BIAS
A2B 06748 56BE0002 (00371)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00372) *
A2C 0674A 58C203EA (00373)    LOAD(BR0,SAS52(1),TF)          IQ=DC BIAS
A2D 0674C 5AD42030 (00374)    LOAD(BR1,DCS(2),TF)            IQ=PTHR(0)
A2E 0674E 5C9A0002 (00375)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A2F 06750 5EF42224 (00376)    LOAD(BR3,DCSDEQ(2),TF)         IQ=PDEQ(0)
A30 06752 60BE0002 (00377)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00378) *
A31 06754 62C203E0 (00379)    LOAD(BR0,SAS55(1),TF)          IQ=VAR
A32 06756 64D42050 (00380)    LOAD(BR1,VARS(2),TF)           IQ=PTHR(0)
A33 06758 669A0002 (00381)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A34 0675A 68F42244 (00382)    LOAD(BR3,VARSDEQ(2),TF)        IQ=PDEQ(0)
A35 0675C 6ABE0002 (00383)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00384) *
A36 0675E 6CC203EC (00385)    LOAD(BR0,SAS61(1),TF)          IQ=PG
A37 06760 6ED42090 (00386)    LOAD(BR1,PGS(2),TF)            IQ=PTHR(0)
A38 06762 709A0002 (00387)    ADD(BR1,WS,TF)                 IQ=PTHR(1)
A39 06764 72F42284 (00388)    LOAD(BR3,PGSDEQ(2),TF)         IQ=PDEQ(0)
A3A 06766 7430002B (00389)    SET(AF3)                       GET READY TO TERMINATE
A3B 06768 76BE0002 (00390)    ADD(BR3,WS,TF,C)               IQ=PDEQ(1),THEN PC1
                   (00391) *
A3C 0676A 78C203EE (00392)    LOAD(BR0,SAS62(1),TF)          IQ=M
A3D 0676C 7A300037 (00393)    SET(W1)                        STALL APS INPUT
A3E 0676E 7C000020 (00394)    NOP                            RELEASED BY APU
A3F 06770 7EC420EB (00395)    LOAD(BR0,INSTRS0(2),TF)        IQ=INSTRS0
```

```
A40 06772 B0300037 (00396)         SET(WI)                       STALL APS INPUT
A41 06774 B2642098 (00397)         LOAD(BR2,PITCHS(2))           PITCH(.) BA
A42 06776 B4500000 (00398) INS0    LOAD(BR1,MSS)                 APU SETS MSS->M
A43 06778 B7290028 (00399)         ADDB(BR2,BR1,TF)              IQ=QTM=PITCH(M)
A44 0677A B8300037 (00400)         SET(WI)                       STALL APS INPUT
A45 0677C BA000020 (00401)         NOP                           RELEASED BY APU
A46 0677E BCC420FA (00402)         LOAD(BR0,INSTRS1(2),TF)       IQ=INSTRS1
A47 06780 BE300037 (00403)         SET(WI)                       STALL APS INPUT
A48 06782 906422BC (00404)         LOAD(BR2,MSDEQ(2))            DEQ(.) BA
A49 06784 92500000 (00405) INS1    LOAD(BR1,MSS)                 APU SETS MSS->DTM
A4A 06786 9429002B (00406)         ADDB(BR2,BR1)                 DEQ(DTM)
A4B 06788 96A9002B (00407)         ADDB(BR2,BR1,TF)              IQ=DRM=DEQ(2*DTM)
                   (00408) *
A4C 0678A 98460021 (00409) KAS     LOAD(BR0,PARCS-2(3))          PARC(.) BA-2
A4D 0678C 9A500007 (00410)         LOAD(BR1,PSIZES)              PARC(.) SIZE-1
A4E 0678E 9CBA0002 (00411) PINCS1  ADD(BR0,WS,TF)                IQ=PARC(I)
A4F 06790 9E194FB1 (00412)         SUBL(BR1,1),JUMPP(PINCS1)     FOR I=0,1...7
A50 06792 A0C2043A (00413)         LOAD(BR0,SAS92(1),TF)         IQ=LTH2
                   (00414) *
A51 06794 A2200031 (00415) DNES1   CLEAR(RI)                     INPUT DONE!
A52 06796 A4000020 (00416)         NOP
                   (00417) *
                   (00418) *  O/DO APS SUBROUTINE
A53 06798 A6205F40 (00419) G105S1  JSN(G105S0,P2)                SET OUTPUT PC(P2)
A54 0679A A8005960 (00420) TOPSP1  JUMP(ENTRYS)                  ENTER @ STALL!
A55 0679C AA9A0002 (00421) ADDS1   ADD(BR1,WS,TF)                IQ=D(K)
A56 0679E AC9A0002 (00422)         ADD(BR1,WS,TF)                IQ=D(K+1)
A57 067A0 AEBA0002 (00423)         ADD(BR3,WS,TF)                IQ=V(K)
A58 067A2 B0BA0002 (00424)         ADD(BR3,WS,TF)                IQ=V(K+1)
A59 067A4 B2300037 (00425) ENTRYS  SET(WI)                       WAIT FOR THRESHOLD ...
A5A 067A6 B4000020 (00426)         NOP                           DECISION
A5B 067A8 B60055AA (00427)         JUMPC(ADDS1,AF2)              AF2=0 THEN K+2,K+3
A5C 067AA B82F0000 (00428)         ADD(BR2,MSS,NA,C)             AF2=1,THEN PC0->PC
A5D 067AC BA20546A (00429)         JUMP(TOPSP1,AF2),CLEAR        RESET LOOP PC & AF2
                   (00430) *
A5E 067AE BC307C40 (00431) G105S0  JSN(G105S3,P3)                SET OUTPUT PC(PC3)
A5F 067B0 BE500007 (00432) TOPSP2  LOAD(BW1,PSIZES)              P(.) SIZE-1
A60 067B2 C0400000 (00433)         LOAD(BW0,[0])                 QPRM(.) BA
A61 067B4 C2700000 (00434)         LOAD(BW3,MSS)                 DUMMY OP FOR EXEC
A62 067B6 C4020000 (00435)         SUB(BW0,MSS)                  QPRM(.) BA-1
A63 067B8 C6660021 (00436)         LOAD(BW2,PARCS-2(3))          P(.) BA-2
A64 067BA C90A0001 (00437) #3      ADD(BW0,HS,TF)                OQ=QPRM(I)
A65 067BC CAAA0002 (00438)         ADD(BW2,WS,TF)                OQ=DEQ(QPRM(I))
A66 067BE CC1164B1 (00439)         SUBL(BW1,1),JUMPP(#3)         FOR I=1,2...8
```

```
A67 067C0 CE442F31 (00440)          LOAD(BW0,QPRMS+DCPOS(2),TE)          QQ=DEQ(DC)=QTDC
A68 067C2 D0F20432 (00441)          LOAD(BW2,SASB8(1),TE)                QQ=DEQ. DCBIAS
A69 067C4 D3442F30 (00442)          LOAD(BW0,QPRMS+VARPOS(2),TE)         QQ=DEQ(VAR)=QTVAR
A6A 067C6 D4F20434 (00443)          LOAD(BW2,SASB9(1),TE)                QQ=DEQ. VAR.
A6B 067C8 D7442F34 (00444)          LOAD(BW0,QPRMS+PGPOS(2),TE)          QQ=DEQ(PG)=QTG
A6C 067CA D8F20436 (00445)          LOAD(BW2,SAS90(1),TE)                QQ=DEQ. PG.
A6D 067CC DA6420F8 (00446)          LOAD(BW2,INSTRS0(2))                 OVERWRITE INSTRS0+1
A6E 067CE DD2F0001 (00447)          ADD(BW2,HS,TE,C)                     AND CHANGE PC TO STALL
A6F 067D0 DE73FFC0 (00448)          LOAD(BW3,APSSMEM(1))                 OVERWRITE INS0
A70 067D2 F0BF0000 (00449)          ADD(BW3,MSS,TE,C)                    AND CHANGE PC TO STALL
A71 067D4 F3442F32 (00450)          LOAD(BW0,QPRMS+MPOS(2),TE)           QQ=DEQ(M)=QTM
A72 067D6 F52F0002 (00451)          ADD(BW2,WS,TE,C)                     OVERWRITE INSTRS1+1
A73 067D8 F6BF0000 (00452)          ADD(BW3,MSS,TE,C)                    OVERWRITE INS1
A74 067DA F8F20438 (00453)          LOAD(BW2,SAS91(1),TE)                QQ=DEQ. PITCH.
                   (00454) *
A75 067DC FA460010 (00455)          LOAD(BW0,A1S-2(3))                   A1(.) BA-2
A76 067DE FC500007 (00456)          LOAD(BW1,FSIZES)                     A1(.) SIZE-1
A77 067E0 FFF203EA (00457)          LOAD(BW2,SAS60(1),TE)                QQ=PROD(8)=FNG
A78 067E2 F0BA0002 (00458) PINCSJ   ADD(BW0,WS,TE)                       QQ=A(J)
A79 067E4 F21178B1 (00459)          SUBL(BW1,1),JUMPP(PINCSJ)            FOR J=0,1,...7
                   (00460) *
A7A 067E6 F4200030 (00461) DNFS0    CLEAR(R0)                            OUTPUT DONE!
A7B 067E8 F6000020 (00462)          NOP
                   (00463) *
                   (00464) *  O/DO APS OUTPUT SUBROUTINE
A7C 067EA F8300032 (00465) G105S3   SET(RA)                              ENABLE APU
A7D 067EC FA200030 (00466) TOPSP3   CLEAR(R0)                            STALL APS OUTPUT
A7E 067EE FC2F0000 (00467)          ADD(BW2,MSS,NA,C)                    PC2->PC
A7F 067F0 FE007D60 (00468)          JUMP(TOPSP3)                         RESET LOOP PC
                   (00469) *
         000067B2  (00470)          G105SA= *C                           ASSIGN VALUE TO CHAIN AN
                   (00471) ;                                             CHOR
    067F2          (00472)          END *A-1
                   (00473) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    067F2 00000000 (00474) G105S1   DATA 1F'0,0'
          00000102 (00475)          G105SZ=*L-G105S
    067F4          (00476)          END
```

```
AI$:        00012 (00006) (00455)
ADD$I:      00055 (00421) (00427)
AF3$TST:    00022 (00181) (00188)
AFDTSORG:   008FB (00007) (00054)
APSSMEM:    1FFC0 (00008) (00448)
BIAS$:      00008 (00160) (00162)
CSPU$NOS:   021FC (00009) (00057)
DC$:        02030 (00114) (00374)
DCSDEQ:     02224 (00025) (00376)
DCPOS:      00009 (00010) (00440)
DNESA:      00058 (00249)
DNESI:      00051 (00415)
DNESO:      0007A (00461)
ENTRYS:     00059 (00420) (00425)
G105$:      066F2 (00056) (00314) (00320) (00475)
G105$1:     00053 (00321) (00419)
G105$3:     0007C (00431) (00465)
G105$A:     067B2 (00317) (00470)
G105$I:     067F2 (00312) (00474)
G105$O:     0005F (00419) (00431)
G105$Z:     00102 (00316) (00475)
H$:         00001 (00012) (00437) (00447)
INSO:       00042 (00140) (00398)
INSI:       00049 (00141) (00405)
INCSJ:      00005 (00159) (00188)
INSTRSO:    020F8 (00140) (00395) (00446)
INSTRS1:    020FA (00141) (00402)
KAS:        0004C (00409)
KTOAS:      00039 (00213)
LOOP$:      00006 (00165) (00176)
M$:         00000 (00013)
MS$:        00000 (00014) (00398) (00405) (00428) (00434) (00435) (00449) (00452) (00467)
MSDEQ:      0228C (00028) (00404)
MPOS:       0000A (00015) (00450)
MQUANS:     00023 (00189)
PARC$:      00023 (00030) (00324) (00409) (00436)
PDEQ$1:     02134 (00017) (00327)
PDEQ$2:     02174 (00018) (00333)
PDEQ$3:     021B4 (00019) (00339)
PDEQ$4:     021D4 (00020) (00345)
PDEQ$5:     021F4 (00021) (00351)
PDEQ$6:     02204 (00022) (00357)
PDEQ$7:     02214 (00023) (00363)
PDEQ$8:     0221C (00024) (00369)
```

```
PGS:       02090 (00130) (00386)
PGSDFO:    02284 (00027) (00388)
PGPOS:     0000C (00016) (00444)
PINCS1:    0004F (00411) (00412)
PINCSJ:    00078 (00458) (00459)
PITCHS:    02098 (00133) (00397)
PSIZES:    00007 (00031) (00410) (00432) (00456)
PVECS1:    01F40 (00068) (00325)
PVECS2:    01F80 (00078) (00331)
PVECS3:    01FC0 (00088) (00337)
PVECS4:    01FF0 (00094) (00343)
PVECS5:    02000 (00100) (00349)
PVECS6:    02010 (00104) (00355)
PVECS7:    02020 (00108) (00361)
PVECS8:    02028 (00111) (00367)
QPRMS:     02F28 (00032) (00440) (00442) (00444) (00450)
QUANS:     065F2 (00055) (00152)
QUANSSA:   00000 (00150) (00154) (00305)
QUANSSZ:   00084 (00151) (00305) (00306)
SAS104:    00452 (00046)
SAS11:     00398 (00033)
SAS38:     003CF (00034) (00323)
SAS39:     003D0 (00035)
SAS52:     003EA (00036) (00373)
SAS55:     003F0 (00037) (00379)
SAS60:     003FA (00038) (00457)
SAS61:     003FC (00039) (00385)
SAS62:     003FF (00040) (00392)
SAS88:     00432 (00041) (00441)
SAS89:     00434 (00042) (00443)
SAS90:     00436 (00043) (00445)
SAS91:     00438 (00044) (00453)
SAS92:     0043A (00045) (00413)
SCLRS:     00002 (00314) (00323)
START:     065F0 (00047) (00144)
SVTS:      00382 (00029) (00033) (00034) (00035) (00036) (00037) (00038) (00039) (00040) (00041)
                 (00042) (00043) (00044) (00045) (00046)
T1S:       00017 (00172) (00177)
T2S:       0001C (00173) (00182)
TOPSP1:    00054 (00420) (00429)
TOPSP2:    0005F (00432)
TOPSP3:    0007D (00466) (00468)
VARS:      02050 (00120) (00380)
VARSDFO:   02244 (00026) (00382)
```

```
VARPOS:     0000B (00049) (00442)
VPOS:       0000B (00048)
WS:         00002 (00050) (00326) (00328) (00330) (00332) (00334) (00336) (00338) (00340) (00342)
                  (00344) (00346) (00348) (00350) (00352) (00354) (00356) (00358) (00360) (00362)
                  (00364) (00366) (00368) (00371) (00375) (00377) (00381) (00383) (00387) (00390)
                  (00411) (00421) (00422) (00423) (00424) (00438) (00451) (00458)
X23RD:      0006F (00227) (00228) (00277)
X4TH:       00069 (00229) (00268)
X5TH:       0006A (00230) (00267) (00269)
X6TH:       00062 (00232) (00259)
X7TH:       00064 (00233) (00258) (00261)
X8TH:       0005D (00235) (00253)
ZS:         00003 (00051)

LINES WITH ERRORS:     0  (MAP VERSION 800101.10)    F=     0
```

```
                    (00001) *  FCB 243... "MPDQPP(Y,U,V)"      DEQUANTIZE SIDEBAND
                    (00002) *                                  W/ K->A TRANSFORM
                    (00003) *                                  ORIGINATED:13-AUG-79
                    (00004) *                                  UPDATED:16-MAY-80
                    (00005) *  DEFINE GLOBAL SYMBOLS
          00000012  (00006)         AIS=D'18'
          00000BFB  (00007)         AFDTSORG=$BFB
          0001FFC0  (00008)         APSSMEM=$1FFC0
          000021FC  (00009)         CSPUSNOS=$21FC
          00000003  (00010)         #M=3
          00000001  (00011)         HS=1
          00000000  (00012)         MS=0
          00000000  (00013)         MSS=0
          00000023  (00014)         PARCS=D'35'
          00000007  (00015)         PSIZES=D'8'-D'1'
                    (00016) *  OBPRM=>OPRM
          00002E36  (00017)         OPRMS=D'11830'
          000003B2  (00018)         SVTS=$03B2
          000003C8  (00019)         SAS11=SVTS+2*D'11'
          00000400  (00020)         SAS39=SVTS+2*D'39'
          0000042A  (00021)         SAS60=SVTS+2*D'60'
          00000462  (00022)         SAS88=SVTS+2*D'88'
          00000464  (00023)         SAS89=SVTS+2*D'89'
          00000466  (00024)         SAS90=SVTS+2*D'90'
          00000468  (00025)         SAS91=SVTS+2*D'91'
          0000046A  (00026)         SAS92=SVTS+2*D'92'
          00006800  (00027)         START=$6800
          00000002  (00028)         WS=2
          00000003  (00029)         ZS=3
                    (00030) *
                    (00031) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000B9A  (00032)         #L=AFDTSORG+3*2*(243-128)              FCB 243
00B9A 001F6802  (00033)         ADDR DEQUS(R7,1)
00B9C 001F68C4  (00034)         ADDR G106S(R7,1)
00B9E 001021FC  (00035)         ADDR CSPUSNOS(,1,0)
                    (00036)         EJECT
```

```
                    (00037) *
                    (00038) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00039) *
                    (00040)          BUS 2
           00002134 (00041)          #I=D'H500'
                    (00042)          EVEN
                    (00043) *
                    (00044) *  PARCOR(1) W/ 5 BITS
02134 107BE6BE (00045) PDECS1  DATA  0.80431E-02,  0.24740E-01,  0.42199E-01,  0.60427E-01
02136 32AAF2BE
02138 566C6DBE
0213A 78C126BE
0213C 0A2D72C0 (00046)         DATA  0.79512E-01,  0.10024E+00,  0.12414E+00,  0.14986E+00
0213E 0CD4AA40
02140 0FF301C0
02142 132E9CC0
02144 16B39040 (00047)         DATA  0.17589E+00,  0.20368E+00,  0.23451E+00,  0.26811E+00
02146 1A122FC0
02148 1E046C40
0214A 22516DC0
0214C 26E9CCC0 (00048)         DATA  0.30401E+00,  0.34440E+00,  0.39167E+00,  0.44480E+00
0214E 2C154CC0
02150 32223E40
02152 38EE34C0
02154 41264 1C0 (00049)        DATA  0.50898E+00,  0.57740E+00,  0.65201E+00,  0.74030E+00
02156 49E83E40
02158 53751040
0215A 5EC22640
0215C 66B30B40 (00050)         DATA  0.80236E+00,  0.88455E+00,  0.97486E+00,  0.10631E+01
0215E 713BEE40
02160 7CCB3640
02162 08813AC1
02164 0971A9C1 (00051)         DATA  0.11805E+01,  0.13001E+01,  0.13851E+01,  0.15023E+01
02166 0A669AC1
02168 0B14AE41
0216A 0C04B5C1
0216C 0CA40D41 (00052)         DATA  0.15805E+01,  0.17080E+01,  0.18691E+01,  0.19972E+01
0216E 0DA9EEC1
02170 0EE3EAC1
02172 0FEA43C1
                    (00053) *
                    (00054) *  PARCOR(2) W/ 5 BITS
02174 0B25EEC0 (00055) PDECS2  DATA  0.87097E-01,  0.26319E+00,  0.36928E+00,  0.46156E+00
02176 21B035C0
```

```
PAGE     3:       ECH 243... "MPDOPP(Y,U,V)"      DEQUANTIZE SIDEBAND

      0217B 2E449140
      0217A 3B1465C0
      0217C 47367A40 (00056)          DATA  0.55635E+00,  0.62403E+00,  0.68141E+00,  0.73631E+00
      0217E 4EE03740
      02180 573B7140
      02182 5E3E67C0
      02184 65532640 (00057)          DATA  0.79160E+00,  0.84597E+00,  0.90653E+00,  0.97650E+00
      02186 6C4BBEC0
      02188 74092CC0
      0218A 7CEDE3C0
      0218C 08624DC1 (00058)          DATA  0.10480E+01,  0.11214E+01,  0.12027E+01,  0.12736E+01
      0218E 08EBA0C1
      02190 099E2141
      02192 0A305541
      02194 0AB4D6C1 (00059)          DATA  0.13383E+01,  0.14072E+01,  0.14643E+01,  0.15197E+01
      02196 0B41F241
      02198 0BB6E2C1
      0219A 0C2B5841
      0219C 0C97BD41 (00060)          DATA  0.15740E+01,  0.16295E+01,  0.16759E+01,  0.17142E+01
      0219E 0D093741
      021A0 0D6B3F41
      021A2 0DB6AE41
      021A4 0E00D1C1 (00061)          DATA  0.17504E+01,  0.17863E+01,  0.18208E+01,  0.18521E+01
      021A6 0E4A57C1
      021A8 0E90EEC1
      021AA 0ED119C1
      021AC 0F0D1B41 (00062)          DATA  0.18814E+01,  0.19108E+01,  0.19426E+01,  0.19807E+01
      021AE 0F4951C1
      021B0 0F8A71C1
      021B2 0FDB7941
                     (00063) *
                     (00064) *  PARCOR(3) W/ 4 BITS
      021B4 1E7B0B40 (00065) PDECS3  DATA  0.23813E+00,  0.39243E+00,  0.51204E+00,  0.61256E+00
      021B6 323B2540
      021B8 41BAB6C0
      021BA 4E685DC0
      021BC 59EBD8C0 (00066)          DATA  0.70251E+00,  0.78737E+00,  0.86998E+00,  0.95262E+00
      021BE 64CBBA40
      021C0 6E5BB140
      021C2 79EE73C0
      021C4 084AC0C1 (00067)          DATA  0.10365E+01,  0.11226E+01,  0.12129E+01,  0.13086E+01
      021C6 08EB15C1
      021C8 09B404C1
      021CA 0A7B0341
```

```
021CC 08428E41 (00068)          DATA  0.14075E+01,  0.15162E+01,  0.16545E+01,  0.18438E+01
021CE 0C212D41
021D0 0D3C6A41
021D2 0EC01A41
                (00069) *
                (00070) *  PARCOR(4) W/ 4 BITS
021D4 2E34C1C0 (00071) PDECS4  DATA  0.31411E+00,  0.50442E+00,  0.63992E+00,  0.75167E+00
021D6 4090D5C0
021D8 51E8E640
021DA 6036B8C0
021DC 6CE50C40 (00072)          DATA  0.85074E+00,  0.94305E+00,  0.10322E+01,  0.11205E+01
021DE 78B5ECC0
021E0 0841E241
021E2 08E6C8C1
021E4 09ACD9C1 (00073)          DATA  0.12094E+01,  0.13000E+01,  0.13929E+01,  0.14888E+01
021E6 0A666641
021E8 0B24A8C1
021EA 0BE90EC1
021EC 0C85DCC1 (00074)          DATA  0.15888E+01,  0.16951E+01,  0.18129E+01,  0.19500E+01
021EE 0D8E90C1
021F0 0E80D1C1
021F2 0F9999C1
                (00075) *
                (00076) *  PARCOR(5) W/ 3 BITS
021F4 38E90940 (00077) PDECS5  DATA  0.44510E+00,  0.66865E+00,  0.82476E+00,  0.95821E+00
021F6 559652C0
021F8 69918C40
021FA 7AA6A040
021FC 08AFEC41 (00078)          DATA  0.10859E+01,  0.12194E+01,  0.13739E+01,  0.15907E+01
021FE 09C154C1
02200 0AFD8E41
02202 0CB9C0C1
                (00079) *
                (00080) *  PARCOR(6) W/ 3 BITS
02204 3B2200C0 (00081) PDECS6  DATA  0.46200E+00,  0.68463E+00,  0.84172E+00,  0.97631E+00
02206 57A1F4C0
02208 6B8D7840
0220A 7CF789C0
0220C 08D70A41 (00082)          DATA  0.11050E+01,  0.12400E+01,  0.13980E+01,  0.16206E+01
0220E 09EB8541
02210 0B2F1AC1
02212 0CF6FD41
                (00083) *
                (00084) *  PARCOR(7) W/ 2 BITS
```

```
02214 320DACC0 (00085) PDFCS7  DATA  0.39739E+00,  0.73761E+00,  0.10127E+01,  0.13204E+01
02216 5E6A0140
02218 2B1A0241
0221A 0A902DC1
               (00086) *
               (00087) *  PARCOR(8) W/ 2 BITS
0221C 4A0E4140 (00088) PDFCS8  DATA  0.57856E+00,  0.86541E+00,  0.11045E+01,  0.13815E+01
0221E 6ECSC140
02220 08D60411
02222 06004FC1
               (00089) *
               (00090) *  DC BIAS W/ 4 BITS
02224 7E937BBF (00091) DCS     DATA  0.62293E-01,  0.18721E+00,  0.31317E+00,  0.44096E+00
02226 17E67F40
02228 2B15F140
0222A 3B7160C0
0222C 492B3BC0 (00092)         DATA  0.57154E+00,  0.70609E+00,  0.84579E+00,  0.99227E+00
0222E 5A612B40
02230 6C42EBC0
02232 7E02B3C0
02234 092F1441 (00093)         DATA  0.11475E+01,  0.13143E+01,  0.14964E+01,  0.16991E+01
02236 0AB3AFC1
02238 0BFBA0C1
0223A 0D97C1C1
0223C 0F74B7C1 (00094)         DATA  0.19319E+01,  0.22117E+01,  0.25759E+01,  0.31324E+01
0223E 1181BFC1
02240 149B7141
02242 190F27C1
               (00095) *
               (00096) *  VARIANCE W/ 5 BITS
02244 23AF1441 (00097) VARS    DATA  0.44600E+01,  0.68023E+01,  0.87943E+01,  0.10868E+02
02246 366B1C41
02248 465AB9C1
0224A 56F1A9C1
0224C 69F9DB41 (00098)         DATA  0.13247E+02,  0.15849E+02,  0.18140E+02,  0.20412E+02
0224E 7ECAC041
02250 0911FBC2
02252 0A34BC42
02254 0B3B3142 (00099)         DATA  0.22439E+02,  0.24372E+02,  0.26458E+02,  0.28495E+02
02256 0C2F9DC2
02258 003A9FC2
0225A 0E3E5C12
0225C 0F293742 (00100)         DATA  0.30322E+02,  0.32035E+02,  0.33741E+02,  0.35525E+02
0225E 10047AC2
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

3 OF 9

AD
A091663

```
02260 10DED942
02262 11C33342
02264 12B70A42 (00101)        DATA  0.37430E+02,  0.39343E+02,  0.41065E+02,  0.42664E+02
02266 13AHF742
02268 14885 1C2
0226A 1554FDC2
0226C 161C6A42 (00102)        DATA  0.44222E+02,  0.45741E+02,  0.47224E+02,  0.48695E+02
0226E 16DFD942
02270 179CAC42
02272 185BF5C2
02274 19156042 (00103)        DATA  0.50167E+02,  0.51633E+02,  0.53034E+02,  0.54385E+02
02276 19D10642
02278 1AB45A42
0227A 1B3147C2
0227C 1BF10642 (00104)        DATA  0.55758E+02,  0.57223E+02,  0.58810E+02,  0.60709E+02
0227E 1C9CBB42
02280 1D67AF42
02282 1E5AC0C2
               (00105) *
               (00106) *  PITCH GAIN W/ 2 BITS
02284 32219640 (00107) PGAINS  DATA  0.39165E+00,  0.58430E+00,  0.74993E+00,  0.89805E+00
02286 4ACA57C0
02288 5FFDB4C0
0228A 72F34D40
               (00108) *
               (00109) *  PITCH W/ 6 BITS
0228C 78000041 (00110) MSDEC   DATA 15.
0228E 08000042 (00111)     DATA 16.,17.,18.,19.,20.,21.,22.,23.,24.,25.,26.,27.,28.,29.,30.,31.
02290 08800042
02292 09000042
02294 09800042
02296 0A000042
02298 0A800042
0229A 0B000042
0229C 0B800042
0229E 0C000042
022A0 0C800042
022A2 0D000042
022A4 0D800042
022A6 0E000042
022A8 0E800042
022AA 0F000042
022AC 0F800042
022AE 10000042 (00112)     DATA 32.,33.,34.,35.,36.,37.,38.,39.,40.,41.,42.,43.,44.,45.,46.,47.
```

```
022B0 10800042
022B2 11000042
022B4 11800042
022B6 12000042
022B8 12800042
022BA 13000042
022BC 13800042
022BE 14000042
022C0 14800042
022C2 15000042
022C4 15800042
022C6 16000042
022C8 16800042
022CA 17000042
022CC 17800042
022CE 18000042 (00113)      DATA 48.,49.,50.,51.,52.,53.,54.,55.,56.,57.,58.,59.,60.,61.,62.
022D0 18800042
022D2 19000042
022D4 19800042
022D6 1A000042
022D8 1A800042
022DA 1B000042
022DC 1B800042
022DE 1C000042
022E0 1C800042
022E2 1D000042
022E4 1D800042
022E6 1E000042
022E8 1E800042
022EA 1F000042
022EC 1F800042 (00114)      DATA 63.,65.,67.,69.,71.,73.,75.,77.,79.,81.,83.,85.,87.,89.,91.,93.
022EE 20800042
022F0 21800042
022F2 22800042
022F4 23800042
022F6 24800042
022F8 25800042
022FA 26800042
022FC 27800042
022FE 28800042
02300 29800042
02302 2A800042
02304 2B800042
02306 2C800042
```

```
0230B 2D800042
0230A 2E800042
                (00115)          EVEN
0230C 5450      (00116) INSTRSO DATA INSO*D'512'+X'50',X'0000'
0230D 0000
                (00117)          BUS 1
       00006800 (00118)          *L=START
                (00119)          EJECT
```

```
                             (00120) *
                             (00121) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                             (00122) *
                             (00123)            EVEN
    06800 0000               (00124)            DATA DEQUSSA                              START ADDRESS
    06801 005D               (00125)            DATA DEQUSSZ                              MODULE SIZE
                             (00126) DEQUS      BEGIN APU(DEQU)
          00000000           (00127)            #A=0
A00 06802 00000000           (00128) DEQUSSA    NOP                                       FOR TIMING PULSE OR CSPU
A01 06804 16801680           (00129)            K(1.0)                                    R=1.0
A02 06806 08910891           (00130)            MOV(R,A1)                                 A1=1.0(BIAS FOR PARCORS)
A03 06808 08FC0000           (00131) INCSJ      MOV(IOA,OO)\NOP                           LEVEL->INSTRS0+1
A04 0680A 901C0004           (00132) #1         JUMPC(#1,EO)                              WAIT FOR OUTPUT TO CLEAR
A05 0680C 20370000           (00133)            CLEAR(WI)\NOP                             RELEASE APS INPUT
A06 0680E 20500000           (00134)            SET(RO)\NOP                               RELEASE APS OUTPUT
A07 06810 08FC0000           (00135)            MOV(IOA,OO)\NOP                           OVERWRITE INS0
A08 06812 901C0008           (00136) #2         JUMPC(#2,EO)                              WAIT FOR OUTPUT TO CLEAR
A09 06814 20370000           (00137)            CLEAR(WI)\NOP                             RELEASE APS INPUT
A0A 06816 08F00000           (00138)            MOV(IOA,A0)\NOP                           A0=DEQ(I)
A0B 06818 9009000D           (00139)            JUMPC(BIASS,AF1)                          END OF PARCORS?
A0C 0681A 08110000           (00140)            MOV(ZERO,A1)\NOP                          YES,REMOVE BIAS
A0D 0681C 49000000           (00141) BIASS      SUB(A0,A1)\NOP                            DEQ(I)-BIAS
A0E 0681E 20500000           (00142)            SET(RO)\NOP                               RELEASE APS OUTPUT
A0F 06820 089C0000           (00143)            MOV(R,OO)\NOP                             OO=D(OPRM(I))
A10 06822 90160003           (00144)            JUMPC(INCSJ,FWI)                          FOR K(J),J=1,...,8;DC,VAR
                             (00145) ;                                                    ,PG,M
A11 06824 20372037           (00146)            CLEAR(WI)                                 RELEASE APS INPUT
                             (00147) *
A12 06826 08F10000           (00148) KTOAS      MOV(IOA,A1)\NOP                           P(1)
A13 06828 0A200000           (00149)            NEG(A1)\NOP                               MAKE IT K(1)
A14 0682A 08880000           (00150)            MOV(R,M0)\NOP
A15 0682C 088F0000           (00151)            MOV(R,M7)\NOP                             K(1)**2 FOR ENERGY
A16 0682E 08910000           (00152)            MOV(R,A1)\NOP                             SAVE K[1]
A17 06830 84600000           (00153)            MUL(M0,M7)\NOP                            SO SQUARE IT
A18 06832 16801680           (00154)            K(+1)                                     NEED THIS AROUND, TOO
A19 06834 08D008D0           (00155)            MOV(IO,A0)                                P(2)
A1A 06836 08950000           (00156)            MOV(R,A5)\NOP                             1.0
A1B 06838 08B608B6           (00157)            MOV(P,A6)                                 K[1]^2
A1C 0683A 4EA04EA0           (00158)            SUB(A5,A6)                                1.0-K[1]^2
A1D 0683C 020D020D           (00159)            MOV(M5),R(A0)                             PROD(1),GET K[2]
A1E 0683E 088F088F           (00160)            MOV(R,M7)                                 K[2]
A1F 06840 088B088B           (00161)            MOV(R,M3)                                 K[2]
A20 06842 30000048           (00162)            CALL(X23RD)                               DO 2ND ITERATION (M=2)
A21 06844 30000048           (00163)            CALL(X23RD)                               DO 3RD ITER (M=3)
```

```
A22 06846 30000042 (00164)          CALL(X4TH)
A23 06848 30000043 (00165)          CALL(X5TH)
A24 0684A 00000260 (00166)          NOP\R(A3)                         GET A[5](3) READY
A25 0684C 3000003B (00167)          CALL(X6TH)                        M=6
A26 0684E 3000003D (00168)          CALL(X7TH)                        M=7
A27 06850 00000280 (00169)          NOP\R(A4)                         GET A[7](4) READY
A28 06852 30000036 (00170)          CALL(X8TH)
A29 06854 08CB0000 (00171)          MOV(10,M3)\NOP                    LTH2 (=SA)
A2A 06856 85A00000 (00172)          MUL(M3,M5)\NOP                    PROD(8)*LTH2
A2B 06858 02200000 (00173)          R(A1)\NOP
A2C 0685A 088C0000 (00174)          MOV(P,DO)\NOP                     OUTPUT ENERGY = LTH2*PRO
                   (00175) ;                                          D(8)
A2D 0685C 025C0000 (00176)          MOV(DO),R(A2)\NOP                 A(1) OUT
A2E 0685E 027C0000 (00177)          MOV(DO),R(A3)\NOP                 A(2) OUT
A2F 06860 029C0280 (00178)          MOV(DO),R(A4)\R(A4)               A(3) OUT
A30 06862 089C027C (00179)          MOV(R,DO)\MOV(DO),R(A3)           A(4) OUT,A(5) OUT
A31 06864 0000025C (00180)          NOP\MOV(DO),R(A2)                 A(6) OUT
A32 06866 0000023C (00181)          NOP\MOV(DO),R(A1)                 A(7) OUT
A33 06868 0000089C (00182)          NOP\MOV(R,DO)                     A(8) OUT
                   (00183) *
A34 0686A 20320000 (00184) DNESA    CLEAR(RA)\NOP                     APU DONE!
A35 0686C 00000000 (00185)          NOP
                   (00186) *
                   (00187) *  K->A APU SUBROUTINES
A36 0686E 0000089B (00188) X8TH     NOP\MOV(R,EXO)                    \A[7](4)
A37 06870 08540854 (00189)          MOV(EXI,A4)                       A[7](4)
A38 06872 85B00000 (00190)          MUL(M3,M4)\NOP                    A[M-1](M-4)*K[M] ((M=8))
A39 06874 08B008B0 (00191)          MOV(P,A0)
                   (00192) *
A3A 06876 1000003D (00193)          JUMP(X7TH)
A3B 06878 00000898 (00194) X6TH     NOP\MOV(R,EXO)                    A[5](3)
A3C 0687A 08530000 (00195)          MOV(EXI,A3)\NOP                   A[5](3)
A3D 0687C 85608560 (00196) X7TH     MUL(M2,M7)\MUL(M2,M7)             A[M-1](M-3)*K[M]\A[M-1](
                   (00197) ;                                          3)*K[M]
A3E 0687E 44000000 (00198)          ADD(A0,A4)\NOP                    SUM FOR A[M](4)
A3F 06880 08940000 (00199)          MOV(R,A4)\NOP                     STORE A[M](4)
A40 06882 08B008B0 (00200)          MOV(P,A0)
                   (00201) *
A41 06884 10000043 (00202)          JUMP(X5TH)
A42 06886 08520000 (00203) X4TH     MOV(EXI,A2)\NOP                   A[3](2)
A43 06888 84F084F0 (00204) X5TH     MUL(M1,M7)\MUL(M1,M7)             A[M-1](M-2)*K[M]\A[M-1](
                   (00205) ;                                          2)*K[M]
A44 0688A 43004300 (00206)          ADD(A0,A3)\ADD(A0,A3)             SUM FOR A[M](3)\A[M](M-3
                   (00207) ;                                          )
```

```
A45 068BC 08980898 (00208)          MOV(R,FXO)\MOV(R,FXO)
A46 068BE 084C084A (00209)          MOV(FXI,M4)\MOV(EXI,M2)                 STORE A[M](M-3)\A[M](3)
A47 06890 08B008B0 (00210)          MOV(P,A0)
                   (00211) *
A48 06892 84608460 (00212) X23BD    MUL(M0,M7)\MUL(M0,M7)                   A[M-1](M-1)*K[M]\A[M-1](
                   (00213) ;                                                1)*K[M]
A49 06894 42134214 (00214)          MOV(A3),ADD(A0,A2)\MOV(A4),ADD(A0,A2)   STORE A[M](3),SUM FOR A[
                   (00215) ;                                                M](2)\A[M](M-3),SUM A[M]
                   (00216) ;                                                (M-2)
A4A 06896 08980898 (00217)          MOV(R,FXO)\MOV(R,FXO)
A4B 06898 084A0849 (00218)          MOV(FXI,M2)\MOV(EXI,M1)                 ATORE A[M](M-2)\A[M](2)
A4C 0689A 08B008B0 (00219)          MOV(P,A0)\MOV(P,A0)
A4D 0689C 41124113 (00220)          MOV(A2),ADD(A0,A1)\MOV(A3),ADD(A0,A1)   STORE A[M](2),SUM FOR A[
                   (00221) ;                                                M](1)\A[M](M-2),SUM A[M]
                   (00222) ;                                                (M-1)
                   (00223) *
A4E 0689E 85F085F0 (00224)          MUL(M3,M7)                              K[M]^2
A4F 068A0 08980898 (00225)          MOV(R,FXO)\MOV(R,FXO)
A50 068A2 0849084B (00226)          MOV(FXI,M1)\MOV(EXI,M0)                 A[M](M-1)\A[M](1)
A51 068A4 08910892 (00227)          MOV(R,A1)\MOV(R,A2)                     A[M](1)\A[M](M-1)
A52 068A6 08F708F7 (00228)          MOV(IOA,A7)                             -A[M](M) (=P[M])
A53 068A8 0AF00AF0 (00229)          NEG(A7)                                 MAKE IT REFLN COEF
A54 068AA 08B608B6 (00230)          MOV(P,A6)                               K[M]^2
A55 068AC 4EA80891 (00231)          MOV(M0),SUB(A5,A6)\MOV(R,A1)            K[M]=A[M](M),1-K[M]^2
A56 068AE 08D708D7 (00232)          MOV(IO,A7)                              P[M+1]
A57 068B0 02EB02EB (00233)          MOV(M3),R(A7)                           1-K[M]^2,GET K[M+1]
A58 068B2 85A085A0 (00234)          MUL(M3,M5)                              (1-K[M]^2)*PROD
A59 068B4 08AD08AD (00235)          MOV(P,M5)                               NEW PROD
A5A 068B6 088B088B (00236)          MOV(R,M3)                               K[M+1]
A5B 068B8 08BF08BF (00237)          MOV(R,M7)                               K[M+1]
A5C 068BA A000A000 (00238)          RETURN
          0000005D (00239)          DEQUSSZ=*A-DEQUSSA
    068BC          (00240)          END DEQUSSZ
                   (00241)          EJECT
```

```
                     (00242) *
                     (00243) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                     (00244) *
                     (00245)          EVEN
    068BC 00006960   (00246)          ADDR G106SI                 POINTER TO CONSTRUCTED I
                     (00247) ;                                    NSTRUCTION BLOCK
    068BE 000068C4   (00248)          ADDR G106$+2*SCLRS          POINTER TO SCALAR BLOCK
    068C0 0000       (00249)          DATA 0                      NUMBER OF SCALARS
    068C1 009E       (00250)          DATA G106$Z                 MODULE SIZE
    068C2 00006924   (00251)          ADDR G106$A                 POINTER TO CHAIN ANCHOR
                     (00252)          EVEN                        END ON WORD BOUNDARY
                     (00253) *
                     (00254) G106$    BEGIN APS(G106)
A00 068C4 00102440   (00255) SCLRS    JSP(G106$1,P1)              SET INPUT PC(PC1)
A01 068C6 02300030   (00256)          SET(RO)                     ENABLE APS OUTPUT
A02 068C8 04442F35   (00257)          LOAD(BRO,QPRMS-1(2))        QPRM(.) BA-1
A03 068CA 06642134   (00258)          LOAD(BR2,PDECS1(2))         PDECS1(.) BA
A04 068CC 090E0001   (00259)          ADD(BRO,HS,TE,C)            IQ=P1 THEN PC=PC1
A05 068CE 0A642174   (00260)          LOAD(BR2,PDECS2(2))         PDECS2(.) BA
A06 068D0 0D0E0001   (00261)          ADD(BRO,HS,TE,C)            IQ=P2 THEN PC=PC1
A07 068D2 0E6421B4   (00262)          LOAD(BR2,PDECS3(2))         PDECS3(.) BA
A08 068D4 110F0001   (00263)          ADD(BRO,HS,TE,C)            IQ=P3 THEN PC=PC1
A09 068D6 126421D4   (00264)          LOAD(BR2,PDECS4(2))         PDECS4(.) BA
A0A 068D8 150F0001   (00265)          ADD(BRO,HS,TE,C)            IQ=P4 THEN PC=PC1
A0B 068DA 166421F4   (00266)          LOAD(BR2,PDECS5(2))         PDECS5(.) BA
A0C 068DC 190E0001   (00267)          ADD(BRO,HS,TE,C)            IQ=P5 THEN PC=PC1
A0D 068DE 1A642204   (00268)          LOAD(BR2,PDECS6(2))         PDECS6(.) BA
A0E 068E0 1D0E0001   (00269)          ADD(BRO,HS,TE,C)            IQ=P6 THEN PC=PC1
A0F 068E2 1E642214   (00270)          LOAD(BR2,PDECS7(2))         PDECS7(.) BA
A10 068E4 210F0001   (00271)          ADD(BRO,HS,TE,C)            IQ=P7 THEN PC=PC1
A11 068E6 2264221C   (00272)          LOAD(BR2,PDECS8(2))         PDECS8(.) BA
A12 068E8 250F0001   (00273)          ADD(BRO,HS,TE,C)            IQ=P8 THEN PC=PC1
A13 068EA 26642244   (00274)          LOAD(BR2,VARS(2))           VAR(.) BA-2
A14 068EC 290F0001   (00275)          ADD(BRO,HS,TE,C)            IQ=VAR THEN PC=PC1
A15 068EE 2A642224   (00276)          LOAD(BR2,DCS(2))            DC(.) BA-2
A16 068F0 2D0F0001   (00277)          ADD(BRO,HS,TE,C)            IQ=DC THEN PC=PC1
A17 068F2 2E64228C   (00278)          LOAD(BR2,MSDEC(2))          M(.) BA
A18 068F4 310F0001   (00279)          ADD(BRO,HS,TE,C)            IQ=QTM THEN PC=PC1
A19 068F6 320A0001   (00280)          ADD(BRO,HS,NA)              NO IQ SINCE IQ=QTV
A1A 068F8 34642284   (00281)          LOAD(BR2,PGAINS(2))         P.GAIN(.) BA-2
A1B 068FA 370F0001   (00282)          ADD(BRO,HS,TE,C)            IQ=PG THEN PC=PC1
A1C 068FC 38300037   (00283)          SET(WI)                     STALL APS INPUT
                     (00284) *
A1D 068FE 3A460021   (00285) KAS      LOAD(BRO,PARCS-2(3))        PARC(.) BA-2
```

```
A1F 06900 3C500007 (00286)         LOAD(BR1,PSIZES)               PARC(.) SIZE-1
A1F 06902 3E8A0002 (00287) PINCSI  ADD(BR0,WS,TF)                 IQ=PARC(I)
A20 06904 40191FB1 (00288)         SUBL(BR1,1),JUMPP(PINCSI)      FOR I=0,1...7
A21 06906 42C2043A (00289)         LOAD(BR0,SAS92(1),TF)          IQ=LTH2
                   (00290) *
A22 06908 44200031 (00291) DNFSI   CLEAR(RI)                      INPUT DONE!
A23 0690A 46000020 (00292)         NOP
                   (00293) *
A24 0690C 48202F40 (00294) G106S1  JSR(G106S0,P2)                 SET OUTPUT PC(P2)
A25 0690E 4A300037 (00295) TOPSP1  SET(WI)                        STALL APS INPUT
A26 06910 4C000020 (00296)         NOP                            RELEASED BY APU
A27 06912 4FF4230C (00297)         LOAD(BR3,INSTRS0(2),TF)        IQ=INSTRS0
A28 06914 50300037 (00298)         SET(WI)                        STALL APS INPUT
A29 06916 52000020 (00299)         NOP                            RELEASED BY APU
A2A 06918 54500000 (00300) INS0    LOAD(BR1,MSS)                  APU SETS MSS->LEVEL
A2B 0691A 5629002B (00301)         ADDB(BR2,BR1)                  D(QPRM(I))
A2C 0691C 58A9002B (00302)         ADDB(BR2,BR1,TF)               IQ=D(QPRM(2I))
A2D 0691E 5A3F0000 (00303)         ADD(BR3,MSS,NA,C)              FOR PC1->PC0
A2E 06920 5C002560 (00304)         JUMP(TOPSP1)                   RESET TOP-OF-LOOP
                   (00305) *
A2F 06922 5F304640 (00306) G106S0  JSR(G106S3,P3)                 SET OUTPUT PC(PC3)
A30 06924 60400000 (00307)         LOAD(BW0,(0))                  PARC(.) BA
A31 06926 62500000 (00308)         LOAD(BW1,MSS)                  PARC(.) SIZE-1
A32 06928 64020000 (00309)         SUB(BW0,MSS)                   PARC(.) BA-1
A33 0692A 663F0000 (00310) PARCS0  ADD(BW3,MSS,NA,C)              PC3->PC
A34 0692C 688A0002 (00311)         ADD(BW0,WS,TF)                 OQ=PARC(I)
A35 0692E 6A1133B1 (00312)         SUBL(BW1,1),JUMPP(PARCS0)      ETC FOR I=1,2...8
A36 06930 6C300029 (00313)         SET(AF1)                       SET "END-OF-PARCORS"
A37 06932 6F3F0000 (00314) VARS0   ADD(BW3,MSS,NA,C)              PC3->PC
A38 06934 70C20434 (00315)         LOAD(BW0,SAS89(1),TF)          OQ=D(QTVAR)=DTVAR
A39 06936 723F0000 (00316) DCS0    ADD(BW3,MSS,NA,C)              PC3->PC
A3A 06938 74C20432 (00317)         LOAD(BW0,SAS88(1),TF)          OQ=D(QTDC)=DTDC
A3B 0693A 763F0000 (00318) MS0     ADD(BW3,MSS,NA,C)              PC3->PC
A3C 0693C 78C2043B (00319)         LOAD(BW0,SAS91(1),TF)          OQ=D(QTM)=DTM
A3D 0693E 7A3F0000 (00320) PGS0    ADD(BW3,MSS,NA,C)              PC3->PC
A3E 06940 7CC20436 (00321)         LOAD(BW0,SAS90(1),TF)          OQ=D(QTG)=DTG
                   (00322) *
A3F 06942 7F460010 (00323)         LOAD(BW0,A1S-2(3))             A1(.) BA-2
A40 06944 80500007 (00324)         LOAD(BW1,PSIZES)               A1(.) SIZE-1
A41 06946 82F203FA (00325)         LOAD(BW2,SAS60(1),TF)          OQ=PROD(8)=ENG
A42 06948 848A0002 (00326) PINCSJ  ADD(BW0,WS,TF)                 OQ=A(J)
A43 0694A 861142B1 (00327)         SUBL(BW1,1),JUMPP(PINCSJ)      FOR J=0,1,...7
                   (00328) *
A44 0694C 88200030 (00329) DNFS0   CLEAR(R0)                      OUTPUT DONE!
```

```
A45 0694F 8A000020 (00330)          NOP
                   (00331) *
A46 06950 8C300032 (00332) G106$3   SET(BA)                                  ENABLE APU
A47 06952 8E64230D (00333) TOPSP3   LOAD(B#2,INSTRS0+1(2),TE)               OVERWRITE INSTRS0+1
A48 06954 90200030 (00334)          CLEAR(R0)                                STALL APS OUTPUT
A49 06956 92000020 (00335)          NOP                                      RELEASED BY APU
A4A 06958 94E3FEC0 (00336)          LOAD(BW2,APSSPEM(1),TE)                  OVERWRITE INS0
A4B 0695A 96200030 (00337)          CLEAR(R0)                                STALL APS OUTPUT
A4C 0695C 983E0000 (00338)          ADD(B#3,MSS,NA,C)                        PC2->PC
A4D 0695E 9A004760 (00339)          JUMP(TOPSP3)                             RESET TOP-OF-LOOP
                   (00340) *
          00006924 (00341)          G106$A= *C                               ASSIGN VALUE TO CHAIN AN
                   (00342) ;                                                 CHOR
    06960          (00343)          END *A-1
                   (00344) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    06960 00000000 (00345) G106$I   DATA 1F'0,0'
          0000009F (00346)          G106$Z=*L-G106$
    06962          (00347)          END
```

```
AIS:        00012 (00006) (00323)
AFDTSORG:   008F8 (00007) (00032)
APSSMEM:    1FFC0 (00008) (00336)
BIAS$:      0000D (00139) (00141)
CSPUSNOS:   021FC (00009) (00035)
DC$:        02224 (00091) (00276)
DCS0:       00039 (00316)
DEQUS:      06802 (00033) (00126)
DEQUSSA:    00000 (00124) (00128) (00239)
DEQUSSZ:    00050 (00125) (00239) (00240)
DNFSA:      00034 (00184)
DNFS1:      00022 (00291)
DNFS0:      00044 (00329)
G106$:      068C4 (00034) (00248) (00254) (00346)
G106$1:     00024 (00255) (00294)
G106$3:     00046 (00306) (00332)
G106$A:     06924 (00251) (00341)
G106$T:     06960 (00246) (00345)
G106$0:     0002F (00294) (00306)
G106$Z:     0009F (00250) (00346)
H$:         00001 (00011) (00259) (00261) (00263) (00265) (00267) (00269) (00271) (00273) (00275)
                  (00277) (00279) (00280) (00282)
INS0:       0002A (00116) (00300)
INCSJ:      00003 (00131) (00144)
INSTRSD:    0230C (00116) (00297) (00333)
KAS:        00010 (00285)
KTDAS:      00012 (00148)
MS:         00000 (00012)
MSS:        00000 (00013) (00300) (00303) (00308) (00309) (00310) (00314) (00316) (00318) (00320)
                  (00338)
MSDEC:      0228C (00110) (00278)
MSU:        0003B (00318)
PARCS:      00023 (00014) (00285)
PARCS0:     00033 (00310) (00312)
PDECS1:     02134 (00045) (00258)
PDECS2:     02174 (00055) (00260)
PDECS3:     02184 (00065) (00262)
PDECS4:     021D4 (00071) (00264)
PDECS5:     021F4 (00077) (00266)
PDECS6:     02204 (00081) (00268)
PDECS7:     02214 (00085) (00270)
PDECS8:     0221C (00088) (00272)
PGS0:       0003D (00320)
PGAINS:     02284 (00107) (00281)
```

```
PINCSI:    0001F (00287) (00288)
PINCSJ:    00042 (00326) (00327)
PSIZES:    00007 (00015) (00286) (00324)
QPRMS:     02F36 (00017) (00257)
SAS11:     0039B (00019)
SAS39:     003D0 (00020)
SAS60:     003FA (00021) (00325)
SAS88:     00432 (00022) (00317)
SAS89:     00434 (00023) (00315)
SAS90:     00436 (00024) (00321)
SAS91:     00438 (00025) (00319)
SAS92:     0043A (00026) (00289)
SCLRS:     00000 (00248) (00255)
START:     06800 (00027) (00118)
SVTS:      00382 (00018) (00019) (00020) (00021) (00022) (00023) (00024) (00025) (00026)
TOPSP1:    00025 (00295) (00304)
TOPSP3:    00047 (00333) (00339)
VARS:      02244 (00097) (00274)
VARSO:     00037 (00314)
WS:        00002 (00028) (00287) (00311) (00326)
X23RD:     00048 (00162) (00163) (00212)
X4TH:      00042 (00164) (00203)
X5TH:      00043 (00165) (00202) (00204)
X6TH:      0003B (00167) (00194)
X7TH:      0003D (00168) (00193) (00196)
X8TH:      00036 (00170) (00188)
ZS:        00003 (00029)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E=    0
```

```
                     (00001) * FCB 244... "MPMWGX(Y,U,V,W)"    EXTRACT M,PG,MOVE X
                     (00002) *                                 ORIGINATED:04-OCT-79
                     (00003) *                                 UPDATED:29-MAY-80
                     (00004) * Y BID IS AUTOCORRELATIONS, USED AS BOTH OUTPUT AND INPUT.
                     (00005) * THESE ARE MOVED FROM BUFFER X1R INTO Y.
                     (00006) * U BID IS INPUT SAMPLES FROM THE A/D.  THE LAST
                     (00007) * FEW POINTS OF THIS BUFFER ARE MOVED TO THE BEGINNING OF BUFFER
                     (00008) * XOM, TO PROVIDE FRAME OVERLAP.
                     (00009) * V BID IS THE ARRAY OF PARCOR COEFFICIENTS (OUTPUT)
                     (00010) * W BID IS ARAY OF PEDICTO COEFFICIENTS(OUTPUT BUT UNUSED
                     (00011) * SUBSEQUENTLY).
                     (00012) *
                     (00013) * THIS ROUTINEFINDS THE LARGEST AUTOCORELATION IN THE RANGE O
                     (00014) * INTERESTAND STORES IT IN THE PITCH PERIOD.  IT ALSO
                     (00015) * CALCULATES THE PITCH GAIN R(PITCH)/R(0).  FINALLY, IT FINDS
                     (00016) * PARCOR COEFFICIENTS USING THE MWLD FUNCTION.
                     (00017) * DEFINE GLOBAL SYMBOLS
           00000BEB  (00018)        AFDTSORG=$BEB
           000021FC  (00019)        CSPUSNOS=$21FC
           00000794  (00020)        DMYS=$794
           0000001B  (00021)        OOF=$1B
           00000003  (00022)        #M=3
           00000001  (00023)        HS=1
           0000005E  (00024)        LIMITS=D'94'
           00000000  (00025)        MSS=0
           0000000A  (00026)        NPS=D'10'
           0000000F  (00027)        OFFSETS=D'15'
           0000004F  (00028)        RANGES=LIMITS-OFFSETS
           000003B2  (00029)        SVTS=$03B2
           000003D0  (00030)        SAS39=SVTS+2*D'39'
           000003FC  (00031)        SAS61=SVTS+2*D'61'
           000003FE  (00032)        SAS62=SVTS+2*D'62'
           0000042E  (00033)        SAS86=SVTS+2*D'86'
           00006CB0  (00034)        START=$6CB0
           00000002  (00035)        WS=2
           00000003  (00036)        ZS=3
                     (00037) * DEFINE HARD BOUND BUFFERS
           00000800  (00038)        X1RS=D'2048'
           00000814  (00039)        XOMS=D'2068'
                     (00040) *
                     (00041) * EXPAND ARRAY FUNCTION DISPATCH TABLE
           00000BA0  (00042)        #L=AFDTSORG+3*2*(244-128)                FCB 244
00BA0 001E6CB4  (00043)        ADDR MWGXS(R7,1)
00BA2 001E6DB4  (00044)        ADDR G110S(R7,1)
```

```
00BA4 001021FC (00045)          ADDR CSPU$NOS(,1,0)
               (00046)          EJECT
```

```
                    (00047) *
                    (00048) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00049) *
          00006CB0 (00050)          #L=START
                    (00051)          EVEN
06CB0 78000041 (00052) OFFSSET DATA 1F'15,0'
                    (00053)          EJECT
```

```
                        (00054) *
                        (00055) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        (00056) *
                        (00057)           EVEN
     06CB2 0000         (00058)           DATA PITCSSA                     START ADDRESS
     06CB3 0094         (00059)           DATA MWGXSS7                     MODULE SIZE
                        (00060) MWGXS     BEGIN APU(MWGX)
           00000000     (00061)           *A=0
A00  06CB4 08EC0000     (00062) PITCSSA   MOV(IOA,OO)\NOP                  R(I)=X1R(I)
A01  06CB6 90160000     (00063)           JUMPC(PITCSSA,FWI)               FOR I=0,1,2...LPCN+1
A02  06CB8 20370000     (00064)           CLEAR(WI)\NOP                    RELEASE APS INPUT
                        (00065) *
A03  06CBA 08D31680     (00066) EXTRCTS   MOV(IO,A3)\K(1)                  OFFSET;R=1.0
A04  06CBC 000008F3     (00067)           NOP\MOV(IOA,A3)                  RITE OFFSET
A05  06CBE 08E70894     (00068)           MOV(IOA,A7)\MOV(R,A4)            A7=UO;A4=1.0
A06  06C90 08160896     (00069)           MOV(ZERO,A6)\MOV(R,A6)           A6=0;A6=1 L;R LOC.
A07  06C92 16A016A0     (00070)           K(2)                             R=2.0
A08  06C94 000008E7     (00071)           NOP\MOV(IOA,A7)                  A7=U1
A09  06C96 089502F5     (00072)           MOV(R,A5)\MOV(A5),R(A7)          A5=2;A5=2,R=U1
A0A  06C98 91160014     (00073)           JUMPS(MXS1,FWI)                  FOR I=0,1,2...LTH-1
A0B  06C9A 08F00000     (00074) #1        MOV(IOA,A0)\NOP                  A0=UE
A0C  06C9C 60F40897     (00075)           MOV(A4),MAX(A7,A0)\MOV(R,A7)     A4=LOC,R=MAX\A7=MAX
A0D  06C9E B11F0028     (00076)           CALLS(MXS4,T2)                   IF NEW MAX
A0E  06CA0 000044A0     (00077)           NOP\ADD(A5,A4)                   R=LOC
A0F  06CA2 91160019     (00078)           JUMPS(MXS2,FWI)                  EXIT IF UBS IS ODD
A10  06CA4 08970BE0     (00079)           MOV(R,A7)\MOV(IOA,A0)            A7=MAX;A0=UO
A11  06CA6 B11F0025     (00080)           CALLS(MXS3,T1)                   IF NEW MAX
A12  06CA8 44A060F4     (00081)           ADD(A5,A4)\MOV(A4),MAX(A7,A0)    R=LOC;A4=LOC,R=MAX
A13  06CAA 9016000B     (00082)           JUMPC(#1,FWI)                    EXIT IF UBS IS EVEN
A14  06CAC 02F00000     (00083) MXS1      R(A7)\NOP                        R=MAX
A15  06CAE 08910898     (00084)           MOV(R,A1)\MOV(R,EXO)             A1=MAX;EXO=MAX
A16  06CB0 B11F0028     (00085)           CALLS(MXS4,T2)                   IF NEW MAX
A17  06CB2 08520000     (00086)           MOV(EXI,A2)\NOP                  A2=MAX
A18  06CB4 1000001D     (00087)           JUMP(MXS5)                       TO COMMON CLEAN-UP
A19  06CB6 000002E0     (00088) MXS2      NOP\R(A7)                        R=MAX
A1A  06CB8 08910898     (00089)           MOV(R,A1)\MOV(R,EXO)             A1=MAX;EXO=MAX
A1B  06CBA B11F0025     (00090)           CALLS(MXS3,T1)                   IF NEW MAX
A1C  06CBC 08520000     (00091)           MOV(EXI,A2)\NOP                  A2=MAX
A1D  06CBE 622043C0     (00092) MXS5      MAX(A1,A2)\ADD(A6,A3)            R=MAX;R=LOC(MAX)+OFFSET
A1E  06CC0 089C0000     (00093)           MOV(R,OO)\NOP                    OO=SA=MAX
A1F  06CC2 911F0023     (00094)           JUMPS(MXS6,T1)                   IF FROM RIGHT BOARD
A20  06CC4 43C00000     (00095)           ADD(A6,A3)\NOP                   R=LOC(MAX)+OFFSET
A21  06CC6 089C0000     (00096)           MOV(R,OO)\NOP                    OO=SB=LOC(MAX)
A22  06CC8 1000003B     (00097)           JUMP(INVSR)                      NOW FORM R'(0)
```

```
A23 06CCA 0000089C (00098) MXS6    NOP\MOV(R,00)                      00=SB=LOC(MAX)
A24 06CCC 10000038 (00099)         JUMP(INVSR)                        NOW FORM 1.0/R(0)
                   (00100) *
                   (00101) *  SUBROUTINES:MXS3,MXS4
A25 06CCE 02800000 (00102) MXS3    R(A4)\NOP                          R=LOC
A26 06CD0 08960000 (00103)         MOV(R,A6)\NOP                      A6=LOC
A27 06CD2 A000A000 (00104)         RETURN
A28 06CD4 00000280 (00105) MXS4    NOP\R(A4)                          R=LOC
A29 06CD6 00000896 (00106)         NOP\MOV(R,A6)                      A6=LOC
A2A 06CD8 A000A000 (00107)         RETURN
                   (00108) *
                   (00109) *  INVERSE SUBR
                   (00110) *RECIP(X) IN M1, X IN M5.  RETURNS 1/X IN P.
A2B 06CDA 84A084A0 (00111) INV:    MUL(M1,M5)
A2C 06CDC 16A016A0 (00112)         K(2)
A2D 06CDE 08B008B0 (00113)         MOV(P,A0)                          X/X'
A2E 06CE0 48314831 (00114)         MOV(A1),SUB(A1,A0)                 2-X/X'
A2F 06CE2 08BE08BE (00115)         MOV(R,M6)
A30 06CE4 84C084C0 (00116)         MUL(M1,M6)                         (2-X/X')/X'
A31 06CE6 08A908A9 (00117)         MOV(P,M1)                          (2-X/X")/X'
A32 06CE8 84848484 (00118)         MOV(A4),MUL(M1,M5)                 (2-X/X')*X/X'
A33 06CEA 08B008B0 (00119)         MOV(P,A0)
A34 06CEC 48204820 (00120)         SUB(A1,A0)                         2-((2-X/X')*X/X')
A35 06CEE 08BE08BE (00121)         MOV(R,M6)
A36 06CF0 84C084C0 (00122)         MUL(M1,M6)                         ((2-X/X')/X')*(2-((2-X/X
                   (00123) :                                          ')*X/X'))
A37 06CF2 A000A000 (00124)         RETURN
A38 06CF4 20370000 (00125) INVSR   CLEAR(WI)\NOP                      RELEASE APS INPUT
A39 06CF6 08D00000 (00126)         MOV(IO,A0)\NOP                     R(0)
A3A 06CF8 2F000000 (00127)         RCP(A0)\NOP
A3B 06CFA 08FD0000 (00128)         MOV(IOA,M5)\NOP
A3C 06CFC 08890889 (00129)         MOV(R,M1)                          1/X'
A3D 06CFE 3000002B (00130)         CALL(INV)                          GET REAL INVERSE OF R(0)
A3E 06D00 08FD0000 (00131)         MOV(IOA,M5)\NOP                    M5=R(MAX)
A3F 06D02 08A80000 (00132)         MOV(P,M0)\NOP                      M0=1.0/R0
A40 06D04 84200000 (00133)         MUL(M0,M5)\NOP                     R(MAX)/R(0)
A41 06D06 08BC0000 (00134)         MOV(P,00)\NOP                      00=R'(MAX)
                   (00135) *
                   (00136) *  XOM(I)=FLOAT(NIN(NTUPS-NP+I)     I=0,1,2...NP-1
A42 06D08 08FF08FF (00137)         MOV(IOA,M7)                        M7=SA=2**15
A43 06D0A 08F20000 (00138) FLTS    MOV(IOA,A2)\NOP                    A2=UE
A44 06D0C 000008F2 (00139)         NOP\MOV(IOA,A2)                    A2=U0
A45 06D0E 32403240 (00140)         NORM(A2)                           NRM<UE,E0>
A46 06D10 08880888 (00141)         MOV(R,M0)                          M0=NRM<UE>;M0=NRM<U0>
```

```
A47 06D12 847C847C (00142)          MOV(OQ),MUL(MO,M7)              OO=SA*NRM<UE(I-1),UO(I-1
                   (00143) ;                                        )>,SA*NRM<UE,UO>
A48 06D14 90160043 (00144)          JUMPC(FLTS,F=1)                 LOOP FOR I=0,1...NP-1
A49 06D16 08BC08BC (00145)          MOV(P,OO)
A4A 06D18 901C004A (00146) #1       JUMPC(#1,FO)                    WAIT FOR OUTPUT TO CLEAR
A4B 06D1A 20372037 (00147)          CLEAR(WI)                       LET APS CONTINUE
                   (00148) *
                   (00149) * GO ON TO MWLD PORTION OF FUNCTION.
                   (00150)          EJECT
```

```
                          *
                          *      MWLD-APU PROGRAM
                          *
                          *
                          * WEINER LEVINSON DURBAN INVERSE
                          *
                          * MATHEMATICS
                          * SEE J. MAKHOUL,LINEAR PREDICTION REVIEW
                          * PROC. IEEE, VOL 63, PP566, APR 1975
                          *
                          * BUFFER DEFINITIONS
                          *
                          * V(K), CORRELATION COEFFICIENTS, V(K)=R(K)
                          * V(K) MUST BE COMPACT 32 BIT FLOATING POINT
                          * VSIZE NOT USED IN COMPUTATION, BUT
                          * VSIZE > USIZE FOR VALID RESULTS.
                          *
                          * U(K),A(K) COEFFICIENTS
                          * U(K),MUST BE COMPACT 32 BIT FLOATING POINT
                          * U(K)=A(K-1),IE,A(0)=1 NOT IN BUFFER
                          *
                          * Y(K), PARTIAL CORRELATIONS AND ERROR
                          * Y(K) MUST BE COMPACT 32 BIT FLOATING POINT
                          * YSIZE=2*USIZE
                          * Y(K)=P(K-1), 0<=K<USIZE
                          * Y(K+USIZE)=F(K-1), USIZE<=K<YSIZE
                          *
                          * A,STABILITY TEST PARAMETER
                          * IF /P(N)/> CONTENTS OF (A), THEN
                          * FOR J>0
                          * A(N+J)=P(N+J)=0, F(N+J)=F(N)
                          *
                          *
                          *
A4C 06D1C 08F70000        MWLDSSA MOV(IOA,A7)\NOP            A7=STABILITY TEST VALUE
A4D 06D1E 080C080C                MOV(ZERO,M4)
A4E 06D20 08CD0000                MOV(IO,M5)\NOP
A4F 06D22 08F50000                MOV(IOA,A5)\NOP            M5=A5=E(0)=R(0)
                          *
                          * MWLD-APU CALCULATION OF S(N)
                          * S(N)=P(N) + SUM[1,N-1] OF R(N-J)A(N-1,J)
                          * REGISTER CONTENTS AT START
                          * M4=P(N-1), M5=A5=F(N-1), A6=1, A7=TEST
                          *
```

```
A50 06D24 2EA02EA0 (00195) #1        RCP(A5)
A51 06D26 08F20000 (00196)          MOV(IOA,A2)\NOP           A2=R(N)
A52 06D28 08080808 (00197)          MOV(ZERO,M0)
A53 06D2A 84008400 (00198)          MUL(M0,M4)                 SETS P = 0
A54 06D2C 02490249 (00199)          MOV(M1), R(A2)            M1=RCP(E(N-1))
A55 06D2E 900A0090 (00200)          JUMPC(MWLDRW,AF2)         FIRST TIME
                   (00201) *
A56 06D30 9048005C (00202)          JUMPC(MWLDSE,AF3),SET     EVEN OR ODD N
                   (00203) *
A57 06D32 2028202B (00204)          CLEAR(AF3)
                   (00205) *
A58 06D34 08FA0000 (00206) #2       MOV(IOA,M2)\NOP           M2=A(N-1,J)
A59 06D36 08EE0000 (00207)          MOV(IOA,M6)\NOP           M6=R(N-J)
A5A 06D38 85508550 (00208)          MOV(A0), MUL(M2,M6)
A5B 06D3A 42124212 (00209)          MOV(A2), ADD(A0,A2)       S+A(N-1,J+1)R(N-J-1)
                   (00210) *
A5C 06D3C 08EB0000 (00211) MWLDSE   MOV(IOA,M3)\NOP           M3=A(N-1,J+1)
A5D 06D3E 08EF0000 (00212)          MOV(IOA,M7)\NOP           M7=R(N-J-1)
A5E 06D40 85F185F1 (00213)          MOV(A1), MUL(M3,M7)
A5F 06D42 42324232 (00214)          MOV(A2), ADD(A1,A2)       S+A(N-1,J)R(N-J)
                   (00215) *
A60 06D44 90160058 (00216)          JUMPC(#2,EW1)
                   (00217) *
A61 06D46 08800880 (00218)          MOV(P,A0)
A62 06D48 42124212 (00219)          MOV(A2), ADD(A0,A2)
                   (00220) *
A63 06D4A 20372037 (00221) MWLDRF   CLEAR(W1)
                   (00222) *
                   (00223) *
                   (00224) * MWLD-APU CALCULATION OF P(N)
                   (00225) * P(N)=-S(N)/E(N-1)
                   (00226) * REGISTER CONTENTS AT START
                   (00227) * M4=P(N-1), A5=M5=E(N-1), A6=1, A7=TEST
                   (00228) * M1=RCP[E(N-1)], R=S(N)
                   (00229) *
                   (00230) *
A64 06D4C 08AA0000 (00231)          MOV(R,M2)\NOP                                  S(N)
A65 06D4E 3000002B (00232)          CALL(INV)                                      GET 1.0/E(N-1)
A66 06D50 08AE08AE (00233)          MOV(P,M6)                                      1.0/E
A67 06D52 A540A540 (00234)          MULN(M2,M6)
A68 06D54 16801680 (00235)          R(1)
A69 06D56 08960896 (00236)          MOV(R,A6)                                      FOLLOWING CODE NEEDS A6=
                   (00237) ;                                                       1
A6A 06D58 08A808A8 (00238)          MOV(P,M0)                 M0=M4=A4=P(N)
```

```
A6B 06D5A 08AC08AC (00239)         MOV(P,M4)
A6C 06D5C 08B408B4 (00240)         MOV(P,A4)
A6D 06D5E 08BC0000 (00241)         MOV(P,OO)\NOP              OO = P(N) = A(N,N)
                   (00242) *
                   (00243) *
                   (00244) *
                   (00245) * MWLD-APU CALCULATION OF A(N,J)
                   (00246) * A(N-1,J)=A(J)
                   (00247) * A(N,J)=A(J)+P(N)A(N-J)
                   (00248) * A(N,N-J)=A(N-J)+P(N)A(J)
                   (00249) * LOOP COMPUTES BOTH A(N,J) AND A(N,N-J)
                   (00250) * FOR N=2M, A(N,M) COMPUTED TWICE
                   (00251) *
                   (00252) * REGISTER CONTENTS AT START
                   (00253) * M0=M4=A4=P=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                   (00254) *
                   (00255) * DATA SEQUENCE
                   (00256) * INPT, [A(N-J+1),A(N-J),A(J)] FOR [1,N/2],WI, A(N/2+1)
                   (00257) * OTPT, DUMP, DUMP [A(N,J), A(N,N-J)]
                   (00258) *
                   (00259) *
A6E 06D60 904A0092 (00260)         JUMPC(MWLDAW,AF2), SET      FIRST TIME AF2 CLEAR
                   (00261) *
A6F 06D62 08F30000 (00262)         MOV(IOA,A3)\NOP            A3=A(N-J+1)
                   (00263) *
A70 06D64 08FA0000 (00264) #3      MOV(IOA,M2)\NOP            M2=A(N-J)
A71 06D66 85118511 (00265)         MOV(A1), MUL(M2,M4)
A72 06D68 433C0000 (00266)         MOV(OO), ADD(A1,A3)\NOP    OO=A(J-1)+P(N)A(N-J+1)
                   (00267) *                                   A(N-J+1)+P(N)A(J-1)
A73 06D6A 08D20000 (00268)         MOV(IO,A2)\NOP             A2=M3=A(J)
A74 06D6C 08FB0000 (00269)         MOV(IOA,M3)\NOP
A75 06D6E 85908590 (00270)         MOV(A0), MUL(M3,M4)
A76 06D70 421C0000 (00271)         MOV(OO), ADD(A0,A2)\NOP    OO=A(N-J+1)+P(N)A(J-1)
A77 06D72 08F30000 (00272)         MOV(IOA,A3)\NOP
A78 06D74 00000000 (00273)         NOP
A79 06D76 90160070 (00274)         JUMPC(#3,FWI)
                   (00275) *
A7A 06D78 20372037 (00276) MWLDAE  CLEAR(WI)
A7B 06D7A 84118411 (00277)         MOV(A1), MUL(M0,M4)
A7C 06D7C 433C0000 (00278)         MOV(OO), ADD(A1,A3)\NOP
                   (00279) *
                   (00280) *
                   (00281) * MWLD-APU CALCULATION OF E(N)
                   (00282) * E(N)=[1-P(N)P(N)]E(N-1)
```

```
                          (00283) *
                          (00284) * REGISTER CONTENTS AT START
                          (00285) *
                          (00286) * M0=M4=A4=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                          (00287) * P=P(N)P(N),R=P(N)=A(N,N)
                          (00288) * DATA SEQUENCE
                          (00289) * OTPT  P(N), E(N)
A7D 06D7E 08800880 (00290)         MOV(P,A0)                    A0=P(N)P(N)
A7E 06D80 48DC0000 (00291)         MOV(QQ), SUB(A6,A0)\NOP      QQ=A(N,N)
A7F 06D82 028A028A (00292)         MOV(M2), R(A4)               M2=1-P(N)P(N)
A80 06D84 85208520 (00293)         MUL(M2,M5)                   E(N-1)* [1-P(N)P(N)]
A81 06D86 74FC0000 (00294)         MOV(QQ), MAXABS(A7,A4)\NOP      QQ=P(N)
A82 06D88 08800880 (00295)         MOV(R,NULL)                  T1 SET IF /P(N)/>TEST
A83 06D8A 086C0000 (00296)         MOV(P,QQ)\NOP                QQ=E(N)
A84 06D8C 20892089 (00297)         SETTG(AF1)                   AF1 SET IF /P(N)/>TEST
A85 06D8E 08850885 (00298)         MOV(P,A5)
A86 06D90 08AD08AD (00299)         MOV(P,M5)
A87 06D92 20482048 (00300)         SET(AF0)                     LET APS/OTPT GO
A88 06D94 911D00RF (00301)         JUMPS(MWLDD,FI)                 GO BACK UNLESS DONE
                   (00302) *
A89 06D96 901E0050 (00303)         JUMPC(#1,T1)
                   (00304) *
                   (00305) *
                   (00306) * MWLD-APU ABORT ROUTINE
                   (00307) * P REGISTER=E(N)
                   (00308) *
                   (00309) *
A8A 06D98 081C0000 (00310) #4      MOV(ZERO,QQ)\NOP             A(N+J)=0
A8B 06D9A 081C0000 (00311)         MOV(ZERO,QQ)\NOP             P(N+J)=0
A8C 06D9C 088C0000 (00312)         MOV(P,QQ)\NOP                E(N+J)=E(N)
A8D 06D9E 901C008A (00313)         JUMPC(#4,F0)
                   (00314) *
A8E 06DA0 20322032 (00315) MWLDD   CLEAR(RA)
A8F 06DA2 1000004C (00316)         JUMP(MWLD$SA)
A90 06DA4 90160090 (00317) MWLDRW  JUMPC(MWLDRW,FWI)
A91 06DA6 10000063 (00318)         JUMP(MWLDRE)
A92 06DA8 08F00000 (00319) MWLDAW  MOV(IOA,NULL)\NOP
A93 06DAA 1000007A (00320)         JUMP(MWLDAE)
                   (00321) *
          00000094 (00322)         MWGX$SZ=#A-P1TC$SA
    06DAC          (00323)         END
                   (00324) *
                   (00325) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                   (00326) *
```

```
                    (00327)         EVEN
    06DAC 00006F82  (00328)         ADDR MWGXSI                    POINTER TO CONSTRUCTED I
                    (00329) ;                                      NSTRUCTION BLOCK
    06DAE 00006F20  (00330)         ADDR G110S+2*MWGXSS            POINTER TO SCALAR BLOCK
    06DB0 0000      (00331)         DATA 0                         NUMBER OF SCALARS
    06DB1 0110      (00332)         DATA MWGXSZ                    MODULE SIZE
    06DB2 00006F9C  (00333)         ADDR MWGXSA                    POINTER TO CHAIN ANCHOR
                    (00334)         EVEN                           END ON WORD BOUNDARY
                    (00335) *
                    (00336) G110S   BEGIN APS(G110)
A00 06DB4 00202440  (00337)         JSN(G110S0,P2)                 SET OUTPUT PC(P2)
A01 06DB6 02300030  (00338)         SET(RO)                        ENABLE OUTPUT ADDRESSING
A02 06DB8 04000020  (00339)         NOP
A03 06DBA 06460800  (00340)         LOAD(BR0,X1RS(3))              X1R(.) BA
A04 06DBC 08020004  (00341)         SUB(BR0,4)                     X1R(.) BA-4
A05 06DBE 0A600000  (00342)         LOAD(BR2,(0))                  DUMMY OP FOR EXEC
A06 06DC0 0C500000  (00343)         LOAD(BR1,MSS)                  R(.) SIZE-1
A07 06DC2 0E220000  (00344)         SUB(BR2,MSS)                   DUMMY OP FOR EXEC
A08 06DC4 10BA0004  (00345) #1      ADD(BR0,4,TF)                  IQ=X1R(I)
A09 06DC6 121908B1  (00346)         SUBL(BR1,1),JUMPP(#1)          FOR I=0,1,2...LPCN+1
A0A 06DC8 14300037  (00347)         SET(WI)                        STALL APS INPUT
A0B 06DCA 16000020  (00348)         NOP                            RELEASED BY APU
                    (00349) *
A0C 06DCC 18460800  (00350)         LOAD(BR0,X1RS(3))              X1R(.) BA
A0D 06DCE 1A020004  (00351)         SUB(BR0,4)                     X1R(.) BA-4
A0E 06DD0 1C390011  (00352)         MOVB(BR3,BR0)                  SAVE X1R(.) BA-4
A0F 06DD2 1E0A003C  (00353)         ADD(BR0,4*OFFSETS)             ADD OFFSET TO X1R(.) BA
A10 06DD4 2050004F  (00354)         LOAD(BR1,RANGES)               X1R(.) SEARCH RANGE
A11 06DD6 22F26C80  (00355)         LOAD(BR2,OFFSSET(1),TF)        IQ=OFFSET
A12 06DD8 24BA0004  (00356) #2      ADD(BR0,4,TF)                  IQ=X1R(I)
A13 06DDA 261912B1  (00357)         SUBL(BR1,1),JUMPP(#2)          FOR I=15,16,...,94
A14 06DDC 28300037  (00358)         SET(WI)                        STALL APS INPUT
A15 06DDE 2A000020  (00359)         NOP                            RELEASED BY APU
                    (00360) *
A16 06DE0 2CBA0004  (00361) INVS1   ADD(BR3,4,TF)                  IQ=X1R(0)
A17 06DE2 2EC203FC  (00362)         LOAD(BR0,SAS61(1),TF)          IQ=SAS61=X1R(MAX)
                    (00363) *
A18 06DE4 30C203D0  (00364) IFLTS   LOAD(BR0,SAS39(1),TF)          IQ=2**15
A19 06DE6 32401028  (00365)         LOAD(BR0,(I1))                 NIN(.) BA
A1A 06DE8 34500000  (00366)         LOAD(BR1,MSS)                  NIN(.) SIZE-1=NTUPS-1
A1B 06DEA 36220000  (00367)         SUB(BR2,MSS)                   DUMMY OP FOR EXEC
A1C 06DEC 3B09002B  (00368)         ADDB(BR0,BR1)                  NIN(.) BA+NTUPS-1
A1D 06DEE 3A02000A  (00369)         SUB(BR0,NPS)                   NIN(.) BA+NTUPS-1-NP
A1E 06DF0 3C600009  (00370)         LOAD(BR2,NPS-1)                MOVE SIZE
```

```
A1F 06DE2 3E0A0001 (00371) IFLTST  ADD(BRO,MS,TE)                               IQ=NIN(1)
A20 06DE4 40291EB1 (00372)         SUBL(BR2,1),JUMPP(IFLTST)                    FOR I=NTUPS-NP+1...NTUPS
                   (00373) ;                                                    -1
A21 06DE6 42300037 (00374)         SET(W1)                                      STALL
A22 06DE8 44000020 (00375)         NOP
                   (00376) *
A23 06DEA 46003660 (00377) DNEST   JUMP(MWGXSS)                                 GO DO MWLD PORTION
                   (00378) *
A24 06DEC 48300032 (00379) G110SO  SET(RA)                                      ENABLE APU
A25 06DEE 4A40001B (00380)         LOAD(BW0,[01)                                R(,) BA
A26 06E00 4C500000 (00381)         LOAD(BW1,MSS)                                R(,) SIZE-1
A27 06E02 4E020000 (00382)         SUB(BW0,MSS)                                 R(,) BA-2
A28 06E04 50BAB006 (00383) #1      ADD(BW0,[B1,TE)                              OQ=R(I)
A29 06E06 52112BB1 (00384)         SUBL(BW1,1),JUMPP(#1)                        FOR I=0,1,2...LPCN+1
                   (00385) *
A2A 06E08 54C203FC (00386) SCLRS   LOAD(BW0,SAS61(1),TE)                        OQ=X1R(MAX)
A2B 06E0A 56C203FE (00387)         LOAD(BW0,SAS62(1),TE)                        OQ=M
                   (00388) *
A2C 06E0C 58C203FC (00389) INVSO   LOAD(BW0,SAS61(1),TE)                        OQ=X1R'(MAX)
                   (00390) *
A2D 06E0E 5AC20794 (00391) OFLTS   LOAD(BW0,DMYS(1),TE)                         OQ=?
A2E 06E10 5CC20794 (00392)         LOAD(BW0,DMYS(1),TE)                         OQ=?
A2F 06E12 5E440814 (00393)         LOAD(BW0,XOMS(2))                            XOM(,) BA
A30 06E14 60500009 (00394)         LOAD(BW1,NPS-1)                              XOM(,) SIZE-1
A31 06E16 62020002 (00395)         SUB(BW0,2)                                   XOM(,) BA-2
A32 06E18 64BA0002 (00396) OFLTSJ  ADD(BW0,WS,TE)                               OQ=XOM(J)
A33 06E1A 661132B1 (00397)         SUBL(BW1,1),JUMPP(OFLTSJ)                    FOR J=0,1,....NP-1
                   (00398) *
A34 06E1C 68200030 (00399) DNESO   CLEAR(RO)                                    OUTPUT DONE!
A35 06E1E 6A000020 (00400)         NOP
                   (00401) *
                   (00402) *
                   (00403) *
                   (00404) *
                   (00405) *
                   (00406) * WEINER - APS INITIALIZATION ROUTINES
                   (00407) *
A36 06E20 6CE2042F (00408) MWGXSS  LOAD(BR2,SAS86(1),L,TE)        INPUT STABILITY TEST
                   (00409) *
A37 06E22 6EC0001F (00410)         LOAD(BRO,[01,L,TE)            INPT R(0)
A38 06E24 70500000 (00411)         LOAD(BR1,MSS)                 NOT USED
A39 06E26 720A0000 (00412)         ADD(BRO,MSS)                  BRO=R(1)
A3A 06E28 74500000 (00413)         LOAD(BR1,0)
A3B 06E2A 76010013 (00414)         MOVB(BW0,BR1,                 BW0=0=2(N-1)
```

```
A3C 06F2C 7B70300A (00415)          LOAD(BR3,(3),L)              BR3=A(1)
A3D 06F2E 7A500000 (00416)          LOAD(BR1,MSS)                BR1=ASIZE-1, NOT USED
A3E 06F30 7C320000 (00417)          SUB(BR3,MSS)                 BR3=A(0)
                   (00418) *
                   (00419) *
                   (00420) * MWLD = APS S(N) INPUT
                   (00421) * REGISTER CONTENTS AT START
                   (00422) * BR0=R(1), BW0=2(N-1), BR3=A(0)=AB-2
                   (00423) *
                   (00424) *
A3F 06F32 7FB9002B (00425) #1       ADDB(BR0,BW0,TF)             INPT R(N)
A40 06F34 80190010 (00426)          MOVB(BR1,BW0)                BR1=2(N-1)
A41 06F36 82290017 (00427)          MOVB(BR2,BR3)
A42 06F38 84194BF2 (00428)          SUBL(BR1,2),JUMPN(MWLSC)
A43 06F3A 86004560 (00429)          JUMP(MWLSF)
                   (00430) *
A44 06F3C 8FB20002 (00431) #2       SUB(BR0,2,TF)                INPT R(N-J)
                   (00432) *
A45 06F3E 8AAA0002 (00433) MWLSF    ADD(BR2,2,TF)               INPT A(J)
A46 06F40 8C1944B2 (00434)          SUBL(BR1,2), JUMPP(#2)
                   (00435) *
A47 06F42 8FB94B72 (00436)          SUBL(BR0,2,TF),JUMP(MWLSC)   INPT R(1)
                   (00437) *
A48 06F44 90300037 (00438) MWLSC    SET(WI)
A49 06F46 92000020 (00439)          NOP(0)
A4A 06F48 9401003A (00440)          ADDL(BW0,2)                  BW0=2N
                   (00441) *
                   (00442) *
                   (00443) *
                   (00444) * MWLD = APS  A(N,J)  CALCULATION INPT
                   (00445) * REGISTER CONTENTS AT START
                   (00446) * BR0=R(1), BW0=2N
                   (00447) * BR2=A(N-1), BR3=A(0)=AB-2
                   (00448) *
                   (00449) *
A4B 06F4A 962A0002 (00450)          ADD(BR2,2)                   BR2=A(N)
A4C 06F4C 98210015 (00451)          MOVB(BW2,BR2)
A4D 06F4F 9A205C40 (00452)          JSN(MWLOT,P2)                SET UP OUTPUT
                   (00453) *
A4E 06F50 9C195250 (00454)          MOVB(BR1,BW0), JUMP(MWLAE)   BR1=2N
                   (00455) *
A4F 06F52 9FAA0000 (00456) #3       ADD(BR2,0,TF)                A(N-1,N-J+1)INPT
A50 06F54 A0A20002 (00457)          SUB(BR2,2,TF)                A(N-1,N-J)INPT
A51 06F56 A2BA0002 (00458)          ADD(BR3,2,TF)                A(N-1,J)INPT
```

```
                       (00459) *
A52 06E58 A4194EB4 (00460) MWLAF   SUBL(BR1,4), JUMPP(#3)
                       (00461) *
A53 06E5A A670302F (00462) MWLT    LOAD(BR3,[31,L)             BR3=A(1)=AB
A54 06E5C AB500000 (00463)         LOAD(BR1,MSS)               ASIZE=N,DONE
A55 06E5E AA320000 (00464)         SUB(BR3,MSS)                BR3=A(0)=AB-2
A56 06E60 AC19002B (00465)         ADDB(BR1,BR1)               BR1=2N-2, DONE
A57 06E62 AEA95B7B (00466)         ADDL(BR2,0,TF), JUMP(#A+1)             CLEANUP OF A(N) CALC*
A58 06E64 B0300037 (00467)         SET(WI)
A59 06E66 B2000020 (00468)         NOP(0)
A5A 06E68 B4193EA0 (00469)         SUBB(BR1,BW0), JUMPP(#1)
A5B 06E6A B62059F1 (00470)         JUMPS(#A-2,RI),CLEAR
                       (00471) *
                       (00472) *
                       (00473) * MWLD - APS OUTPUT
                       (00474) *
                       (00475) * REGISTER CONTENTS AT START
                       (00476) * BW0=2N, BW2=A(N)
                       (00477) *
                       (00478) *
A5C 06E6C B8300030 (00479) MWLOT   SET(RO)
A5D 06E6E BAB10014 (00480)         MOVB(BW3,BW2,TF)            BW3=A(N), OUTPUT
A5E 06E70 BC210020 (00481)         SUBB(BW2,BW0)               BW2=A(0)
A5F 06E72 BED20794 (00482)         LOAD(BW1,DMYS(1),TF)       OTPT DUMP
A60 06E74 C091003B (00483)         ADDL(BW1,0,TF)              OTPT DUMP
A61 06E76 C2116450 (00484)         MOVB(BW1,BW0), JUMP(MWLOE)  BW1=2N
                       (00485) *
A62 06E78 C4AA0002 (00486) #4      ADD(BW2,2,TF)               OTPT A(N,J)
A63 06E7A C6B20002 (00487)         SUB(BW3,2,TF)               OTPT A(N,N-J)
                       (00488) *
A64 06E7C C8116284 (00489) MWLOE   SUBL(BW1,4),JUMPP(#4)
A65 06E7E CA602024 (00490)         LOAD(BW2,[21,L)             BW2=P(1)=PB
A66 06E80 CC500000 (00491)         LOAD(BW1,MSS)
A67 06E82 CE220000 (00492)         SUB(BW2,MSS)                BW2=P(0)=PB-2
A68 06E84 D01A0001 (00493)         ADD(BW1,1)                  BW1=PSIZE=2ASIZE
A69 06E86 D2A1002B (00494)         ADDB(BW2,BW0,TF)            OTPT P(N)
A6A 06E88 D4A1002A (00495)         ADDB(BW2,BW1,TF)            OTPT E(N)
                       (00496) *
A6B 06E8A D6000020 (00497) #5      NOP(0)
A6C 06E8C D8006BAB (00498)         JUMPC(#5,AF0)               WAIT ON APU
                       (00499) *
A6D 06E8E DA206E6B (00500)         JUMP(#A+1,AF0),CLEAR
A6E 06E90 DC007DA9 (00501)         JUMPC(MWLOD,AF1)            JUMP TO DONE IF NO ABORT
                       (00502) *
```

```
A6F 06F92 DF200031 (00503)          CLEAR(R1)
A70 06F94 F0390010 (00504)          MOVB(BR3,BW0)           BR3 = 2N
A71 06F96 F2290010 (00505)          MOVB(BR2,BW0)           BR2 = 2N
A72 06F98 F4390022 (00506)          SUBB(BR3,BW1)           BR3=2N-ASIZE
A73 06F9A F6390031 (00507)          SUBL(BR3,1)             BR3=2N-ASIZE-1
                   (00508) *
A74 06F9C F840301F (00509)          LOAD(BW0,131)           BW0 = A(1)
A75 06F9E FA700000 (00510)          LOAD(BW3,MSS)           NOT USED
A76 06FA0 FC020000 (00511)          SUB(BW0,MSS)            BW0 = A(0)
A77 06FA2 FF017C6D (00512)          ADDB(BW0,BR2),JUMP(MWLOC)   BW0 = A(N)
                   (00513) *
A78 06FA4 F0BA0002 (00514) #6       ADD(BW0,2,TF)           OTPT A(N+J)=0
A79 06FA6 F2210022 (00515)          SUBB(BW2,BW1)           BW2=P(N+J-1)
A7A 06FA8 F4AA0002 (00516)          ADD(BW2,2,TF)           OTPT P(N+J)=0
A7B 06FAA F6A1002A (00517)          ADDB(BW2,BW1,TF)        OTPT E(N+J)=E(N)
A7C 06FAC F83978FA (00518) MWLOC    ADDL(BR3,2), JUMPN(#6)
                   (00519) *
A7D 06FAE FA200030 (00520) MWLOD    CLEAR(R0)
A7E 06FB0 FC000020 (00521)          NOP
                   (00522) *
          00006F9C (00523)          MWGXSA=#C                                   ASSIGN VALUE TO CHAIN AN
                   (00524) ;                                                    CHOR
    06FB2          (00525)          END #A-1
                   (00526) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    06FB2 00000000 (00527) MWGXSI   DATA 9F'0,0'
     ...
          00000110 (00528)          MWGXSZ=#L-G110S
    06FC4          (00529)          END
```

```
AFDTSORG:  008FB (00018) (00042)
CSPUSNOS:  021FC (00019) (00045)
DMYS:      00794 (00020) (00391) (00392) (00482)
DNFS1:     00023 (00377)
DNFS0:     00034 (00399)
EXTRCTS:   00003 (00066)
FLTS:      00043 (00138) (00144)
G110S:     06D84 (00044) (00330) (00336) (00528)
G110S0:    00024 (00337) (00379)
HS:        00001 (00023) (00371)
IFLTS:     0001B (00364)
IFLTS1:    0001F (00371) (00372)
INV:       0002B (00111) (00130) (00232)
INVS1:     00016 (00361)
INVS0:     0002C (00389)
INVSP:     00038 (00097) (00099) (00125)
LIMITS:    0005F (00024) (00028)
MSS:       00000 (00025) (00343) (00344) (00366) (00367) (00381) (00382) (00411) (00412) (00416)
                 (00417) (00463) (00464) (00491) (00492) (00510) (00511)
MWGXS:     06C84 (00043) (00060)
MWGXSA:    06F9C (00333) (00523)
MWGXSI:    06F82 (00328) (00527)
MWGXSS:    00036 (00330) (00377) (00408)
MWGXSSZ:   00094 (00059) (00322)
MWGXSZ:    00110 (00332) (00528)
MWLAF:     00052 (00454) (00460)
MWLDSSA:   0004C (00185) (00316)
MWLDAF:    0007A (00276) (00320)
MWLDAW:    00092 (00260) (00319)
MWLDD:     0008F (00301) (00315)
MWLDRF:    00063 (00221) (00318)
MWLDRW:    00090 (00200) (00317) (00317)
MWLDSF:    0005C (00202) (00211)
MWLOC:     0007C (00512) (00518)
MWLOD:     0007D (00501) (00520)
MWLOF:     00064 (00484) (00489)
MWLOT:     0005C (00452) (00479)
MWLSC:     00048 (00428) (00436) (00438)
MWLSF:     00045 (00429) (00433)
MWLT:      00053 (00462)
MXS1:      00014 (00073) (00083)
MXS2:      00019 (00078) (00088)
MXS3:      00025 (00080) (00090) (00102)
MXS4:      00028 (00076) (00085) (00105)
```

```
MXS5:      0001D (00087) (00092)
MXS6:      00023 (00094) (00098)
NPS:       0000A (00026) (00369) (00370) (00394)
OFFSSET:   06C80 (00052) (00355)
OFFSETS:   0000F (00027) (00028) (00353)
OFLTS:     0002D (00391)
OFLTSJ:    00032 (00396) (00397)
OOF:       0001F (00021)
PITCSSA:   00000 (00058) (00062) (00063) (00322)
RANGES:    0004F (00028) (00354)
SAS39:     00300 (00030) (00364)
SAS61:     003FC (00031) (00362) (00386) (00389)
SAS62:     003FE (00032) (00387)
SAS86:     0042F (00033) (00408)
SCLRS:     0002A (00386)
START:     06C80 (00034) (00050)
SVTS:      00382 (00029) (00030) (00031) (00032) (00033)
WS:        00002 (00035) (00396)
XIRS:      00800 (00038) (00340) (00350)
XOMS:      00814 (00039) (00393)
ZS:        00003 (00036)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   F=    0
```

```
                   (00001) '  FCB 245... "MPESTV(U)"  CREATE PITCH & VOCAL TRACT RESPONSE
                   (00002) *                                ORIGINATED:01-OCT-79
                   (00003) *                                UPDATED:09-JUL-80
                   (00004) *  DEFINE GLOBAL SYMBOLS
                   (00005)         OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1F
                   (00006)         OPADD DOF, (3 .LS. 10) + (12 .LS. 5) + $1B
          00000BE8 (00007)         AFDT$ORG=$BE8
          0001FFC0 (00008)         APS$MEM=$1FFC0
          00000007 (00009)         ASIZE$=D'8'-D'1'
          000021FC (00010)         CSPU$NOS=$21FC
          00000794 (00011)         DMY$=$794
          00000003 (00012)         #M=3
          00000001 (00013)         H$=1
          000000FF (00014)         LTHM1$=D'256'-D'1'
          00000000 (00015)         M$=0
          00000000 (00016)         M$$=0
          00000382 (00017)         SVT$=$0382
          000003CF (00018)         SA$38=SVT$+2*D'38'
          0000043A (00019)         SA$92=SVT$+2*D'92'
          00000436 (00020)         SA$90=SVT$+2*D'90'
          00000438 (00021)         SA$91=SVT$+2*D'91'
          00000444 (00022)         SA$97=SVT$+2*D'97'
          00000414 (00023)         SA$73=SVT$+2*D'73'
          000001FF (00024)         SIZE$=D'512'-D'1'
          00006B70 (00025)         START=$6B70
          00000002 (00026)         W$=2
          00000004 (00027)         WW$=4
          00000800 (00028)         X1$=D'2048'
          00000800 (00029)         X1P$=X1$
                   (00030) *
                   (00031) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000BA6 (00032)         #L=AFDT$ORG+3*2*(245-128)                      FCB 245
00BA6 001F6B72 (00033)         ADDR VOCD$(R7,1)
00BA8 001F6C02 (00034)         ADDR G107$(R7,1)
00BAA 001021FC (00035)         ADDR CSPU$NOS(,1,0)
                   (00036)         EJECT
```

```
            (00037) *
            (00038) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
            (00039) *
   00006B70 (00040)         *L=START
            (00041)         EVEN
            (00042)         EJECT
```

```
                        (00043) *
                        (00044) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        (00045) *
                        (00046)            EVEN
     06B70 0000         (00047)            DATA VOCOSSA                                       START ADDRESS
     06B71 0044         (00048)            DATA VOCOSSZ                                       MODULE SIZE
                        (00049) VOCOS      BEGIN APU(VOCO)
           00000000     (00050)            #A=0
A00  06B72 16801680     (00051) VOCOSSA    K(1)
A01  06B74 08E608E0     (00052)            MOV(IOA,A6)\MOV(IOA,A0)                            512=N
A02  06B76 08E008E3     (00053)            MOV(IOA,A0)\MOV(IOA,A3)                            M=PITCH PERIOD
A03  06B78 08910892     (00054)            MOV(R,A1)\MOV(R,A2)                                1.0
A04  06B7A 22944800     (00055)            MOV(A4),INCRE(A4)\SUB(A0,A3)                       1;16\N-M
A05  06B7C 08C80000     (00056)            MOV(IO,M0)\NOP                                     G
A06  06B7E 08CC0000     (00057)            MOV(IO,M4)\NOP                                     G**P,P=1
A07  06B80 49940250     (00058)            MOV(A4),SUB(A4,A1)\MOV(A0),R(A2)                   16;15\N-M;1
A08  06B82 08D50000     (00059)            MOV(IO,A5)\NOP                                     GSUM=G**1
A09  06B84 08EC4A7C     (00060)            MOV(IOA,OO)\MOV(OO),SUB(A3,A2)                     G\1.0;M-1
A0A  06B86 000008E5     (00061)            NOP\MOV(IOA,A5)                                    A1
A0B  06B88 49940AB3     (00062)            MOV(A4),SUB(A4,A1)\MOV(A3),NEG(A5)                 15;14\M-1;-A1
A0C  06B8A 081C08E6     (00063)            MOV(ZERO,OO)\MOV(IOA,A6)                           Z1 OUT\A2
A0D  06B8C 08980ADC     (00064)            MOV(R,EXO)\MOV(OO),NEG(A6)                         14\-A1 OUT;-A2
A0E  06B8E 089408E5     (00065)            MOV(R,A4)\MOV(IOA,A5)                              14\A3
A0F  06B90 081C0851     (00066)            MOV(ZERO,OO)\MOV(EXT,A1)                           Z2 OUT\14
A10  06B92 4C000ABC     (00067)            SUB(A0,A4)\MOV(OO),NEG(A5)                         M-15\-A2 OUT;-A3
A11  06B94 081C08E6     (00068)            MOV(ZERO,OO)\MOV(IOA,A6)                           Z3 OUT\A4
A12  06B96 08900ADC     (00069)            MOV(R,A0)\MOV(OO),NEG(A6)                          M-14\-A3 OUT;-A4
A13  06B98 911E0042     (00070)            JUMPS(NOGD,T1)                                     FOR DEBUGGING
A14  06B9A 840008E5     (00071)            MUL(M0,M4)\MOV(IOA,A5)                             G**2 FOR NEXT ITER\A5
A15  06B9C 081C0ABC     (00072)            MOV(ZERO,OO)\MOV(OO),NEG(A5)                       Z4 OUT\-A4 OUT,-A5
A16  06B9E 081C08E6     (00073)            MOV(ZERO,OO)\MOV(IOA,A6)                           Z5 OUT\A6
A17  06BA0 08920ADC     (00074)            MOV(R,A2)\MOV(OO),NEG(A6)                          M-15=M'\-A5 OUT;-A6
A18  06BA2 081C08E5     (00075)            MOV(ZERO,OO)\MOV(IOA,A5)                           Z6 OUT\A7
A19  06BA4 00000ABC     (00076)            NOP\MOV(OO),NEG(A5)                                -A6 OUT;-A7
A1A  06BA6 081C08E6     (00077)            MOV(ZERO,OO)\MOV(IOA,A6)                           Z7 OUT\A8
A1B  06BA8 00000ADC     (00078)            NOP\MOV(OO),NEG(A6)                                -A7 OUT;-A8
A1C  06BAA 081C0000     (00079)            MOV(ZERO,OO)\NOP                                   Z8 OUT
A1D  06BAC 4900491C     (00080)            SUB(A0,A1)\MOV(OO),SUB(A0,A1)                      M'-1\-A8 OUT;N-M-14
A1E  06BAE 1000002A     (00081)            JUMP(COMM)
A1F  06BB0 08950000     (00082) #1         MOV(R,A5)\NOP                                      UPDATE SUMG
A20  06BB2 081C081C     (00083)            MOV(ZERO,OO)                                       Z1 OUT
A21  06BB4 081C081C     (00084)            MOV(ZERO,OO)                                       Z2 OUT
A22  06BB6 081C081C     (00085)            MOV(ZERO,OO)                                       Z3 OUT
A23  06BB8 84000000     (00086)            MUL(M0,M4)\NOP                                     G**(P+1)
```

```
 A24 06BBA 081C081C (00087)         MOV(ZERO,OO)                                Z4 OUT
 A25 06BBC 081C081C (00088)         MOV(ZERO,OO)                                Z5 OUT
 A26 06BBE 081C081C (00089)         MOV(ZERO,OO)                                Z6 OUT
 A27 06BC0 081C081C (00090)         MOV(ZERO,OO)                                Z7 OUT
 A28 06BC2 49004900 (00091)         SUB(A0,A1)                                  M'-1 FOR LATER TEST\N'-1
                    (00092) ;                                                   4
 A29 06BC4 081C081C (00093)         MOV(ZERO,OO)                                Z8 OUT
 A2A 06BC6 081C081C (00094) COMM    MOV(ZERO,OO)                                Z9 OUT
 A2B 06BC8 081C081C (00095)         MOV(ZERO,OO)                                Z10 OUT
 A2C 06BCA 08AC0000 (00096)         MOV(P,M4)\NOP                               G**P
 A2D 06BCC 081C081C (00097)         MOV(ZERO,OO)                                Z11 OUT
 A2E 06BCE 081C081C (00098)         MOV(ZERO,OO)                                Z12 OUT
 A2F 06BD0 081C081C (00099)         MOV(ZERO,OO)                                Z13 OUT
 A30 06BD2 08B60000 (00100)         MOV(P,A6)\NOP                               G**P, TO ADD TO SUMG
 A31 06BD4 081C081C (00101) #2      MOV(ZERO,OO)                                Z14+ OUT
 A32 06BD6 49104A10 (00102)         MOV(A0),SUB(A0,A1)\MOV(A0),SUB(A0,A2)       M'-1-L;M'-1-L-1\N'-14-L;
                    (00103) ;                                                   N'-15-L
 A33 06BD8 901F0031 (00104)         JUMPC(#2,T1)                                LOOP TIL M'-1-L .LE. 0
                    (00105) *MUST BE TIME FOR PULSE
 A34 06BDA 02400890 (00106)         R(A2)\MOV(R,A0)                             RESTORE M'\COUNT PULSE A
                    (00107) ;                                                   S OUTPUT
 A35 06BDC 08BC081C (00108)         MOV(P,OO)\MOV(ZERO,OO)                      PUT OUT PULSE
 A36 06BDE 46B00000 (00109)         MOV(A0),ADD(A5,A6)\NOP                      M';ADD PULSE TO SUMG
 A37 06BE0 901F001F (00110)         JUMPC(#1,T2)                                LOOP IF WE NEED MORE PUL
                    (00111) ;                                                   SES
 A38 06BE2 08954060 (00112)         MOV(R,A5)\ADD(A3,A0)                        SUMG\N'+M-1 IS .LT. M, #
 A39 06BE4 081C081C (00113) #3      MOV(ZERO,OO)
 A3A 06BE6 00004A10 (00114)         NOP\MOV(A0),SUB(A0,A2)                      SAVE COUNT;DECREMENT
 A3B 06BE8 901F0039 (00115)         JUMPC(#3,T2)                                LOOP UNTIL WE HAVE ENOUG
                    (00116) ;                                                   H ZEROS
 A3C 06BEA 089C0000 (00117)         MOV(R,OO)\NOP                               SUMG OUT
 A3D 06BEC 20322032 (00118)         CLEAR(RA)
 A3E 06BEE 00000000 (00119)         NOP
 A3F 06BF0 00000000 (00120)         NOP
 A40 06BF2 00000000 (00121)         NOP
 A41 06BF4 00000000 (00122)         NOP
 A42 06BF6 10000042 (00123) NOGD    JUMP(NOGD)
 A43 06BF8 00000000 (00124)         NOP
           00000044 (00125)         VOCOSSZ=#A-VOCOSSA
     06BFA          (00126)         END VOCOSSZ
                    (00127)         EJECT
```

```
                    *
                    *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    *
                             EVEN
    06BFA 00006C34           ADDR G107SI                 POINTER TO CONSTRUCTED I
                    ;                                    NSTRUCTION BLOCK
    06BFC 00006C04           ADDR G107S+2*SCLRS          POINTER TO SCALAR BLOCK
    06BFE 0000               DATA 0                      NUMBER OF SCALARS
    06BFF 0034               DATA G107SZ                 MODULE SIZE
    06C00 00006C0C           ADDR G107SA                 POINTER TO CHAIN ANCHOR
                             EVEN                        END ON WORD BOUNDARY
                    *
                    G107S    BEGIN APS(G107)
A00 06C02 00200D40           JSN(G107S0,P2)              SET OUTPUT PC(P2)
A01 06C04 02300030  SCLRS    SET(R0)                     ENABLE OUTPUT ADDRESSING
A02 06C06 04C2043A           LOAD(BR0,SAS92(1),TF)       512
A03 06C08 06C20438           LOAD(BR0,SAS91(1),TF)       M
A04 06C0A 08C20436           LOAD(BR0,SAS90(1),TF)       G
A05 06C0C 0AC01000           LOAD(BR0,(1),TF)            A1
A06 06C0E 0C500000           LOAD(BR1,MSS)               DUMMY
A07 06C10 0E500000           LOAD(BR1,MSS)               DUMMY
A08 06C12 10500006           LOAD(BR1,6)                 COUNT A'S, NEED 8 ALL TO
                    ;                                    LD
A09 06C14 128A0002  #1       ADD(BR0,2,TF)               A(N)
A0A 06C16 141909B1           SUBL(BR1,1),JUMPP(#1)
A0B 06C18 16200031           CLEAR(RI)
A0C 06C1A 18000020           NOP
                    *
A0D 06C1C 1A300032  G107S0   SET(RA)                     ENABLE APU
A0E 06C1E 1CC60802           LOAD(BW0,X1S+2(3),TF)       X1(.) BA+2 (=X1I(0))
A0F 06C20 1E5001FE           LOAD(BW1,SIZES-1)           X1(.) SIZE-2
A10 06C22 20820002           SUB(BW0,2,TF)               X1R(0)
A11 06C24 228A0006  #1       ADD(BW0,6,TF)               X1I(N)
A12 06C26 24820002           SUB(BW0,2,TF)               X1R(N)
A13 06C28 261111B1           SUBL(BW1,1),JUMPP(#1)
                    *  LOOP  FOR N=1,2,...255
A14 06C2A 28C20794           LOAD(BW0,DMYS(1),TF)        TWO DUMMIES OUT
A15 06C2C 2AC20794           LOAD(BW0,DMYS(1),TF)
A16 06C2E 2CC20414           LOAD(BW0,SAS73(1),TF)       GSUM = ENERGY OUT
A17 06C30 2E200030           CLEAR(R0)
A18 06C32 30000020           NOP
                    *
          00006C0C           G107SA= *C                  ASSIGN VALUE TO CHAIN AN
                    ;                                    CHOR
```

```
06C34          (00172)          END #A-1
               (00173) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
06C34 00000000 (00174) G107$I  DATA 1F'0.0'
      00000034 (00175)         G107$Z=*L-G107$
06C36          (00176)         END
```

```
AEDTSURG:   00HER (00007) (00032)
APSSMEM:    1FFC0 (00008)
ASIZES:     00007 (00009)
COMM:       0002A (00081) (00094)
CSPUSNOS:   021FC (00010) (00035)
DMYS:       00794 (00011) (00164) (00165)
G107S:      06C02 (00034) (00134) (00140) (00175)
G107SA:     06C0C (00137) (00170)
G107S1:     06C34 (00132) (00174)
G107S0:     0000D (00141) (00156)
G107SZ:     00034 (00136) (00175)
HS:         00001 (00013)
LTHMIS:     000FF (00014)
MS:         00000 (00015)
MSS:        00000 (00016) (00147) (00148)
NOGD:       00042 (00070) (00123) (00123)
SAS3R:      003CF (00018)
SAS73:      00414 (00023) (00166)
SAS90:      00436 (00020) (00145)
SAS91:      00439 (00021) (00144)
SAS92:      0043A (00019) (00143)
SAS97:      00444 (00022)
SCLRS:      00001 (00134) (00142)
SIZES:      001FF (00024) (00158)
START:      06070 (00025) (00040)
SVTS:       00382 (00017) (00018) (00019) (00020) (00021) (00022) (00023)
VOCUS:      06872 (00033) (00049)
VOCOSSA:    00000 (00047) (00051) (00125)
VOCUSSZ:    00044 (00048) (00125) (00126)
WS:         00002 (00026)
WWS:        00004 (00027)
X1S:        00800 (00028) (00029) (00157)
X1RS:       00800 (00029)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E=    0
```

```
                     ' FCB 247... "MPSSRT(Y)"  SORT DCT1(,) COEFFICIENTS
                     *                                 ORIGINATED:13-DEC-79
                     *                                 UPDATED:30-MAY-80
                     * DEFINE GLOBAL SYMBOLS
                             OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                             OPADD DOE, (3 .LS. 10) + (12 .LS. 5) + $1B
                             OPADD WAFNE, (1 .LS. 10) + (12 .LS. 5) + $1A
           000008E8          AFDTSORG=$8E8
           0001FFC0          A2SSMEM=$1FFC0
           000021FC          CSPUSNOS=$21FC
           00000928          DCTS=D'2344'
           00000200          DCT1S=D'512'
           00000400          DCT2S=D'1024'
           00000794          DMYS=$794
           00000003          #M=3
           00000001          HS=1
           00000100          INCRES=D'256'
           00001F24          INTGRS=$1F24
           00000000          IORDRS=D'0'
           00000502          ISVTS=$502
           00000502          ISAS00=ISVTS+D'0'
           00000503          ISAS01=ISVTS+D'1'
           00000000          MSS=0
           0000000A          OVRSLAP=D'10'
           000000FF          SIZES=D'256'-D'1'
           00000382          SVTS=$0382
           00008300          START=$8300
           00000C00          TABLS0=D'3072'
           00000D00          TABLS1=TABLS0+INCRES
           00000E00          TABLS2=TABLS1+INCRES
           00000F00          TABLS3=TABLS2+INCRES
           00000A00          TMP2S=D'2560'
           00000C00          TMP4S=D'3072'
           00000002          WS=2
           00000003          ZS=3
                     *
                     * EXPAND ARRAY FUNCTION DISPATCH TABLE
           00000BB2          #L=AFDTSORG+3*2*(247-128)                 FCB 247
     00BB2 001F8308          ADDR SORTS(R7,1)
     00BB4 001F8422          ADDR G106S(R7,1)
     00BB6 001021FC          ADDR CSPUSNOS(,1,0)
                             EJECT
```

```
                 (00043) *
                 (00044) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                 (00045) *
                 (00046)          BUS 2
      00001F14   (00047)          *L=D'7700'
                 (00048)          EVEN
01F14 3C50       (00049) INSTRS0 DATA INS0*D'512'+X'50',X'0000'
01F15 0000
01F16 3450       (00050) INSTRS1 DATA INS1*D'512'+X'50',X'0000'
01F17 0000
01F18 2C50       (00051) INSTRS2 DATA INS2*D'512'+X'50',X'0000'
01F19 0000
01F1A 2450       (00052) INSTRS3 DATA INS3*D'512'+X'50',X'0000'
01F1B 0000
01F1C 662F       (00053) INSTRS4 DATA INDS4*D'512'+X'2E',X'0000'
01F1D 0000
01F1E 6A2F       (00054) INSTRS5 DATA INDS5*D'512'+X'2F',X'0000'
01F1F 0000
01F20 562F       (00055) INSTRS6 DATA INDS6*D'512'+X'2F',X'0000'
01F21 0000
01F22 5A2F       (00056) INSTRS7 DATA INDS7*D'512'+X'2E',X'0000'
01F23 0000
                 (00057)          BUS 1
      00008300   (00058)          *L=START
                 (00059) ;T1$        DATA 1F'0.713'
                 (00060) ;T2$        DATA 1F'2.5'
                 (00061) ;T3$        DATA 1F'7.5'
08300 BF77A2C0   (00062) T1$      DATA 1F'-0.48802602'
08302 0A934F41   (00063) T2$      DATA 1F'1.32192810'
08304 17414FC1   (00064) T3$      DATA 1F'2.9068906'
                 (00065)          EJECT
```

```
                         *
                         *>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                         *
                                 EVEN
    08306 0000                   DATA SORTSSA                           START ADDRESS
    08307 0089                   DATA SORTSSZ                           MODULE SIZE
                         SORTS   BEGIN APU(SORT)
          00000000               #A=0
A00 08308 08F10000       SORTSSA MOV(IOA,A1)\NOP                        A1=THRSH1
A01 0830A 20250000               CLEAR(G1)\NOP                          RESET STALL FLAG
A02 0830C 000008F5               NOP\MOV(IOA,A5)                        A5=THRSH2
A03 0830E 20500000               SET(RO)\NOP                            ENABLE OUTPUT ADDRESSING
A04 08310 08F30000               MOV(IOA,A3)\NOP                        A3=THRSH3
A05 08312 000008F7               NOP\MOV(IOA,A7)                        A7=1
A06 08314 00000810               NOP\MOV(ZERO,A0)                       A0=0
A07 08316 00000811               NOP\MOV(ZERO,A1)                       A1=0
A08 08318 00000812               NOP\MOV(ZERO,A2)                       A2=0
A09 0831A 00000813               NOP\MOV(ZERO,A3)                       A3=0
A0A 0831C 08F008F4               MOV(IOA,A0)\MOV(IOA,A4)                A0=D=DCT1(I);A4=D
A0B 0831E 49004D80               SUB(A0,A1)\SUB(A4,A5)                  D-THRSH1;D-THRSH2
A0C 08320 08800880       INCS1   MOV(R,NULL)                            FORCE END
A0D 08322 911E0030               JUMPS(LVLS0,T1)                        IF T1=1,LEVEL 0
A0E 08324 911F0026               JUMPS(LVLS1,T2)                        IF T2=1,LEVEL 1
A0F 08326 46000000               SUB(A0,A3)\NOP                         D-THRSH3
A10 08328 08800000               MOV(R,NULL)\NOP                        FORCE END
A11 0832A 911F001C               JUMPS(LVLS2,T1)                        IF T1=1,LEVEL 2
A12 0832C 20480000       LVLS3   SET(AF3)\NOP                           INFORM APS OF LVL3
A13 0832E 20530000               SET(UN)\NOP                            FIXED POINT ADD
A14 08330 00004760               NOP\ADD(A3,A7)                         LVL3 CNT+1
A15 08332 08F008F4               MOV(IOA,A0)\MOV(IOA,A4)                A0=D=DCT1(I+1);A4=D
A16 08334 08FC0893               MOV(IOA,OO)\MOV(R,A3)                  OO=I;LVL3=LVL3+1
A17 08336 20330000               CLEAR(UN)\NOP                          RESUME FLOATING POINT!
A18 08338 49004D80               SUB(A0,A1)\SUB(A4,A5)                  D-THRSH1;D-THRSH2
A19 0833A 910B0019       #1      JUMPS(#1,AF3)
A1A 0833C 9016000C               JUMPC(INCS1,FWI)                       FOR I=0,1,2...LTH-1
A1B 0833E 1045003A               JUMP(CNTSOUT,G1),SET                   GET LEVEL COUNTS
A1C 08340 204A0000       LVLS2   SET(AF2)\NOP                           INFORM APS OF LVL2
A1D 08342 20530000               SET(UN)\NOP                            FIXED POINT ADD
A1E 08344 00004740               NOP\ADD(A2,A7)                         LVL2 CNT+1
A1F 08346 08F008F4               MOV(IOA,A0)\MOV(IOA,A4)                A0=D=DCT1(I+1);A4=D
A20 08348 08FC0892               MOV(IOA,OO)\MOV(R,A2)                  OO=I;LVL2=LVL2+1
A21 0834A 20330000               CLEAR(UN)\NOP                          RESUME FLOATING POINT!
A22 0834C 49004D80               SUB(A0,A1)\SUB(A4,A5)                  D-THRSH1;D-THRSH2
A23 0834E 910A0023       #1      JUMPS(#1,AF2)
```

```
A24 08350 9016000C (00110)          JUMPC(INCS1,FW1)              FOR I=0,1,2...LTH-1
A25 08352 1045003A (00111)          JUMP(CNTSOUT,G1),SET          GET LEVEL COUNTS
A26 08354 20490000 (00112) LVLS1    SET(AF1)\NOP                  INFORM APS OF LVL1
A27 08356 20530000 (00113)          SET(UN)\NOP                   FIXED POINT ADD
A28 08358 00004720 (00114)          NOP\ADD(A1,A7)                LVL1 CNT+1
A29 0835A 08F00BF4 (00115)          MOV(IQA,A0)\MOV(IQA,A4)       A0=D=DCT1(I+1);A4=D
A2A 0835C 08FC0891 (00116)          MOV(IQA,OQ)\MOV(R,A1)         OQ=1;LVL1=LVL1+1
A2B 0835E 20330000 (00117)          CLEAR(UN)\NOP                 RESUME FLOATING POINT!
A2C 08360 49004DB0 (00118)          SUB(A0,A1)\SUB(A4,A5)         D-THRSH1;D-THRSH2
A2D 08362 9109002D (00119) #1       JUMPS(#1,AF1)
A2E 08364 9016000C (00120)          JUMPC(INCS1,FW1)              FOR I=0,1,2...LTH-1
A2F 08366 1045003A (00121)          JUMP(CNTSOUT,G1),SET          GET LEVEL COUNTS
A30 08368 20480000 (00122) LVLS0    SET(AF0)\NOP                  INFORM APS OF LVL0
A31 0836A 20530000 (00123)          SET(UN)\NOP                   FIXED POINT ADD
A32 0836C 00004700 (00124)          NOP\ADD(A0,A7)                LVL0 CNT+1
A33 0836E 08F00BF4 (00125)          MOV(IQA,A0)\MOV(IQA,A4)       A0=D=DCT1(I+1);A4=D
A34 08370 08FC0890 (00126)          MOV(IQA,OQ)\MOV(R,A0)         OQ=I;LVL0=LVL0+1
A35 08372 20330000 (00127)          CLEAR(UN)\NOP                 RESUME FLOATING POINT!
A36 08374 49004DB0 (00128)          SUB(A0,A1)\SUB(A4,A5)         D-THRSH1;D-THRSH2
A37 08376 91080037 (00129) #1       JUMPS(#1,AF0)
A38 08378 9016000C (00130)          JUMPC(INCS1,FW1)              FOR I=0,1,2...LTH-1
A39 0837A 20450000 (00131)          SET(G1)\NOP                   GET LEVEL COUNTS
A3A 0837C 00000200 (00132) CNTSOUT  NOP\R(A0)                     R=LVL0 CNT
A3B 0837E 0000023C (00133)          NOP\MOV(OQ),R(A1)             R=LVL1 CNT
A3C 08380 0000025C (00134)          NOP\MOV(OQ),R(A2)             R=LVL2 CNT
A3D 08382 0000027C (00135)          NOP\MOV(OQ),R(A3)             R=LVL3 CNT
A3E 08384 0000089C (00136)          NOP\MOV(R,OQ)                 OQ=LVL3 CNT(0 UPTO LTH)
A3F 08386 081C081C (00137)          MOV(ZERO,OQ)                  OQ=0 FOR INSTR$4+1...
A40 08388 081C081C (00138)          MOV(ZERO,OQ)                  THRU INSTR$7+1
A41 0838A 00000000 (00139) #1       NOP                           WAIT FOR...
A42 0838C 901C0041 (00140)          JUMPC(#1,FO)                  "LEVELS" TO SETTLE
A43 0838E 20370000 (00141)          CLEAR(WI)\NOP                 RELEASE APS INPUT
A44 08390 20500000 (00142)          SET(RO)\NOP                   RELEASE APS OUTPUT
A45 08392 08FC0000 (00143)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS0
A46 08394 08FC0000 (00144)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS1
A47 08396 08FC0000 (00145)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS2
A48 08398 08FC0000 (00146)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS3
A49 0839A 00000000 (00147) #2       NOP                           WAIT FOR...
A4A 0839C 901C0049 (00148)          JUMPC(#2,FO)                  "WRITES" TO SETTLE
A4B 0839E 20370000 (00149)          CLEAR(WI)\NOP                 RELEASE APS INPUT
A4C 083A0 20500000 (00150)          SET(RO)\NOP                   RELEASE APS OUTPUT
A4D 083A2 08FC0000 (00151) ORDRS1   MOV(IQA,OQ)\NOP               OQ=IORDR(I)
A4E 083A4 08FC0000 (00152)          MOV(IQA,OQ)\NOP               OQ=IORDR(I+1)
A4F 083A6 08FC0000 (00153)          MOV(IQA,OQ)\NOP               OQ=IORDR(I+2)
```

```
A50 083A8 08FC0000 (00154)          MOV(IOA,OO)\NOP          OO=IORDR(I+3)
A51 083AA 9016004D (00155)          JUMPC(ORDRS1,FWI)        FOR I=0,1,2...LTH-1
A52 083AC 20280000 (00156)          CLEAR(AFO)\NOP           CLEAR "SKIP" FLG
A53 083AE 081C081C (00157)          MOV(ZERO,OO)             OO=BUFFER ZONE=0
A54 083B0 081C081C (00158)          MOV(ZERO,OO)             OO=BUFFER ZONE=0
A55 083B2 081C081C (00159)          MOV(ZERO,OO)             OO=BUFFER ZONE=0
A56 083B4 081C081C (00160)          MOV(ZERO,O )             OO=BUFFER ZONE=0
A57 083B6 081C081C (00161)          MOV(ZERO,OO)             OO=BUFFER ZONE=0
                   (00162) *
A58 083B8 20370000 (00163) SHUFLSV  CLEAR(WI)\NOP            RELEASE APS INPUT
A59 083BA 20530000 (00164)          SET(UN)\NOP              FIXED POINT ADD
A5A 083BC 08F00000 (00165)          MOV(IOA,A0)\NOP          A0=J
A5B 083BE 000008F0 (00166)          NOP\MOV(IOA,A0)          A0=K
A5C 083C0 40004000 (00167)          ADD(A0,A0)               2J;2K
A5D 083C2 08F10000 (00168)          MOV(IOA,A1)\NOP          A1=J'
A5E 083C4 000008F1 (00169)          NOP\MOV(IOA,A1)          A1=K'
A5F 083C6 413C413C (00170)          MOV(OO),ADD(A1,A1)       2J->$6+1,2J';2K->$7+1,2K
                                                             '
                   (00171) ;
A60 083C8 08FC0000 (00172)          MOV(IOA,OO)\NOP          INSTR$4 INTO INDS4
A61 083CA 000008FC (00173)          NOP\MOV(IOA,OO)          INSTR$5 INTO INDS5
A62 083CC 08FC0000 (00174)          MOV(IOA,OO)\NOP          NULL=TMP2(J*)
A63 083CE 08FC0000 (00175)          MOV(IOA,OO)\NOP          NULL=DCT1(J*)
A64 083D0 0 C0000 (00176)          MOV(IOA,OO)\NOP          NULL=TMP2(K*)
A65 083D2 08FC0000 (00177)          MOV(IOA,OO)\NOP          NULL=DCT1(K*)
A66 083D4 08F20000 (00178)          MOV(IOA,A2)\NOP          A2=J''
A67 083D6 000008F2 (00179)          NOP\MOV(IOA,A2)          A2=K''
A68 083D8 425C425C (00180)          MOV(OO),ADD(A2,A2)       2J'->$4+1,2J'';2K'->$5+1
                                                             ,2K''
                   (00181) ;
A69 083DA 08FC0000 (00182)          MOV(IOA,OO)\NOP          INSTR$6 INTO INDS6
A6A 083DC 000008FC (00183)          NOP\MOV(IOA,OO)          INSTR$7 INTO INDS7
A6B 083DE 08FC0000 (00184)          MOV(IOA,OO)\NOP          NULL=TMP2(J**)
A6C 083E0 08FC0000 (00185)          MOV(IOA,OO)\NOP          NULL=DCT1(J**)
A6D 083E2 08FC0000 (00186)          MOV(IOA,OO)\NOP          NULL=TMP2(K**)
A6E 083E4 08FC0000 (00187)          MOV(IOA,OO)\NOP          NULL=DCT1(K**)
A6F 083E6 911D0086 (00188) SHUFLS   JUMPS(DNESA,FI)          I/O HAS SETTLED! DONE?
A70 083E8 901C0070 (00189) STALS    JUMPC(STALS,FO)          WAIT FOR OUTPUT TO CLEAR
A71 083EA 20480000 (00190)          SET(AFO)\NOP             SET OUTPUT "SKIP" FLG
A72 083EC 20500000 (00191)          SET(RO)\NOP              RELEASE APS OUTPUT
A73 083EE 08F10000 (00192)          MOV(IOA,A1)\NOP          A1=J'
A74 083F0 000008F1 (00193)          NOP\MOV(IOA,A1)          A1=K'
A75 083F2 413C413C (00194)          MOV(OO),ADD(A1,A1)       2J->$6+1,2J';2K->$7+1,2K
                                                             '
                   (00195) ;
A76 083F4 08FC0000 (00196)          MOV(IOA,OO)\NOP          INSTR$4 INTO INDS4
A77 083F6 000008FC (00197)          NOP\MOV(IOA,OO)          INSTR$5 INTO INDS5
```

```
A78 083F8 08FC0000 (00198)          MOV(IOA,00)\NOP              DCT2(I)=TMP2(J*)
A79 083FA 08FC0000 (00199)          MOV(IOA,00)\NOP              TMP4(I)=DCT1(J*)
A7A 083FC 000008FC (00200)          NOP\MOV(IOA,00)              DCT2(I+1)=TMP2(K*)
A7B 083FE 000008FC (00201)          NOP\MOV(IOA,00)              TMP4(I+1)=DCT1(K*)
A7C 08400 08F20000 (00202)          MOV(IOA,A2)\NOP              A2=J''
A7D 08402 000008F2 (00203)          NOP\MOV(IOA,A2)              A2=K''
A7E 08404 425C425C (00204)          MOV(00),ADD(A2,A2)           2J'->$4+1,2J'';2K'->$5+1
                   (00205) ;                                     ,2K''
A7F 08406 08FC0000 (00206)          MOV(IOA,00)\NOP              INSTRS6 INTO INDS6
A80 08408 000008FC (00207)          NOP\MOV(IOA,00)              INSTRS7 INTO INDS7
A81 0840A 08FC0000 (00208)          MOV(IOA,00)\NOP              DCT2(I+2)=TMP2(J*)
A82 0840C 08FC0000 (00209)          MOV(IOA,00)\NOP              TMP4(I+2)=DCT1(J*)
A83 0840E 000008FC (00210)          NOP\MOV(IOA,00)              DCT2(I+3)=TMP2(K*)
A84 08410 000008FC (00211)          NOP\MOV(IOA,00)              TMP4(I+3)=DCT1(K*)
A85 08412 1000006F (00212)          JUMP(SHUFLS)                 "JUMP(...)" SETTLES I/O
                   (00213)
A86 08414 20300000 (00214) DNESA    CLEAR(R0)\NOP                APS CANNOT TURN ITSELF O
                   (00215) ;                                     FF!
A87 08416 20320000 (00216)          CLEAR(RA)\NOP                APU DONE!
A88 08418 00000000 (00217)          NOP
          00000089 (00218)          SORTSSZ=#A-SORTSSA
    0841A          (00219)          END SORTSSZ
                   (00220)          EJECT
```

```
                        (00221) *
                        (00222) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        (00223) *
                        (00224)          EVEN
    0841A 00008516      (00225)          ADDR G106$1                   POINTER TO CONSTRUCTED I
                        (00226) ;                                      NSTRUCTION BLOCK
    0841C 00008424      (00227)          ADDR G106$+2*SCLR$            POINTER TO SCALAR BLOCK
    0841E 0000          (00228)          DATA 0                        NUMBER OF SCALARS
    0841F 00F6          (00229)          DATA G106$Z                   MODULE SIZE
    08420 00008468      (00230)          ADDR G106$A                   POINTER TO CHAIN ANCHOR
                        (00231)          EVEN                          END ON WORD BOUNDARY
                        (00232) *
                        (00233) G106$    BEGIN APS(G106)
A00 08422 00103940 (00234)          JSN(G106$1,P1)                SET INPUT PC(P1)
A01 08424 02C28300 (00235) SCLR$    LOAD(BR0,T1$(1),TF)           IQ=T1
A02 08426 04C28302 (00236)          LOAD(BR0,T2$(1),TF)           IQ=T2
A03 08428 06C28304 (00237)          LOAD(BR0,T3$(1),TF)           IQ=T3
A04 0842A 09420503 (00238)          LOAD(BR0,1SAS01(1),TF)        IQ=1
A05 0842C 0AC60AD0 (00239)          LOAD(BR0,TMP2$(3),TF)         IQ=TMP2(0)
A06 0842E 0C5000FF (00240)          LOAD(BR1,SIZE$)               TMP2(.) SIZE-1
A07 08430 0E641F23 (00241)          LOAD(BR2,INTGR$-1(2))         TABL(.) BA-1
A08 08432 108A0002 (00242) IINC$I   ADD(BR0,W$,TF)                IQ=TMP2(I+1)
A09 08434 132A0001 (00243)          ADD(BR2,H$,TF)                IQ=TABL(I)=I
A0A 08436 141908B1 (00244)          SUBL(BR1,1),JUMPP(IINC$I)     FOR I=0,1,2...LTH-1
A0B 08438 16300037 (00245)          SET(W1)                       STALL APS INPUT
A0C 0843A 18500002 (00246)          LOAD(BR1,2)                   3+1=4 OVERWRITES
A0D 0843C 1AC41F14 (00247)          LOAD(BR0,INSTR$0(2),TF)       IQ="LOAD(BR1,M$$)"
A0E 0843E 1C8A0002 (00248) #1       ADD(BR0,W$,TF)                IQ="LOAD(BR1,M$$)"
A0F 08440 1F190FB1 (00249)          SUBL(BR1,1),JUMPP(#1)         INS2,INS1 & INS0
A10 08442 20300037 (00250)          SET(W1)                       STALL APS INPUT
A11 08444 22460FFF (00251) L3$SU    LOAD(BR0,TABL$3-1(3))         TABLE3(.) BA-1
A12 08446 24500000 (00252) INS3     LOAD(BR1,M$$)                 M$$=LEVEL 3 COUNT
A13 08448 261915F1 (00253) TSTSL3   SUBL(BR1,1),JUMPN(L2$SU)      PICKUP TILL NO MORE
A14 0844A 29091379 (00254)          ADDL(BR0,H$,TF),JUMP(TSTSL3)  IQ=TABLE3(I)
A15 0844C 2A460DFF (00255) L2$SU    LOAD(BR0,TABL$2-1(3))         TABLE2(.) BA-1
A16 0844E 2C500000 (00256) INS2     LOAD(BR1,M$$)                 M$$=LEVEL 2 COUNT
A17 08450 2F1919F1 (00257) TSTSL2   SUBL(BR1,1),JUMPN(L1$SU)      PICKUP TILL NO MORE
A18 08452 31091779 (00258)          ADDL(BR0,H$,TF),JUMP(TSTSL2)  IQ=TABLE2(I)
A19 08454 32460CFF (00259) L1$SU    LOAD(BR0,TABL$1-1(3))         TABLE1(.) BA-1
A1A 08456 34500000 (00260) INS1     LOAD(BR1,M$$)                 M$$=LEVEL 1 COUNT
A1B 08458 36191DF1 (00261) TSTSL1   SUBL(BR1,1),JUMPN(L0$SU)      PICKUP TILL NO MORE
A1C 0845A 39091B79 (00262)          ADDL(BR0,H$,TF),JUMP(TSTSL1)  IQ=TABLE1(I)
A1D 0845C 3A460BFF (00263) L0$SU    LOAD(BR0,TABL$0-1(3))         TABLE0(.) BA-1
A1F 0845E 3C500000 (00264) INS0     LOAD(BR1,M$$)                 M$$=LEVEL 0 COUNT
```

```
A1F 08460 3F1921F1 (00265) TSTSLO  SUBL(BR1,1),JUMPN(STALSI)           PICKUP TILL NO MORE
A20 08462 41091F79 (00266)         ADDL(BRO,HS,TF),JUMP(TSTSLO)        IQ=TABLEO(I)
A21 08464 42300037 (00267) STALSI  SET(WI)                             STALL APS INPUT
A22 08466 44500103 (00268)         LOAD(BF1,SIZES+4)                   IORDR(,) SIZE-1+4
                   (00269) *
A23 08468 47400000 (00270)         LOAD(BRO,IO),TF)                    IORDR(,) BA,IQ=IORDR=J
A24 0846A 48600000 (00271)         LOAD(BR2,MSS)                       DUMMY OP FOR EXEC
A25 0846C 4B0A0000 (00272)         ADD(BRO,MSS,TF)                     IQ=IORDR(I)=K
A26 0846E 4D0A0001 (00273) INDSJ   ADD(BRO,HS,TF)                      IQ=J'=IORDR(I+2)
A27 08470 4F0A0001 (00274)         ADD(BRO,HS,TF)                      IQ=K'=IORDR(I+3)
A28 08472 50F41F1C (00275)         LOAD(BR2,INSTRS4(2),TF)             OVERWRITE INDS4
A29 08474 52F41F1F (00276)         LOAD(BR2,INSTRS5(2),TF)             OVERWRITE INDS5
A2A 08476 54600000 (00277)         LOAD(BR2,MSS)                       RESET BR2 TO 0!
A2B 08478 562F0000 (00278) INDS6   ADD(BR2,MSS,NA,C)                   MSS=2J,PC->PC1
A2C 0847A 58600000 (00279)         LOAD(BP2,MSS)                       RESET BR2 TO 0!
A2D 0847C 5A2F0000 (00280) INDS7   ADD(BR2,MSS,NA,C)                   MSS=2J,PC->PC1
A2E 0847E 5D0A0001 (00281)         ADD(BRO,HS,TF)                      IQ=J''
A2F 08480 5F0A0001 (00282)         ADD(BRO,HS,TF)                      IQ=K''
A30 08482 60F41F20 (00283)         LOAD(BR2,INSTRS6(2),TF)             OVERWRITE INDS6
A31 08484 62F41F22 (00284)         LOAD(BR2,INSTRS7(2),TF)             OVERWRITE INDS7
A32 08486 64600000 (00285)         LOAD(BR2,MSS)                       RESET BR2 TO 0!
A33 08488 662F0000 (00286) INDS4   ADD(BR2,MSS,NA,C)                   MSS=2J,PC->PC1
A34 0848A 68600000 (00287)         LOAD(BR2,MSS)                       RESET BR2 TO 0!
A35 0848C 6A2F0000 (00288) INDS5   ADD(BR2,MSS,NA,C)                   MSS=2J,PC->PC1
A36 0848E 6C1926B4 (00289)         SUBL(BR1,4),JUMPP(INDSJ)            FOR I=0,1,2...LTH+1
                   (00290) *
A37 08490 6E200031 (00291) DNESI   CLEAR(RI)                           INPUT DONE!
A38 08492 70000020 (00292)         NOP
A39 08494 72204040 (00293) G106S1  JSN(G106S0,P2)                      SET OUTPUT PC
A3A 08496 74760A00 (00294) TOPSP1  LOAD(BR3,TMP2S(3))                  TMP2(,) BA
A3B 08498 76B9002D (00295)         ADDB(BR3,BR2,TF)                    IQ=TMP2(J* OR K*)
A3C 0849A 78760200 (00296)         LOAD(BR3,DCT1S(3))                  DCT1(,) BA
A3D 0849C 7AB9002D (00297)         ADDB(BR3,BR2,TF)                    IQ=DCT1(J* OR K*)
A3E 0849E 7C2F0000 (00298)         ADD(BR2,MSS,NA,C)                   PC->PC0
A3F 084A0 7E003A60 (00299)         JUMP(TOPSP1)                        RESET PC AND SETTLE
                   (00300) *
A40 084A2 80306D40 (00301) G106S0  JSN(G106S3,P3)                      SET OUTPUT PC(PC3)
A41 084A4 82460BFF (00302)         LOAD(BW0,TABLS0-1(3))               TABLE0(,) BA-1
A42 084A6 84560CFF (00303)         LOAD(BW1,TABLS1-1(3))               TABLE1(,) BA-1
A43 084A8 86660DFF (00304)         LOAD(BW2,TABLS2-1(3))               TABLE2(,) BA-1
A44 084AA 88760EFF (00305)         LOAD(BW3,TABLS3-1(3))               TABLE3(,) BA-1
A45 084AC 8A0053F5 (00306) FLGSTST JUMPS(SETSCNT,G1)                   G1=1,SET LEVEL COUNTS
A46 084AE 8C004BF8 (00307)         JUMPS(INSRTS0,AF0)                  AF0=1 THEN LEVEL 0
A47 084B0 8E004DF9 (00308)         JUMPS(INSRTS1,AF1)                  AF1=1 THEN LEVEL 1
```

```
A48 084B2 90004FFA (00309)          JUMPS(INSRTS2,AF2)                AF2=1 THEN LEVEL 2
A49 084B4 92005IFB (00310)          JUMPS(INSRTS3,AF3)                AF3=1 THEN LEVEL 3
A4A 084B6 94004560 (00311)          JUMP(FLGSTST)                     RE-TEST ALL FLAGS
A4B 084B8 9620002B (00312) INSRTS0  CLEAR(AF0)                        RELEASE APU
A4C 084BA 99014579 (00313)          ADDL(BW0,HS,TF),JUMP(FLGSTST)     OQ=TABLE0(II),II=II+1
A4D 084BC 9A200029 (00314) INSRTS1  CLEAR(AF1)                        RELEASE APU
A4E 084BE 9D114579 (00315)          ADDL(BW1,HS,TF),JUMP(FLGSTST)     OQ=TABLE1(JJ),JJ=JJ+1
A4F 084C0 9E20002A (00316) INSRTS2  CLEAR(AF2)                        RELEASE APU
A50 084C2 A1214579 (00317)          ADDL(BW2,HS,TF),JUMP(FLGSTST)     OQ=TABLE2(KK),KK=KK+1
A51 084C4 A220002B (00318) INSRTS3  CLEAR(AF3)                        RELEASE APU
A52 084C6 A5314579 (00319)          ADDL(BW3,HS,TF),JUMP(FLGSTST)     OQ=TABLE3(LL),LL=LL+1
A53 084C8 A6500007 (00320) SETSCNT  LOAD(BW1,7)                       8 "INSTR" WORDS...
A54 084CA A8441F13 (00321)          LOAD(BW0,INSTRS0-1(2))            TO BE SET OR CLEARED
A55 084CC AB0A0002 (00322) #1       ADD(BW0,WS,TF)                    OQ=INSTRS#+1
A56 084CE AC1155B1 (00323)          SUBL(BW1,1),JUMPP(#1)             FOR I=0,1,2...7
A57 084D0 AF200030 (00324)          CLEAR(RO)                         STALL APS OUTPUT
A58 084D2 B0500003 (00325)          LOAD(BW1,3)                       3+1=4 SCATTER WRITES
A59 084D4 B2C3FFC0 (00326) #2       LOAD(BW0,APSSMEM(1),TF)           INSTRS# OVERWRITE
A5A 084D6 B41159B1 (00327)          SUBL(BW1,1),JUMPP(#2)             FOR I=0,1...3
A5B 084D8 B6200030 (00328)          CLEAR(RO)                         STALL APS OUTPUT
A5C 084DA B8000020 (00329)          NOP                               RELEASED BY APU
A5D 084DC BB460000 (00330)          LOAD(BW0,IORDRS(3),TF)            IORDR(.) BA,OQ=IORDR(0)
A5E 084DE BC500108 (00331)          LOAD(BW1,SIZES+OVRSLAP-1)         IORDR(.) SIZE-2+OVERLAP
A5F 084E0 BF0A0001 (00332) RDSOUT   ADD(BW0,HS,TF)                    OQ=IORDR(I)
A60 084E2 C0115FB1 (00333)          SUBL(BW1,1),JUMPP(RDSOUT)         FOR I=1,2...LTH-1;+10 0'
                   (00334) ;                                          S
                   (00335) *
A61 084E4 C24603FF (00336)          LOAD(BW0,DCT2S-2(3))              DCT2(.) BA-2
A62 084E6 C4760BFF (00337)          LOAD(BW3,TMP4S-2(3))              TMP4(.) BA-2
A63 084E8 C6000020 (00338) INDS1    NOP                               LET "JUMP(...)" SETTLE
A64 084EA C9641F21 (00339)          LOAD(BW2,INSTRS6+1(2),TE)         SET MSS IN INSTRS6
A65 084EC CB641F23 (00340)          LOAD(BW2,INSTRS7+1(2),TF)         SET MSS IN INSTRS7
A66 084EE CCF3FFC0 (00341)          LOAD(BW2,APSSMEM(1),TF)           OVERWRITE INDS4
A67 084F0 CFAF0000 (00342)          ADD(BW2,MSS,TF,C)                 OVERWRITE INDS5;PC3->PC
A68 084F2 D1641F1D (00343)          LOAD(BW2,INSTRS4+1(2),TE)         SET MSS IN INSTRS4
A69 084F4 D3641F1F (00344)          LOAD(BW2,INSTRS5+1(2),TE)         SET MSS IN INSTRS5
A6A 084F6 D4F3FFC0 (00345)          LOAD(BW2,APSSMEM(1),TF)           OVERWRITE INDS6
A6B 084F8 D6AF0000 (00346)          ADD(BW2,MSS,TF,C)                 OVERWRITE INDS7;PC3->PC
A6C 084FA DB206370 (00347)          JUMP(INDS1,RO),CLEAR              RESET & STALL
A6D 084FC DA300037 (00348) G106S3   SET(RA)                           ENABLE APU
A6E 084FE DC006020 (00349) TOPSP3   NOP                               LET "JUMP(...)" SETTLE!
A6F 08500 DF0076AB (00350)          JUMPC(DAYSFIL,AF0)                WAIT FOR "SKIP" FLAG
A70 08502 E0BA0002 (00351)          ADD(BW0,WS,TF)                    OQ=DCT2(J*)
A71 08504 E2BA0002 (00352)          ADD(BW3,WS,TF)                    OQ=TMP4(J*)=DCT1T(J*)
```

```
A72 08506 F48A0002 (00353)          ADD(BW0,WS,TF)                    00=DCT2(K*)
A73 08508 F6BA0002 (00354)          ADD(BW3,WS,TF)                    00=TMP4(K*)=DCT1T(K*)
A74 0850A F82F0000 (00355) CHNGSPC ADD(BW2,MSS,C)                     NO IO! CHANGE PC
A75 0850C FA006F60 (00356)          JUMP(TOPSP3)                      RESET PC AND SETTLE
A76 0850E FCF20794 (00357) DMYSFIL LOAD(BW2,DMYS(1),TF)               00=?
A77 08510 FEAA0000 (00358)          ADD(BW2,MSS,TF)                   00=?
A78 08512 F0AA0000 (00359)          ADD(BW2,MSS,TF)                   00=?
A79 08514 F2A1747B (00360)          ADDL(BW2,MSS,TF),JUMP(CHNGSPC)    00=? THEN CHANGE PC
                   (00361) *
          00008468 (00362)          G106SA= #C                        ASSIGN VALUE TO CHAIN AN
                   (00363) ;                                          CHOR
    08516          (00364)          END #A-1
                   (00365) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    08516 00000000 (00366) G106SI  DATA 1F'0,0'
          000000F6 (00367)          G106SZ=#L-G106S
    08518          (00368)          END
```

```
AFDTSORG:   00HF8 (00008) (00038)
APSSMFM:    1FFC0 (00009) (00326) (00341) (00345)
CHNGSPC:    00074 (00355) (00360)
CNTSOUT:    0003A (00101) (00111) (00121) (00132)
CSPUSNOS:   021FC (00010) (00041)
DCTS:       00928 (00011)
DCT1S:      00200 (00012) (00296)
DCT2S:      00400 (00013) (00336)
DMYS:       00794 (00014) (00357)
DMYSFIL:    00076 (00350) (00357)
DNESA:      00086 (00188) (00214)
DNESI:      00037 (00291)
FLGSTST:    00045 (00306) (00311) (00313) (00315) (00317) (00319)
G106S:      08422 (00040) (00227) (00233) (00367)
G106S1:     00039 (00234) (00293)
G106S3:     0006D (00301) (00348)
G106SA:     08468 (00230) (00362)
G106SI:     08516 (00225) (00366)
G106SD:     00040 (00293) (00301)
G106SZ:     000F6 (00229) (00367)
HS:         00001 (00016) (00243) (00254) (00258) (00262) (00266) (00273) (00274) (00281) (00282)
                  (00313) (00315) (00317) (00319) (00332)
IINCS1:     00008 (00242) (00244)
INS0:       0001F (00049) (00264)
INS1:       0001A (00050) (00260)
INS2:       00016 (00051) (00256)
INS3:       00012 (00052) (00252)
INCS1:      0000C (00086) (00100) (00110) (00120) (00130)
INCRES:     00100 (00017) (00029) (00030) (00031)
INDS4:      00033 (00053) (00286)
INDS5:      00035 (00054) (00288)
INDS6:      0002B (00055) (00278)
INDS7:      0002D (00056) (00280)
INDSI:      00063 (00338) (00347)
INDSJ:      00026 (00273) (00289)
INSRTS0:    0004B (00307) (00312)
INSRTS1:    0004D (00308) (00314)
INSRTS2:    0004F (00309) (00316)
INSRTS3:    00051 (00310) (00318)
INSTRS0:    01F14 (00049) (00247) (00321)
INSTRS1:    01F16 (00050)
INSTRS2:    01F18 (00051)
INSTRS3:    01F1A (00052)
INSTRS4:    01F1C (00053) (00275) (00343)
```

```
INSTR$5:   01F1F (00054) (00276) (00344)
INSTR$6:   01F20 (00055) (00283) (00339)
INSTR$7:   01F22 (00056) (00284) (00340)
INTGR$:    01F24 (00018) (00241)
IORDR$:    00000 (00019) (00330)
ISAS00:    00502 (00021)
ISAS01:    00503 (00022) (00238)
ISVTS:     00502 (00020) (00021) (00022)
L0$SU:     0001D (00261) (00263)
L1$SU:     00019 (00257) (00259)
L2$SU:     00015 (00253) (00255)
L3$SU:     00011 (00251)
LVLS0:     00030 (00087) (00122)
LVLS1:     00026 (00088) (00112)
LVLS2:     0001C (00091) (00102)
LVLS3:     00012 (00092)
M$S:       00000 (00023) (00252) (00256) (00260) (00264) (00271) (00272) (00277) (00278) (00279)
                 (00280) (00285) (00286) (00287) (00288) (00298) (00342) (00346) (00355) (00358)
                 (00359) (00360)
ORDR$1:    00040 (00151) (00155)
OVRSLAP:   0000A (00024) (00331)
RPSOUT:    0005F (00332) (00333)
SCLR$:     00001 (00227) (00235)
SETSCNT:   00053 (00306) (00320)
SHUFLS:    0006F (00188) (00212)
SHUFLSV:   00058 (00163)
SIZES:     000FF (00025) (00240) (00268) (00331)
SORTS:     08308 (00039) (00072)
SORTSSA:   00000 (00070) (00074) (00218)
SORTSSZ:   00089 (00071) (00218) (00219)
STALS:     00070 (00189) (00189)
STALSI:    00021 (00265) (00267)
START:     08300 (00027) (00058)
SVTS:      00382 (00026)
T1S:       08300 (00062) (00235)
T2S:       08302 (00063) (00236)
T3S:       08304 (00064) (00237)
TABLS0:    00C00 (00028) (00029) (00263) (00302)
TABLS1:    00000 (00029) (00030) (00259) (00303)
TABLS2:    00F00 (00030) (00031) (00255) (00304)
TABLS3:    00F00 (00031) (00251) (00305)
TMP2S:     09A00 (00032) (00239) (00294)
TMP4S:     00C00 (00033) (00337)
TOPSP1:    0003A (00294) (00299)
```

```
PAGE   13:      FCH 247... "MPSSRT(Y)"  SORT DCTI(.) COEFFICIENTS

  TOPSP3:     0006F (00349) (00356)
  TSTSL0:     0001F (00265) (00266)
  TSTSL1:     0001B (00261) (00262)
  TSTSL2:     00017 (00257) (00258)
  TSTSL3:     00013 (00253) (00254)
  WS:         00002 (00034) (00242) (00248) (00322) (00351) (00352) (00353) (00354)
  ZS:         00003 (00035)

LINES WITH ERRORS:    0  (MAP VERSION R00101.10)   E=    0
```

```
                 '  FCB 248... "MPIDCM(Y,U,V)"      MAKHOUL IDCT CORRECTION
                 *                                  ORIGINATED:16-JAN-80
                 *                                  UPDATED:17-JUN-80
                 *  DEFINE GLOBAL SYMBOLS
                          OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                          OPADD OOF, (3 .LS. 10) + (12 .LS. 5) + $1B
                          OPADD WAFNE, (1 .LS. 10) + (12 .LS. 5) + $1A
                          OPADD EO, (3 .LS. 10) + (12 .LS. 5) + $1C
      00000BE8            AFDTSORG=$BE8
      00000400            COSZS=D'1024'
      000021FC            CSPUSNOS=$21FC
      00000794            DMYS=$794
      00000400            DRDCTS=D'1024'
      00000003            MM=3
      00000001            HS=1
      00000000            MSS=0
      000000FF            SIZES=D'256'-D'1'
      00000600            SINZS=D'1536'
      00000382            SVTS=$0382
      000003B8            SAS27=SVTS+2*D'27'
      000003E6            SAS50=SVTS+2*D'50'
      00000446            SAS98=SVTS+2*D'98'
      0000044A            SAS100=SVTS+2*D'100'
      0000044C            SAS101=SVTS+2*D'101'
      00007150            START=$7150
      00000002            WS=2
      00000004            WWS=4
      00000005            WWHS=5
      00000006            WWWS=6
      00000003            ZS=3
      00000007            ZZHS=7
                 *
                 *  EXPAND ARRAY FUNCTION DISPATCH TABLE
      00000BE8            *L=AFDTSORG+3*2*(248-128)                        FCB 248
00BE8 001F7166            ADDR IDCMS(R7,1)
00BEA 001F722A            ADDR G201S(R7,1)
00BEC 001021FC            ADDR CSPUSNOS(,1,0)
                          EJECT
```

```
                        (00039) *
                        (00040) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                        (00041) *
               00007150 (00042)          *L=START
                        (00043)          EVEN
07150 2000003B          (00044) TABLES   DATA 2.3841859E-07                    2**(-22)
07152 40000044          (00045)          DATA 3.2768000E+04                    2**15
07154 0820003D          (00046)          DATA 1.5497208E-05                    4*65*16**(-6)
07156 3D86415C          (00047)          DATA 0.24957274BE+34                  C5
07158 126900D7          (00048)          DATA 0.71220889 7E+27                 C4
0715A 725511D1          (00049)          DATA 0.263632333E+21                  C3
0715C 1EBD5ACC          (00050)          DATA 0.675972338E+14                  C2
0715E 8800003B          (00051) TABLES2  DATA -5.9604646E-08                   -16**(-6)
07160 58B93D46          (00052)          DATA 0.116291788E+08                  C1
07162 7FFFFF40          (00053)          DATA 0.9999999250                     C0
                        (00054)          EJECT
```

```
                          (00055) *
                          (00056) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00057) *
                          (00058)          EVEN
     07164 0000           (00059)          DATA IDCM$SA                                START ADDRESS
     07165 005F           (00060)          DATA IDCM$SZ                                MODULE SIZE
                          (00061) IDCMS    BEGIN APU(IDCM)
           00000000       (00062)          #A=0
                          (00063) *
                          (00064) *  YR(K)=YR(N-K)=[COS(K)*DCT(K)+SIN(K)*DCT(N-K)]
                          (00065) *  YI(K)=-YI(N-K)=[SIN(K)*DCT(K)-COS(K)*DCT(N-K)]
                          (00066) *  WHERE COS(K) IS COS(2*PI*K/2N), SIN SIMILAR.
 A00 07166 16601680       (00067) IDCM$SA K(1)
 A01 07168 08EC0000       (00068)          MOV(IOA,00)\NOP                             DCT(0)=C(0)
 A02 0716A 08EC08EC       (00069)          MOV(IOA,M4)                                 C(N/2)
 A03 0716C 08E808E8       (00070)          MOV(IOA,M0)                                 COS(N/2)
 A04 0716E 00008400       (00071)          NOP\MUL(M0,M4)                              COS(N/2)*C(N/2)
 A05 07170 000006B0       (00072)          NOP\MOV(P,A0)                               COS*C (=.5*DCT(N/2)
 A06 07172 088F400F       (00073)          MOV(R,M7)\MOV(M7),ADD(A0,A0)                STORE 1;2*COS*C
 A07 07174 081C089C       (00074)          MOV(ZERO,00)\MOV(R,00)                      YI(0)\YR(N/2)
 A08 07176 081C0000       (00075)          MOV(ZERO,00)\NOP                            YI(N/2)
 A09 07178 08EC08EC       (00076)          MOV(IOA,M4)                                 C(K) (K STARTS AT N/2-1)
 A0A 0717A 08ED08ED       (00077)          MOV(IOA,M5)                                 C(N-K)
 A0B 0717C 08E808E8       (00078)          MOV(IOA,M0)                                 COS(K)
 A0C 0717E 08E908E9       (00079)          MOV(IOA,M1)                                 SIN(K)
 A0D 07180 84008480       (00080)          MUL(M0,M4)\MUL(M1,M4)                       COS*C(K)\SIN*C(K)
 A0E 07182 84B08430       (00081)          MOV(A0),MUL(M1,M5)\MOV(A0),MUL(M0,M5)       SIN*C(N-K)\COS*C(N-K)
 A0F 07184 08EC08EC       (00082)          MOV(IOA,M4)                                 NEXT C(K)
 A10 07186 0BA00BA0       (00083)          MOV(P,NULL)                                 MAKE SURE ITS DONE
 A11 07188 08ED08ED       (00084)          MOV(IOA,M5)                                 NEXT C(N-K)
 A12 0718A 08E808E8       (00085)          MOV(IOA,M0)                                 NEXT COS(K)
 A13 0718C 08E908E9       (00086)          MOV(IOA,M1)                                 NEXT SIN(K)
 A14 0718E 84118491       (00087) IDCLP    MOV(A1),MUL(M0,M4)\MOV(A1),MUL(M1,M4)       LAST TIMES C(N-K) TERMS,
 A15 07190 41004900       (00088)          ADD(A0,A1)\SUB(A0,A1)                       MAKE YR(K)\YI(K)
 A16 07192 088B088B       (00089)          MOV(R,M3)                                   NEED TO TAKE HALF OF THE
                          (00090) ;                                                    SE RESULTS
 A17 07194 85F085F0       (00091)          MOV(A0),MUL(M3,M7)                          STORE COS*C(K),YR(K)*.5\
                          (00092) ;                                                    SIN*C(K);.5*YI(K)
 A18 07196 84B8843B       (00093)          MOV(EXO),MUL(M1,M5)\MOV(EXO),MUL(M0,M5)     SAVE YR,YI;SIN*C(K)\COS*
                          (00094) ;                                                    C(N-K)
 A19 07198 0853085C       (00095)          MOV(EXI,A3)\MOV(EXI,00)                     YI(K)\YR(K) OUT
 A1A 0719A 0A600000       (00096)          NEG(A3)\NOP                                 GET YI(N-K)
 A1B 0719C 085C085C       (00097)          MOV(EXI,00)\MOV(EXI,00)                     YI(K)\YR(N-K (=YR(K))
 A1C 0719E 089C0000       (00098)          MOV(R,00)\NOP                               YI(N-K)
```

```
A1D 071A0 08FC08FC (00099)          MOV(IOA,M4)                      NEXT C(K)
A1E 071A2 08FD08FD (00100)          MOV(IOA,M5)                      NEXT C(N-K)
A1F 071A4 08F808F8 (00101)          MOV(IOA,M0)                      NEXT COS(K)
A20 071A6 08F908F9 (00102)          MOV(IOA,M1)                      NEXT SIN(K)
A21 071A8 00000000 (00103)          NOP
A22 071AA 90160014 (00104)          JUMPC(IDCLP,FWI)
A23 071AC 20372037 (00105)          CLEAR(WI)
                   (00106) *
A24 071AE 00000000 (00107)          NOP
A25 071B0 08F80000 (00108) SMULS    MOV(IOA,M0)\NOP                  M0=VAR''
A26 071B2 08FD0000 (00109)          MOV(IOA,M5)\NOP                  M5=1.0/20.0=0.05
A27 071B4 84200000 (00110)          MUL(M0,M5)\NOP                   0.05*VAR''
A28 071B6 08BC0000 (00111)          MOV(P,OQ)\NOP                    OQ=VAR'''=0.05*VAR''
A29 071B8 20372037 (00112)          CLEAR(WI)                        RELEASE APS INPUT
                   (00113) *
A2A 071BA 08FB08FB (00114) EXPS     MOV(IOA,M3)                      M3=SA TEMPORARILY
A2B 071BC 08FC08FC (00115)          MOV(IOA,M4)                      M4=2**(-22) TEMPORARILY
A2C 071BE 85808580 (00116)          MUL(M3,M4)
A2D 071C0 08FD08FD (00117)          MOV(IOA,M5)                      M5=2**(15)
A2E 071C2 08F408F4 (00118)          MOV(IOA,A4)                      A4=4*65*16**(-6)
A2F 071C4 08E908E9 (00119)          MOV(IOA,M1)                      M1=C5
A30 071C6 08F708F7 (00120)          MOV(IOA,A7)                      A7=C4
A31 071C8 08F608F6 (00121)          MOV(IOA,A6)                      A6=C3
A32 071CA 08F508F5 (00122)          MOV(IOA,A5)                      A5=C2
A33 071CC 08AC08AC (00123)          MOV(P,M4)                        M4=SA*2**(-22)
A34 071CE 08FA0000 (00124) #1       MOV(IOA,M2)\NOP                  M2=X(0,K)=VAR'''
A35 071D0 851C0000 (00125)          MOV(OQ),MUL(M2,M4)\NOP           OQ=X(19,K-2)=VAR,X(1,K)
A36 071D2 3A6B3A6B (00126)          MOV(M3),ALIGN(A3)                A3=X(9,K-1),X(7,K-1)
A37 071D4 1F731F73 (00127)          MOV(A3),EXP(A3)                  A3=X(7,K-1),X(10,K-1)
A38 071D6 85D185D1 (00128)          MOV(A1),MUL(M3,M6)               A1=X(1,K),X(11,K-1)
A39 071D8 3A283A28 (00129)          MOV(M0),ALIGN(A1)                M0=X(10,K-1),X(2,K)
A3A 071DA 32523252 (00130)          MOV(A2),NORM(A2)                 A2=X(2,K),X(3A,K)
A3B 071DC 08B308B3 (00131)          MOV(P,A3)                        A3=X(11,K-1)
A3C 071DE 46704670 (00132)          MOV(A0),ADD(A3,A6)               A0=X(3A,K),X(12,K-1)
A3D 071E0 022B022B (00133)          MOV(M3),R(A1)                    M3=X(12,K-1),SET T BIT B
                   (00134) ;                                         Y X(3A,K)
A3E 071E2 85C085C0 (00135)          MUL(M3,M6)                       X(13,K-1)
A3F 071E4 08F208F2 (00136)          MOV(IOA,A2)                      A2=-16**(-6)
A40 071E6 5A005A00 (00137)          ADDLT(A0,A2)                     X(3B,K)
A41 071E8 08B308B3 (00138)          MOV(P,A3)                        A3=X(13,K-1)
A42 071EA 45704570 (00139)          MOV(A0),ADD(A3,A5)               A0=X93B,K),X(14,K-1)
A43 071EC 440B440B (00140)          MOV(M3),ADD(A0,A4)               M3=X(A4,K-1),X(4,K)
A44 071EE 85C085C0 (00141)          MUL(M3,M6)                       X(15,K-1)
A45 071F0 08F208F2 (00142)          MOV(IOA,A2)                      A2=C1
```

```
A46 071F2 08BA08BA (00143)         MOV(R,M2)                M2=X(4,K)
A47 071F4 85338533 (00144)         MOV(A3),MUL(M2,M5)       A3=X(15,K-1),X(5,K)
A48 071F6 43404340 (00145)         ADD(A2,A3)               X(16,K-1)
A49 071F8 48284828 (00146)         MOV(M3),SUB(A1,A0)       M3=X(16,K-1),X(6,K)
A4A 071FA 850385D3 (00147)         MOV(A3),MUL(M3,M6)       A3=X(5,K),X(17,K-1)
A4B 071FC 08F208F2 (00148)         MOV(IOA,A2)              A2=C0
A4C 071FE 08B008B0 (00149)         MOV(R,NULL)
A4D 07200 08B108B1 (00150)         MOV(P,A1)                A1=X(17,K-1)
A4E 07202 08BE08BE (00151)         MOV(R,M6)                M6=X(6,K)
A4F 07204 84C084C0 (00152)         MUL(M1,M6)               X(8,K)
A50 07206 42204220 (00153)         ADD(A1,A2)               X(18,K-1)
A51 07208 08BF08BF (00154)         MOV(R,M7)                M7=X(18,K-1)
A52 0720A 84718471 (00155)         MOV(A1),MUL(M0,M7)       A1=X(8,K),X(19,K-1)
A53 0720C 47204720 (00156)         ADD(A1,A7)               X(9,K)
A54 0720E 90160034 (00157)         JUMPC(#1,FWI)            WAIT FOR DELAY
A55 07210 20372037 (00158)         CLEAR(WI)                RELEASE APS INPUT
                   (00159) *
A56 07212 08F80000 (00160) DCS     MOV(IOA,M0)\NOP          M0=DC'
A57 07214 08FD08FD (00161)         MOV(IOA,M5)              M5=VAR
A58 07216 000008F8 (00162)         NOP\MOV(IOA,M0)          M0=2**(-MLONG)
A59 07218 84208420 (00163)         MUL(M0,M5)               DC'*VAR;2**(-MLONG)*VAR
A5A 0721A 08BC08BC (00164)         MOV(P,O0)                O0=DC=DC'*VAR;VAR'=2**(-
                   (00165) ;                                MLONG)*VAR
                   (00166) *
A5B 0721C 20320000 (00167) DNESA   CLEAR(RA)\NOP            APU DONE!
A5C 0721E 00000000 (00168)         NOP
A5D 07220 10000000 (00169)         JUMP(0)
          0000005E (00170)         IDCMSS7=#A-IDCMSSA
    07222          (00171)         END IDCMSS7
                   (00172)         EJECT
```

```
                          (00173) *
                          (00174) *>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00175) *
                          (00176)         EVEN
     07222 000072C6       (00177)         ADDR G201$I                   POINTER TO CONSTRUCTED I
                          (00178) ;                                     NSTRUCTION BLOCK
     07224 0000722C       (00179)         ADDR G201$+2*SCLR$            POINTER TO SCALAR BLOCK
     07226 0000           (00180)         DATA 0                        NUMBER OF SCALARS
     07227 00A2           (00181)         DATA G201$Z                   MODULE SIZE
     07228 0000729A       (00182)         ADDR G201$A                   POINTER TO CHAIN ANCHOR
                          (00183)         EVEN                          END ON WORD BOUNDARY
                          (00184) *
                          (00185) G201$   BEGIN APS(G201)
A00  0722A 00203740       (00186)         JSN(G201$0,P2)                SET OUTPUT PC(P2)
A01  0722C 02300030       (00187) SCLR$   SET(RO)                       ENABLE OUTPUT ADDRESSING
A02  0722E 04402000       (00188)         LOAD(BR0,[2])                 COS(0)
A03  07230 06500000       (00189)         LOAD(BR1,MSS)                 N-1
A04  07232 08600000       (00190)         LOAD(BR2,MSS)                 DUMMY
A05  07234 0AE01006       (00191)         LOAD(BR2,[1],TF)              C(0)
A06  07236 0C700000       (00192)         LOAD(BR3,MSS)                 DUMMY
A07  07238 0E700000       (00193)         LOAD(BR3,MSS)                 DUMMY
A08  0723A 10190039       (00194)         ADDL(BR1,1)                   N
A09  0723C 12A9002B       (00195)         ADDB(BR2,BR1,TF)              C(N/2)
A0A  0723E 1489002B       (00196)         ADDB(BR0,BR1,TF)              COS(N/2) (=PI/4)
A0B  07240 1619002B       (00197)         ADDB(BR1,BR1)                 2N
A0C  07242 18090023       (00198)         SUBB(BR0,BR1)                 COS(-N/2)
A0D  07244 1A390015       (00199)         MOVB(BR3,BR2)                 C(N/2)
A0E  07246 1C09003A       (00200)         ADDL(BR0,2)                   COS(-N/2+1)
A0F  07248 1E190035       (00201)         SUBL(BR1,5)                   4*(N-1)-1
                          (00202) * LOOP
A10  0724A 20190039       (00203) #1      ADDL(BR1,1)                   FIX FROM LOOP TEST
A11  0724C 22A20002       (00204)         SUB(BR2,2,TF)                 C(K)
A12  0724E 24BA0002       (00205)         ADD(BR3,2,TF)                 C(N-K)
A13  07250 2689002B       (00206)         ADDB(BR0,BR1,TF)              COS(K)
A14  07252 28090023       (00207)         SUBB(BR0,BR1)
A15  07254 2A8A0200       (00208)         ADD(BR0,2*256,TF)             SIN(K)=COS(N-K)
A16  07256 2C0201FE       (00209)         SUB(BR0,2*256-2)              C(-NEW K)
A17  07258 2E1910B5       (00210)         SUBL(BR1,5),JUMPP(#1)
A18  0725A 30500006       (00211)         LOAD(BR1,6)
A19  0725C 32C20794       (00212) #2      LOAD(BR0,DMYS(1),TF)          NEED 8 DUMMIES
A1A  0725E 341919B1       (00213)         SUBL(BR1,1),JUMPP(#2)
A1B  07260 36C20794       (00214)         LOAD(BR0,DMYS(1),TF)
A1C  07262 38300037       (00215)         SET(WI)                       STALL APS INPUT
A1D  07264 3A000020       (00216)         NOP                           NOTE:6 DUMMIES ARE NEEDE
```

```
                    (00217) ;                                           D
                    (00218) *
A1E 07266 3CC2041C (00219) IMULS   LOAD(BRO,SAS101(1),TF)               IQ=VAR''
A1F 07268 3EC20446 (00220)         LOAD(BRO,SAS98(1),TF)                IQ=1.0/20.=0.05
A20 0726A 40300037 (00221)         SET(WI)                              STALL APS INPUT
A21 0726C 42000020 (00222)         NOP
                    (00223) *
A22 0726E 44C203BB (00224) IEXPS   LOAD(BRO,SAS27(1),TF)                IQ=LOG16(10.0)
A23 07270 46C27150 (00225)         LOAD(BRO,TABLES(1),TF)               IQ=2**(-22)
A24 07272 48BA0002 (00226)         ADD(BRO,WS,TF)                       IQ=2**15
A25 07274 4ABA0002 (00227)         ADD(BRO,WS,TF)                       IQ=4*65*16**(-6)
A26 07276 4CBA0002 (00228)         ADD(BRO,WS,TF)                       IQ=C5
A27 07278 4EBA0002 (00229)         ADD(BRO,WS,TF)                       IQ=C4
A28 0727A 50BA0002 (00230)         ADD(BRO,WS,TF)                       IQ=C3
A29 0727C 52BA0002 (00231)         ADD(BRO,WS,TF)                       IQ=C2
A2A 0727E 54700002 (00232)         LOAD(BR3,WS)                         3LOOPS @ 4 IQ'S=12 IQ'S
A2B 07280 56E2044C (00233) #1      LOAD(BR2,SAS101(1),TF)               IQ=VAR'''
A2C 07282 58C2715F (00234)         LOAD(BRO,TABLES2(1),TF)              IQ=-16**(-6)
A2D 07284 5ABA0002 (00235)         ADD(BRO,WS,TF)                       IQ=C1
A2E 07286 5CBA0002 (00236)         ADD(BRO,WS,TF)                       IQ=C0
A2F 07288 5E392BB1 (00237)         SUBL(BR3,1),JUMPP(#1)                FOR I=0,1...3
A30 0728A 60300037 (00238)         SET(WI)
A31 0728C 62000020 (00239)         NOP
                    (00240) *
A32 0728E 64C2044A (00241) IDCS    LOAD(BRO,SAS100(1),TF)               IQ=DC'
A33 07290 66C2044C (00242)         LOAD(BRO,SAS101(1),TF)               IQ=VAR
A34 07292 6BE203F6 (00243)         LOAD(BR2,SAS50(1),TF)                SA=2**(-KLONG)
                    (00244) *
A35 07294 6A200031 (00245) DNEST   CLEAR(RI)                            INPUT DONE!
A36 07296 6C000020 (00246)         NOP
                    (00247) *
A37 07298 6E300032 (00248) G201SD  SET(RA)                              ENABLE APU
A38 0729A 70C00066 (00249)         LOAD(BWO,[O],TF)                     YR(0)
A39 0729C 72500000 (00250)         LOAD(BW1,MSS)                        N-1
A3A 0729E 74600000 (00251)         LOAD(BW2,MSS)                        DUMMY
A3B 072A0 76BA0002 (00252)         ADD(BWO,2,TF)                        YI(0)
A3C 072A2 7B11002A (00253)         ADDB(BW1,BW1)                        2N-2
A3D 072A4 7AB1002A (00254)         ADDB(BWO,BW1,TF)                     YR(N/2)
A3E 072A6 7CBA0002 (00255)         ADD(BWO,2,TF)                        YI(N/2)
A3F 072A8 7E210010 (00256)         MOVB(BW2,BWO)                        YI(N/2)
A40 072AA 80110033 (00257)         SUBL(BW1,3)                          4*(N/2-1)-1
A41 072AC 82B20006 (00258) #2      SUB(BWO,6,TF)                        YR(K)
A42 072AE 84BA0002 (00259)         ADD(BWO,2,TF)                        YI(K)
A43 072B0 86AA0002 (00260)         ADD(BW2,2,TF)                        YR(N-K)
```

```
 A44 072B2 B6AA0002 (00261)          ADD(BW2,2,TF)                              YI(N-K)
 A45 072B4 BA1141B4 (00262)          SUBL(BW1,4),JUMPP(#2)
                    (00263) *
 A46 072B6 BCC2044C (00264) DMULS    LOAD(BW0,SAS101(1),TF)                     DQ=VAR'''
                    (00265) *
 A47 072B8 BEC20794 (00266) DEXPS    LOAD(BW0,DMYS(1),TF)                       DQ=?
 A48 072BA 90C20794 (00267)          LOAD(BW0,DMYS(1),TF)                       DQ=?
 A49 072BC 92C2044C (00268)          LOAD(BW0,SAS101(1),TF)                     DQ=VAR
                    (00269) *
 A4A 072BE 94C2044A (00270) DDCS     LOAD(BW0,SAS100(1),TF)                     DQ=DC
 A4B 072C0 96C2044C (00271)          LOAD(BW0,SAS101(1),TF)                     DQ=VAR
                    (00272) *
 A4C 072C2 98200030 (00273) DNFSO    CLEAR(R0)                                  OUTPUT DONE!
 A4D 072C4 9A000020 (00274)          NOP
                    (00275) *
           0000729A (00276)          G201$A= #C                                 ASSIGN VALUE TO CHAIN AN
                    (00277) ;                                                   CHOR
     072C6          (00278)          END #A-1
                    (00279) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
     072C6 00000000 (00280) G201$I   DATA 3F'0,0'
      ...
           000000A2 (00281)          G201$Z=#L-G201$
     072CC          (00282)          END
```

```
PAGE     9:      FCB 24H... "WPIDCM(Y,U,V)"      MAKHOUL IDCT CORRECTION

  AFDTSORG:   008FB (00009) (00034)
  COS7$:      00400 (00010)
  CSPU$NOS:   021FC (00011) (00037)
  DC$:        00056 (00160)
  DMY$:       00794 (00012) (00212) (00214) (00266) (00267)
  DNF$A:      0005B (00167)
  DNF$I:      00035 (00245)
  DNF$O:      0004C (00273)
  DRDCTS:     00400 (00013)
  FXP$:       0002A (00114)
  G201$:      0722A (00036) (00179) (00185) (00281)
  G201$A:     0729A (00182) (00276)
  G201$I:     072C6 (00177) (00280)
  G201$O:     00037 (00186) (00248)
  G201$Z:     000A2 (00181) (00281)
  H$:         00001 (00015)
  IDC$:       00032 (00241)
  IDCLP:      00014 (00087) (00104)
  IDCM$:      07166 (00035) (00061)
  IDCM$SA:    00000 (00059) (90067) (00170)
  IDCM$SZ:    0005F (00060) (00170) (00171)
  IFXP$:      00022 (00224)
  IMUL$:      0001F (00219)
  M$$:        00000 (00016) (00189) (00190) (00192) (00193) (00250) (00251)
  ODC$:       0004A (00270)
  OFXP$:      00047 (00266)
  OMUL$:      00046 (00264)
  SA$100:     0044A (00023) (00241) (00270)
  SA$101:     0044C (00024) (00219) (00233) (00242) (00264) (00268) (00271)
  SA$27:      003BB (00020) (00224)
  SA$50:      003F6 (00021) (00243)
  SA$98:      00446 (00022) (00220)
  SCLR$:      00001 (00179) (00187)
  SINZ$:      00600 (00018)
  SIZF$:      000FF (00017)
  SMUL$:      00025 (00108)
  START:      07150 (00025) (00042)
  SVT$:       003B2 (00019) (00020) (00021) (00022) (00023) (00024)
  TABLE$:     07150 (00044) (00225)
  TABLE$2:    0715F (00051) (00234)
  W$:         00002 (00026) (00226) (00227) (00228) (00229) (00230) (00231) (00232) (00235) (00236)
  WW$:        00004 (00027)
  WWH$:       00005 (00028)
  WWW$:       00006 (00029)
```

```
PAGE   10:      FCH 248... "MPIDCM(Y,U,V)"      MAKHOUL IDCT CORRECTION

  ZS:        00003 (00030)
  ZZMS:      00007 (00031)

LINES WITH ERRORS:    0  (MAP VERSION R00101.10)   E=    0
```

```
                   (00001) '  FCB 249... "MPASRT(U)"  SORT DCT1(.) COEFFICIENTS
                   (00002) *                                    ORIGINATED:13-DEC-79
                   (00003) *                                    UPDATED:27-MAY-80
                   (00004) *  DEFINE GLOBAL SYMBOLS
                   (00005)         OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                   (00006)         OPADD DOF, (3 .LS. 10) + (12 .LS. 5) + $1B
                   (00007)         OPADD WAFNE, (1 .LS. 10) + (12 .LS. 5) + $1A
         00000BEB  (00008)         AFDTSORG=$BEB
         0001FFC0  (00009)         APSSMEM=$1FFC0
         000021FC  (00010)         CSPUSNOS=$21FC
         00000928  (00011)         DCTS=D'2344'
         00000200  (00012)         DCT1S=D'512'
         00000400  (00013)         DCT2S=D'1024'
         00000928  (00014)         DCTMS=D'2344'
         00000794  (00015)         DMYS=$794
         00000003  (00016)         #M=3
         00000001  (00017)         HS=1
         00000100  (00018)         INCRES=D'256'
         00000000  (00019)         IORDRS=D'0'
         00000502  (00020)         ISVTS=$502
         00000502  (00021)         ISAS00=ISVTS+D'0'
         00000503  (00022)         ISAS01=ISVTS+D'1'
         00000000  (00023)         MSS=0
         0000000A  (00024)         OVRSLAP=D'10'
         000000FF  (00025)         SIZES=D'256'-D'1'
         00000382  (00026)         SVTS=$0382
         00007300  (00027)         START=$7300
         00000C00  (00028)         TABLS0=D'3072'
         00000D00  (00029)         TABLS1=TABLS0+INCRES
         00000E00  (00030)         TABLS2=TABLS1+INCRES
         00000F00  (00031)         TABLS3=TABLS2+INCRES
         00000A00  (00032)         TMP2S=D'2560'
         00000E00  (00033)         TMP3S=D'3584'
         00000C00  (00034)         TMP4S=D'3072'
         00000002  (00035)         WS=2
         00000003  (00036)         ZS=3
                   (00037) *
                   (00038) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
         00000BBE  (00039)         #L=AFDTSORG+3*2*(249-128)                    FCB 249
00BBE    001F730B  (00040)         ADDR SORTS(R7,1)
00BC0    001F7430  (00041)         ADDR G106S(R7,1)
00BC2    001021FC  (00042)         ADDR CSPUSNOS(,1,0)
                   (00043)         EJECT
```

```
                    (00044) *
                    (00045) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00046) *
                    (00047)          BUS 2
          00001E14 (00048)          *L=D'7700'
                    (00049)          EVEN
01E14 3C50         (00050) INSTRS0 DATA INS0*D'512'+X'50',X'0000'
01E15 0000
01E16 3450         (00051) INSTRS1 DATA INS1*D'512'+X'50',X'0000'
01E17 0000
01E18 2C50         (00052) INSTRS2 DATA INS2*D'512'+X'50',X'0000'
01E19 0000
01E1A 2450         (00053) INSTRS3 DATA INS3*D'512'+X'50',X'0000'
01E1B 0000
01E1C 662F         (00054) INSTRS4 DATA INDS4*D'512'+X'2F',X'0000'
01E1D 0000
01E1E 6A2F         (00055) INSTRS5 DATA INDS5*D'512'+X'2F',X'0000'
01E1F 0000
01E20 562E         (00056) INSTRS6 DATA INDS6*D'512'+X'2E',X'0000'
01E21 0000
01E22 5A2E         (00057) INSTRS7 DATA INDS7*D'512'+X'2E',X'0000'
01E23 0000
01E24 0000         (00058) INTGRS   DATA D'0',D'1',D'2',D'3',D'4',D'5',D'6',D'7'
01E25 0001
01E26 0002
01E27 0003
01E28 0004
01E29 0005
01E2A 0006
01E2B 0007
01E2C 0008         (00059)          DATA D'8',D'9',D'10',D'11',D'12',D'13',D'14',D'15'
01E2D 0009
01E2E 000A
01E2F 000B
01E30 000C
01E31 000D
01E32 000E
01E33 000F
01E34 0010         (00060)          DATA D'16',D'17',D'18',D'19',D'20',D'21',D'22',D'23'
01E35 0011
01E36 0012
01E37 0013
01E38 0014
01E39 0015
```

```
01E3A 0016
01E3B 0017
01E3C 0018     (00061)          DATA D'24',D'25',D'26',D'27',D'28',D'29',D'30',D'31'
01E3D 0019
01E3E 001A
01E3F 001B
01E40 001C
01E41 001D
01E42 001E
01E43 001F
01E44 0020     (00062)          DATA D'32',D'33',D'34',D'35',D'36',D'37',D'38',D'39'
01E45 0021
01E46 0022
01E47 0023
01E48 0024
01E49 0025
01E4A 0026
01E4B 0027
01E4C 0028     (00063)          DATA D'40',D'41',D'42',D'43',D'44',D'45',D'46',D'47'
01E4D 0029
01E4E 002A
01E4F 002B
01E50 002C
01E51 002D
01E52 002E
01E53 002F
01E54 0030     (00064)          DATA D'48',D'49',D'50',D'51',D'52',D'53',D'54',D'55'
01E55 0031
01E56 0032
01E57 0033
01E58 0034
01E59 0035
01E5A 0036
01E5B 0037
01E5C 0038     (00065)          DATA D'56',D'57',D'58',D'59',D'60',D'61',D'62',D'63'
01E5D 0039
01E5E 003A
01E5F 003B
01E60 003C
01E61 003D
01E62 003E
01E63 003F
01E64 0040     (00066)          DATA D'64',D'65',D'66',D'67',D'68',D'69',D'70',D'71'
01E65 0041
```

```
01E66 0042
01E67 0043
01E68 0044
01E69 0045
01E6A 0046
01E6B 0047
01E6C 0048      (00067)         DATA D'72',D'73',D'74',D'75',D'76',D'77',D'78',D'79'
01E6D 0049
01E6E 004A
01E6F 004B
01E70 004C
01E71 004D
01E72 004E
01E73 004F
01E74 0050      (00068)         DATA D'80',D'81',D'82',D'83',D'84',D'85',D'86',D'87'
01E75 0051
01E76 0052
01E77 0053
01E78 0054
01E79 0055
01E7A 0056
01E7B 0057
01E7C 0058      (00069)         DATA D'88',D'89',D'90',D'91',D'92',D'93',D'94',D'95'
01E7D 0059
01E7E 005A
01E7F 005B
01E80 005C
01E81 005D
01E82 005E
01E83 005F
01E84 0060      (00070)         DATA D'96',D'97',D'98',D'99',D'100',D'101',D'102',D'103'
01E85 0061
01E86 0062
01E87 0063
01E88 0064
01E89 0065
01E8A 0066
01E8B 0067
01E8C 0068      (00071)         DATA D'104',D'105',D'106',D'107',D'108',D'109',D'110',D'111'
01E8D 0069
01E8E 006A
01E8F 006B
01E90 006C
01E91 006D
```

```
01F92 006E
01F93 006F
01F94 0070      (00072)         DATA D'112',D'113',D'114',D'115',D'116',D'117',D'118',D'119'
01F95 0071
01F96 0072
01F97 0073
01F98 0074
01F99 0075
01F9A 0076
01F9B 0077
01F9C 0078      (00073)         DATA D'120',D'121',D'122',D'123',D'124',D'125',D'126',D'127'
01F9D 0079
01F9E 007A
01F9F 007B
01FA0 007C
01FA1 007D
01FA2 007E
01FA3 007F
01FA4 0080      (00074)         DATA D'128',D'129',D'130',D'131',D'132',D'133',D'134',D'135'
01FA5 0081
01FA6 0082
01FA7 0083
01FA8 0084
01FA9 0085
01FAA 0086
01FAB 0087
01FAC 0088      (00075)         DATA D'136',D'137',D'138',D'139',D'140',D'141',D'142',D'143'
01FAD 0089
01FAE 008A
01FAF 008B
01FB0 008C
01FB1 008D
01FB2 008E
01FB3 008F
01FB4 0090      (00076)         DATA D'144',D'145',D'146',D'147',D'148',D'149',D'150',D'151'
01FB5 0091
01FB6 0092
01FB7 0093
01FB8 0094
01FB9 0095
01FBA 0096
01FBB 0097
01FBC 0098      (00077)         DATA D'152',D'153',D'154',D'155',D'156',D'157',D'158',D'159'
01FBD 0099
```

```
01FBE 009A
01FBF 009B
01FC0 009C
01FC1 009D
01FC2 009E
01FC3 009F
01FC4 00A0        (00078)           DATA D'160',D'161',D'162',D'163',D'164',D'165',D'166',D'167'
01FC5 00A1
01FC6 00A2
01FC7 00A3
01FC8 00A4
01FC9 00A5
01FCA 00A6
01FCB 00A7
01FCC 00A8        (00079)           DATA D'168',D'169',D'170',D'171',D'172',D'173',D'174',D'175'
01FCD 00A9
01FCE 00AA
01FCF 00AB
01FD0 00AC
01FD1 00AD
01FD2 00AE
01FD3 00AF
01FD4 00B0        (00080)           DATA D'176',D'177',D'178',D'179',D'180',D'181',D'182',D'183'
01FD5 00B1
01FD6 00B2
01FD7 00B3
01FD8 00B4
01FD9 00B5
01FDA 00B6
01FDB 00B7
01FDC 00B8        (00081)           DATA D'184',D'185',D'186',D'187',D'188',D'189',D'190',D'191'
01FDD 00B9
01FDE 00BA
01FDF 00BB
01FE0 00BC
01FE1 00BD
01FE2 00BE
01FE3 00BF
01FE4 00C0        (00082)           DATA D'192',D'193',D'194',D'195',D'196',D'197',D'198',D'199'
01FE5 00C1
01FE6 00C2
01FE7 00C3
01FE8 00C4
01FE9 00C5
```

```
01EEA 00C6
01EEB 00C7
01EEC 00C8      (00083)         DATA D'200',D'201',D'202',D'203',D'204',D'205',D'206',D'207'
01EED 00C9
01EEE 00CA
01EEF 00CB
01EF0 00CC
01EF1 00CD
01EF2 00CE
01EF3 00CF
01EF4 00D0      (00084)         DATA D'208',D'209',D'210',D'211',D'212',D'213',D'214',D'215'
01EF5 00D1
01EF6 00D2
01EF7 00D3
01EF8 00D4
01EF9 00D5
01EFA 00D6
01EFB 00D7
01EFC 00D8      (00085)         DATA D'216',D'217',D'218',D'219',D'220',D'221',D'222',D'223'
01EFD 00D9
01EFE 00DA
01EFF 00DB
01F00 00DC
01F01 00DD
01F02 00DE
01F03 00DF
01F04 00E0      (00086)         DATA D'224',D'225',D'226',D'227',D'228',D'229',D'230',D'231'
01F05 00E1
01F06 00E2
01F07 00E3
01F08 00E4
01F09 00E5
01F0A 00E6
01F0B 00E7
01F0C 00E8      (00087)         DATA D'232',D'233',D'234',D'235',D'236',D'237',D'238',D'239'
01F0D 00E9
01F0E 00EA
01F0F 00EB
01F10 00EC
01F11 00ED
01F12 00EE
01F13 00EF
01F14 00F0      (00088)         DATA D'240',D'241',D'242',D'243',D'244',D'245',D'246',D'247'
01F15 00F1
```

```
01F16 00F2
01F17 00F3
01F18 00F4
01F19 00F5
01F1A 00F6
01F1B 00F7
01F1C 00F8      (00089)          DATA D'248',D'249',D'250',D'251',D'252',D'253',D'254',D'255'
01F1D 00F9
01F1E 00FA
01F1F 00FB
01F20 00FC
01F21 00FD
01F22 00FE
01F23 00FF
                (00090)         BUS 1
     00007300   (00091)         #I=START
                (00092) ;T1$       DATA 1F'0.713'
                (00093) ;T2$       DATA 1F'2.5'
                (00094) ;T3$       DATA 1F'7.5'
07300 BE77A2C0  (00095) T1$     DATA 1F'-0.48802602'
07302 0A934F41  (00096) T2$     DATA 1F'1.32192810'
07304 17414FC1  (00097) T3$     DATA 1F'2.9068906'
                (00098)         EJECT
```

```
                       (00099) *
                       (00100) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                       (00101) *
                       (00102)          EVEN
    07306 0000         (00103)          DATA SORTSSA                       START ADDRESS
    07307 0090         (00104)          DATA SORTSSZ                       MODULE SIZE
                       (00105) SORTS    BEGIN APU(SORT)
          00000000     (00106)          #A=0
A00 07308 08F10000     (00107) SORTSSA  MOV(IQA,A1)\NOP                    A1=THRSH1
A01 0730A 20250000     (00108)          CLEAR(G1)\NOP                      RESET STALL FLAG
A02 0730C 000008F5     (00109)          NOP\MOV(IQA,A5)                    A5=THRSH2
A03 0730E 20500000     (00110)          SET(RO)\NOP                        ENABLE OUTPUT ADDRESSING
A04 07310 08F30000     (00111)          MOV(IQA,A3)\NOP                    A3=THRSH3
A05 07312 000008F7     (00112)          NOP\MOV(IQA,A7)                    A7=1
A06 07314 00000810     (00113)          NOP\MOV(ZERO,A0)                   A0=0
A07 07316 00000811     (00114)          NOP\MOV(ZERO,A1)                   A1=0
A08 07318 00000812     (00115)          NOP\MOV(ZERO,A2)                   A2=0
A09 0731A 00000813     (00116)          NOP\MOV(ZERO,A3)                   A3=0
A0A 0731C 08F008F4     (00117)          MOV(IQA,A0)\MOV(IQA,A4)            A0=D=DCT1(I);A4=D
A0B 0731E 49004080     (00118)          SUB(A0,A1)\SUB(A4,A5)              D-THRSH1;D-THRSH2
A0C 07320 08800880     (00119) INCSI    MOV(R,NULL)                        FORCE END
A0D 07322 911E0030     (00120)          JUMPS(LVLS0,T1)                    IF T1=1,LEVEL 0
A0E 07324 911F0026     (00121)          JUMPS(LVLS1,T2)                    IF T2=1,LEVEL 1
A0F 07326 48000000     (00122)          SUB(A0,A3)\NOP                     D-THRSH3
A10 07328 08800000     (00123)          MOV(R,NULL)\NOP                    FORCE END
A11 0732A 911E001C     (00124)          JUMPS(LVLS2,T1)                    IF T1=1,LEVEL 2
A12 0732C 204B0000     (00125) LVLS3    SET(AF3)\NOP                       INFORM APS OF LVL3
A13 0732E 20530000     (00126)          SET(UN)\NOP                        FIXED POINT ADD
A14 07330 00004760     (00127)          NOP\ADD(A3,A7)                     LVL3 CNT+1
A15 07332 08F008F4     (00128)          MOV(IQA,A0)\MOV(IQA,A4)            A0=D=DCT1(I+1);A4=D
A16 07334 08FC0893     (00129)          MOV(IQA,OQ)\MOV(R,A3)              OQ=I;LVL3=LVL3+1
A17 07336 20330000     (00130)          CLEAR(UN)\NOP                      RESUME FLOATING POINT!
A18 07338 49004080     (00131)          SUB(A0,A1)\SUB(A4,A5)              D-THRSH1;D-THRSH2
A19 0733A 91080019     (00132) *1       JUMPS(*1,AF3)
A1A 0733C 9016000C     (00133)          JUMPC(INCSI,FWI)                   FOR I=0,1,2...LTH-1
A1B 0733E 1045003A     (00134)          JUMP(CNTSOUT,G1),SET               GET LEVEL COUNTS
A1C 07340 204A0000     (00135) LVLS2    SET(AF2)\NOP                       INFORM APS OF LVL2
A1D 07342 20530000     (00136)          SET(UN)\NOP                        FIXED POINT ADD
A1E 07344 00004740     (00137)          NOP\ADD(A2,A7)                     LVL2 CNT+1
A1F 07346 08F008F4     (00138)          MOV(IQA,A0)\MOV(IQA,A4)            A0=D=DCT1(I+1);A4=D
A20 07348 08FC0892     (00139)          MOV(IQA,OQ)\MOV(R,A2)              OQ=I;LVL2=LVL2+1
A21 0734A 20330000     (00140)          CLEAR(UN)\NOP                      RESUME FLOATING POINT!
A22 0734C 49004080     (00141)          SUB(A0,A1)\SUB(A4,A5)              D-THRSH1;D-THRSH2
A23 0734E 910A0023     (00142) *1       JUMPS(*1,AF2)
```

```
A24 07350 9016000C (00143)          JUMPC(INCS1,FWI)              FOR I=0,1,2...LTH-1
A25 07352 1045003A (00144)          JUMP(CNTSOUT,G1),SET          GET LEVEL COUNTS
A26 07354 20490000 (00145) LVLS1    SET(AF1)\NOP                  INFORM APS OF LVL1
A27 07356 20530000 (00146)          SET(UN)\NOP                   FIXED POINT ADD
A28 07358 00004720 (00147)          NOP\ADD(A1,A7)                LVL1 CNT+1
A29 0735A 08F008F4 (00148)          MOV(IQA,A0)\MOV(IQA,A4)       A0=D=DCT1(I+1);A4=D
A2A 0735C 08FC0891 (00149)          MOV(IQA,OQ)\MOV(R,A1)         OQ=I;LVL1=LVL1+1
A2B 0735E 20330000 (00150)          CLEAR(UN)\NOP                 RESUME FLOATING POINT!
A2C 07360 49004D80 (00151)          SUB(A0,A1)\SUB(A4,A5)         D-THRSH1;D-THRSH2
A2D 07362 9109002D (00152) #1       JUMPS(#1,AF1)
A2E 07364 9016000C (00153)          JUMPC(INCS1,FWI)              FOR I=0,1,2...LTH-1
A2F 07366 1045003A (00154)          JUMP(CNTSOUT,G1),SET          GET LEVEL COUNTS
A30 07368 20480000 (00155) LVLS0    SET(AF0)\NOP                  INFORM APS OF LVL0
A31 0736A 20530000 (00156)          SET(UN)\NOP                   FIXED POINT ADD
A32 0736C 00004700 (00157)          NOP\ADD(A0,A7)                LVL0 CNT+1
A33 0736E 08F008F4 (00158)          MOV(IQA,A0)\MOV(IQA,A4)       A0=D=DCT1(I+1);A4=D
A34 07370 08FC0890 (00159)          MOV(IQA,OQ)\MOV(R,A0)         OQ=I;LVL0=LVL0+1
A35 07372 20330000 (00160)          CLEAR(UN)\NOP                 RESUME FLOATING POINT!
A36 07374 49004D80 (00161)          SUB(A0,A1)\SUB(A4,A5)         D-THRSH1;D-THRSH2
A37 07376 91080037 (00162) #1       JUMPS(#1,AF0)
A38 07378 9016000C (00163)          JUMPC(INCS1,FWI)              FOR I=0,1,2...LTH-1
A39 0737A 20450000 (00164)          SET(G1)\NOP                   GET LEVEL COUNTS
A3A 0737C 00000200 (00165) CNTSOUT  NOP\R(A0)                     R=LVL0 CNT
A3B 0737E 0000023C (00166)          NOP\MOV(OQ),R(A1)             R=LVL1 CNT
A3C 07380 0000025C (00167)          NOP\MOV(OQ),R(A2)             R=LVL2 CNT
A3D 07382 0000027C (00168)          NOP\MOV(OQ),R(A3)             R=LVL3 CNT
A3E 07384 0000089C (00169)          NOP\MOV(R,OQ)                 OQ=LVL3 CNT(0 UPTO LTH)
A3F 07386 081C081C (00170)          MOV(ZERO,OQ)                  OQ=0 FOR INSTRS4+1...
A40 07388 081C081C (00171)          MOV(ZERO,OQ)                  THRU INSTRS7+1
A41 0738A 00000000 (00172) #1       NOP                           WAIT FOR...
A42 0738C 901C0041 (00173)          JUMPC(#1,FO)                  "LEVELS" TO SETTLE
A43 0738E 20370000 (00174)          CLEAR(WI)\NOP                 RELEASE APS INPUT
A44 07390 20500000 (00175)          SET(RO)\NOP                   RELEASE APS OUTPUT
A45 07392 08FC0000 (00176)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS0
A46 07394 08FC0000 (00177)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS1
A47 07396 08FC0000 (00178)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS2
A48 07398 08FC0000 (00179)          MOV(IQA,OQ)\NOP               SCATTER-WRITE INTO INS3
A49 0739A 00000000 (00180) #2       NOP                           WAIT FOR...
A4A 0739C 901C0049 (00181)          JUMPC(#2,FO)                  "WRITES" TO SETTLE
A4B 0739E 20370000 (00182)          CLEAR(WI)\NOP                 RELEASE APS INPUT
A4C 073A0 20500000 (00183)          SET(RO)\NOP                   RELEASE APS OUTPUT
A4D 073A2 08FC0000 (00184) ORDRS1   MOV(IQA,OQ)\NOP               OQ=IORDR(I)
A4E 073A4 08FC0000 (00185)          MOV(IQA,OQ)\NOP               OQ=IORDR(I+1)
A4F 073A6 08FC0000 (00186)          MOV(IQA,OQ)\NOP               OQ=IORDR(I+2)
```

```
A50 073A8 0HFC0000 (00187)         MOV(IOA,OO)\NOP          OO=IORDR(I+3)
A51 073AA 9016004D (00188)         JUMPC(ORDRS1,FWI)        FOR I=0,1,2...LTH-1
A52 073AC 202H0000 (00189)         CLEAR(AFO)\NOP           CLEAR "SKIP" FLG
A53 073AE 081C081C (00190)         MOV(ZERO,OO)             OO=BUFFER ZONE=0
A54 073B0 081C081C (00191)         MOV(ZERO,OO)             OO=BUFFER ZONE=0
A55 073B2 081C081C (00192)         MOV(ZERO,OO)             OO=BUFFER ZONE=0
A56 073B4 081C081C (00193)         MOV(ZERO,OO)             OO=BUFFER ZONE=0
A57 073B6 081C081C (00194)         MOV(ZERO,OO)             OO=BUFFER ZONE=0
                   (00195) *
A58 073B8 20370000 (00196) SHUFLSV CLEAR(WI)\NOP            RELEASE APS INPUT
A59 073BA 20530000 (00197)         SET(UN)\NOP              FIXED POINT ADD
A5A 073BC 08F00000 (00198)         MOV(IOA,A0)\NOP          A0=J
A5B 073BE 000008F0 (00199)         NOP\MOV(IOA,A0)          A0=K
A5C 073C0 40004000 (00200)         ADD(A0,A0)               2J;2K
A5D 073C2 08F10000 (00201)         MOV(IOA,A1)\NOP          A1=J'
A5E 073C4 000008F1 (00202)         NOP\MOV(IOA,A1)          A1=K'
A5F 073C6 413C413C (00203)         MOV(OO),ADD(A1,A1)       2J->S6+1,2J';2K->S7+1,2K
                   (00204) ;                                '
A60 073C8 08FC0000 (00205)         MOV(IOA,OO)\NOP          INSTRS4 INTO INDS4
A61 073CA 000008FC (00206)         NOP\MOV(IOA,OO)          INSTRS5 INTO INDS5
A62 073CC 08FC0000 (00207)         MOV(IOA,OO)\NOP          NULL=TMP2(J*)
A63 073CE 08FC0000 (00208)         MOV(IOA,OO)\NOP          NULL=DCT1(J*)
A64 073D0 08FC0000 (00209)         MOV(IOA,OO)\NOP          NULL=DCTM(J*)
A65 073D2 08FC0000 (00210)         MOV(IOA,OO)\NOP          NULL=TMP2(K*)
A66 073D4 08FC0000 (00211)         MOV(IOA,OO)\NOP          NULL=DCT1(K*)
A67 073D6 08FC0000 (00212)         MOV(IOA,OO)\NOP          NULL=DCTM(K*)
A68 073D8 08F20000 (00213)         MOV(IOA,A2)\NOP          A2=J''
A69 073DA 000008F2 (00214)         NOP\MOV(IOA,A2)          A2=K''
A6A 073DC 425C425C (00215)         MOV(OO),ADD(A2,A2)       2J'->S4+1,2J'';2K'->S5+1
                   (00216) ;                                ,2K''
A6B 073DE 08FC0000 (00217)         MOV(IOA,OO)\NOP          INSTRS6 INTO INDS6
A6C 073E0 000008FC (00218)         NOP\MOV(IOA,OO)          INSTRS7 INTO INDS7
A6D 073E2 08FC0000 (00219)         MOV(IOA,OO)\NOP          NULL=TMP2(J**)
A6E 073E4 08FC0000 (00220)         MOV(IOA,OO)\NOP          NULL=DCT1(J**)
A6F 073E6 08FC0000 (00221)         MOV(IOA,OO)\NOP          NULL=DCTM(J**)
A70 073E8 08FC0000 (00222)         MOV(IOA,OO)\NOP          NULL=TMP2(K**)
A71 073EA 08FC0000 (00223)         MOV(IOA,OO)\NOP          NULL=DCT1(K**)
A72 073EC 08FC0000 (00224)         MOV(IOA,OO)\NOP          NULL=DCTM(K**)
A73 073EE 911D008E (00225) SHUFLS  JUMPS(DNESA,FI)          I/O HAS SETTLED! DONE?
A74 073F0 901C0074 (00226) STALS   JUMPC(STALS,EO)          WAIT FOR OUTPUT TO CLEAR
A75 073F2 204B0000 (00227)         SET(AFO)\NOP             SET OUTPUT "SKIP" FLG
A76 073F4 20500000 (00228)         SET(RO)\NOP              RELEASE APS OUTPUT
A77 073F6 08F10000 (00229)         MOV(IOA,A1)\NOP          A1=J'
A78 073F8 000008F1 (00230)         NOP\MOV(IOA,A1)          A1=K'
```

```
A79 073FA 413C413C (00231)          MOV(00),ADD(A1,A1)          2J->$6+1,2J';2K->$7+1,2K
                   (00232) ;                                    '
A7A 073FC 0BFC0000 (00233)          MOV(IOA,00)\NOP             INSTRS4 INTO INDS4
A7B 073FE 00000BFC (00234)          NOP\MOV(IOA,00)             INSTRS5 INTO INDS5
A7C 07400 0BFC0000 (00235)          MOV(IOA,00)\NOP             DCT2(I)=TMP2(J*)
A7D 07402 0BFC0000 (00236)          MOV(IOA,00)\NOP             TMP4(I)=DCT1(J*)
A7E 07404 0BFC0000 (00237)          MOV(IOA,00)\NOP             TMP3(I)=DCTM(J*)
A7F 07406 00000BFC (00238)          NOP\MOV(IOA,00)             DCT2(I+1)=TMP2(K*)
A80 07408 00000BFC (00239)          NOP\MOV(IOA,00)             TMP4(I+1)=DCT1(K*)
A81 0740A 00000BFC (00240)          NOP\MOV(IOA,00)             TMP3(I+1)=DCTM(K*)
A82 0740C 0BF20000 (00241)          MOV(IOA,A2)\NOP             A2=J''
A83 0740E 00000BF2 (00242)          NOP\MOV(IOA,A2)             A2=K''
A84 07410 425C425C (00243)          MOV(00),ADD(A2,A2)          2J'->$4+1,2J'';2K'->$5+1
                   (00244) ;                                    ,2K''
A85 07412 0BFC0000 (00245)          MOV(IOA,00)\NOP             INSTRS6 INTO INDS6
A86 07414 00000BFC (00246)          NOP\MOV(IOA,00)             INSTRS7 INTO INDS7
A87 07416 0BFC0000 (00247)          MOV(IOA,00)\NOP             DCT2(I+2)=TMP2(J*)
A88 07418 0BFC0000 (00248)          MOV(IOA,00)\NOP             TMP4(I+2)=DCT1(J*)
A89 0741A 0BFC0000 (00249)          MOV(IOA,00)\NOP             TMP3(I+2)=DCTM(J*)
A8A 0741C 00000BFC (00250)          NOP\MOV(IOA,00)             DCT2(I+3)=TMP2(K*)
A8B 0741E 00000BFC (00251)          NOP\MOV(IOA,00)             TMP4(I+3)=DCT1(K*)
A8C 07420 00000BFC (00252)          NOP\MOV(IOA,00)             TMP3(I+3)=DCTM(K*)
A8D 07422 10000073 (00253)          JUMP(SHUFLS)                "JUMP(...)" SETTLES I/O
                   (00254)
A8E 07424 20320000 (00255) DNESA    CLEAR(RA)\NOP               APU DONE!
A8F 07426 00000000 (00256)          NOP
          00000090 (00257)          SORTSSZ=#A-SORTSSA
    07428          (00258)          END SORTSSZ
                   (00259)          EJECT
```

```
                          (00260) *
                          (00261) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00262) *
                          (00263)          EVEN
    0742B 00007530 (00264)          ADDR G106$1                   POINTER TO CONSTRUCTED I
                          (00265) ;                                         NSTRUCTION BLOCK
    0742A 00007432 (00266)          ADDR G106$+2*SCLRS            POINTER TO SCALAR BLOCK
    0742C 0000     (00267)          DATA 0                        NUMBER OF SCALARS
    0742D 0102     (00268)          DATA G106$Z                   MODULE SIZE
    0742F 00007476 (00269)          ADDR G106$A                   POINTER TO CHAIN ANCHOR
                          (00270)          EVEN                          END ON WORD BOUNDARY
                          (00271) *
                          (00272) G106$    BEGIN APS(G106)
A00 07430 00103940 (00273)          JSN(G106$1,P1)                SET INPUT PC(P1)
A01 07432 02C27300 (00274) SCLRS    LOAD(BR0,T1S(1),TF)           IQ=T1
A02 07434 04C27302 (00275)          LOAD(BR0,T2S(1),TF)           IQ=T2
A03 07436 06C27304 (00276)          LOAD(BR0,T3S(1),TF)           IQ=T3
A04 07438 09420503 (00277)          LOAD(BR0,ISAS01(1),TF)        IQ=1
A05 0743A 0AC60A00 (00278)          LOAD(BR0,TMP2S(3),TF)         IQ=TMP2(0)
A06 0743C 0C5000FF (00279)          LOAD(BR1,SIZES)               TMP2(.) SIZE-1
A07 0743E 0E641F23 (00280)          LOAD(BR2,INTGRS-1(2))         TABL(.) BA-1
A08 07440 108A0002 (00281) IINCSI   ADD(BR0,WS,TF)                IQ=TMP2(I+1)
A09 07442 132A0001 (00282)          ADD(BR2,HS,TF)                IQ=TABL(I)=I
A0A 07444 141908B1 (00283)          SUBL(BR1,1),JUMPP(IINCSI)     FOR I=0,1,2...LTH-1
A0B 07446 16300037 (00284)          SET(#1)                       STALL APS INPUT
A0C 07448 18500002 (00285)          LOAD(BR1,2)                   3+1=4 OVERWRITES
A0D 0744A 1AC41F14 (00286)          LOAD(BR0,INSTRS0(2),TF)       IQ="LOAD(BR1,M$$)"
A0E 0744C 1C8A0002 (00287) #1       ADD(BR0,WS,TF)                IQ="LOAD(BR1,M$$)"
A0F 0744E 1E190EB1 (00288)          SUBL(BR1,1),JUMPP(#1)         INS2,INS1 & INS0
A10 07450 20300037 (00289)          SET(#1)                       STALL APS INPUT
A11 07452 22460EFF (00290) L3$SU    LOAD(BR0,TABL$3-1(3))         TABLE3(.) BA-1
A12 07454 24500000 (00291) INS3     LOAD(BR1,M$$)                 M$$=LEVEL 3 COUNT
A13 07456 261915E1 (00292) TSTSL3   SUBL(BR1,1),JUMPN(L2$SU)      PICKUP TILL NO MORE
A14 07458 29091379 (00293)          ADDL(BR0,HS,TF),JUMP(TSTSL3)  IQ=TABLE3(I)
A15 0745A 2A4600FF (00294) L2$SU    LOAD(BR0,TABL$2-1(3))         TABLE2(.) BA-1
A16 0745C 2C500000 (00295) INS2     LOAD(BR1,M$$)                 M$$=LEVEL 2 COUNT
A17 0745E 2F1919F1 (00296) TSTSL2   SUBL(BR1,1),JUMPN(L1$SU)      PICKUP TILL NO MORE
A18 07460 31091779 (00297)          ADDL(BR0,HS,TF),JUMP(TSTSL2)  IQ=TABLE2(I)
A19 07462 32460CFF (00298) L1$SU    LOAD(BR0,TABL$1-1(3))         TABLE1(.) BA-1
A1A 07464 34500000 (00299) INS1     LOAD(BR1,M$$)                 M$$=LEVEL 1 COUNT
A1B 07466 361910F1 (00300) TSTSL1   SUBL(BR1,1),JUMPN(L0$SU)      PICKUP TILL NO MORE
A1C 07468 39091B79 (00301)          ADDL(BR0,HS,TF),JUMP(TSTSL1)  IQ=TABLE1(I)
A1D 0746A 3A460BFF (00302) L0$SU    LOAD(BR0,TABL$0-1(3))         TABLE0(.) BA-1
A1E 0746C 3C500000 (00303) INS0     LOAD(BR1,M$$)                 M$$=LEVEL 0 COUNT
```

```
A1F 0746F 3F1921F1 (00304) TSTSL0  SUBL(BR1,1),JUMPN(STALSI)          PICKUP TILL NO MORE
A20 07470 41091F79 (00305)         ADDL(BR0,HS,TF),JUMP(TSTSL0)       IO=TABLE0(I)
A21 07472 42300037 (00306) STALSI  SET(WI)                            STALL APS INPUT
A22 07474 44500103 (00307)         LOAD(BR1,SIZFS+4)                  IORDR(.) SIZE-1+4
                   (00308) *
A23 07476 47400000 (00309)         LOAD(BR0,IOI,TF)                   IORDR(.) A,IO=IORDR=J
A24 07478 48800000 (00310)         LOAD(BR2,MSS)                      DUMMY OP FOR EXEC
A25 0747A 480A0000 (00311)         ADD(BR0,MSS,TF)                    IO=IORDR(I)=K
A26 0747C 4D0A0001 (00312) INDSJ   ADD(BR0,HS,TF)                     IO=J'=IORDR(I+2)
A27 0747E 4E0A0001 (00313)         ADD(BR0,HS,TF)                     IO=K'=IORDR(I+3)
A28 07480 50E41F1C (00314)         LOAD(BR2,INSTRS4(2),TF)            OVERWRITE INDS4
A29 07482 52F41F1E (00315)         LOAD(BR2,INSTRS5(2),TF)            OVERWRITE INDS5
A2A 07484 54600000 (00316)         LOAD(BR2,MSS)                      RESET BR2 TO 0!
A2B 07486 562F0000 (00317) INDS6   ADD(BR2,MSS,NA,C)                  MSS=2J,PC->PC1
A2C 07488 58600000 (00318)         LOAD(BR2,MSS)                      RESET BR2 TO 0!
A2D 0748A 5A2F0000 (00319) I' S7   ADD(BR2,MSS,NA,C)                  MSS=2J,PC->PC1
A2E 0748C 5D0A0001 (00320)         ADD(BR0,HS,TF)                     IO=J''
A2F 0748E 5F0A0001 (00321)         ADD(BR0,HS,TF)                     IO=K''
A30 07490 60E41F20 (00322)         LOAD(BR2,INSTRS6(2),TF)            OVERWRITE INDS6
A31 07492 62F41F22 (00323)         LOAD(BR2,INSTRS7(2),TF)            OVERWRITE INDS7
A32 07494 64600000 (00324)         LOAD(BR2,MSS)                      RESET BR2 TO 0!
A33 07496 662F0000 (00325) INDS4   ADD(BR2,MSS,NA,C)                  MSS=2J,PC->PC1
A34 07498 68600000 (00326)         LOAD(BR2,MSS)                      RESET BR2 TO 0!
A35 0749A 6A2F0000 (00327) INDS5   ADD(BR2,MSS,NA,C)                  MSS=2J,PC->PC1
A36 0749C 6C1926B4 (00328)         SUBL(BR1,4),JUMPP(INDSJ)           FOR I=0,1,2...LTH+1
                   (00329) *
A37 0749E 6E200031 (00330) DNFSI   CLEAR(RI)                          INPUT DONE!
A38 074A0 70000020 (00331)         NOP
A39 074A2 72204240 (00332) G106S1  JSN(G106S0,P2)                     SET OUTPUT PC
A3A 074A4 74760A00 (00333) TOPSP1  LOAD(BR3,TMP2S(3))                 TMP2(.) BA
A3B 074A6 76B9002D (00334)         ADDB(BR3,BR2,TF)                   IO=TMP2(J* OR K*)
A3C 074A8 78740928 (00335)         LOAD(BR3,DCTMS(2))                 DCTM(.) BA
A3D 074AA 7AB9002D (00336)         ADDB(BR3,BR2,TF)                   IO=DCTM(J* OR K*)
A3E 074AC 7C760200 (00337)         LOAD(BR3,DCT1S(3))                 DCT1(.) BA
A3F 074AE 7EB9002D (00338)         ADDB(BR3,BR2,TF)                   IO=DCT1(J* OR K*)
A40 074B0 802F0000 (00339)         ADD(BR2,MSS,NA,C)                  PC->PC0
A41 074B2 82003A60 (00340)         JUMP(TOPSP1)                       RESET PC AND SETTLE
                   (00341) *
A42 074B4 84307040 (00342) G106S0  JSN(G106S3,P3)                     SET OUTPUT PC(PC3)
A43 074B6 86460BFF (00343)         LOAD(BW0,TABLS0-1(3))              TABLE0(.) BA-1
A44 074B8 88560CFF (00344)         LOAD(BW1,TABLS1-1(3))              TABLE1(.) BA-1
A45 074BA 8A660DFF (00345)         LOAD(BW2,TABLS2-1(3))              TABLE2(.) BA-1
A46 074BC 8C760EFF (00346)         LOAD(BW3,TABLS3-1(3))              TABLE3(.) BA-1
A47 074BE 8E0055F5 (00347) FLGSTST JUMPS(SETSCNT,G1)                  G1=1,SET LEVEL COUNTS
```

```
A48 074C0 90004DFB (00348)          JUMPS(INSRTS0,AF0)                       AF0=1 THEN LEVEL 0
A49 074C2 92004FF9 (00349)          JUMPS(INSRTS1,AF1)                       AF1=1 THEN LEVEL 1
A4A 074C4 940051FA (00350)          JUMPS(INSRTS2,AF2)                       AF2=1 THEN LEVEL 2
A4B 074C6 960053FB (00351)          JUMPS(INSRTS3,AF3)                       AF3=1 THEN LEVEL 3
A4C 074C8 98004760 (00352)          JUMP(FLGSTST)                            RE-TEST ALL FLAGS
A4D 074CA 9A20002B (00353) INSRTS0 CLEAR(AF0)                               RELEASE APU
A4E 074CC 9D014779 (00354)          ADDL(BW0,HS,TE),JUMP(FLGSTST)            OO=TABLE0(II),II=II+1
A4F 074CE 9F200029 (00355) INSRTS1 CLEAR(AF1)                               RELEASE APU
A50 074D0 A1114779 (00356)          ADDL(BW1,HS,TE),JUMP(FLGSTST)            OO=TABLE1(JJ),JJ=JJ+1
A51 074D2 A220002A (00357) INSRTS2 CLEAR(AF2)                               RELEASE APU
A52 074D4 A5214779 (00358)          ADDL(BW2,HS,TE),JUMP(FLGSTST)            OO=TABLE2(KK),KK=KK+1
A53 074D6 A620002B (00359) INSRTS3 CLEAR(AF3)                               RELEASE APU
A54 074D8 A9314779 (00360)          ADDL(BW3,HS,TE),JUMP(FLGSTST)            OO=TABLE3(LL),LL=LL+1
A55 074DA AA500007 (00361) SETSCNT LOAD(BW1,7)                              8 "INSTR" WORDS...
A56 074DC AC441F13 (00362)          LOAD(BW0,INSTRS0-1(2))                   TO BE SET OR CLEARED
A57 074DE AF0A0002 (00363) #1       ADD(BW0,WS,TE)                           OO=INSTRS#+1
A58 074E0 B01157B1 (00364)          SUBL(BW1,1),JUMPP(#1)                    FOR I=0,1,2...7
A59 074E2 B2200030 (00365)          CLEAR(RO)                                STALL APS OUTPUT
A5A 074E4 B4500003 (00366)          LOAD(BW1,3)                              3+1=4 SCATTER WRITES
A5B 074E6 B6C3FFC0 (00367) #2       LOAD(BW0,APSSMEM(1),TE)                  INSTRS# OVERWRITE
A5C 074E8 B8115BB1 (00368)          SUBL(BW1,1),JUMPP(#2)                    FOR I=0,1...3
A5D 074EA BA200030 (00369)          CLEAR(RO)                                STALL APS OUTPUT
A5E 074EC BC000020 (00370)          NOP                                      RELEASED BY APU
A5F 074EE BE460000 (00371)          LOAD(BW0,IORDRS(3),TE)                   IORDR(.) BA,OO=IORDR(0)
A60 074F0 C0500108 (00372)          LOAD(BW1,SIZES+OVRSLAP-1)                IORDR(.) SIZE-2+OVERLAP
A61 074F2 C30A0001 (00373) RDSOUT  ADD(BW0,HS,TE)                           OO=IORDR(I)
A62 074F4 C41161B1 (00374)          SUBL(BW1,1),JUMPP(RDSOUT)                FOR I=1,2...LTH-1;+10 0'
                   (00375) ;                                                 S
                   (00376) *
A63 074F6 C64603FE (00377)          LOAD(BW0,DCT2S-2(3))                     DCT2(.) BA-2
A64 074F8 C8560DFF (00378)          LOAD(BW1,TMP3S-2(3))                     TMP3(.) BA-2
A65 074FA CA760BFE (00379)          LOAD(BW3,TMP4S-2(3))                     TMP4(.) BA-2
A66 074FC CC000020 (00380) INDSI   NOP                                      LET "JUMP(...)" SETTLE
A67 074FE CF641F21 (00381)          LOAD(BW2,INSTRS6+1(2),TE)                SET MSS IN INSTRS6
A68 07500 D1641F23 (00382)          LOAD(BW2,INSTRS7+1(2),TE)                SET MSS IN INSTRS7
A69 07502 D2E3FFC0 (00383)          LOAD(BW2,APSSMEM(1),TE)                  OVERWRITE INDS4
A6A 07504 D4AE0000 (00384)          ADD(BW2,MSS,TE,C)                        OVERWRITE INDS5;PC3->PC
A6B 07506 D7641F1D (00385)          LOAD(BW2,INSTRS4+1(2),TE)                SET MSS IN INSTRS4
A6C 07508 D9641F1F (00386)          LOAD(BW2,INSTRS5+1(2),TE)                SET MSS IN INSTRS5
A6D 0750A DAE3FFC0 (00387)          LOAD(BW2,APSSMEM(1),TE)                  OVERWRITE INDS6
A6E 0750C DCAE0000 (00388)          ADD(BW2,MSS,TE,C)                        OVERWRITE INDS7;PC3->PC
A6F 0750E DF206670 (00389)          JUMP(INDSI,RO),CLEAR                     RESET & STALL
A70 07510 F0300032 (00390) G106S3  SET(RA)                                  ENABLE APU
A71 07512 F2007AAB (00391) TOPSP3  JUMPC(DMYSFIL,AF0)                       WAIT FOR "SKIP" FLAG
```

```
A72 07514 F4HA0002 (00392)         ADD(BW0,WS,TF)                  00=DCT2(J*)
A73 07516 F69A0002 (00393)         ADD(BW1,WS,TF)                  00=TMP3(J*)=DCTT(J*)
A74 07518 F8BA0002 (00394)         ADD(BW3,WS,TF)                  00=TMP4(J*)=DCT1T(J*)
A75 0751A FABA0002 (00395)         ADD(BW0,WS,TF)                  00=DCT2(K*)
A76 0751C FC9A0002 (00396)         ADD(BW1,WS,TF)                  00=TMP3(K*)=DCTT(K*)
A77 0751E FEBA0002 (00397)         ADD(BW3,WS,TF)                  00=TMP4(K*)=DCT1T(K*)
A78 07520 F02F0000 (00398) CHNGSPC ADD(BW2,MSS,C)                  NO IO! CHANGE PC
A79 07522 F2007160 (00399)         JUMP(TOPSP3)                    RESET PC AND SETTLE
A7A 07524 F4F20794 (00400) DMYSFIL LOAD(BW2,DMYS(1),TF)            00=?
A7B 07526 F6AA0000 (00401)         ADD(BW2,MSS,TF)                 00=?
A7C 07528 F8AA0000 (00402)         ADD(BW2,MSS,TF)                 00=?
A7D 0752A FAAA0000 (00403)         ADD(BW2,MSS,TF)                 00=?
A7E 0752C FCAA0000 (00404)         ADD(BW2,MSS,TF)                 00=?
A7F 0752E FEA17878 (00405)         ADDL(BW2,MSS,TF),JUMP(CHNGSPC)  00=? THEN CHANGE PC
                   (00406) *
          00007476 (00407)         G106$A= #C                      ASSIGN VALUE TO CHAIN AN
                   (00408) ;                                       CHOR
    07530          (00409)         END #A-1
                   (00410) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    07530 00000000 (00411) G106$I  DATA 1F'0,0'
          00000102 (00412)         G106$Z=#L-G106$
    07532          (00413)         END
```

```
AFDTSORG:  008FB (00008) (00039)
APSSMEM:   1FFC0 (00009) (00367) (00383) (00387)
CHNGSPC:   00078 (00398) (00405)
CNTSOUT:   0003A (00134) (00144) (00154) (00165)
CSPUSNOS:  021FC (00010) (00042)
DCTS:      00928 (00011)
DCT1S:     00200 (00012) (00337)
DCT2S:     00400 (00013) (00377)
DCTMS:     00928 (00014) (00335)
DMYS:      00794 (00015) (00400)
DMYSFIL:   0007A (00391) (00400)
DNFSA:     0008F (00225) (00255)
DNFSI:     00037 (00330)
FLGSTST:   00047 (00347) (00352) (00354) (00356) (00358) (00360)
G106S:     07430 (00041) (00266) (00272) (00412)
G106S1:    00039 (00273) (00332)
G106S3:    00070 (00342) (00390)
G106SA:    07476 (00269) (00407)
G106SI:    07530 (00264) (00411)
G106SO:    00042 (00332) (00342)
G106SZ:    00102 (00266) (00412)
HS:        00001 (00017) (00282) (00293) (00297) (00301) (00305) (00312) (00313) (00320) (00321)
                 (00354) (00356) (00358) (00360) (00373)
IINCSI:    00008 (00281) (00283)
INS0:      0001F (00050) (00303)
INS1:      0001A (00051) (00299)
INS2:      00016 (00052) (00295)
INS3:      00012 (00053) (00291)
INCSI:     0000C (00119) (00133) (00143) (00153) (00163)
INCRES:    00100 (00018) (00029) (00030) (00031)
INDS4:     00033 (00054) (00325)
INDS5:     00035 (00055) (00327)
INDS6:     00028 (00056) (00317)
INDS7:     0002D (00057) (00319)
INDSI:     00066 (00380) (00389)
INDSJ:     00026 (00312) (00328)
INSRTS0:   0004D (00348) (00353)
INSRTS1:   0004F (00349) (00355)
INSRTS2:   00051 (00350) (00357)
INSRTS3:   00053 (00351) (00359)
INSTRS0:   01F14 (00050) (00286) (00362)
INSTRS1:   01F16 (00051)
INSTRS2:   01F18 (00052)
INSTRS3:   01F1A (00053)
```

```
INSTRS4:   01F1C (00054) (00314) (00385)
INSTRS5:   01E1F (00055) (00315) (00386)
INSTRS6:   01F20 (00056) (00322) (00381)
INSTRS7:   01F22 (00057) (00323) (00382)
INTGRS:    01F24 (00058) (00280)
IORDRS:    00000 (00019) (00371)
ISAS00:    00502 (00021)
ISAS01:    00503 (00022) (00277)
ISVTS:     00502 (00020) (00021) (00022)
L0SSU:     0001D (00300) (00302)
L1SSU:     00019 (00296) (00298)
L2SSU:     00015 (00292) (00294)
L3SSU:     00011 (00290)
LVLS0:     00030 (00120) (00155)
LVLS1:     00026 (00121) (00145)
LVLS2:     0001C (00124) (00135)
LVLS3:     00012 (00125)
MSS:       00000 (00023) (00291) (00295) (00299) (00303) (00310) (00311) (00316) (00317) (00318)
                 (00319) (00324) (00325) (00326) (00327) (00339) (00384) (00388) (00398) (00401)
                 (00402) (00403) (00404) (00405)
ORDRSI:    0004D (00184) (00188)
OVRSLAP:   0000A (00024) (00372)
RDSOUT:    00061 (00373) (00374)
SCLRS:     00001 (00266) (00274)
SFTSCNT:   00055 (00347) (00361)
SHUFLS:    00073 (00225) (00253)
SHUFLSV:   00058 (00196)
SIZFS:     000FF (00025) (00279) (00307) (00372)
SORTS:     07308 (00040) (00105)
SORTSSA:   00000 (00103) (00107) (00257)
SORTSSZ:   00090 (00104) (00257) (00258)
STALS:     00074 (00226) (00226)
STALSI:    00021 (00304) (00306)
START:     07300 (00027) (00091)
SVTS:      00382 (00026)
T1S:       07300 (00095) (00274)
T2S:       07302 (00096) (00275)
T3S:       07304 (00097) (00276)
TABLS0:    00C00 (00028) (00029) (00302) (00343)
TABLS1:    00D00 (00029) (00030) (00298) (00344)
TABLS2:    00E00 (00030) (00031) (00294) (00345)
TABLS3:    00F00 (00031) (00290) (00346)
TMP2S:     00A00 (00032) (00278) (00333)
TMP3S:     00F00 (00033) (00378)
```

```
PAGE   19:       FCR 249... "MPASRT(U)"  SORT DCT1(.) COEFFICIENTS

  TMP4S:      00C00 (00034) (00379)
  TOPSP1:     0003A (00333) (00340)
  TOPSP3:     00071 (00391) (00399)
  TSTSL0:     0001F (00304) (00305)
  TSTSL1:     0001A (00300) (00301)
  TSTSL2:     00017 (00296) (00297)
  TSTSL3:     00013 (00292) (00293)
  *S:         00002 (00035) (00281) (00287) (00363) (00392) (00393) (00394) (00395) (00396) (00397)
  ZS:         00003 (00036)

LINES WITH ERRORS:    0  (MAP VERSION R00101.10)   F=    0
```

```
                    (00001) * FCB 250... "MPESTO(Y,U,V,W))"   QUANTIZE DCT PARAMETERS
                    (00002) *                                 ORIGINATED:25-JUL-79
                    (00003) *                                 UPDATED:29-MAY-80
                    (00004) * DEFINE GLOBAL SYMBOLS
                    (00005)        OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                    (00006)        OPADD DQF, (3 .LS. 10) + (12 .LS. 5) + $1B
                    (00007)        OPADD WAFNF, (1 .LS. 10) + (12 .LS. 5) + $1A
                    (00008)        OPADD EO, (3 .LS. 10) + (12 .LS. 5) + $1C
          000008E8  (00009)        AFDTSORG=$8E8
          0001FFC0  (00010)        APSSMEM=$1FFC0
          000021FC  (00011)        CSPUSNOS=$21FC
          00000794  (00012)        DMYS=$0794
          00000003  (00013)        #M=3
          00000001  (00014)        HS=1
          00000000  (00015)        MS=0
          00000000  (00016)        MSS=0
          00000502  (00017)        ISVTS=$502
          00000503  (00018)        ISAS001=ISVTS+D'1'
          00000382  (00019)        SVTS=$0382
          00000398  (00020)        SAS11=SVTS+2*D'11'
          000003CE  (00021)        SAS38=SVTS+2*D'38'
          000003D0  (00022)        SAS39=SVTS+2*D'39'
          00007600  (00023)        START=$7600
          00000E00  (00024)        TMP3S=D'3584'
          00000C00  (00025)        TMP4S=D'3072'
          00000002  (00026)        WS=2
          00000003  (00027)        ZS=3
                    (00028) *
                    (00029) * EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000BC4  (00030)        #L=AFDTSORG+3*2*(250-128)               FCB 250
00BC4     001E7602  (00031)        ADDR QUANS(R7,1)
00BC6     001E76AE  (00032)        ADDR G104S(R7,1)
00BC8     001021FC  (00033)        ADDR CSPUSNOS(,1,0)
                    (00034)        EJECT
```

```
                      (00035) *
                      (00036) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      (00037) *
                      (00038)          EVEN
                      (00039)          BUS 2
             00001CB4 (00040)          #L=D'7300'
01CB4 1E50            (00041) INSTRSO DATA INSO*D'512'+X'50',X'0000'
01CB5 0000
                      (00042)          EVEN
01CB6 00000000 (00043) NLS0     DATA 1F'0.0'
01CB8 08000041 (00044) NLS1     DATA 1F'1.0'
01CBA 10000041 (00045) NLS2     DATA 1F'2.0'
01CBC 20000041 (00046) NLS3     DATA 1F'4.0'
01CBE 40000041 (00047) NLS4     DATA 1F'8.0'
01C90 08000042 (00048) NLS5     DATA 1F'16.0'
                      (00049) *  0,1 LEVEL
01C92 12A05F49 (00050) DTHRS01 DATA 16F'1.0E10'
 ...
                      (00051) *  2 LEVEL
01CB2 0903F441 (00052) DTHRS2  DATA 1F'1.1269',15F'1.0E10'
01CB4 12A05F49
 ...
                      (00053) *  3 LEVEL
01CD2 443FE5C0 (00054) DTHRS3  DATA 1F'0.5332',1F'1.2527',1F'2.3796',13F'1.0E10'
01CD4 0A05B7C1
01CD6 13096BC1
01CD8 12A05F49
 ...
                      (00055) *  4 LEVEL
01CF2 21D7DBC0 (00056) DTHRS4  DATA 1F'0.2644',1F'0.5667',1F'0.9198',1F'1.3444'
01CF4 4B89A040
01CF6 758C01C0
01CF8 0AC154C1
01CFA 0F055341 (00057)         DATA 1F'1.8776',1F'2.5971',1F'3.724',9F'1.0E10'
01CFC 14C6DC41
01CFE 1DCAC0C1
01D00 12A05F49
 ...
                      (00058) *  5 LEVEL
01D12 10EBEDC0 (00059) DTHRS5  DATA 1F'0.1322',1F'0.2732',1F'0.4243',1F'0.5870'
01D14 22F837C0
01D16 364F7640
01D18 4B2200C0
01D1A 61B089C0 (00060)         DATA 1F'0.7632',1F'0.9555',1F'1.1669',1F'1.4019'
```

```
      01D1C 7A40D2C0
      01D1E 0955CFC1
      01D20 0B371741
      01D22 00549541 (00061)        DATA 1E'1.6663',1E'1.9687',1E'2.3217',1E'2.7463'
      01D24 0FBEF5C1
      01D26 12920741
      01D28 15FB6C41
      01D2A 1A3C6A41 (00062)        DATA 1E'3.2795',1E'3.9990',1E'5.1259',1.0E10
      01D2C 1FF0F3C1
      01D2E 2901D7C1
      01D30 12A05F49
                     (00063)        BUS 1
            00007600 (00064)        #L=START
                     (00065)        EVEN
                     (00066)        EJECT
```

```
                          (00067) *
                          (00068) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00069) *
                          (00070)          EVEN
    07600 00              (00071)          DATA QUANSSA                   START ADDRESS
    07601 00              (00072)          DATA QUANSSZ                   MODULE SIZE
                          (00073) QUANS    BEGIN APU(QUAN)
          00 0000         (00074)          *A=0
A00 07602 08 0000         (00075) QUANSSA  MOV(IOA,A1)\NOP                FOR BITS=0 TEST
A01 07604 0A200000        (00076)          NEG(A1)\NOP                    WANT A1=8001 NOT FFFF!
A02 07606 16811681        (00077)          MOV(A1),K(2,0)                 A1=-1,2,0
A03 07608 08920892        (00078)          MOV(R,A2)                      A2=2.0,LE & LO INCRE.
A04 0760A 088F088F        (00079)          MOV(R,M7)                      M7=2.0,FORCE Q(K-1)<0
A05 0760C 08000000        (00080) INCSJ    MOV(IO,A0)\NOP                 A0=IBIT(I)=BITS
A06 0760E 40200000        (00081)          AD (A1,A0)\NOP                 IF BITS=0, THEN ZERO FIL
                          (00082) ;                                       L!
A07 07610 08FC0000        (00083)          MOV(IOA,OO)\NOP                OTHERWISE GO W/ SCATTER-
                          (00084) ;                                       WRITES
A08 07612 08800000        (00085)          MOV(R,NULL)\NOP                SAVE TIME BY NOT QUANTIZ
                          (00086) ;                                       ING
A09 07614 901F0012        (00087)          JUMPC(BITSOK,T1)               T1=0 THEN BITS.NE.0
A0A 07616 20490000        (00088) ZEROS    SET(AF1)\NOP                   BITS=0 SO 0 FILL TMP1S
A0B 07618 901C0008        (00089) #1       JUMPC(#1,F0)                   WAIT FOR LAST PUTDOWN
A0C 0761A 20500000        (00090)          SET(RO)\NOP                    RELEASE APS OUTPUT
A0D 0761C 081C0000        (00091) CLRS     MOV(ZERO,OO)\NOP               AND ZERO FILL REMAINDER
A0E 0761E 00000000        (00092)          NOP                            *** NOP FOR RACE
A0F 07620 00000000        (00093)          NOP                            *** PREVENTION
A10 07622 901C000D        (00094)          JUMPC(CLRS,F0)                 BEFORE "IFIX" OF RESULTS
A11 07624 10500043        (00095)          JUMP(FIXSO,RO),SET             BITS=0 QUANT. IS USELESS
A12 07626 901C0012        (00096) BITSOK   JUMPC(BITSOK,F0)               CLEAR BITS PUTDOWN
A13 07628 20370000        (00097)          CLEAR(WI)\NOP                  RELEASE APS INPUT
A14 0762A 20500000        (00098)          SET(RO)\NOP                    RELEASE APS OUTPUT
A15 0762C 08FC0000        (00099)          MOV(IOA,OO)\NOP                INSO SCATTER-WRITE
A16 0762E 901C0016        (00100) #2       JUMPC(#2,F0)                   1 PUTDOWNS
A17 07630 20370000        (00101)          CLEAR(WI)\NOP                  RELEASE APS INPUT
A18 07632 20500000        (00102)          SET(RO)\NOP                    RELEASE APS OUTPUT
A19 07634 06A006B0        (00103)          K(-2,0)\K(-1,0)                -2.0;-1.0
A1A 07636 088A088A        (00104)          MOV(R,M2)                      M2<0 BOTH SIDES
A1B 07638 08F008F0        (00105)          MOV(IOA,A0)                    DCT(J)
A1C 0763A 12131213        (00106)          MOV(A3),ABS(A0)                LE''=-2.0,/DCT(J)/;LO''=
                          (00107) ;                                       -1.0,/DCT(J)/
A1D 0763C 08F608F6        (00108)          MOV(IOA,A6)                    A6=NL(1,2,4,8,16)
A1E 0763E 08900890        (00109)          MOV(R,A0)                      A0=/DCT(J)/
A1F 07640 911F0022        (00110)          JUMPS(DCTSN,T1)                IF T1=1 DCT(J).LT.0
```

```
A20 07642 02600260 (00111)          R(A3)                  R=LE'';R=LO''
A21 07644 10000023 (00112)          JUMP(CLRSFLG)          RESET "FLAG JAMS"
A22 07646 43C043C0 (00113) DCTSN    ADD(A6,A3)             LE''+NL;LO''+NL
A23 07648 8560F560 (00114) CLRSFLG  MUL(M2,M7)             P<0 FORCING T-BIT->0 FOR
A24 0764A 08F808F8 (00115)          MOV(IOA,M0)            DCT1(J)
A25 0764C 08FC0000 (00116)          MOV(IOA,M4)\NOP        D(K)
A26 0764E 000008FC (00117)          NOP\MOV(IOA,M4)        D(K+1)
A27 07650 84148414 (00118) LOOPS    MOV(A4),MUL(M0,M4)     A4=Q(K-2),Q(K);A4=Q(K-1)
                   (00119) ;                               ,Q(K+1)
A28 07652 4C134 13 (00120)          MOV(A3),SUB(A0,A4)     LE'=LE''+2,T(K-2);LO'=LO
                   (00121) ;                               ''+2,T(K-1)
A29 07654 08FD0000 (00122)          MOV(IOA,M5)\NOP        D(K+2)
A2A 07656 000008FD (00123)          NOP\MOV(IOA,M5)        D(K+3)
A2B 07658 08800880 (00124)          MOV(R,NULL)            FORCE COMPLETION
A2C 0765A 911F0039 (00125)          JUMPS(T3S,T1)          IS T1=1 ON T(K-4)?
A2D 0765C 911F003F (00126)          JUMPS(T4S,T2)          IS T2=1 ON T(K-3)?
A2E 0765E 43404340 (00127)          ADD(A2,A3)             LE'+2;LO'+2
A2F 07660 20370000 (00128)          CLEAR(W1)\NOP          RELEASE APS INPUT
A30 07662 84358435 (00129)          MOV(A5),MUL(M0,M5)     A5=Q(K),Q(K+2);A5=Q(K+1)
                   (00130) ;                               ,Q(K+3)
A31 07664 4D134D13 (00131)          MOV(A3),SUB(A0,A5)     LE''=LE'+2,T(K);LO''=LO'
                   (00132) ;                               +2,T(K+1)
A32 07666 08FC0000 (00133)          MOV(IOA,M4)\NOP        D(K+4)
A33 07668 000008FC (00134)          NOP\MOV(IOA,M4)        D(K+5)
A34 0766A 08800880 (00135)          MOV(R,NULL)            FORCE COMPLETION
A35 0766C 911F0039 (00136)          JUMPS(T3S,T1)          IS T1=1 ON T(K-4)?
A36 0766E 911F003F (00137)          JUMPS(T4S,T2)          IS T2=1 ON T(K-3)?
A37 07670 43104340 (00138)          ADD(A2,A3)             LE''+2;LO''+2
A38 07672 10370027 (00139)          JUMP(LOOPS,W1),CLEAR   K=K+4 FOR K<LTH
A39 07674 02600000 (00140) T3S      R(A3)\NOP              LE
A3A 07676 20480000 (00141)          SET(AF0)\NOP           INFORM APS
A3B 07678 089C0000 (00142)          MOV(R,O0)\NOP          O0=LE
A3C 0767A 91080043 (00143)          JUMPS(FIXSQ,AF3)       "FIX" ALL V VALUES @ ONC
                   (00144) ;                               E
A3D 0767C 10370005 (00145)          JUMP(INCSJ,W1),CLEAR   FOR I=01,2...LTH-1
A3E 0767E 00000260 (00146) T4S      NOP\R(A3)              LO
A3F 07680 20480000 (00147)          SET(AF0)\NOP           INFORM APS
A40 07682 0000089C (00148)          NOP\MOV(R,O0)          O0=LO
A41 07684 91080043 (00149)          JUMPS(FIXSQ,AF3)       "FIX" ALL V VALUES @ ONC
                   (00150) ;                               E
A42 07686 10370005 (00151)          JUMP(INCSJ,W1),CLEAR   FOR I=01,2...LTH-1
                   (00152) *
A43 07688 20370000 (00153) FIXSQ    CLEAR(W1)\NOP          RELEASE APS INPUT
A44 0768A 08FF08FF (00154)          MOV(IOA,M7)            M7=SCALAR A
```

```
        ... "M(ESTO(Y,U,V,W))"    QUANTIZE DCT PARAMETERS

[illegible]          [illegible] (00155)          MOV(IQA,A7)                    A7=SCALAR B
[illegible]          [illegible] (00156) #1       MOV(NULL),ADDST(A0,A7)         R''=P'+(-1) T*B
[illegible]          [illegible] (00157)          MOV(P,NULL)                    COMPLETE MULT.
[illegible]          [illegible] (00158)          MOV(IQA,M0)\NOP                M0=UF
[illegible]          [illegible] (00159)          MOV(A1),ALIGN(A1)              A1=R'',R''=FIXED(=Y'')
[illegible]          [illegible] (00160)          NOP\MOV(IQA,M0)                M0=U0
[illegible]          [illegible] (00161)          MOV(A0),MUL(M0,M7)             A0=P',P=SA*U
[illegible]       021C021C (00162)          MOV(OO),R(A0)                  OUT=Y'',R=P'
[illegible]       901C0046 (00163)          JUMPC(#1,F0)                   FOR I=0,1,2...LTH-1
                       (00164) *
A4[illegible] 0769E 20260000 (00165) DNESA    CLEAR(G2)\NOP                  END ANALYZER TIMING
A4[illegible] 076A0 20320000 (00166)          CLEAR(RA)\NOP                  APU DONE!
A50 076A2 00000000 (00167)          NOP
A51 076A4 10000000 (00168)          JUMP(0)
          00000052 (00169)          QUANSSZ=*A-QUANSSA
    076A6          (00170)          END QUANSSZ
                   (00171)          EJECT
```

```
                    (00172) *
                    (00173) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00174) *
                    (00175)          EVEN
    076A6  00007776 (00176)          ADDR G104$1                    POINTER TO CONSTRUCTED I
                    (00177) ;                                       NSTRUCTION BLOCK
    076A8  0000772A (00178)          ADDR G104$+2*SCLRS             POINTER TO SCALAR BLOCK
    076AA  0000     (00179)          DATA 0                         NUMBER OF SCALARS
    076AB  0000     (00180)          DATA G104$Z                    MODULE SIZE
    076AC  00007760 (00181)          ADDR G104$A                    POINTER TO CHAIN ANCHOR
                    (00182)          EVEN                           END ON WORD BOUNDARY
                    (00183) *
                    (00184) G104$    BEGIN APS(G104)
A00 076AE  0204840  (00185)          JSN(G104$0,P2)                 SET OUTPUT PC(P2)
A01 076B0  02300030 (00186)          SET(R0)                        ENABLE APS OUTPUT
A02 076B2  05420503 (00187)          LOAD(BR0,ISA$001(1),TF)        IQ=1
A03 076B4  06701000 (00188)          LOAD(BR3,I1)                   IBIT(.) BA
A04 076B6  08500000 (00189)          LOAD(BR1,MS$)                  DUMMY OP FOR EXEC
A05 076B8  0A320000 (00190)          SUB(BR3,MS$)                   IBIT(.) BA-1
A06 076BA  0C460BEE (00191)          LOAD(BR0,TMP3$-2(3))           TMP3(.)->DCTT(.) BA-2
A07 076BC  0E660BEE (00192)          LOAD(BR2,TMP4$-2(3))           TMP4(.)->DCT1T(.) BA-2
A08 076BE  11AA0001 (00193) IINCS1   ADD(BR3,HS,TF)                 IQ=IBIT(I)
A09 076C0  12300037 (00194)          SET(WI)                        LET APS OUTPUT CATCH-UP
A0A 076C2  14200028 (00195)          CLEAR(AF0)                     RESET APU DECISION
A0B 076C4  16003EE9 (00196)          JUMPS(SCLRS,AF1)               IF AF1=1 THEN ZERO FILL!
A0C 076C6  18D41C84 (00197)          LOAD(BR1,INSTR$0(2),TF)        IQ="LOAD(BR1,MS$)"
A0D 076C8  1A300037 (00198)          SET(WI)                        LET APS OUTPUT CATCH-UP
A0E 076CA  1C000020 (00199)          NOP
A0F 076CC  1E500000 (00200) INS0     LOAD(BR1,MS$)                  APU SETS MS$=BITS
A10 076CE  201933F1 (00201)          SUBL(BR1,1),JUMPN(B$0)         B$0 FOR BITS=0
A11 076D0  22192FF1 (00202)          SUBL(BR1,1),JUMPN(B$1)         B$1 FOR BITS=1
A12 076D2  241927F1 (00203)          SUBL(BR1,1),JUMPN(B$2)         B$2 FOR BITS=2
A13 076D4  261921F1 (00204)          SUBL(BR1,1),JUMPN(B$3)         B$3 FOR BITS=3
A14 076D6  28191BF1 (00205)          SUBL(BR1,1),JUMPN(B$4)         B$4 FOR BITS=4
A15 076D8  2A8A0002 (00206) B$5      ADD(BR0,WS,TF)                 IQ=DCTT(J)
A16 076DA  2CD41C90 (00207)          LOAD(BR1,NLS5(2),TF)           IQ=NL=16
A17 076DC  2EAA0002 (00208)          ADD(BR2,WS,TF)                 IQ=DCT1T(J)
A18 076DE  30D41D12 (00209)          LOAD(BR1,DTHR$5(2),TF)         IQ=DTHR(5 LEVEL)
A19 076E0  329A0002 (00210)          ADD(BR1,WS,TF)                 IQ=DTHR(OFFSET+I)
A1A 076E2  34003B60 (00211)          JUMP(ADDS1)                    LET APU CATCH-UP
A1B 076E4  368A0002 (00212) B$4      ADD(BR0,WS,TF)                 IQ=DCTT(J)
A1C 076E6  38D41C8E (00213)          LOAD(BR1,NLS4(2),TF)           IQ=NL=8
A1D 076E8  3AAA0002 (00214)          ADD(BR2,WS,TF)                 IQ=DCT1T(J)
A1E 076EA  3CD41CF2 (00215)          LOAD(BR1,DTHR$4(2),TF)         IQ=DTHR(4 LEVEL)
```

```
A1F 076EC 3E9A0002 (00216)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I)
A20 076EE 40003B60 (00217)        JUMP(ADDS1)                  LET APU CATCH-UP
A21 076F0 428A0002 (00218) RS3    ADD(BR0,WS,TF)               IO=DCTT(J)
A22 076F2 44041C8C (00219)        LOAD(BR1,NLS3(2),TF)         IO=NL=4
A23 076F4 46AA0002 (00220)        ADD(BR2,WS,TF)               IO=DCT1T(J)
A24 076F6 48041CD2 (00221)        LOAD(BR1,DTHRS3(2),TF)       IO=DTHR(3 LEVEL)
A25 076F8 4A9A0002 (00222)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I)
A26 076FA 4C003B60 (00223)        JUMP(ADDS1)                  LET APU CATCH-UP
A27 076FC 4E8A0002 (00224) RS2    ADD(BR0,WS,TF)               IO=DCTT(J)
A28 076FE 50041C8A (00225)        LOAD(BR1,NLS2(2),TF)         IO=NL=2
A29 07700 52AA0002 (00226)        ADD(BR2,WS,TF)               IO=DCT1T(J)
A2A 07702 54041CB2 (00227)        LOAD(BR1,DTHRS2(2),TF)       IO=DTHR(2 LEVEL)
A2B 07704 569A0002 (00228)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I)
A2C 07706 58003B60 (00229)        JUMP(ADDS1)                  LET APU CATCH-UP
A2D 07708 5A8A0002 (00230) RS1    ADD(BR0,WS,TF)               IO=DCTT(J)
A2E 0770A 5C041C88 (00231)        LOAD(BR1,NLS1(2),TF)         IO=NL=1
A2F 0770C 5EAA0002 (00232)        ADD(BR2,WS,TF)               IO=DCT1T(J)
A30 0770E 60041C92 (00233)        LOAD(BR1,DTHRS01(2),TF)      IO=DTHR(1 LEVEL)
A31 07710 629A0002 (00234)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I)
A32 07712 64003B60 (00235)        JUMP(ADDS1)                  LET APU CATCH-UP
A33 07714 668A0002 (00236) RS0    ADD(BR0,WS,TF)               IO=DCTT(J)
A34 07716 68041C86 (00237)        LOAD(BR1,NLS0(2),TF)         IO=NL=0
A35 07718 6AAA0002 (00238)        ADD(BR2,WS,TF)               IO=DCT1T(J)
A36 0771A 6C041C92 (00239)        LOAD(BR1,DTHRS01(2),TF)      IO=DTHR(0 LEVEL)
A37 0771C 6E9A0002 (00240)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I)
A38 0771E 709A0002 (00241) ADDS1  ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I+1)
A39 07720 729A0002 (00242)        ADD(BR1,WS,TF)               IO=DTHR(OFFSET+I+2)
A3A 07722 74300037 (00243)        SET(W1)                      WAIT FOR THRESHOLD DEC.
A3B 07724 76000020 (00244)        NOP                          RELEASED BY APU
A3C 07726 78003BA8 (00245)        JUMPC(ADDS1,AF0)             APU DECISION ON AF0
A3D 07728 7A0008AB (00246)        JUMPC(IINCS1,AF3)            FOR I=0,1,2...LTH-1
                   (00247) *
A3E 0772A 7CC203CF (00248) SCLRS  LOAD(BR0,SAS3B(1),TF)        IO=2**(-15)
A3F 0772C 7EC2039B (00249)        LOAD(BR0,SAS11(1),TF)        IO=2**(-16)
A40 0772E 8040207A (00250)        LOAD(BF0,(2))                TMP1(.) BA
A41 07730 82500000 (00251)        LOAD(BR1,MSS)                TMP1(.) SIZE-1
A42 07732 84020000 (00252)        SUB(BR0,MSS)                 TMP1(.) BA-2
A43 07734 868A0002 (00253) IFIXS1 ADD(BR0,WS,TF)               IO=TMP1(I)
A44 07736 888A0002 (00254)        ADD(BR0,WS,TF)               IO=TMP1(I+1)
A45 07738 8A1943B2 (00255)        SUBL(BR1,2),JUMPP(IFIXS1)    FOR I=0,1,2...LTH-1
                   (00256) *
A46 0773A 8C200031 (00257) DNES1  CLEAR(R1)                    INPUT DONE!
A47 0773C 8E000020 (00258)        NOP
                   (00259) *
```

```
A48 0773E 90300032 (00260) G104SO  SET(RA)                                 ENABLE APU
A49 07740 92402012 (00261)         LOAD(RW0,(2))                           TMP1(.) BA
A4A 07742 94500000 (00262)         LOAD(RW1,MSS)                           TMP1(.) SIZE-1
A4B 07744 96020000 (00263)         SUB(RW0,MSS)                            TMP1(.) BA-2
A4C 07746 9964ICH5 (00264) OINCSJ  LOAD(RW2,INSTRSO+1(2),TF)               MSS=BITS IN INSO
A4D 07748 9A200030 (00265)         CLEAR(RO)                               STALL APS OUTPUT
A4E 0774A 9C000020 (00266)         NOP                                     RELEASED BY APU
A4F 0774C 9E0056E9 (00267)         JUMPS(CLRSTMP,AF1)                      IF AF1=1 THEN ZERO FILL
A50 0774E A0E3EEC0 (00268)         LOAD(RW2,APSSMEM(1),TF)                 INSTRSO INTO APS INPUT
A51 07750 A2200030 (00269)         CLEAR(RO)                               STALL APS OUTPUT
A52 07752 A4000020 (00270)         NOP
A53 07754 A6RA0002 (00271)         ADD(RW0,WS,TF)                          OO=TMP3(J)
A54 07756 A8114CB1 (00272)         SUBL(RW1,1),JUMPP(OINCSJ)               FOR J=0,1...LTH-1
A55 07758 AA305968 (00273)         JUMP(OFIXS,AF3),SET                     SIGNAL END TO APS INPUT
                   (00274) *
A56 0775A ACRA0002 (00275) CLRSTMP ADD(RW0,WS,TF)                          OO=TMP3(J)
A57 0775C AE1156B1 (00276)         SUBL(RW1,1),JUMPP(CLRSTMP)              FOR J=?...LTH-1
A58 0775E B0200030 (00277)         CLEAR(RO)                               STALL APS OUTPUT
                   (00278) *
A59 07760 B2400020 (00279) OFIXS   LOAD(RW0,(0))                           ODCT(.) BA
A5A 07762 B4500000 (00280)         LOAD(RW1,MSS)                           ODCT(.) SIZE-1
A5B 07764 B6020000 (00281)         SUB(RW0,MSS)                            ODCT(.) BA-1
A5C 07766 B8700003 (00282)         LOAD(RW3,3)                             3+1=4 DUMMIES
A5D 07768 BAE20794 (00283) #1      LOAD(RW2,DMYS(1),TF)                    OO=?
A5E 0776A BC315DB1 (00284)         SUBL(RW3,1),JUMPP(#1)                   4 TIMES
A5F 0776C BE0A0001 (00285) OFIXSI  ADD(RW0,HS,TF)                          OO=ODCT(I)
A60 0776E C10A0001 (00286)         ADD(RW0,HS,TF)                          OO=ODCT(I+1)
A61 07770 C2115EB2 (00287)         SUBL(RW1,2),JUMPP(OFIXSI)               FOR I=0,1,2...LTH-1
                   (00288) *
A62 07772 C4200030 (00289) DNESO   CLEAR(RO)                               OUTPUT DONE!
A63 07774 C6000020 (00290)         NOP
                   (00291) *
          00007760 (00292)         G104SA= #C                              ASSIGN VALUE TO CHAIN AN
                   (00293) ;                                               CHOR
    07776          (00294)         END #A-1
                   (00295) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    07776 00000000 (00296) G104SI  DATA 4E'0,0'
    ...
          00000000 (00297)         G104SZ=#L-G104S
    0777E          (00298)         END
```

```
ADDSI:     0003B (00211) (00217) (00223) (00229) (00235) (00241) (00245)
AFDTSORG:  008EB (00009) (00030)
APSSMEM:   1FFC0 (00010) (00268)
BS0:       00033 (00201) (00236)
BS1:       0002D (00202) (00230)
BS2:       00027 (00203) (00224)
BS3:       00021 (00204) (00218)
BS4:       0001B (00205) (00212)
BS5:       00015 (00206)
BITSOK:    00012 (00087) (00096) (00096)
CLRS:      0000D (00091) (00094)
CLRSFLG:   00023 (00112) (00114)
CLRSTEP:   00056 (00267) (00275) (00276)
CSPUSNOS:  021EC (00011) (00033)
DCTSN:     00022 (00110) (00113)
DMYS:      00794 (00012) (00283)
DNFSA:     0004F (00165)
DNFSI:     00046 (00257)
DFFSO:     00062 (00289)
DTHRS01:   01C92 (00050) (00233) (00239)
DTHRS2:    01CB2 (00052) (00227)
DTHRS3:    01CD2 (00054) (00221)
DTHRS4:    01CF2 (00056) (00215)
DTHRS5:    01D12 (00059) (00209)
FIXSU:     00043 (00095) (00143) (00149) (00153)
G104S:     076AF (00032) (00178) (00184) (00297)
G104SA:    07760 (00181) (00292)
G104SI:    07776 (00176) (00296)
G104SO:    0004B (00185) (00260)
G104SZ:    000D0 (00180) (00297)
HS:        00001 (00014) (00193) (00285) (00286)
IFIXSI:    00043 (00253) (00255)
IINCSI:    0000B (00193) (00246)
IRSO:      0000F (00041) (00200)
INCSJ:     00005 (00080) (00145) (00151)
INSTRSO:   01CB4 (00041) (00197) (00264)
ISASOO1:   00503 (00018) (00187)
ISVTS:     00502 (00017) (00018)
LOOPS:     00027 (00118) (00139)
MS:        00000 (00015)
MSS:       00000 (00016) (00189) (00190) (00200) (00251) (00252) (00262) (00263) (00280) (00281)
NLS0:      01CB6 (00043) (00237)
NLS1:      01CB8 (00044) (00231)
NLS2:      01CBA (00045) (00225)
```

```
NLS3:      01CBC (00046) (00219)
NLS4:      01CBE (00047) (00213)
NLS5:      01C90 (00048) (00207)
OFIXS:     00059 (00273) (00279)
OFIXS1:    0005E (00285) (00287)
OINCSJ:    0004C (00264) (00272)
QUANS:     07602 (00031) (00073)
QUANSSA:   00000 (00071) (00075) (00169)
QUANSSZ:   00052 (00072) (00169) (00170)
SAS11:     0039B (00020) (00249)
SAS38:     003CE (00021) (00218)
SAS39:     003D0 (00022)
SCLRS:     0003E (00178) (00196) (00248)
START:     07600 (00023) (00064)
SVTS:      003B2 (00019) (00020) (00021) (00022)
T3S:       00039 (00125) (00136) (00140)
T4S:       0003E (00126) (00137) (00146)
TMP3S:     00E00 (00024) (00191)
TMP4S:     00C00 (00025) (00192)
WS:        00002 (00026) (00206) (00208) (00210) (00212) (00214) (00216) (00218) (00220) (00222)
                 (00224) (00226) (00228) (00230) (00232) (00234) (00236) (00238) (00240) (00241)
                 (00242) (00253) (00254) (00271) (00275)
ZS:        00003 (00027)
ZEROS:     0000A (00088)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)    E-    0
```

```
                   * FCB 251... "MPESTD(Y,U,V,W)"    DEQUANTIZE DCT PARAMETERS
                   *                                 ORIGINATED:14-JAN-80
                   *                                 UPDATED:07-APR-80
                   * DEFINE GLOBAL SYMBOLS
                          OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                          OPADD DOF, (3 .LS. 10) + (12 .LS. 5) + $1B
                          OPADD WAFNF, (1 .LS. 10) + (12 .LS. 5) + $1A
                          OPADD FD, (3 .LS. 10) + (12 .LS. 5) + $1C
        000008E8          AFDTSORG=$8E8
        0001FFC0          APSSMEM=$1FFC0
        000021FC          CSPUSNOS=$21FC
        00000400          DEDCTS=D'1024'
        00000003          MM=3
        00000001          HS=1
                   * IBITS=D'2088'
                   * MIB'T2=>IBIT
        00002A28          IBITS=D'10792'
        00000000          MS=0
        00000000          MSS=0
        00000382          SVTS=$0382
        000003CE          SAS38=SVTS+2*D'38'
        000003D0          SAS39=SVTS+2*D'39'
        000077F0          START=$77F0
                   * TMP4S=D'3072'
                   * DCT1T2=>TMP4
        00002528          TMP4S=D'9512'
        00000002          WS=2
        00000003          ZS=3
                   *
                   * EXPAND ARRAY FUNCTION DISPATCH TABLE
        000008CA          *L=AFDTSORG+3*2*(251-128)                 FCB 251
   08CA 001F77F2          ADDR QUANS(R7,1)
   08CC 001F7874          ADDR G104S(R7,1)
   08CE 001021FC          ADDR CSPUSNOS(,1,0)
                          EJECT
```

```
                    (00036) *
                    (00037) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00038) *
                    (00039)                  BUS 2
          00001D4C  (00040)                  #L=D'7500'
                    (00041)                  EVEN
01D4C 1E50          (00042) INSTRS1 DATA INS1*D'512'+X'50',X'0000'
01D4D 0000
01D4E 4A9A          (00043) INSTRS2 DATA INS2*D'512'+X'9A',X'0000'
01D4F 0000
01D50 E0AA          (00044) INSTRS4 DATA INDS4*D'512'+X'AA',X'0000'
01D51 0000
01D52 E4AA          (00045) INSTRS5 DATA INDS5*D'512'+X'AA',X'0000'
01D53 0000
01D54 CEAA          (00046) INSTRS6 DATA INDS6*D'512'+X'AA',X'0000'
01D55 0000
01D56 D2AA          (00047) INSTRS7 DATA INDS7*D'512'+X'AA',X'0000'
01D57 0000
01D58 00000000      (00048) DECS0   DATA 1E'0.0'
01D5A 00000000      (00049)         DATA 1E'-0.0'
01D5C 5AB240C0      (00050) DECS1   DATA 1E'0.7071'
01D5E DAB240C0      (00051)         DATA 1E'-0.7071'
01D60 35B573C0      (00052) DECS2   DATA 1E'0.4196',1E'1.8340'
01D62 0EAC0841
01D64 B5B573C0      (00053)         DATA 1E'-0.4196',1E'-1.8340'
01D66 8EAC0841
01D68 1DE00040      (00054) DECS3   DATA 1E'0.2334',1E'0.8330',1E'1.6725',1E'3.0867'
01D6A 6A9EBE40
01D6C 0D6147C1
01D6E 18B1BEC1
01D70 9DE00D40      (00055)         DATA 1E'-0.2334',1E'-0.8330',1E'-1.6725',1E'-3.0867'
01D72 EA9EBE40
01D74 8D6147C1
01D76 98B1BEC1
01D78 0EDE3B40      (00056) DECS4   DATA 1E'0.1240',1E'0.4048',1E'0.7287',1E'1.1110'
01D7A 33D07CC0
01D7C 5D460AC0
01D7E 08E353C1
01D80 0C9E55C1      (00057)         DATA 1E'1.5778',1E'2.1773',1E'3.0169',1E'4.4311'
01D82 116B1C41
01D84 18229C41
01D86 2372E4C1
01D88 8EDE3B40      (00058)         DATA 1E'-0.1240',1E'-0.4048',1E'-0.7287',1E'-1.1110'
01D8A B3D07CC0
```

```
01D8C 0D460AC0
01D8E 88E353C1
01D90 8C9E55C1 (00059)        DATA 1E'-1.5778',1E'-2.1773',1E'-3.0169',1E'-4.4311'
01D92 91681C41
01D94 98229C41
01D96 A372E4C1
01D98 084126C0 (00060) DECS5  DATA 1E'0.0640',1E'0.2004',1E'0.3461',1E'0.5025'
01D9A 19A66540
01D9C 2C4D0140
01D9E 40516EC0
01DA0 55F3B640 (00061)        DATA 1E'0.6715',1E'0.8550',1E'1.0559',1E'1.2779'
01DA2 6D70A3C0
01DA4 08727BC1
01DA6 0A3923C1
01DA8 0C350B41 (00062)        DATA 1E'1.5259',1E'1.8068',1E'2.1306',1E'2.5129'
01DAA 0E7453C1
01DAC 11087B41
01DAE 141A6B41
01DB0 17D66CC1 (00063)        DATA 1E'2.9797',1E'3.5793',1E'4.4188',1E'5.8330'
01DB2 1CA26B41
01DB4 2359B3C1
01DB6 2FA9EBC1
01DB8 883126C0 (00064)        DATA 1E'-0.0640',1E'-0.2004',1E'-0.3461',1E'-0.5025'
01DBA 99A66540
01DBC AC3D0140
01DBE C0516EC0
01DC0 D5F3B640 (00065)        DATA 1E'-0.6715',1E'-0.8550',1E'-1.0559',1E'-1.2779'
01DC2 ED70A3C0
01DC4 88727BC1
01DC6 8A3923C1
01DC8 8C350B41 (00066)        DATA 1E'-1.5259',1E'-1.8068',1E'-2.1306',1E'-2.5129'
01DCA 8E7453C1
01DCC 91087B41
01DCE 941A6B41
01DD0 97D66CC1 (00067)        DATA 1E'-2.9797',1E'-3.5793',1E'-4.4188',1E'-5.8330'
01DD2 9CA26B41
01DD4 A359B3C1
01DD6 AFA9EBC1
               (00068)        BUS 1
      000077E0 (00069)        #L=START
               (00070)        EVEN
               (00071)        EJECT
```

```
                            (00072) *
                            (00073) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                            (00074) *
                            (00075)          EVEN
    077F0 0000              (00076)          DATA QUANSSA                   START ADDRESS
    077F1 0045              (00077)          DATA QUANSSZ                   MODULE SIZE
                            (00078) QUANS    BEGIN APU(QUAN)
                   00000000 (00079)          #A=0
A00 077F2 20530000 (00080) QUANSSA SET(UN)\NOP                FIXED PT. OP.
A01 077F4 08FC0000 (00081) STARTS  MOV(IQA,OO)\NOP            IBIT(I)->INSTRS1+1
A02 077F6 90180002 (00082) #1      JUMPC(#1,OOF)              WAIT4 INSTRS1+1 REWRITE
A03 077F8 20370000 (00083)         CLEAR(WI)\NOP              RELEASE APS INPUT
A04 077FA 20500000 (00084) LOOPS   SET(RO)\NOP                RELEASE APS OUTPUT
A05 077FC 08FC0000 (00085)         MOV(IQA,OO)\NOP            INSTRS1 INTO INS1
A06 077FE 08FF0000 (00086)         MOV(IQA,M7)\NOP            M7=TMP4(I)=DCT1(J)
A07 077F0 08F00000 (00087)         MOV(IQA,A0)\NOP            A0=M=QDCT(I)
A08 077F2 40000000 (00088)         ADD(A0,A0)\NOP             2M
A09 077F4 90180009 (00089) #2      JUMPC(#2,OOF)              WAIT4 INS1 REWRITE
A0A 077F6 20500000 (00090)         SET(RO)\NOP                RELEASE APS OUTPUT
A0B 077F8 20370000 (00091)         CLEAR(WI)\NOP              RELEASE APS INPUT
A0C 077FA 089C0000 (00092)         MOV(R,OO)\NOP              2M->INSTRS2+1
A0D 077FC 901 000D (00093) #5      JUMPC(#5,OQF)              WAIT4 INSTRS2+1 REWRITE
A0E 077FE 9017000E (00094) #7      JUMPC(#7,WI)               WAIT4 APS TO STALL!
A0F 07800 20370000 (00095)         CLEAR(WI)\NOP              RELEASE APS INPUT
A10 07802 20500000 (00096)         SET(RO)\NOP                RELEASE APS OUTPUT
A11 07804 08FC0000 (00097)         MOV(IQA,OO)\NOP            INSTRS2 INTO INS2
A12 07806 20330000 (00098)         CLEAR(UN)\NOP              FLOATING PT. OP.
A13 07808 90180013 (00099) #6      JUMPC(#6,OOF)              WAIT4 INS2 REWRITE
A14 0780A 20370000 (00100)         CLEAR(WI)\NOP              RELEASE APS INPUT
A15 0780C 20500000 (00101)         SET(RO)\NOP                RELEASE APS OUTPUT
A16 0780E 08F80000 (00102)         MOV(IQA,M0)\NOP            M0=DEC#(2M)
A17 07810 84500000 (00103)         MUL(M0,M7)\NOP             DEC$#(2M)*TMP4(I)
A18 07812 08FC0000 (00104)         MOV(IQA,OO)\NOP            IBIT(I+1)->INSTRS1+1
A19 07814 088C0000 (00105)         MOV(P,OO)\NOP              OO=TMP1(I)=>DRDCT(J)
A1A 07816 20530000 (00106)         SET(UN)\NOP                FIXED PT. OP.
A1B 07818 9018001B (00107) #3      JUMPC(#3,OOF)              WAIT4 FOR 2 OO'S TO SETT
                   (00108) ;                                  LE
A1C 0781A 20370000 (00109)         CLEAR(WI)\NOP              RELEASE APS INPUT
A1D 0781C 90080004 (00110)         JUMPC(LOOPS,AF3)           FOR I=0,1,2...LTH-1
                   (00111) *
A1E 0781E 20500000 (00112) SHUFLSV SET(RO)\NOP                RELEASE APS OUTPUT
A1F 07820 081C081C (00113)         MOV(ZERO,OO)               INSTRS4+1=0;INSTRS5+1=0
A20 07822 081C081C (00114)         MOV(ZERO,OO)               INSTRS6+1=0;INSTRS7+1=0
A21 07824 20370000 (00115)         CLEAR(WI)\NOP              RELEASE APS INPUT
```

```
A22 07826 08F0C000 (00116)         MOV(IQA,A0)\NOP               A0=J
A23 07828 000008F0 (00117)         NOP\MOV(IQA,A0)               A0=K
A24 0782A 40004000 (00118)         ADD(A0,A0)                    2J;2K
A25 0782C 08F10000 (00119)         MOV(IQA,A1)\NOP               A1=J'
A26 0782E 000008F1 (00120)         NOP\MOV(IQA,A1)               A1=K'
A27 07830 413C413C (00121)         MOV(DQ),ADD(A1,A1)            2J->INSTR$6+1,2J';2K->IN
                   (00122) ;                                     STR$7+1,2K'
A28 07832 08FC0000 (00123)         MOV(IQA,DQ)\NOP               INSTR$4 INTO INDS4
A29 07834 000008FC (00124)         NOP\MOV(IQA,DQ)               INSTR$5 INTO INDS5
A2A 07836 08F00000 (00125)         MOV(IQA,NULL)\NOP             NULL=TMP1(J*)
A2B 07838 000008F0 (00126)         NOP\MOV(IQA,NULL)             NULL=TMP1(K*)
A2C 0783A 08F20000 (00127)         MOV(IQA,A2)\NOP               A2=J''
A2D 0783C 000008F2 (00128)         NOP\MOV(IQA,A2)               A2=K''
A2E 0783E 425C425C (00129)         MOV(DQ),ADD(A2,A2)            2J'->INSTR$4+1,2J'';2K'-
                   (00130) ;                                     >INSTR$5+1,2K''
A2F 07840 08FC0000 (00131)         MOV(IQA,DQ)\NOP               INSTR$6 INTO INDS6
A30 07842 000008FC (00132)         NOP\MOV(IQA,DQ)               INSTR$7 INTO INDS7
A31 07844 08F00000 (00133)         MOV(IQA,NULL)\NOP             NULL=TMP1(J*)
A32 07846 000008F0 (00134)         NOP\MOV(IQA,NULL)             NULL=TMP1(K*)
A33 07848 08F10000 (00135) SHUFLS  MOV(IQA,A1)\NOP               A1=J'
A34 0784A 000008F1 (00136)         NOP\MOV(IQA,A1)               A1=K'
A35 0784C 413C413C (00137)         MOV(DQ),ADD(A1,A1)            2J->INSTR$6+1,2J';2K->IN
                   (00138) ;                                     STR$7+1,2K'
A36 0784E 08FC0000 (00139)         MOV(IQA,DQ)\NOP               INSTR$4 INTO INDS4
A37 07850 000008FC (00140)         NOP\MOV(IQA,DQ)               INSTR$5 INTO INDS5
A38 07852 08FC0000 (00141)         MOV(IQA,DQ)\NOP               DRDCT(I)=TMP1(J*)
A39 07854 000008FC (00142)         NOP\MOV(IQA,DQ)               DRDCT(I+1)=TMP1(K*)
A3A 07856 08F20000 (00143)         MOV(IQA,A2)\NOP               A2=J''
A3B 07858 000008F2 (00144)         NOP\MOV(IQA,A2)               A2=K''
A3C 0785A 425C425C (00145)         MOV(DQ),ADD(A2,A2)            2J'->INSTR$4+1,2J'';2K'-
                   (00146) ;                                     >INSTR$5+1,2K''
A3D 0785C 08FC0000 (00147)         MOV(IQA,DQ)\NOP               INSTR$6 INTO INDS6
A3E 0785E 000008FC (00148)         NOP\MOV(IQA,DQ)               INSTR$7 INTO INDS7
A3F 07860 08FC0000 (00149)         MOV(IQA,DQ)\NOP               DRDCT(I+2)=TMP1(J*)
A40 07862 000008FC (00150)         NOP\MOV(IQA,DQ)               DRDCT(I+3)=TMP1(K*)
A41 07864 90100033 (00151)         JUMPC(SHUFLS,F1)              FOR I=0,1,2,..LTH-1
                   (00152) *
A42 07866 20420000 (00153) DNESA   CLEAR(RA)\NOP                 APU DONE!
A43 07868 00000000 (00154)         NOP
A44 0786A 10000000 (00155)         JUMP(0)
          00000045 (00156)         QUANSSZ=*A-QUANSSA
    0786C          (00157)         END QUANSSZ
                   (00158)         EJECT
```

```
                       (00159) *
                       (00160) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                       (00161) *
                       (00162)         EVEN
    07B6C 00007960     (00163)         ADDR G104S1                POINTER TO CONSTRUCTED I
                       (00164) *                                  NSTRUCTION BLOCK
    07B6E 00007B76     (00165)         ADDR G104S+2*SCLRS         POINTER TO SCALAR BLOCK
    07B70 0000         (00166)         DATA 0                     NUMBER OF SCALARS
    07B71 00E6         (00167)         DATA G104S7                MODULE SIZE
    07B72 0000792C     (00168)         ADDR G104SA                POINTER TO CHAIN ANCHOR
                       (00169)         EVEN                       END ON WORD BOUNDARY
                       (00170) *
                       (00171) G104S   BEGIN APS(G104)
A00 07B74 00204340     (00172)         JSN(G104S0,P2)             SET OUTPUT PC(PC2)
A01 07B76 02300030     (00173) SCLRS   SET(R0)                    ENABLE APS OUTPUT
A02 07B78 04601000     (00174)         LOAD(BR2,[1])              QDCT(,) BA
A03 07B7A 06500000     (00175)         LOAD(BR1,MSS)              DUMMY OP FOR EXEC
A04 07B7C 08220000     (00176)         SUB(BR2,MSS)               QDCT(,) BA-1
A05 07B7E 0A742A27     (00177)         LOAD(BR3,IBITS-1(2))       IBIT(,) BA-1
A06 07B80 0C442526     (00178)         LOAD(BR0,TMP4S-2(2))       TMP4(,) BA-2
A07 07B82 0E3A0001     (00179)         ADD(BR3,BS,TE)             IQ=IBIT(I)
A08 07B84 10300037     (00180)         SET(WI)                    STALL APS INPUT
A09 07B86 12000020     (00181)         NOP                        RELEASED BY APU
A0A 07B88 14D41D4C     (00182) LOOPS1  LOAD(BR1,INSTRS1(2),TE)    REWRITE INS1
A0B 07B8A 16BA0002     (00183)         ADD(BR0,WS,TE)             IQ=TMP4(I)=>DCT1(J)
A0C 07B8C 192A0001     (00184)         ADD(BR2,BS,TE)             IQ=QDCT(I)
A0D 07B8E 1A300037     (00185)         SET(WI)                    STALL APS INPUT
A0E 07B90 1C310011     (00186)         MOVB(BW3,BR0)              SAVE TMP4(,) ADDRESS
A0F 07B92 1E500000     (00187) INS1    LOAD(BR1,MSS)              APU SETS MSS=BITS
A10 07B94 20191FF1     (00188)         SUBL(BR1,1),JUMPN(BS0)     BS0 FOR BITS=0
A11 07B96 22191DF1     (00189)         SUBL(BR1,1),JUMPN(BS1)     BS1 FOR BITS=1
A12 07B98 24191BF1     (00190)         SUBL(BR1,1),JUMPN(BS2)     BS2 FOR BITS=2
A13 07B9A 261919F1     (00191)         SUBL(BR1,1),JUMPN(BS3)     BS3 FOR BITS=3
A14 07B9C 281917F1     (00192)         SUBL(BR1,1),JUMPN(BS4)     BS4 FOR BITS=4
A15 07B9E 2A541D98     (00193) BS5     LOAD(BR1,DECS5(2))         DECS5(,) BA
A16 07BA0 2C302177     (00194)         JUMP(STALS2,WI),SET        STALL APS INPUT
A17 07BA2 2E541D78     (00195) BS4     LOAD(BR1,DECS4(2))         DECS4(,) BA
A18 07BA4 30302177     (00196)         JUMP(STALS2,WI),SET        STALL APS INPUT
A19 07BA6 32541D68     (00197) BS3     LOAD(BR1,DECS3(2))         DECS3(,) BA
A1A 07BA8 34302177     (00198)         JUMP(STALS2,WI),SET        STALL APS INPUT
A1B 07BAA 36541D60     (00199) BS2     LOAD(BR1,DECS2(2))         DECS2(,) BA
A1C 07BAC 38302177     (00200)         JUMP(STALS2,WI),SET        STALL APS INPUT
A1D 07BAE 3A541D5C     (00201) BS1     LOAD(BR1,DECS1(2))         DECS1(,) BA
A1E 07BB0 3C302177     (00202)         JUMP(STALS2,WI),SET        STALL APS INPUT
```

```
A1F 07BB2 3E541058 (00203) BS0      LOAD(BR1,DECS0(2))             DECS0(.) BA
A20 07BB4 40300037 (00204)          SET(W1)                        STALL APS INPUT
A21 07BB6 32000020 (00205) STALS2   NOP                            RELEASED BY APU
A22 07BB8 44C4104E (00206)          LOAD(BR0,INSTRS2(2),TF)        REWRITE INS2
A23 07BBA 46300037 (00207)          SET(W1)                        STALL APS INPUT
A24 07BBC 48090016 (00208)          MOVB(BR0,B*3)                  RECALL TMP4(.) ADDRESS
A25 07BBE 4A9A0000 (00209) INS2     ADD(BR1,MSS,TF)                APU SETS MSS,IO=2M
A26 07BC0 4C3A0001 (00210)          ADD(BR3,MS,TF)                 IO=IBIT(I+1)
A27 07BC2 4E300037 (00211)          SET(W1)                        STALL APS INPUT
A28 07BC4 50000020 (00212)          NOP                            RELEASED BY APU
A29 07BC6 5200AAAB (00213)          JUMPC(LOOPS1,AF3)              FOR I=0,1,2,..LTH-1
A2A 07BC8 54300037 (00214)          SET(W1)                        STALL APS INPUT
A2B 07BCA 56000020 (00215)          NOP                            RELEASED BY APU
                   (00216)
                   (00217) *
A2C 07BCC 59403054 (00218)          LOAD(BR0,I3),TF)               IORDR(.) BA,IO=IORDR=J
A2D 07BCE 5A500000 (00219)          LOAD(BR1,MSS)                  IORDR(.) SIZE-1
A2E 07BD0 5D0A0000 (00220)          ADD(BR0,MSS,TF)                IO=IORDR(1)=K
A2F 07BD2 5E602006 (00221)          LOAD(BR2,I21)                  TMP1(.) BA
A30 07BD4 60700000 (00222)          LOAD(BR3,MSS)                  DUMMY OP FOR EXEC
A31 07BD6 62220000 (00223)          SUB(BR2,MSS)                   TMP1(.) BA-2
A32 07BD8 6322000B (00224)          SUB(BR2,D'8')                  TMP1(.) BA-10.
A33 07BDA 6619003C (00225)          ADDL(BR1,4)                    1 EXTRA LOOP IS NEEDED
A34 07BDC 690A0001 (00226) INDSJ    ADD(BR0,MS,TF)                 IO=J'=IORDR(I+2)
A35 07BDE 6B0A0001 (00227)          ADD(BR0,MS,TF)                 IO=K'=IORDR(I+3)
A36 07BE0 6CF41050 (00228)          LOAD(BR3,INSTRS4(2),TF)        REWRITE INDS4
A37 07BE2 6EF41052 (00229)          LOAD(BR3,INSTRS5(2),TF)        REWRITE INDS5
A38 07BE4 70AA0002 (00230)          ADD(BR2,WS,TF)                 IO=TMP1(I)
A39 07BE6 72AA0002 (00231)          ADD(BR2,WS,TF)                 IO=TMP1(I+1)
A3A 07BE8 750A0001 (00232)          ADD(BR0,MS,TF)                 IO=J''
A3B 07BEA 770A0001 (00233)          ADD(BR0,MS,TF)                 IO=K''
A3C 07BEC 78F41054 (00234)          LOAD(BR3,INSTRS6(2),TF)        REWRITE INDS6
A3D 07BEE 7AF41056 (00235)          LOAD(BR3,INSTRS7(2),TF)        REWRITE INDS7
A3E 07BF0 7CAA0002 (00236)          ADD(BR2,WS,TF)                 IO=TMP1(I+2)
A3F 07BF2 7EAA0002 (00237)          ADD(BR2,WS,TF)                 IO=TMP1(I+3)
A40 07BF4 80193484 (00238)          SUBL(BR1,4),JUMPP(INDSJ)       FOR I=0,1,2,..LTH+1
                   (00239) *
A41 07BF6 82200031 (00240) DNES1    CLEAR(RI)                      INPUT DONE!
A42 07BF8 84000020 (00241)          NOP
                   (00242) *
A43 07BFA 86300032 (00243) G104S0   SET(RA)                        ENABLE APU
A44 07BFC 8830202A (00244)          LOAD(BW0,I21)                  TMP1(.) BA
A45 07BFE 8A500000 (00245)          LOAD(BW1,MSS)                  TMP1(.) SIZE-1
A46 07900 8C020000 (00246)          SUB(BW0,MSS)                   TMP1(.) BA-2
```

PAGE 8: FCW 251... "MPESTD (,D,V,W)" DEQUANTIZE DCT PARAMETERS

```
A47 07902 8E641D3D (00247)          LOAD(BW2,INSTRS1+1(2),TF)        SET MSS IN INSTRS1
A48 07904 90200030 (00248)          CLEAR(R0)                        STALL APS OUTPUT
A49 07906 92000020 (00249)          NOP                              RELEASED BY APU
A4A 07908 94E3FFC0 (00250) OINCSJ   LOAD(BW2,APSSMEM(1),TF)          REWRITE INS1
A4B 0790A 96200030 (00251)          CLEAR(R0)                        STALL APS OUTPUT
A4C 0790C 98000020 (00252)          NOP                              RELEASED BY APU
A4D 0790E 9A641D3F (00253)          LOAD(BW2,INSTRS2+1(2),TF)        SET MSS=2M IN INSTRS2
A4E 07910 9C200030 (00254)          CLEAR(R0)                        STALL APS OUTPUT
A4F 07912 9E000020 (00255)          NOP                              RELEASED BY APU
A50 07914 A0E3FFC0 (00256)          LOAD(BW2,APSSMEM(1),TF)          REWRITE INS2
A51 07916 A2200030 (00257)          CLEAR(R0)                        STALL APS OUTPUT
A52 07918 A4000020 (00258)          NOP                              RELEASED BY APU
A53 0791A A7641D40 (00259)          LOAD(BW2,INSTRS1+1(2),TF)        SET MSS IN INSTRS1
A54 0791C AEBA0002 (00260)          ADD(BW0,WS,TF)                   00=TMP1(I)=>DRDCT(J)
A55 0791E AA114AB1 (00261)          SUBL(BW1,1),JUMPP(OINCSJ)        FOR I=0,1...LTH-1
A56 07920 AC300028 (00262)          SET(AF3)                         SIGNAL END TO APS INPUT
A57 07922 AE200030 (00263)          CLEAR(R0)                        STALL APS OUTPUT
                   (00263) *
A58 07924 B1441D51 (00265)          LOAD(BW0,INSTRS4+1(2),TF)        00=0!
A59 07926 B3441D53 (00266)          LOAD(BW0,INSTRS5+1(2),TF)        00=0!
A5A 07928 B5341D55 (00267)          LOAD(BW0,INSTRS6+1(2),TF)        00=0!
A5B 0792A B7441D57 (00268)          LOAD(BW0,INSTRS7+1(2),TF)        00=0!
A5C 0792C B8400030 (00269)          LOAD(BW0,[0])                    DUMMY OP FOR EXEC
A5D 0792E BA500000 (00270)          LOAD(BW1,MSS)                    DRDCT(,) SIZE-1
A5E 07930 BC020000 (00271)          SUB(BW0,MSS)                     DUMMY OP FOR EXEC
A5F 07932 BE700002 (00272)          LOAD(BW3,2)                      4 INTERMEDIARY DUMMIES..
A60 07934 C011003C (00273)          ADDL(BW1,4)                      & 1 EXTRA LOOP
A61 07936 C3641D55 (00274) INDS1    LOAD(BW2,INSTRS6+1(2),TF)        SET MSS IN INSTRS6
A62 07938 C5641D57 (00275)          LOAD(BW2,INSTRS7+1(2),TF)        SET MSS IN INSTRS7
A63 0793A C6E3FFC0 (00276)          LOAD(BW2,APSSMEM(1),TF)          OVERWRITE INDS4
A64 0793C C8E3FFC0 (00277)          LOAD(BW2,APSSMEM(1),TF)          OVERWRITE INDS5
A65 0793E CA316AB1 (00278)          SUBL(BW3,1),JUMPP(DMYS1)         1 ST 4 00'S ARE ?
A66 07940 CC660100 (00279)          LOAD(BW2,DRDCTS(3))              DRDCT(,) BA
A67 07942 CEAA0000 (00280) INDS6    ADD(BW2,MSS,TF)                  00=DRDCT(J*)
A68 07944 D0660400 (00281)          LOAD(BW2,DRDCTS(3))              DRDCT(,) BA
A69 07946 D2AA0000 (00282) INDS7    ADD(BW2,MSS,TF)                  00=DRDCT(K*)
A6A 07948 D5641D51 (00283) DMYS1    LOAD(BW2,INSTRS4+1(2),TF)        SET MSS IN INSTRS4
A6B 0794A D7641D53 (00284)          LOAD(BW2,INSTRS5+1(2),TF)        SET MSS IN INSTRS5
A6C 0794C D8E3FFC0 (00285)          LOAD(BW2,APSSMEM(1),TF)          OVERWRITE INDS6
A6D 0794E DAE3FFC0 (00286)          LOAD(BW2,APSSMEM(1),TF)          OVERWRITE INDS7
A6E 07950 DC3173B1 (00287)          SUBL(BW3,1),JUMPP(DMYS2)         1 ST 4 00'S ARE ?
A6F 07952 DE660400 (00288)          LOAD(BW2,DRDCTS(3))              DRDCT(,) BA
A70 07954 E0AA0000 (00289) INDS4    ADD(BW2,MSS,TF)                  00=DRDCT(J*)
A71 07956 E2660100 (00290)          LOAD(BW2,DRDCTS(3))              DRDCT(,) BA
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

4 OF 9

AD
A091663

```
A72 0795B F4AA0000 (00291) INDS5    ADD(BW2,MSS,TE)                          DQ=DRDCT(K*)
A73 0795A F6116184 (00292) DMYS2    SUBL(BW1,4),JUMPP(INDS1)                 FOR I=-3,-2...LTH-1
                   (00293) *
A74 0795C FB200030 (00294) DNESO    CLEAR(RO)                                OUTPUT DONE!
A75 0795F FA000020 (00295)          NOP
                   (00296) *
        0000792C (00297)          G104SA= *C                               ASSIGN VALUE TO CHAIN AN
                   (00298) ;                                                 CHOR
    07960          (00299)          END *A-1
                   (00300) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    07960 00000000 (00301) G104SI   DATA 5F'0,0'
     ...
          000000F6 (00302)          G104SZ=*L-G104S
    0796A          (00303)          END
```

```
AFDTSORG:   00BFB (00009) (00031)
APSSMEM:    1FFC0 (00010) (00250) (00256) (00276) (00277) (00285) (00286)
BS0:        0001F (00188) (00203)
BS1:        0001D (00189) (00201)
BS2:        0001B (00190) (00199)
BS3:        00019 (00191) (00197)
BS4:        00017 (00192) (00195)
BS5:        00015 (00193)
CSPUSNOS:   021FC (00011) (00034)
DFCS0:      01D58 (00048) (00203)
DFCS1:      01D5C (00050) (00201)
DFCS2:      01D60 (00052) (00199)
DFCS3:      01D6B (00054) (00197)
DFCS4:      01D78 (00056) (00195)
DFCS5:      01D98 (00060) (00193)
DMYS1:      0006A (00278) (00283)
DMYS2:      00073 (00287) (00292)
DNFSA:      00042 (00153)
DNFSI:      00041 (00240)
DNFSU:      00074 (00294)
DRDCTS:     00400 (00012) (00279) (00281) (00288) (00290)
G104S:      07874 (00033) (00165) (00171) (00302)
G104SA:     0792C (00168) (00297)
G104SI:     07960 (00163) (00301)
G104SU:     00043 (00172) (00243)
G104SZ:     000F6 (00167) (00302)
HS:         00001 (00014) (00179) (00184) (00210) (00226) (00227) (00232) (00233)
IBITS:      02A28 (00017) (00177)
INS1:       0000F (00042) (00187)
INS2:       00025 (00043) (00209)
INDS4:      00070 (00044) (00289)
INDS5:      00072 (00045) (00291)
INDS6:      00067 (00046) (00280)
INDS7:      00069 (00047) (00282)
INDSI:      00061 (00274) (00292)
INDSJ:      00034 (00226) (00238)
INSTRS1:    01D4C (00042) (00182) (00247) (00259)
INSTRS2:    01D4F (00043) (00206) (00253)
INSTRS4:    01D50 (00044) (00228) (00265) (00283)
INSTRS5:    01D52 (00045) (00229) (00266) (00284)
INSTRS6:    01D54 (00046) (00234) (00267) (00274)
INSTRS7:    01D56 (00047) (00235) (00268) (00275)
LOOPS:      00004 (00084) (00110)
LOOPSI:     0000A (00182) (00213)
```

```
 MS:        00000 (00018)
 MSS:       00000 (00019) (00175) (00176) (00187) (00209) (00219) (00220) (00222) (00223) (00245)
                  (00246) (00270) (00271) (00280) (00282) (00289) (00291)
 DINCSJ:    0004A (00250) (00261)
 QUANS:     077F2 (00032) (00078)
 QUANSSA:   00000 (00076) (00080) (00156)
 QUANSSZ:   00045 (00077) (00156) (00157)
 SAS38:     003CF (00021)
 SAS39:     003D0 (00022)
 SCLRS:     00001 (00165) (00173)
 SHUFLS:    00033 (00135) (00151)
 SHUFLSV:   0001F (00112)
 STALS2:    00021 (00194) (00196) (00198) (00200) (00202) (00205)
 START:     077E0 (00023) (00069)
 STARTS:    00001 (00081)
 SVTS:      003B2 (00020) (00021) (00022)
 TMP4S:     02528 (00026) (00178)
 WS:        00002 (00027) (00183) (00230) (00231) (00236) (00237) (00260)
 ZS:        00003 (00028)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   F=    0
```

```
                   (00001) *  FCB 252... "MPVDNM(Y,U,V)"      VAR.,DC,INTERPOLATE,FIX
                   (00002) *                                  ORIGINATED:18-JAN-80
                   (00003) *                                  UPDATED:29-MAY-80
                   (00004) *  DEFINE GLOBAL SYMBOLS
                   (00005)         OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1F
                   (00006)         OPADD OUF, (3 .LS. 10) + (12 .LS. 5) + $1B
                   (00007)         OPADD WAFNF, (1 .LS. 10) + (12 .LS. 5) + $1A
                   (00008)         OPADD FO, (3 .LS. 10) + (12 .LS. 5) + $1C
          00000BFB (00009)         AFDTSORG=$BFB
          000021FC (00010)         CSPUSNOS=$21FC
          00000794 (00011)         DMYS=$0794
          00000003 (00012)         #M=3
          00000001 (00013)         US=1
          00000000 (00014)         MS=0
          00000000 (00015)         MSS=0
          00000F16 (00016)         NOUTS=D'3862'
          000000F5 (00017)         NSIZES=D'246'-D'1'
          00000382 (00018)         SVTS=$0382
          00000398 (00019)         SAS11=SVTS+2*D'11'
          000003CF (00020)         SAS38=SVTS+2*D'38'
          000003D0 (00021)         SAS39=SVTS+2*D'39'
          0000044A (00022)         SAS100=SVTS+2*D'100'
          0000044C (00023)         SAS101=SVTS+2*D'101'
          00007980 (00024)         START=$7980
          00000002 (00025)         WS=2
          00000004 (00026)         WWS=4
          00000800 (00027)         X2S=D'2048'
          000000FF (00028)         XSIZES=D'256'-D'1'
          00000800 (00029)         YDMS=D'2048'
          00000009 (00030)         YSIZES=D'10'-D'1'
          00000003 (00031)         ZS=3
                   (00032) *
                   (00033) *  EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000BD0 (00034)         #L=AFDTSORG+3*2*(252-128)                      FCB 252
00BD0 001F7998 (00035)         ADDR VDINS(R7,1)
00BD2 001F7A12 (00036)         ADDR G100S(R7,1)
00BD4 001021FC (00037)         ADDR CSPUSNOS(,1,0)
                   (00038)         EJECT
```

```
                 (00039) *
                 (00040) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                 (00041) *
      00007980   (00042)             #L=START
                 (00043)             EVEN
07980 00000000   (00044) FS0         DATA 1F'0.0'
07982 0CCCCCC0   (00045) FS1         DATA 1F'0.1'
07984 199999C0   (00046) FS2         DATA 1F'0.2'
07986 26666640   (00047) FS3         DATA 1F'0.3'
07988 33333340   (00048) FS4         DATA 1F'0.4'
0798A 40000040   (00049) FS5         DATA 1F'0.5'
0798C 4CCCCCC0   (00050) FS6         DATA 1F'0.6'
0798E 599999C0   (00051) FS7         DATA 1F'0.7'
07990 66666640   (00052) FS8         DATA 1F'0.8'
07992 73333340   (00053) FS9         DATA 1F'0.9'
07994 08000041   (00054) FSA         DATA 1F'1.0'
                 (00055)             EJECT
```

```
                       (00056) *
                       (00057) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                       (00058) *
                       (00059)            EVEN
    07996 0000         (00060)            DATA VDIN$SA                           START ADDRESS
    07997 0039         (00061)            DATA VDIN$SZ                           MODULE SIZE
                       (00062) VDIN$      BEGIN APU(VDIN)
          00000000     (00063)            #A=0
A00 07998 00000000     (00064) VDIN$SA    NOP                                    START-UP
A01 0799A 08F708F5     (00065)            MOV(IQA,A7)\MOV(IQA,A5)                .1
A02 0799C 47F01680     (00066)            ADD(A7,A7)\R(1)
A03 0799E 08FF08FF     (00067)            MOV(IQA,M7)                            VAR
A04 079A0 08F308F3     (00068)            MOV(IQA,A3)                            DC
A05 079A2 08980898     (00069)            MOV(R,EXO)                             .2\1.0
A06 079A4 0AF746B6     (00070)            MOV(A7),NEG(A7)\MOV(A6),ADD(A5,A6)     .2;-.2\1.0;1,1
A07 079A6 08560657     (00071)            MOV(EXI,A6)\MOV(EXI,A7)                1.0\.2
A08 079A8 08FF08FF     (00072)            MOV(IQA,M6)                            SA (=2**16)
A09 079AA 47D50AB6     (00073)            MOV(A5),ADD(A6,A7)\MOV(A6),NEG(A5)     -.2=NMPY(-2);1.2\1.1=OMP
                       (00074) ;                                                 Y(-1);.1
A0A 079AC 08F208F2     (00075)            MOV(IQA,A2)                            SB
A0B 079AE 08F80000     (00076)            MOV(IQA,M3)\NOP                        NEW(0)
A0C 079B0 08960895     (00077)            MOV(R,A6)\MOV(R,A5)                    1.2=OMPY(-2)\.1=NMPY(-1)
A0D 079B2 000008FB     (00078)            NOP\MOV(IQA,M3)
                       (00079) *LOOP FOR I=0,2,...,N
A0E 079B4 85F185F1     (00080) #1         MOV(A1),MUL(M3,M7)                     MNEW(I-2);VAR*NEW(I)
A0F 079B6 411C411C     (00081)            MOV(OQ),ADD(A0,A1)                     OUTPUT FIX(I-4);MOLD(I-2
                       (00082) ;                                                 )+MNEW(I-2)=INT(I-2)
A10 079B8 08F80000     (00083)            MOV(IQA,M0)\NOP                        OLD(I)
A11 079BA 000008F8     (00084)            NOP\MOV(IQA,M0)                        OLD(I+1)
A12 079BC 4FCA4FCA     (00085)            MOV(M2),SUB(A6,A7)                     INT(I-2);OMPY(I-2)-.2=OM
                       (00086) ;                                                 PY(I)
A13 079BE 85518551     (00087)            MOV(A1),MUL(M2,M6)                     VAR*NEW(I);INT(I-2)*SA
A14 079C0 088C088C     (00088)            MOV(R,M4)                              OMPY(I)
A15 079C2 47B647B6     (00089)            MOV(A6),ADD(A5,A7)                     OMPY(I);NMPY(I-2)+.2=NMP
                       (00090) ;                                                 Y(I)
A16 079C4 088D088D     (00091)            MOV(R,M5)                              NMPY(I)
A17 079C6 43354335     (00092)            MOV(A5),ADD(A1,A3)                     NMPY(I);VAR*NEW(I)+DC=NE
                       (00093) ;                                                 W'(I)
A18 079C8 84148414     (00094)            MOV(A4),MUL(M0,M4)                     INT(I-2)*SA;OLD(I)*OMPY(
                       (00095) ;                                                 I)=MOLD(I)
A19 079CA 02890289     (00096)            MOV(M1),R(A4)                          NEW'(I);INT(I-2)*SA
A1A 079CC 08F80000     (00097)            MOV(IQA,M3)\NOP                        NEW(I+2)
A1B 079CE 000008FB     (00098)            NOP\MOV(IQA,M3)                        NEW(I+3)
A1C 079D0 52805280     (00099)            ADDST(A4,A2)                           ROUND INT(I-2)*SA
```

```
A1D 079D2 84808480 (00100)            MOV(A0),MUL(M1,M5)                               MOLD(I);NEW'(I)*NMPY(I)=
                   (00101) ;                                                           MNEW(I)
A1E 079D4 3A943A94 (00102)            MOV(A4),ALIGN(A4)                                FIX(I-2)
A1F 079D6 9016000F (00103)            JUMPC(#1,FWI)
A20 079D8 20372037 (00104)            CLEAR(WI)
A21 079DA 00000000 (00105)            NOP
A22 079DC 089C089C (00106)            MOV(R,OO)                                        OUTPUT FIX(N-2)\FIX(N-1)
                   (00107) *LOOP FOR I=N+2, N+4, ...., N2+2
A23 079DE 85F485F4 (00108) #2         MOV(A4),MUL(M3,M7)                               NEW'(I-4)*SA;NEW(I)*VAR
A24 079E0 02890289 (00109)            MOV(M1),R(A4)                                    NEW'(I-2);NEW'(I-4)*SA
A25 079E2 52805280 (00110)            ADDST(A4,A2)                                     ROUND NEW'(I-4)*SA
A26 079E4 84D184D1 (00111)            MOV(A1),MUL(M1,M6)                               NEW(I)*VAR);NEW'(I-2)*SA
A27 079E6 3A943A94 (00112)            MOV(A4),ALIGN(A4)                                FIX(I-4)
A28 079E8 08E80000 (00113)            MOV(IOA,M3)\NOP                                  NEW(I+2)
A29 079EA 000008EB (00114)            NOP\MOV(IOA,M3)                                  NEW(I+3)
A2A 079EC 433C433C (00115)            MOV(OO),ADD(A1,A3)                               OUTPUT FIX(I-4);NEW(I)*V
                   (00116) ;                                                           AR+DC=NEW'(I)
A2B 079EE 90160023 (00117)            JUMPC(#2,FWI)
A2C 079F0 20372037 (00118)            CLEAR(WI)
A2D 079F2 00000000 (00119)            NOP
                   (00120) *LOOP FOR I=N2+2, N2+4, ...., N3-2
A2E 079F4 85F085F0 (00121) #3         MUL(M3,M7)                                       NEW(I)*VAR
A2F 079F6 08810881 (00122)            MOV(P,A1)                                        NEW(I)*VAR
A30 079F8 433C433C (00123)            MOV(OO),ADD(A1,A3)                               OUTPUT NEW'(I-2);NEW(I)*
                   (00124) ;                                                           VAR+DC=NEW'(I)
A31 079FA 08EB0000 (00125)            MOV(IOA,M3)\NOP                                  NEW(I+2)
A32 079FC 000008EB (00126)            NOP\MOV(IOA,M3)                                  NEW(I+3)
A33 079FE 9010002F (00127)            JUMPC(#3,FI)
A34 07A00 089C089C (00128)            MOV(R,OO)                                        NEW'(N3-2)\NEW'(N3-1)
                   (00129) *
A35 07A02 00000000 (00130)            NOP                                              PLACE FOR TIMING PATCH
A36 07A04 20260000 (00131) DNESA      CLEAR(G2)\NOP                                    END SYNTHESIZER TIMING
A37 07A06 20322032 (00132)            CLEAR(RA)
A38 07A08 00000000 (00133)            NOP
                   (00134) *
          00000039 (00135)            VDINSSZ=*A-VDINSSA
    07A0A          (00136)            END VDINSSZ
                   (00137)            EJECT
```

```
                      (00138) *
                      (00139) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      (00140) *
                      (00141)         EVEN
    07A0A 00007AA2    (00142)         ADDR G100$I                          POINTER TO CONSTRUCTED I
                      (00143) ;                                            NSTRUCTION BLOCK
    07A0C 00007A14    (00144)         ADDR G100$+2*SCLP$                   POINTER TO SCALAR BLOCK
    07A0E 0000        (00145)         DATA 0                               NUMBER OF SCALARS
    07A0F 0092        (00146)         DATA G100$Z                          MODULE SIZE
    07A10 00007A9C    (00147)         ADDR G100$A                          POINTER TO CHAIN ANCHOR
                      (00148)         EVEN                                 END ON WORD BOUNDARY
                      (00149) *
                      (00150) G100$   BEGIN APS(G100)
A00 07A12 00202F40    (00151)         JSN(G100$0,P2)                       SET OUTPUT PC(PC2)
A01 07A14 02300030    (00152) SCLP$   SET(R0)                              ENABLE APS OUTPUT
A02 07A16 04C27982    (00153)         LOAD(BR0,FS1(1),TF)                  .1
A03 07A18 06C2044C    (00154)         LOAD(BR0,SAS101(1),TF)               VAR
A04 07A1A 08C2044A    (00155)         LOAD(BR0,SAS100(1),TF)               DC
A05 07A1C 0AC203CF    (00156)         LOAD(BR0,SAS38(1),TF)                SA=2**-15
A06 07A1E 0CC2039B    (00157)         LOAD(BR0,SAS11(1),TF)                SB=2**-16
A07 07A20 0EC60800    (00158)         LOAD(BR0,X2S(3),TF)                  X2(,) BA,IO=X2R(0)
A08 07A22 105000FF    (00159)         LOAD(BR1,XSIZES)                     X2(,) SIZE-1
A09 07A24 126407FF    (00160)         LOAD(BR2,YOMS-2(2))                  YOM(,) BA-2
A0A 07A26 14700069    (00161)         LOAD(BR3,YSIZES)                     YOM(,) SIZE-1
A0B 07A28 16210017    (00162)         MOVB(BW2,BR3)
A0C 07A2A 18390011    (00163)         MOVB(BR3,BR0)                        I(-1)
A0D 07A2C 1A39002B    (00164)         ADDB(BR3,BR1)                        I(-1)+N-1
A0E 07A2E 1C39002B    (00165)         ADDB(BR3,BR1)                        I(-1)+2N-2
A0F 07A30 1E39002B    (00166)         ADDB(BR3,BR1)                        I(-1)+3N-3
A10 07A32 20B9002B    (00167)         ADDB(BR3,BR1,TF)                     R(N-1)=NEW(1)
A11 07A34 22190024    (00168)         SUBB(BR1,BW2)                        NEWZ-OLDZ=FIXZ
A12 07A36 24190024    (00169)         SUBB(BR1,BW2)                        NEWZ-2*OLDZ+1
                      (00170) *LOOP FOR I=0,2,...,N-2 WHERE N=OLDZ
A13 07A38 26AA0002    (00171) #1      ADD(BR2,2,TF)                        OLD(I)
A14 07A3A 28AA0002    (00172)         ADD(BR2,2,TF)                        OLD(I+1)
A15 07A3C 2ABA0004    (00173)         ADD(BR0,4,TF)                        NEW(I+2)
A16 07A3E 2CB90034    (00174)         SUBL(BR3,4,TF)                       NEW(I+2)
A17 07A40 2E2113B2    (00175)         SUBL(BW2,2),JUMPP(#1)
A18 07A42 30B20004    (00176)         SUB(BR0,4,TF)                        DUMMY
A19 07A44 32BA0004    (00177)         ADD(BR3,4,TF)                        DUMMY
A1A 07A46 34BA0004    (00178)         ADD(BR0,4,TF)                        NEW(N)
A1B 07A48 36B90034    (00179)         SUBL(BR3,4,TF)                       NEW(N+1)
A1C 07A4A 38300037    (00180)         SET(W1)                              STALL APS INPUT
                      (00181) *LOOP FOR I=0,2,...,N2-N-4
```

```
A1D 07A4C 3A190034 (00182)          SUBL(BR1,4)                          N2=N-3
A1E 07A4E 3CHA0004 (00183) #2       ADD(BR0,4,TE)                        NEW(N+I+2)
A1F 07A50 3EB90034 (00184)          SUBL(BR3,4,TE)
A20 07A52 40191EB2 (00185)          SUBL(BR1,2),JUMPP(#2)
A21 07A54 42020004 (00186)          SUB(BR0,4)
A22 07A56 4439003C (00187)          ADDL(BR3,4)
A23 07A58 46500002 (00188)          LOAD(BR1,2)
                   (00189) *LOOP FOR I=N2-4,N2-2,...,N2 OR 6 DUMMIES
A24 07A5A 488A0004 (00190) #3       ADD(BR0,4,TE)                        NEW(I)
A25 07A5C 4AB90034 (00191)          SUBL(BR3,4,TE)
A26 07A5E 4C1924B1 (00192)          SUBL(BR1,1),JUMPP(#3)
A27 07A60 4E300037 (00193)          SET(WI)                              STALL APS INPUT
A28 07A62 50640802 (00194)          LOAD(BR2,YOMS+2(2))                  YOM(,) BA+2
A29 07A64 52500009 (00195)          LOAD(BR1,YSIZES)                     YOM(,) SIZE-1
                   (00196) *LOOP FOR I=0,2,...N2+OLDZ-2=N3-2
A2A 07A66 548A0004 (00197) #4       ADD(BR0,4,TE)
A2B 07A68 56B90034 (00198)          SUBL(BR3,4,TE)
A2C 07A6A 58192AB2 (00199)          SUBL(BR1,2),JUMPP(#4)
                   (00200) *
A2D 07A6C 5A200031 (00201) DNESI    CLEAR(RI)                            INPUT DONE!
A2E 07A6E 5C000020 (00202)          NOP
                   (00203) *
A2F 07A70 5E300032 (00204) G100SO   SET(RA)                              ENABLE APU
A30 07A72 60500003 (00205)          LOAD(BW1,3)                          3+1=4 DUMMIES
A31 07A74 63420794 (00206) DMYSO    LOAD(BW0,DMYS(1),TE)                 OO=?
A32 07A76 64113181 (00207)          SUBL(BW1,1),JUMPP(DMYSO)             FOR I=0,1,..3
A33 07A78 66440F15 (00208)          LOAD(BW0,NOUTS-1(2))                 NOUT(,) BA-1
A34 07A7A 68500000 (00209)          LOAD(BW1,MSS)                        NOUT(,) SIZE-1
A35 07A7C 6A700009 (00210)          LOAD(BW3,YSIZES)                     YOM(,) SIZE-1
                   (00211) *  INTERPOLATED OUTPUTS
A36 07A7E 6D0A0001 (00212) #1       ADD(BW0,HS,TE)                       OO=NOUT(I)
A37 07A80 6F313681 (00213)          SUBL(BW3,1),JUMPP(#1)                FOR I=0,1,...NSIZE-1
A38 07A82 71720794 (00214)          LOAD(BW3,DMYS(1),TE)                 OO=?
A39 07A84 73720794 (00215)          LOAD(BW3,DMYS(1),TE)                 OO=?
A3A 07A86 75720794 (00216)          LOAD(BW3,DMYS(1),TE)                 OO=?
A3B 07A88 77720794 (00217)          LOAD(BW3,DMYS(1),TE)                 OO=?
                   (00218) *  NON-INTERPOLATED OUTPUTS
A3C 07A8A 785000FB (00219)          LOAD(BW1,NSIZES-YSIZES-1)            NSIZE-YSIZE-1
A3D 07A8C 7B0A0001 (00220) #2       ADD(BW0,HS,TE)                       OO=NOUT(I)
A3E 07A8E 7C113DB1 (00221)          SUBL(BW1,1),JUMPP(#2)                FOR I=YSIZE...NSIZE-1
                   (00222) *  NORMALIZED MEMORY
A3F 07A90 7E4407FF (00223)          LOAD(BW0,YOMS-2(2))                  YOM(,) BA-2
A40 07A92 80500009 (00224)          LOAD(BW1,YSIZES)                     Y(,) SIZE-1
A41 07A94 828A0002 (00225) #3       ADD(BW0,WS,TE)                       OO=YOM(I)
```

```
A42 07A96 84114181 (00226)          SUBL(BW1,1),JUMPP(#3)                 FOR I=0,1...YSIZE-1
                   (00227) *
A43 07A98 86200030 (00228) DNESO    CLEAR(R0)                             OUTPUT DONE!
A44 07A9A 88000020 (00229)          NOP
A45 07A9C 8A400000 (00230)          LOAD(BW0,101)
A46 07A9E 8C500000 (00231)          LOAD(BW1,MSS)
A47 07AA0 8E020000 (00232)          SUB(BW0,MSS)
                   (00233) *
          00007A9C (00234)          G100$A= #C                            ASSIGN VALUE TO CHAIN AN
                   (00235) ;                                              CHDR
    07AA2          (00236)          END #A-1
                   (00237) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    07AA2 00000000 (00238) G100SI   DATA 1F'0,0'
          00000092 (00239)          G100SZ=#L-G100$
    07AA4          (00240)          END
```

```
AFDTSORG:   00HER (00009) (00034)
CSPUSNOS:   021FC (00010) (00037)
DMYS:       00794 (00011) (00206) (00214) (00215) (00216) (00217)
DMYSD:      00031 (00206) (00207)
DNFSA:      00036 (00131)
DNFSI:      0002D (00201)
DNFSD:      00043 (00228)
FS0:        07980 (00044)
FS1:        07982 (00045) (00153)
FS2:        07984 (00046)
FS3:        07986 (00047)
FS4:        07988 (00048)
FS5:        0798A (00049)
FS6:        0798C (00050)
FS7:        0798E (00051)
FS8:        07990 (00052)
FS9:        07992 (00053)
FSA:        07994 (00054)
G100S:      07A12 (00036) (00144) (00150) (00239)
G100SA:     07A9C (00147) (00234)
G100SI:     07AA2 (00142) (00238)
G100SD:     0002F (00151) (00204)
G100SZ:     00092 (00146) (00239)
HS:         00001 (00013) (00212) (00220)
MS:         00000 (00014)
MSS:        00000 (00015) (00209) (00231) (00232)
NOUTS:      00F16 (00016) (00208)
NSIZES:     000F5 (00017) (00219)
SAS100:     0044A (00022) (00155)
SAS101:     0044C (00023) (00154)
SAS11:      0039R (00019) (00157)
SAS38:      003CF (00020) (00156)
SAS39:      003D0 (00021)
SCLRS:      00001 (00144) (00152)
START:      07980 (00024) (00042)
SVTS:       00382 (00018) (00019) (00020) (00021) (00022) (00023)
VDINS:      0799R (00035) (00062)
VDINSSA:    00000 (00060) (00064) (00135)
VDINSSZ:    00039 (00061) (00135) (00136)
WS:         00002 (00025) (00225)
WWS:        00004 (00026)
X2S:        00800 (00027) (00158)
XSIZES:     000FF (00028) (00159)
YOMS:       00800 (00029) (00160) (00194) (00223)
```

```
PAGE    9:      FCH 252... "MPVDNM(Y,U,V)"     VAR.,DC,INTERPOLATE,FIX

  YSIZES:     00009 (00030) (00161) (00195) (00210) (00219) (00224)
  ZS:         00003 (00031)

LINES WITH ERRORS:    0  (MAP VERSION R00101.10)   E=    0
```

```
PAGE    1:           *L=$7A1D

        00007A1D (00001)        *L=$7A1D
  07A1D 7986     (00002)        DATA $7986                           CHANGE FROM ISA TO LOCAL
        00007A1F (00003)        *L=$7A1F
  07A1F 7988     (00004)        DATA $7988
        00007986 (00005)        *L=$7986
  07986 2000     (00006)        DATA $2000,$3D                       2**-14
  07987 003D
  07988 1000     (00007)        DATA $1000,$3D                       2**-15
  07989 003D
  0798A          (00008)        END
```

PAGE 2: *L=S7AID

LINES WITH ERRORS: 0 (MAP VERSION H00101.10) E= 0

```
                    * FCB 253... "MPBASP(Y,U,V,W)"     BASIS SPECTRUM CALCULATION
                    *                                  ORIGINATED:31-OCT-79
                    *                                  UPDATED:21-MAR-80
                    * DEFINE GLOBAL SYMBOLS
                           OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                           OPADD DOF, (3 .LS. 10) + (12 .LS. 5) + $1B
       000006EB            AFDTSORG=$6EB
       000021FC            CSPUSNOS=$21FC
       00000794            DMYS=$794
       00000003            #M=3
       00000001            HS=1
       00000000            MSS=0
       00000382            SVTS=$0382
       00000382            SAS00=SVTS+2*D'00'
       00000386            SAS02=SVTS+2*D'02'
       0000038E            SAS06=SVTS+2*D'06'
       0000039A            SAS12=SVTS+2*D'12'
       000003EE            SAS54=SVTS+2*D'54'
       000003FA            SAS60=SVTS+2*D'60'
       00000414            SAS73=SVTS+2*D'73'
       0000041A            SAS76=SVTS+2*D'76'
       0000041C            SAS77=SVTS+2*D'77'
       00000420            SAS79=SVTS+2*D'79'
       00000422            SAS80=SVTS+2*D'80'
       00007B90            START=$7B90
       00000002            WS=2
       00000004            WWS=4
       00000003            ZS=3
       00000007            ZZHS=7
                    *
                    * EXPAND ARRAY FUNCTION DISPATCH TABLE
       000008D6            #L=AFDTSORG+3*2*(253-128)               FCB 253
00BD6  001E7BA6            ADDR BASS(R7,1)
00BD8  001E7CB6            ADDR G204S(R7,1)
00BDA  001021FC            ADDR CSPUSNOS(,1,0)
                           EJECT
```

```
                    (00037) *
                    (00038) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00039) *
           00007B90 (00040)          #L=START
                    (00041)          EVEN
                    (00042) *  SPECIAL CONSTANTS FOR EXPONENTIATION
07B90 2000003B (00043) TABLES  DATA 2.3841859E-07                    2**(-22)
07B92 40000044 (00044)         DATA 3.2768000E+04                    2**15
07B94 0820003D (00045)         DATA 1.5497208E-05                    4*65*16**(-6)
07B96 3D86415C (00046)         DATA 0.24957274BE+34                  C5
07B98 126900D7 (00047)         DATA 0.71220889 7E+27                 C4
07B9A 725511D1 (00048)         DATA 0.263632333E+21                  C3
07B9C 1F8D5ACC (00049)         DATA 0.675972338E+14                  C2
07B9E 8800003B (00050) TABLES2 DATA -5.9604646E-08                   -16**(-6)
07BA0 58B93D46 (00051)         DATA 0.116291788E+08                  C1
07BA2 7FFFFF40 (00052)         DATA 0.9999999250                     C0
                    (00053)          EJECT
```

```
                          (00054) *
                          (00055) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                          (00056) *
                          (00057)          EVEN
     07BA4 0000           (00058)          DATA BASSSA                                      START ADDRESS
     07BA5 0084           (00059)          DATA BASSSZ                                      MODULE SIZE
                          (00060) BASS     BEGIN APU(BAS)
           00000000       (00061)          #A=0
                          (00062) *
                          (00063) *  PHASE 1 FINDS THE PRODUCT OF THE SQUARED MAGNITUDES OF TWO
                          (00064) *  COMPLEX BUFFERS, MULTIPLIES THESE PRODUCTS BY A SCALE FACTOR
                          (00065) *  G, WHICH IS SB (=ENG)/SA^2 (=EGR^2).  THE TWO COMPLEX BUFFERS
                          (00066) *  MUST BE CONTIGUOUS IN MEMORY.  W IS A DOUBLE LENGTH BUFFER
                          (00067) *  CONTAINING BOTH. (IN THE COMMENTS, W REFERS TO 1ST HALF,
                          (00068) *  V TO 2ND HALF). THE RECIPROCAL IF MAG-SQD (W) IS USED TO FORM
                          (00069) *  THE PRODUCT.  U IS THE OUTPUT BUFFER, WHICH GETS THE
                          (00070) *  PRODUCTS.  SC IS AN OUTPUT SCALAR, 1.0/SA (=EGR')
                          (00071) *  WR(0) IS TAKEN TO BE 2**12
                          (00072) *  EQ:  Y(N)=(ENG/W^2(N))*(V^2(N)/EGR^2)
                          (00073) *
                          (00074) *
A00 07BA6 16A00000 (00075) BASSSA  K(2)\NOP                                         FIND 1/SA (=1/EGR),DO IT
A01 07BA8 08D00000 (00076)         MOV(IO,A0)\NOP                                   EGR.
A02 07BAA 2E162E16 (00077)         MOV(A6),RCP(A0)                                  2;1./E' (=F0)
A03 07BAC 08FF0000 (00078)         MOV(IQA,M6)\NOP                                  EGR (=C)
A04 07BAE 08B808B8 (00079)         MOV(R,M0)                                        1./E' (=F0)
A05 07BB0 84408440 (00080)         MUL(M0,M6)                                       E/E' (=F0*C)
A06 07BB2 08B108B1 (00081)         MOV(P,A1)                                        F0*C
A07 07BB4 49C049C0 (00082)         SUB(A6,A1)                                       2-F0*C
A08 07BB6 08BC08BC (00083)         MOV(R,M4)                                        2-F0*C
A09 07BB8 84008400 (00084)         MUL(M0,M4)                                       F0(2-F0*C) (=F1)
A0A 07BBA 08AB08AB (00085)         MOV(P,M0)                                        F1
A0B 07BBC 84408440 (00086)         MUL(M0,M6)                                       F1*C
A0C 07BBE 08EF0000 (00087)         MOV(IQA,M7)\NOP                                  SB (=ENG)
A0D 07BC0 08B108B1 (00088)         MOV(P,A1)                                        F1*C
A0E 07BC2 49C049C0 (00089)         SUB(A6,A1)                                       2-F1*C
A0F 07BC4 08BC08BC (00090)         MOV(R,M4)                                        2-F1*C
A10 07BC6 84008400 (00091)         MUL(M0,M4)                                       F1(2-F1*C)
A11 07BC8 08AB08AB (00092)         MOV(P,M3)                                        1.0/SA
A12 07BCA 08AD08AD (00093)         MOV(P,M5)
A13 07BCC 85BC85A0 (00094)         MOV(OO),MUL(M3,M5)\MUL(M3,M5)                    OUTPUT EGR';1.0/EGR^25
A14 07BCE 08C80000 (00095)         MOV(IO,M0)\NOP                                   VR(0)
A15 07BD0 08FC0000 (00096)         MOV(IQA,M4)\NOP                                  VR(0)
A16 07BD2 08AB08AB (00097)         MOV(P,M3)                                        1.0/EGR^2
```

```
A17 07BD4 85F085F0 (00098)          MUL(M3,M7)                                    ENG/EGR^2 (=A)
A18 07BD6 08AF08AF (00099)          MOV(P,M7)
                   (00100) *
                   (00101) *  LOOP FOR N=0,1,2,...Z+2
A19 07BD8 841B84B0 (00102) MLP      MOV(EXO),MUL(M0,M4)\MOV(A0),MUL(M1,M5)        V(N-2)^2;VR(N)^2\WR(N-1)
                   (00103) ;                                                      ^2;WI(N-1)^2
A1A 07BDA 08C9084A (00104)          MOV(IQ,M1)\MOV(EX1,M2)                        VI(N)\V(N-2)^2*A
A1B 07BDC 08FD089C (00105)          MOV(IQA,M5)\MOV(R,00)                         VI(N)\L(N-3) OUT
A1C 07BDE 088A08CB (00106)          MOV(R,M2)\MOV(IQ,M0)                          V(N-1)^2\WR(N)
A1D 07BE0 000008FC (00107)          NOP\MOV(IQA,M4)                               WR(N)
A1E 07BE2 84B08571 (00108)          MOV(A0),MUL(M1,M5)\MOV(A1),MUL(M2,M7)         VR(N)^2;VI(N)^2\WI(N-1)^
                   (00109) ;                                                      2;V(N-2)^2*A*W'(N-2)^2
A1F 07BE4 00004100 (00110)          NOP\ADD(A0,A1)                                W(N-1)^2
A20 07BE6 000008C9 (00111)          NOP\MOV(IQ,M1)                                WI(N)
A21 07BE8 000008FD (00112)          NOP\MOV(IQA,M5)                               WI(N)
A22 07BEA 08C80000 (00113)          MOV(IQ,M0)\NOP                                VR(N+1)
A23 07BEC 08FC2F52 (00114)          MOV(IQA,M4)\MOV(A2),RCP(A2)                   VR(N+1)\W(N-1)^2;W'(N-1)
                   (00115) ;                                                      ^2
A24 07BEE 85718413 (00116)          MOV(A1),MUL(M2,M7)\MOV(A3),MUL(M0,M4)         VI(N)^2;V(N-1)^2*A\M(N-2
                   (00117) ;                                                      );WR(N)^2
A25 07BF0 41000F6F (00118)          ADD(A0,A1)\MOV(M7),LGS(A3)                    V(N)^2\W'(N-1)^2;L(N-2)
A26 07BF2 90160019 (00119)          JUMPC(MLP,FWI)
A27 07BF4 20372037 (00120)          CLEAR(WI)
A28 07BF6 00000000 (00121)          NOP
                   (00122) *
                   (00123) *  TMP2(I)=0.5*LOG2(DCT1(I)')=LOG2(SQRT(DCT1(I)')
                   (00124) *NOW DO LOG REFORMATTING
                   (00125) *  PICK UP TWO FAKE LOGS AT A TIME,MANTISSA AND EXPONENT
                   (00126) *  SEPARATELY.
A29 07BF8 16801680 (00127)          K(1)
A2A 07BFA 08FD08FD (00128)          MOV(IQA,M5)                                   KD=2 (1/2 *LOG2(16))
A2B 07BFC 16F01604 (00129)          K(8)\MOV(A4),K(4)                             STORE 1.0
A2C 07BFE 08F00000 (00130)          MOV(IQA,A0)\NOP                               M0
A2D 07C00 000008F0 (00131)          NOP\MOV(IQA,A0)                               M1
A2E 07C02 22852285 (00132)          MOV(A5),INCRE(A5)                             GET 128.\64.
A2F 07C04 08FC4595 (00133)          MOV(IQA,M4)\MOV(A5),ADD(A4,A5)                E0\GET 65.
A30 07C06 000008FC (00134)          NOP\MOV(IQA,M4)                               E1
A31 07C08 0898089B (00135)          MOV(R,EX0)\MOV(R,EX0)                         GET 128,65 ON BOTH SIDES
A32 07C0A 08880896 (00136)          MOV(R,M0)\MOV(R,A6)                           128.\65.
A33 07C0C 08560848 (00137)          MOV(EXI,A6)\MOV(EXI,M0)                       65.\128.
A34 07C0E 84008400 (00138)          ABSMUL(M0,M4)                                 128.*E0\128.*E1
A35 07C10 4F004F00 (00139)          SUB(A0,A6)                                    M0-65\M1-65
A36 07C12 08910891 (00140)          MOV(R,A1)                                     M0-65\M1-65
A37 07C14 08F0080B (00141)          MOV(IQA,A0)\MOV(ZERO,M3)                      M2
```

```
A38 07C16 08080BF0 (00142)         MOV(ZERO,M3)\MOV(IOA,A0)               M3
A39 07C18 08B20BB2 (00143)         MOV(P,A2)                              128*E0\128*E1
A3A 07C1A 08FC0815 (00144)         MOV(IOA,M4)\MOV(ZERO,A5)               E2
A3B 07C1C 08150BFC (00145)         MOV(ZERO,A5)\MOV(IOA,M4)               E3
A3C 07C1E 85808580 (00146)         MUL(M3,M4)                             GET 0 IN P
A3D 07C20 42204220 (00147)         ADD(A1,A2)                             M0-65+128E0\SAME FOR M1,
                   (00148) ;                                              E1
                   (00149) *  NOW LOOP FOR N=0,2,...YZ/2
A3E 07C22 08B40BB4 (00150) LP:     MOV(P,A4)                              STORE A*L(N-2(N-1))
A3F 07C24 B41C841C (00151)         MOV(OO),ABSMUL(M0,M4)                  OUTPUT A*L(N-2);128*(E(N
                   (00152) ;                                              +2)\E(N+3))
A40 07C26 4F094F09 (00153)         MOV(M1),SUB(A0,A6)                     L(N(N+1));M(N+2(3))-65
A41 07C28 08F00891 (00154)         MOV(IOA,A0)\MOV(R,A1)                  M(N+4)\M(N+3)-65
A42 07C2A 45914591 (00155)         MOV(A1),ADD(A4,A5)                     M(N)-65;SUM+A*L(N-2)\M(N
                   (00156) ;                                              +1)-65;SUM+A*L(N-1)
A43 07C2C 000008F0 (00157)         NOP\MOV(IOA,A0)                        M(N+5)
A44 07C2E 84B284B2 (00158)         MOV(A2),MUL(M1,M5)                     128*E(N+2(N+3));L(N(N+1)
                   (00159) ;                                              )*SD
A45 07C30 42354235 (00160)         MOV(A5),ADD(A1,A2)                     STORE SUM;M(N+2)-65+128*
                   (00161) ;                                              E(N+2)\(N+3)
A46 07C32 08FC0000 (00162)         MOV(IOA,M4)\NOP                        E(N+4)
A47 07C34 000008EC (00163)         NOP\MOV(IOA,M4)                        E(N+5)
A48 07C36 00000000 (00164)         NOP                                    BE SAFE?
A49 07C38 9016003F (00165)         JUMPC(LP,FW1)                          ENOUGH?
A4A 07C3A 20372037 (00166)         CLEAR(W1)
A4B 07C3C 00000000 (00167)         NOP
A4C 07C3E 000002A0 (00168)         NOP\R(A5)                              ODD SUM
A4D 07C40 08F40000 (00169)         MOV(IOA,A4)\NOP                        BTLTH
A4E 07C42 4DB0089B (00170)         SUB(A4,A5)\MOV(R,FX0)                  BTLTH-EVEN SUM\ODD SUM
A4F 07C44 08530000 (00171)         MOV(FX1,A3)\NOP                        ODDSUM
A50 07C46 4BB50000 (00172)         MOV(A5),SUB(A5,A3)\NOP                 BTLTH-EVEN-ODD
A51 07C48 08F80000 (00173)         MOV(IOA,M0)\NOP                        1./LTH
A52 07C4A 08BC0000 (00174)         MOV(R,M4)\NOP                          BTLTH-SUM
A53 07C4C 84008400 (00175)         MUL(M0,M4)                             (BTLTH-SUM)/LTH
A54 07C4E 08BC0000 (00176)         MOV(P,OO)\NOP                          OUTPUT IT (CTEMP)
                   (00177) *
                   (00178) *
                   (00179) *  DCT1(I)''=2,0**TMP2(I)=SQRT(DCT1(I)*PWEIT(I))
A55 07C50 08EB08EB (00180) EXPS    MOV(IOA,M3)                            M3=SA TEMPORARILY
A56 07C52 08FC08FC (00181)         MOV(IOA,M4)                            M4=2**(-22) TEMPORARILY
A57 07C54 85808580 (00182)         MUL(M3,M4)
A58 07C56 08FD08FD (00183)         MOV(IOA,M5)                            M5=2**(15)
A59 07C58 08F408F4 (00184)         MOV(IOA,A4)                            A4=4*65*16**(-6)
A5A 07C5A 08E908E9 (00185)         MOV(IOA,M1)                            M1=C5
```

```
A5B 07C5C 0BF70BF7 (00186)        MOV(IOA,A7)              A7=C4
A5C 07C5E 0BF60BF6 (00187)        MOV(IOA,A6)              A6=C3
A5D 07C60 0BF50BF5 (00188)        MOV(IOA,A5)              A5=C2
A5E 07C62 0BAC0BAC (00189)        MOV(P,M4)                M4=SA*2**(-22)
A5F 07C64 0BFA0000 (00190) #1     MOV(IOA,M2)\NOP          M2=X(0,K)
A60 07C66 00000BFA (00191)        NOP\MOV(IOA,M2)
A61 07C68 851C851C (00192)        MOV(D0),MUL(M2,M4)       D0=X(19,K-2),X(1,K)
A62 07C6A 3A6B3A6B (00193)        MOV(M3),ALIGN(A3)        A3=X(9,K-1),X(7,K-1)
A63 07C6C 1F731F73 (00194)        MOV(A3),EXP(A3)          A3=X(7,K-1),X(10,K-1)
A64 07C6E 85D185D1 (00195)        MOV(A1),MUL(M3,M6)       A1=X(1,K),X(11,K-1)
A65 07C70 3A2B3A2B (00196)        MOV(M0),ALIGN(A1)        M0=X(10,K-1),X(2,K)
A66 07C72 32523252 (00197)        MOV(A2),NORM(A2)         A2=X(2,K),X(3A,K)
A67 07C74 0BB30BB3 (00198)        MOV(P,A3)                A3=X(11,K-1)
A68 07C76 46704670 (00199)        MOV(A0),ADD(A3,A6)       A0=X(3A,K),X(12,K-1)
A69 07C78 022B022B (00200)        MOV(M3),B(A1)            M3=X(12,K-1),SET T BIT B
                   (00201) ;                               Y X(3A,K)
A6A 07C7A 85C085C0 (00202)        MUL(M3,M6)               X(13,K-1)
A6B 07C7C 0BF20BF2 (00203)        MOV(IOA,A2)              A2=-16**(-6)
A6C 07C7E 5A005A00 (00204)        ADDLT(A0,A2)             X(3B,K)
A6D 07C80 0BB30BB3 (00205)        MOV(P,A3)                A3=X(13,K-1)
A6E 07C82 45704570 (00206)        MOV(A0),ADD(A3,A5)       A0=X(3B,K),X(14,K-1)
A6F 07C84 440B440B (00207)        MOV(M3),ADD(A0,A4)       M3=X(A4,K-1),X(4,K)
A70 07C86 85C085C0 (00208)        MUL(M3,M6)               X(15,K-1)
A71 07C88 0BF20BF2 (00209)        MOV(IOA,A2)              A2=C1
A72 07C8A 0B8A0B8A (00210)        MOV(R,M2)                M2=X(4,K)
A73 07C8C 8533B533 (00211)        MOV(A3),MUL(M2,M5)       A3=X(15,K-1),X(5,K)
A74 07C8E 43404340 (00212)        ADD(A2,A3)               X(16,K-1)
A75 07C90 4B2B4B2B (00213)        MOV(M3),SUB(A1,A0)       M3=X(16,K-1),X(6,K)
A76 07C92 85D385D3 (00214)        MOV(A3),MUL(M3,M6)       A3=X(5,K),X(17,K-1)
A77 07C94 0BF20BF2 (00215)        MOV(IOA,A2)              A2=C0
A78 07C96 0B800B80 (00216)        MOV(R,NULL)
A79 07C98 0BB10BB1 (00217)        MOV(P,A1)                A1=X(17,K-1)
A7A 07C9A 0B8F0B8F (00218)        MOV(R,M6)                M6=X(6,K)
A7B 07C9C B4C084C0 (00219)        MUL(M1,M6)               X(8,K)
A7C 07C9E 42204220 (00220)        ADD(A1,A2)               X(18,K-1)
A7D 07CA0 0B8F0B8F (00221)        MOV(R,M7)                M7=X(18,K-1)
A7E 07CA2 84718471 (00222)        MOV(A1),MUL(M0,M7)       A1=X(8,K),X(19,K-1)
A7F 07CA4 47204720 (00223)        ADD(A1,A7)               X(9,K)
A80 07CA6 901C005F (00224)        JUMPC(#1,F0)             FOR I=0,1,2...LTH-1
                   (00225) *
AB1 07CA8 20320000 (00226) DNESA  CLEAR(BA)\NOP            APU DONE!
AB2 07CAA 00000000 (00227)        NOP
AB3 07CAC 10000000 (00228)        JUMP(0)
          00000084 (00229)        BASSSZ=$A-BASSSA
```

```
07CAF          (00230)          END BASSSZ
               (00231)          EJECT
```

```
                    (00232) *
                    (00233) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00234) *
                    (00235)        EVEN
    07CAF 00007D9A  (00236)        ADDR G204SI                          POINTER TO CONSTRUCTED I
                    (00237) ;                                           NSTRUCTION BLOCK
    07CB0 00007CB6  (00238)        ADDR G204S                           POINTER TO SCALAR BLOCK
    07CB2 0000      (00239)        DATA 0                               NUMBER OF SCALARS
    07CB3 00F2      (00240)        DATA G204SZ                          MODULE SIZE
    07CB4 00007DB6  (00241)        ADDR G204SA                          POINTER TO CHAIN ANCHOR
                    (00242)        EVEN                                 END ON WORD BOUNDARY
                    (00243) *
                    (00244) G204S  BEGIN APS(G204)
A00 07CB6 00204D40  (00245)        JSN(G204S0,P2)                       SET OUTPUT PC(P2)
A01 07CB8 02300030  (00246)        SET(RO)                              ENABLE OUTPUT ADDRESSING
A02 07CBA 04C20414  (00247)        LOAD(BRO,SAS73(1),TF)                EGR
A03 07CBC 06C203FA  (00248)        LOAD(BRO,SAS60(1),TF)                ENG
A04 07CBE 08403000  (00249)        LOAD(BRO,[3])                        WRO = WBASE
A05 07CC0 0A600000  (00250)        LOAD(BR2,MSS)                        VZ-1
A06 07CC2 0C700000  (00251)        LOAD(BR3,MSS)                        DUMMY, SPACING MUST BE 4
A07 07CC4 0E390011  (00252)        MOVB(BR3,BRO)                        NEED BASE FOR V START
A08 07CC6 1039002D  (00253)        ADDB(BR3,BR2)
A09 07CC8 1239002D  (00254)        ADDB(BR3,BR2)                        VR(0)-2
A0A 07CCA 14190017  (00255)        MOVB(BR1,BR3)                        VR(0)-2
A0B 07CCC 169A0002  (00256)        ADD(BR1,2,TF)                        VR(0)
A0C 07CCE 181A0002  (00257)        ADD(BR1,2)                           SKIP VI(0)
A0D 07CD0 1AF203B2  (00258)        LOAD(BR3,SAS00(1),TF)                VI0 = 0
A0E 07CD2 1CF2039A  (00259)        LOAD(BR3,SAS12(1),TF)                WRO = 2**12
A0F 07CD4 1EF203B2  (00260)        LOAD(BR3,SAS00(1),TF)                WI0 = 0
A10 07CD6 200A0002  (00261)        ADD(BRO,2)                           POINT TO WI0
A11 07CD8 22290032  (00262)        SUBL(BR2,2)                          VZ-3
                    (00263) *  LOOP FOR N = 1,2,3,...,VZ/2-1
A12 07CDA 249A0002  (00264) #1     ADD(BR1,2,TF)                        VR(N)
A13 07CDC 269A0002  (00265)        ADD(BR1,2,TF)                        VI(N)
A14 07CDE 288A0002  (00266)        ADD(BRO,2,TF)                        WR(N)
A15 07CE0 2A8A0002  (00267)        ADD(BRO,2,TF)                        WI(N)
A16 07CE2 2C2912B2  (00268)        SUBL(BR2,2),JUMPP(#1)
A17 07CE4 2E60000A  (00269)        LOAD(BR2,10)                         NEED 13 DUMMIES TOTAL
A18 07CE6 30C20794  (00270)        LOAD(BRO,DMYS(1),TF)
A19 07CE8 32890011  (00271) #2     MOVB(BRO,BRO,TF)
A1A 07CEA 34291981  (00272)        SUBL(BR2,1),JUMPP(#2)
A1B 07CEC 36890011  (00273)        MOVB(BRO,BRO,TF)                     ONE MORE MAKES 9
A1C 07CEE 38390037  (00274)        SET(W1)
A1D 07CF0 3A000020  (00275)        NOP(0)
```

```
                        (00276) *
                        (00277) *CHANGE FORMAT OF LOGS
 A1E 07CF2 3CC20386 (00278)         LOAD(BR0,SAS02(1),TF)          1/2 * LOG2(16) = 2
 A1F 07CF4 3E401036 (00279)         LOAD(BR0,(1))
 A20 07CF6 40500000 (00280)         LOAD(BR1,MSS)                  PWEIT Z-1
 A21 07CF8 42600000 (00281)         LOAD(BR2,MSS)                  DUMMY
 A22 07CFA 44681006 (00282)         LOAD(BR2,(1),S)                NEED TO ADDRESS IT AS SH
                        (00283) :                                  ORT, TOO
 A23 07CFC 46700000 (00284)         LOAD(BR3,MSS)                  DUMMY
 A24 07CFE 48700000 (00285)         LOAD(BR3,MSS)                  DUMMY
 A25 07D00 4A090032 (00286)         SUBL(BR0,2)                    UBASE -2
 A26 07D02 4C290031 (00287)         SUBL(BR2,1)                    U-1
 A27 07D04 4E190031 (00288)         SUBL(BR1,1)                    UZ-2
                        (00289) *  LOOP FOR N = 0,2,...UZ-2
 A28 07D06 510A0002 (00290) #3      ADD(BR0,2,TF)                  M(N)
 A29 07D08 530A0002 (00291)         ADD(BR0,2,TF)                  M(N+1)
 A2A 07D0A 55AA0002 (00292)         ADD(BR2,2,TO)                  E(N)
 A2B 07D0C 57AA0002 (00293)         ADD(BR2,2,TO)                  E(N+1)
 A2C 07D0E 58192BB2 (00294)         SUBL(BR1,2),JUMPP(#3)
 A2D 07D10 5A500004 (00295)         LOAD(BR1,4)                    SIX PAIRS OF DUMMIES (10
                        (00296) :                                  +2)
 A2E 07D12 5D090011 (00297) #4      MOVB(BR0,BR0,TF)               "M" AND "E"
 A2F 07D14 5F090011 (00298)         MOVB(BR0,BR0,TF)
 A30 07D16 60192FB1 (00299)         SUBL(BR1,1),JUMPP(#4)
 A31 07D18 63090011 (00300)         MOVB(BR0,BR0,TF)
 A32 07D1A 65090011 (00301)         MOVB(BR0,BR0,TF)
 A33 07D1C 66300037 (00302)         SET(WI)                        WAIT FOR APU
 A34 07D1E 68000020 (00303)         NOP(0)
 A35 07D20 6AC20420 (00304)         LOAD(BR0,SAS79(1),TF)          RTLTH
 A36 07D22 6CC203FF (00305)         LOAD(BR0,SAS54(1),TF)          1/LTH
                        (00306) *
 A37 07D24 6FD203BF (00307) EXPSI   LOAD(BR1,SAS06(1),TF)          IQ=0.25
 A38 07D26 70D27B90 (00308)         LOAD(BR1,TABLES(1),TF)         IQ=2**(-22)
 A39 07D28 72600005 (00309)         LOAD(BR2,5)                    IQ=2**15,4*65*16**(-6)
 A3A 07D2A 749A0002 (00310) #1      ADD(BR1,WS,TF)                 C5,C4,C3,C2
 A3B 07D2C 76293AB1 (00311)         SUBL(BR2,1),JUMPP(#1)          6 CONSTANTS TOTAL!
 A3C 07D2E 78402034 (00312)         LOAD(BR0,(2))                  TMP2(,) BA
 A3D 07D30 7A700000 (00313)         LOAD(BR3,MSS)                  TMP2(,) SIZE-1
 A3E 07D32 7C020000 (00314)         SUB(BR0,MSS)                   TMP2(,) BA-2
 A3F 07D34 7E8A0002 (00315) #2      ADD(BR0,WS,TF)                 IQ=TMP2(I)
 A40 07D36 808A0002 (00316)         ADD(BR?,WS,TF)                 IQ=TMP2(I+1)
 A41 07D38 829A0002 (00317)         ADD(BR1,WS,TF)                 IQ=-16**(-6)
 A42 07D3A 849A0002 (00318)         ADD(BR1,WS,TF)                 IQ=C1
 A43 07D3C 869A0002 (00319)         ADD(BR1,WS,TF)                 IQ=C0
```

```
A44 07D3F BB190036 (00320)          SUBL(BR1,6)                       -16**(-6) BA
A45 07D40 BA393EB2 (00321)          SUBL(BR3,2),JUMPP(#2)             FOR I=0,1,2...LTH-1
A46 07D42 BCC20794 (00322)          LOAD(BR0,DMYS(1),TF)              IQ=?
A47 07D44 BEC20794 (00323)          LOAD(BR0,DMYS(1),TF)              IQ=?
A48 07D46 909A0002 (00324)          ADD(BR1,WS,TF)                    IQ=-16**(-6)
A49 07D48 929A0002 (00325)          ADD(BR1,WS,TF)                    IQ=C1
A4A 07D4A 949A0002 (00326)          ADD(BR1,WS,TF)                    IQ=C0
                   (00327) *
A4B 07D4C 96200031 (00328) DNESI    CLEAR(RI)                         INPUT DONE!
A4C 07D4E 98000020 (00329)          NOP
                   (00330) *
A4D 07D50 9A300032 (00331) G204SO   SET(RA)                           ENABLE APU
A4E 07D52 9CE20414 (00332)          LOAD(BW3,SAS73(1),TF)             EGR' = 1/EGR
A4F 07D54 9EC20794 (00333)          LOAD(BW0,DMYS(1),TF)              NEED 3 DUMMIES OUT
A50 07D56 A0810010 (00334)          MOVB(BW0,BW0,TF)
A51 07D58 A2810010 (00335) #3       MOVB(BW0,BW0,TF)
A52 07D5A A4C0102C (00336)          LOAD(BW0,(1),TF)                  U(0)
A53 07D5C A6500000 (00337)          LOAD(BW1,MSS)                     UZ-1
A54 07D5E A8600000 (00338)          LOAD(BW2,MSS)                     DUMMY, SPACING MUST BE 2
A55 07D60 AA8A0002 (00339)          ADD(BW0,2,TF)                     U(1)
A56 07D62 AC110032 (00340)          SUBL(BW1,2)
                   (00341) *  LOOP FOR N = 2,4,...UZ-2
A57 07D64 AE8A0002 (00342) #5       ADD(BW0,2,TF)                     U(N)
A58 07D66 B08A0002 (00343)          ADD(BW0,2,TF)                     U(N+1)
A59 07D68 B21157B2 (00344)          SUBL(BW1,2),JUMPP(#5)
                   (00345) *
                   (00346) *  OUTPUT FLT PT LOGS
A5A 07D6A B4C20794 (00347)          LOAD(BW0,DMYS(1),TF)
A5B 07D6C B6810010 (00348)          MOVB(BW0,BW0,TF)                  2 DUMMIES
A5C 07D6E B8C02014 (00349)          LOAD(BW0,(2),TF)                  TMP2(0)
A5D 07D70 BA500000 (00350)          LOAD(BW1,MSS)                     YZ-1
A5E 07D72 BC600000 (00351)          LOAD(BW2,MSS)                     DUMMY
A5F 07D74 BE8A0002 (00352)          ADD(BW0,2,TF)                     Y(1)
A60 07D76 C0110032 (00353)          SUBL(BW1,2)
                   (00354) *  LOOP FOR N=2,3,...YZ-2
A61 07D78 C28A0002 (00355) #6       ADD(BW0,2,TF)                     Y(N)
A62 07D7A C48A0002 (00356)          ADD(BW0,2,TF)                     Y(N+1)
A63 07D7C C61161B2 (00357)          SUBL(BW1,2),JUMPP(#6)
A64 07D7E C8C20422 (00358)          LOAD(BW0,SASB0(1),TF)             NORMALIZED SUM OF LOGS
                   (00359) *
A65 07D80 CAC20794 (00360) EXPSO    LOAD(BW0,DMYS(1),TF)              OQ=?
A66 07D82 CCC20794 (00361)          LOAD(BW0,DMYS(1),TF)              OQ=?
A67 07D84 CEC20794 (00362)          LOAD(BW0,DMYS(1),TF)              OQ=?
A68 07D86 D0C0001B (00363)          LOAD(BW0,(0),TF)                  DCTM1(.) BA,OQ=DCTM1(0)
```

```
A69 07D88 E2700000 (00364)           LOAD(BW3,MSS)                       DCTM1(.) SIZE-1
A6A 07D8A D4026000 (00365)           SUB(BW0,MSS)                        DCTM1(.) BA-2
A6B 07D8C E6BA0002 (00366) #1        ADD(BW0,WS,TF)                      OO=DCTM1(I)
A6C 07D8E D8BA0002 (00367)           ADD(BW0,WS,TF)                      OO=DCTM1(I+1)
A6D 07D90 DABA0002 (00368)           ADD(BW0,WS,TF)                      OO=DCTM1(I+2)
A6E 07D92 ECBA0002 (00369)           ADD(BW0,WS,TF)                      OO=DCTM1(I+3)
A6F 07D94 FE316BB4 (00370)           SUBL(BW3,4),JUMPP(#1)               FOR I=1,2...LTH-1
                   (00371) *
A70 07D96 E0200030 (00372) DNESU     CLEAR(PO)                           OUTPUT DONE!
A71 07D98 E2000020 (00373)           NOP
                   (00374) *
          00007D86 (00375)           G204SA= #C                          ASSIGN VALUE TO CHAIN AN
                   (00376) ;                                             CHOR
    07D9A          (00377)           END #A-1
                   (00378) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    07D9A 00000000 (00379) G204SI    DATA 7F'0,0'
     ...
          000000F2 (00380)           G204SZ=#L-G204S
    07DAB          (00381)           END
```

```
AEDTSORG:  008E8 (00007) (00032)
BASS:      07BA6 (00033) (00060)
BASSSA:    00000 (00058) (00075) (00229)
BASSS7:    00084 (00059) (00229) (00230)
CSPUSNOS:  021FC (00008) (00035)
DMYS:      00794 (00009) (00270) (00322) (00323) (00333) (00347) (00360) (00361) (00362)
DNFSA:     00081 (00226)
DNFSI:     00048 (00328)
DNFSO:     00070 (00372)
EXPS:      00055 (00180)
EXPSI:     00037 (00307)
EXPSO:     00065 (00360)
G204S:     07C86 (00034) (00238) (00244) (00380)
G204SA:    07D86 (00241) (00375)
G204SI:    07D9A (00236) (00379)
G204SO:    0004D (00245) (00331)
G204SZ:    000F2 (00240) (00380)
HS:        00001 (00011)
LP:        0003E (00150) (00165)
MSS:       00000 (00012) (00250) (00251) (00280) (00281) (00284) (00285) (00313) (00314) (00337)
                 (00338) (00350) (00351) (00364) (00365)
MLP:       00019 (00102) (00119)
SAS00:     00382 (00014) (00258) (00260)
SAS02:     00386 (00015) (00278)
SAS06:     0038E (00016) (00307)
SAS12:     0039A (00017) (00259)
SAS54:     003FF (00018) (00305)
SAS60:     003EA (00019) (00248)
SAS73:     00414 (00020) (00247) (00332)
SAS76:     0041A (00021)
SAS77:     0041C (00022)
SAS79:     00420 (00023) (00304)
SAS80:     00422 (00024) (00358)
START:     07B90 (00025) (00040)
SVTS:      00382 (00013) (00014) (00015) (00016) (00017) (00018) (00019) (00020) (00021) (00022)
                 (00023) (00024)
TABLES:    07B90 (00043) (00308)
TABLES2:   07B9E (00050)
WS:        00002 (00026) (00310) (00315) (00316) (00317) (00318) (00319) (00324) (00325) (00326)
                 (00366) (00367) (00368) (00369)
WWS:       00004 (00027)
ZS:        00003 (00028)
ZZHS:      00007 (00029)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)    E=    0
```

```
                   '  FCB 254... "MPSCAN(Y,U,V)"      SCAN DCT BASIS SPECTRUM
                   *                                  ORIGINATED:02-NOV-79
                   *                                  UPDATED:25-MAY-80
                   *  DEFINE GLOBAL SYMBOLS
                           OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                           OPADD DUE, (3 .LS. 10) + (12 .LS. 5) + $18
      000000BE8            AFDTSORG=$BE8
      0001FFC0             APSSMEM=$1FFC0
      000021FC             CSPUSNOS=$21FC
      00000794             DMYS=$794
      00000003             #M=3
      00000001             HS=1
      00000000             MSS=0
      00000382             SVTS=$0382
      0000039A             SAS12=SVTS+2*D'12'
      0000039C             SAS13=SVTS+2*D'13'
      00000420             SAS79=SVTS+2*D'79'
      00000422             SAS80=SVTS+2*D'80'
      00000426             SAS82=SVTS+2*D'82'
      00000428             SAS83=SVTS+2*D'83'
      0000042A             SAS84=SVTS+2*D'84'
      00007F10             START=$7F10
      000000FF             SIZES=D'256'-D'1'
      00000800             TMP1S=D'2048'
      00000A00             TMP2S=D'2560'
      00000002             WS=2
      00000004             WWS=4
      00000003             ZS=3
      00000007             ZZHS=7
                   *
                   *  EXPAND ARRAY FUNCTION DISPATCH TABLE
      00000BDC             #L=AFDTSORG+3*2*(254-128)                          FCB 254
00BDC 001F7F12             ADDR SCAS(R7,1)
00BDE 001F7F24             ADDR G205S(R7,1)
00BE0 001021FC             ADDR CSPUSNOS(,1,0)
                           EJECT
```

```
                (00037) *
                (00038) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                (00039) *
                (00040)           EVEN
       00007E10 (00041)           #L=START
                (00042)           EJECT
```

```
                      *
                      *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      *
                                EVEN
    07F10 0000                  DATA SCASSA                    START ADDRESS
    07F11 0085                  DATA SCASSZ                    MODULE SIZE
                      SCAS      BEGIN APU(SCA)
          00000000              #A=0
                      *  TMP2(I)=DCTM2(I)+CTEMP  I=0,1,2...ITOT
                      *  TMP2(I)=0 I>ITOT
A00 07F12 08F708F7    SCASSA    MOV(IOA,A7)                    A7=CTEMP
A01 07F14 08130000              MOV(ZERO,A3)\NOP               A3=ITOT
A02 07F16 16801680              R(1,0)                         R=1,0
A03 07F18 08150815              MOV(ZERO,A5)                   A5=0,0
A04 07F1A 08910000              MOV(R,A1)\NOP                  A1=1,0
A05 07F1C 20282028              CLEAR(AF0)                     RESET "CANCEL ZERO FILL"
A06 07F1E 08F20000    VADDS     MOV(IOA,A2)\NOP                A2=UE
A07 07F20 000008F2              NOP\MOV(IOA,A2)                A2=UO
A08 07F22 47404740              ADD(A2,A7)                     UE+CTEMP;UO+CTEMP
A09 07F24 08950895              MOV(R,A5)                      SET T1 & T2
A0A 07F26 911F001C              JUMPS(VDNS1,T1)                T1=1 FOR UE+CTEMP<0
A0B 07F28 02400000              R(A2)\NOP                      R=UE
A0C 07F2A 433C0000              MOV(OO),ADD(A1,A3)\NOP         UE=UE THEN ITOT+1
A0D 07F2C 911F0014              JUMPS(VDNS2,T2)                T2=1 FOR UO+CTEMP<0
A0E 07F2E 43330000              MOV(A3),ADD(A1,A3)\NOP         NO,THEN ITOT+1 AGAIN
A0F 07F30 08930000              MOV(R,A3)\NOP                  A3=ITOT+1
A10 07F32 00000240              NOP\R(A2)                      R=UO
A11 07F34 0000089C              NOP\MOV(R,OO)                  UO=UO
A12 07F36 90160006              JUMPC(VADDS,FW1)               LOOP FOR I=1,2...LTH-1
A13 07F38 20482048              SET(AF0)                       ADJUST FOR I=LTH-1
A14 07F3A 089302A0    VDNS2     MOV(R,A3)\R(A5)                A3=ITOT+1;R=UO
A15 07F3C 0000089B              NOP\MOV(R,FXO)                 MOVE OVER
A16 07F3E 08550000              MOV(FXI,A5)\NOP                A5=UO(?)+CTEMP
A17 07F40 91080021              JUMPS(VADDSDN,AF0)             SET "CANCEL ZERO FILL" F
                      ;                                        LAG
A18 07F42 081C0000    #2        MOV(ZERO,OO)\NOP               UO(?)=0
A19 07F44 91160021              JUMPS(VADDSDN,FW1)             LAST SAMPLE?
A1A 07F46 08F00000              MOV(IOA,NULL)\NOP              NO,THROW AWAY NEXT ONE
A1B 07F48 1000001B              JUMP(#2)                       THEN CLEAR SAME LOCATION
A1C 07F4A 081C081C    VDNS1     MOV(ZERO,OO)                   U(?)=0,0
A1D 07F4C 91160021              JUMPS(VADDSDN,FW1)             LAST EVEN,ODD SAMPLES?
A1E 07F4E 08F00000              MOV(IOA,NULL)\NOP              NO, THEN THROW AWAY...
A1F 07F50 000008F0              NOP\MOV(IOA,NULL)              A PAIR @ A TIME
A20 07F52 1000001C              JUMP(VDNS1)                    THEN CLEAR SAME LOCATION
```

```
                    (00087) ;                                                S
A21 07F54 02A00000  (00088) VADDSUN R(A5)\NOP                                A5=UE(2)+CTEMP
A22 07F56 089C0000  (00089)         MOV(R,O0)\NOP                            O0=UE(2)+CTEMP
A23 07F58 02600000  (00090)         P(A3)\NOP                                A3=ITOT
A24 07F5A 089C0000  (00091)         MOV(R,O0)\NOP                            O0=ITOT
A25 07F5C 20372037  (00092)         CLEAR(WI)                                RELEASE APS INPUT
                    (00093) *
                    (00094) *  ITOT'=1.0/ITOT
A26 07F5E 08860000  (00095) SINVS   MOV(R,M6)\NOP                            M6=SC=ITOT
A27 07F60 08900000  (00096)         MOV(R,A0)\NOP                            A0=ITOT
A28 07F62 16A016A0  (00097)         K(2)                                     P=2.0;R=2.0
A29 07F64 08960000  (00098)         MOV(R,A6)\NOP                            A6=2
A2A 07F66 2F000000  (00099)         RCP(A0)\NOP                              R1=1/C+DEL(=F0)
A2B 07F68 08880000  (00100)         MOV(R,M0)\NOP                            M0=F0
A2C 07F6A 84400000  (00101)         MUL(M0,M6)\NOP                           P1=F0*C
A2D 07F6C 08810000  (00102)         MOV(P,A1)\NOP                            A1=F0*C
A2E 07F6E 49C00000  (00103)         SUB(A6,A1)\NOP                           R2=2-F0*C
A2F 07F70 088C0000  (00104)         MOV(R,M4)\NOP                            M4=R2
A30 07F72 84000000  (00105)         MUL(M0,M4)\NOP                           P2=F0*R2(=F1)
A31 07F74 08A80000  (00106)         MOV(P,M0)\NOP                            M0=F1
A32 07F76 84400000  (00107)         MUL(M0,M6)\NOP                           P3=F1*C
A33 07F78 08810000  (00108)         MOV(P,A1)\NOP                            A1=P3
A34 07F7A 49C00000  (00109)         SUB(A6,A1)\NOP                           R3=2-F1*C
A35 07F7C 088C0000  (00110)         MOV(R,M4)\NOP                            M4=R3
A36 07F7E 84000000  (00111)         MUL(M0,M4)\NOP                           P4=F1*(2-F1*C)(=F2)
A37 07F80 08AF0000  (00112)         MOV(P,M7)\NOP                            M7=1.0/ITOT
                    (00113) *
                    (00114) *  CTEMP=(BTLTH-SLOG)/ITOT
A38 07F82 08F60000  (00115) SSUMS   MOV(IOA,A6)\NOP                          A6=BTLTH
A39 07F84 08100810  (00116)         MOV(ZERO,A0)                             A0=0.0
A3A 07F86 02000200  (00117)         R(A0)                                    R=0.0
A3B 07F88 08F04717  (00118) #1      MOV(IOA,A0)\MOV(A7),ADD(A0,A7)           A0=UE;A7=SUMO(-2),A7+UO(
                    (00119) ;                                                -1)
A3C 07F8A 00000000  (00120)         NOP                                      ?
A3D 07F8C 91160048  (00121)         JUMPS(SS2,FWI)
A3E 07F8E 471708F1  (00122)         MOV(A7),ADD(A0,A7)\MOV(IOA,A1)           A7=SUME(-1),A7+UE;A1=UO
A3F 07F90 00000000  (00123)         NOP                                      ?
A40 07F92 91160049  (00124)         JUMPS(SS3,FWI)
A41 07F94 08F14737  (00125)         MOV(IOA,A1)\MOV(A7),ADD(A1,A7)           A1=UE(+1);A7=SUMO(-1),A7
                    (00126) ;                                                +SUMO
A42 07F96 00000000  (00127)         NOP                                      ?
A43 07F98 9116004A  (00128)         JUMPS(SS4,FWI)
A44 07F9A 473708F0  (00129)         MOV(A7),ADD(A1,A7)\MOV(IOA,A0)           A7=SUME,A7+UF(+1);A0=UO(
                    (00130) ;                                                +1)
```

```
A45 07F9C 00000000 (00131)          NOP                                              ?
A46 07F9E 9016003B (00132)          JUMPC(B1,FWI)
A47 07FA0 08104717 (00133)          MOV(ZERO,A0)\MOV(A7),ADD(A0,A7)
A48 07FA2 47170811 (00134) SS2      MOV(A7),ADD(A0,A7)\MOV(ZERO,A1)
A49 07FA4 08114737 (00135) SS3      MOV(ZERO,A1)\MOV(A7),ADD(A1,A7)
A4A 07FA6 47370000 (00136) SS4      MOV(A7),ADD(A1,A7)\NOP
A4B 07FA8 08900898 (00137)          MOV(R,A0)\MOV(R,FX0)                             A0=SUME;EXO=SUMO
A4C 07FAA 08510000 (00138)          MOV(FXI,A1)\NOP                                  A1=SUMO
A4D 07FAC 41004100 (00139)          ADD(A0,A1)                                       (SUME+SUMO)
A4E 07FAE 48D30000 (00140)          MOV(A3),SUB(A6,A3)\NOP                           A3=(SUME+SUMO),BTLTH-A3
A4F 07FB0 08880888 (00141)          MOV(R,M0)                                        M0=BTLTH-(SUME+SUMO)
A50 07FB2 84600000 (00142)          MUL(M0,M7)\NOP                                   ITOT'*(BTLTH-(SUME+SUMO)
                   (00143) ;                                                         )
A51 07FB4 088C0000 (00144)          MOV(P,D0)\NOP                                    SA=CTEMP=(BTLTH-SLOG)/IT
                   (00145) ;                                                         OT
A52 07FB6 20372037 (00146)          CLEAR(FI)                                        RELEASE APS INPUT
                   (00147) *
                   (00148) *  TMP2(I)=TMP2(I)+CTEMP+0.5          I=0,1,2...LTH-1
A53 07FB8 16601660 (00149)          K(0.5)                                           R=.5
A54 07FBA 08880000 (00150)          MOV(P,EXO)\NOP                                   EXO=CTEMP
A55 07FBC 08800850 (00151)          MOV(P,A0)\MOV(FXI,A0)                            A0=CTEMP
A56 07FBE 41110000 (00152)          MOV(A1),ADD(A0,A1)\NOP                           A1=0.5,CTEMP+0.5
A57 07FC0 47570000 (00153)          MOV(A7),ADD(A2,A7)\NOP                           A7=CTEMP+0.5, ADD FIRST
                   (00154) ;                                                         AND CONST.
A58 07FC2 08140000 (00155)          MOV(ZERO,A4)\NOP                                 MIN
A59 07FC4 08F308F6 (00156)          MOV(IQA,A3)\MOV(IQA,A6)                          MAX\OLD SUM=MAX
A5A 07FC6 4F600000 (00157)          SUB(A3,A7)\NOP                                   SUB CONST, SINCE WE ADD
                   (00158) ;                                                         IT LATER
A5B 07FC8 08920000 (00159)          MOV(R,A2)\NOP                                    THIS IS NOW OLD VALUE
A5C 07FCA 08D30000 (00160) VADDS2   MOV(IQ,A3)\NOP                                   A3=NEW
A5D 07FCC 4A600000 (00161)          SUB(A3,A2)\NOP                                   NEW-OLD
A5E 07FCE 08F80000 (00162)          MOV(IQA,FXO)\NOP                                 SAVE NEW
A5F 07FD0 476002C0 (00163)          ADD(A3,A7)\R(A6)                                 NEW SUM\OLD SUM
A60 07FD2 901F006E (00164)          JUMPC(NOK,T1)                                    JUMPIF NEW>OLD
A61 07FD4 64850000 (00165) OK       MOV(A5),MAX(A5,A4)\NOP                           NEWSUM,ZERO
A62 07FD6 00000858 (00166)          NOP\MOV(FXI,FXO)                                 NEW INPUT
A63 07FD8 08520000 (00167)          MOV(FXI,A2)\NOP                                  NEW INPUT
A64 07FDA 08980000 (00168)          MOV(R,EXO)\NOP                                   OLD SUM <-- NEW SUM
A65 07FDC 911F006A (00169)          JUMPS(ZRO,T1)                                    JUMPIF 0.GE.NEWSUM
A66 07FDE 089C0856 (00170)          MOV(R,D0)\MOV(FXI,A6)                            OUTPUT NEW SUM,OLDSUM <-
                   (00171) ;                                                         - NEWSUM
A67 07FE0 9016005C (00172)          JUMPC(VADDS2,FWI)                                LOOP
A68 07FE2 10000070 (00173)          JUMP(VADDS2F)
A69 07FE4 08F00000 (00174) ZROLP    MOV(IQA,A0)\NOP                                  DUMMY TO USE UP INPUT
```

```
A6A 07FF6 081C0000 (00175) ZRO     MOV(ZERO,OO)\NOP
A6B 07FF8 90160069 (00176)         JUMPC(ZROLP,FWI)              KEEP GOING TIL INPUT DON
                   (00177) ;                                     E
A6C 07FFA 20372037 (00178)         CLEAR(WI)                     LET APS GO
A6D 07FFC 10000071 (00179)         JUMP(VINPTS)                  GO ON TO NEXT RTN
A6E 07FFE 0000089C (00180) NOK     NOP\MOV(R,OO)                 OLD SUM OUT
A6F 07FF0 9016005C (00181)         JUMPC(VADDS2,FWI)             LOOP
A70 07FF2 20372037 (00182) VADDS2F CLEAR(WI)                     RELEASE APS INPUT
                   (00183) *
                   (00184) *  TMP1(I)=INTEGER PART(TMP2(I))   I=0,1,2...LTH-1
A71 07FF4 08FF08FF (00185) VINPTS  MOV(IQA,M7)                   M7=16**(6)
A72 07FF6 08FF08FF (00186)         MOV(IQA,M6)                   M6=16**(-6)
A73 07FF8 08170817 (00187)         MOV(ZERO,A7)                  A7=SA=0,0
A74 07FFA 08E80000 (00188)         MOV(IQA,M0)\NOP               M0=X(0,I)
A75 07FFC 08E60000 (00189)         MOV(IQA,A6)\NOP               A6=X(0,I-4)
A76 07FFE 000008E8 (00190)         NOP\MOV(IQA,M0)               M0=X(0,I+1)
A77 07F00 000008E6 (00191)         NOP\MOV(IQA,A6)               A6=X(0,I-3)
A78 07F02 84518451 (00192) #1      MOV(A1),MUL(M0,M6)            A1=X(4,I-2),P=X(1,I)
A79 07F04 02C902C9 (00193)         MOV(M1),R(A6)                 M1=X(3,I-1),T1=SIGN(X(0,
                   (00194) ;                                     I-1))
A7A 07F06 5F205F20 (00195)         MOV(NULL),ADDLT(A1,A7)        R=X(5,I-2)
A7B 07F08 84F084F0 (00196)         MOV(A0),MUL(M1,M7)            A0=X(1,I),P=X(4,I-1)
A7C 07F0A 3A1C3A1C (00197)         MOV(OO),ALIGN(A0)             OO=X(5,I-2),R=X(2,I)
A7D 07F0C 08E80000 (00198)         MOV(IQA,M0)\NOP               M0=X(0,I+2)
A7E 07F0E 08E60000 (00199)         MOV(IQA,A6)\NOP               A6=X(0,I-2)
A7F 07F10 000008E8 (00200)         NOP\MOV(IQA,M0)               X=(0,I+3)
A80 07F12 000008E6 (00201)         NOP\MOV(IQA,A6)               X=(0,I-1)
A81 07F14 32103210 (00202)         MOV(A0),NORM(A0)              A0=X(2,I),R=X(3,I)
A82 07F16 901C0078 (00203)         JUMPC(#1,F0)                  FOR I=0,1,2...LTH-1
                   (00204) *
A83 07F18 20320000 (00205) DNESA   CLEAR(RA)\NOP                 APU DONE!
A84 07F1A 00000000 (00206)         NOP
          00000085 (00207)         SCASSZ=#A-SCASSA
    07F1C          (00208)         END SCASSZ
                   (00209)         EJECT
```

```
                    (00210) *
                    (00211) *>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                    (00212) *
                    (00213)          EVEN
    07F1C 00007FA2  (00214)          ADDR G205$I                         POINTER TO CONSTRUCTED I
                    (00215) ;                                            NSTRUCTION BLOCK
    07F1E 00007F28  (00216)          ADDR G205$+2*SCLR$                  POINTER TO SCALAR BLOCK
    07F20 0000      (00217)          DATA 0                              NUMBER OF SCALARS
    07F21 0080      (00218)          DATA G205$Z                         MODULE SIZE
    07F22 00007F2A  (00219)          ADDR G205$A                         POINTER TO CHAIN ANCHOR
                    (00220)          EVEN                                END ON WORD BOUNDARY
                    (00221) *
                    (00222) G205$    BEGIN APS(G205)
A00 07F24 00102040  (00223)          JSN(G205$I,P1)                      SET INPUT PC(P1)
A01 07F26 02300030  (00224)          SET(RO)                             ENABLE OUTPUT ADDRESSING
A02 07F28 04F20422  (00225) SCLR$    LOAD(BR2,SAS80(1),TF)               IQ=CTEMP
A03 07F2A 06C00000  (00226)          LOAD(BR0,I01,TF)                    IQ=DCTM2(0)
A04 07F2C 08500000  (00227)          LOAD(BR1,MSS)                       DCTM2(.) SIZE-1
A05 07F2E 0ABF0002  (00228)          ADD(BR0,WS,TF,C)                    IQ=DCTM2(1),CHANGE PC
                    (00229) *
A06 07F30 0CF20420  (00230) ISUM$S   LOAD(BR2,SAS79(1),TF)               IQ=BTLTH
A07 07F32 0FC60A00  (00231)          LOAD(BR0,TMP2S(3),TF)               IQ=TMP2(0)
A08 07F34 105000FF  (00232)          LOAD(BR1,SIZES)                     TMP2(.) SIZE-1
A09 07F36 12BF0002  (00233)          ADD(BR0,WS,TF,C)                    IQ=TMP2(1),CHANGE PC
                    (00234) *
A0A 07F38 14F2042A  (00235) IADDS    LOAD(BR2,SAS84(1),TF)               IQ=5.0
A0B 07F3A 16C60A00  (00236)          LOAD(BR0,TMP2S(3),TF)               IQ=TMP2(0)
A0C 07F3C 185000FF  (00237)          LOAD(BR1,SIZES)                     TMP2(.) SIZE-1
A0D 07F3E 1ABF0002  (00238)          ADD(BR0,WS,TF,C)                    IQ=TMP2(1),CHANGE PC
                    (00239) *
A0E 07F40 1CF2039A  (00240) IINPTS   LOAD(BR2,SAS12(1),TF)               IQ=16**6
A0F 07F42 1EF2039C  (00241)          LOAD(BR2,SAS13(1),TF)               IQ=16**(-6)
A10 07F44 204609FF  (00242)          LOAD(BR0,TMP2S-2(3))                TMP2(.) BA-2
A11 07F46 226609FF  (00243)          LOAD(BR2,TMP2S-2(3))                TMP2(.) BA-2
A12 07F48 24700003  (00244)          LOAD(BR3,3)                         SET UP LOOP W/ ...
A13 07F4A 26BA0002  (00245) #1       ADD(BR0,WS,TF)                      IQ=TMP2(I)
A14 07F4C 28AA0000  (00246)          ADD(BR2,MSS,TF)                     IQ=TMP2(-1)
A15 07F4E 2A3913B1  (00247)          SUBL(BR3,1),JUMPP(#1)               FOR I=0,1,2,3
A16 07F50 2C5000FB  (00248)          LOAD(BR1,SIZES-4)                   TMP2(.) SIZE-1-4
A17 07F52 2EBA0002  (00249) INTPSI   ADD(BR0,WS,TF)                      IQ=TMP2(I+4)
A18 07F54 30AA0002  (00250)          ADD(BR2,WS,TF)                      IQ=TMP2(I)
A19 07F56 321917B1  (00251)          SUBL(BR1,1),JUMPP(INTPSI)           FOR I=0,1,2...LTH-6
A1A 07F58 34700005  (00252)          LOAD(BR3,5)                         SET UP LOOP W/ ...
A1B 07F5A 36BA6000  (00253) #2       ADD(BR0,MSS,TF)                     IQ=?
```

```
A1C 07F5C 38AA0002 (00254)          ADD(BR2,WS,TF)                  IQ=TMP2(LTH-4)
A1D 07F5E 3A391BB1 (00255)          SUBL(BR3,1),JUMPP(#2)           FOR I=0,1,...5
                   (00256) *
A1E 07F60 3C200031 (00257) DNESI    CLEAR(RI)                       INPUT DONE!
A1F 07F62 3E000020 (00258)          NOP
A20 07F64 40202740 (00259) G205S1   JSN(G205S0,P2)                  SET OUTPUT PC(PC2)
A21 07F66 42190032 (00260) TOPSP1   SUBL(BR1,2)                     SIZE-1-2
A22 07F68 44BA0002 (00261) #1       ADD(BR0,WS,TF)                  IQ=DCTM2(I) OR TMP2(I)
A23 07F6A 461922B1 (00262)          SUBL(BR1,1),JUMPP(#1)           FOR I=2,3,...LTH-2
A24 07F6C 48300037 (00263)          SET(WI)                         STALL BEFORE JUMP! CRITI
                   (00264) ;                                        CAL!!!
A25 07F6E 4A2F0000 (00265)          ADD(BR2,MS$,NA,C)               PC->PC0
A26 07F70 4C002160 (00266)          JUMP(TOPSP1)                    LET "CHANGE" SETTLE
                   (00267) *
A27 07F72 4E303940 (00268) G205S0   JSN(G205S3,P3)                  SET OUTPUT PC(PC3)
A28 07F74 50C60A00 (00269)          LOAD(BW0,TMP2S(3),TF)           OQ=TMP2(0)
A29 07F76 525000FD (00270)          LOAD(BW1,SIZES-2)               TMP2(.) SIZE-3
A2A 07F78 54BE0002 (00271)          ADD(BW0,WS,TF,C)                OQ=TMP2(1),CHANGE PC
A2B 07F7A 56F20426 (00272)          LOAD(BW2,SASB2(1),TF)           OQ=U(7)+CTEMP
A2C 07F7C 58F20428 (00273)          LOAD(BW2,SASB3(1),TF)           OQ=ITOT
                   (00274) *
A2D 07F7E 5AF20422 (00275) OSUMS    LOAD(BW2,SASB0(1),TF)           OQ=CTEMP
                   (00276) *
A2E 07F80 5CC60A00 (00277) OADDS    LOAD(BW0,TMP2S(3),TF)           OQ=TMP2(0)
A2F 07F82 5E5000FD (00278)          LOAD(BW1,SIZES-2)               TMP2(.) SIZE-3
A30 07F84 60BE0002 (00279)          ADD(BW0,WS,TF,C)                OQ=TMP2(1),CHANGE PC
                   (00280) *
A31 07F86 62700003 (00281) DINPTS   LOAD(BW3,3)                     3+1=4 DUMMIES
A32 07F88 64F20794 (00282) #2       LOAD(BW2,DMYS(1),TF)            OQ=?
A33 07F8A 663132B1 (00283)          SUBL(BW3,1),JUMPP(#2)           FOR I=0,1,..3
A34 07F8C 68C60B00 (00284)          LOAD(BW0,TMP1S(3),TF)           OQ=TMP1(0)
A35 07F8E 6A5000FD (00285)          LOAD(BW1,SIZES-2)               TMP1(.) SIZE-3
A36 07F90 6CBE0002 (00286)          ADD(BW0,WS,TF,C)                OQ=TMP1(1),CHANGE PC
                   (00287) *
A37 07F92 6E200030 (00288) DNESO    CLEAR(RO)                       OUTPUT DONE!
A38 07F94 70000020 (00289)          NOP
A39 07F96 72300032 (00290) G205S3   SET(PA)                         ENABLE APU
A3A 07F98 74110031 (00291) TOPSP3   SUBL(BW1,1)                     SIZE-1-1
A3B 07F9A 76BA0002 (00292) #1       ADD(BW0,WS,TF)                  OQ=TMP2(I) OR TMP1(I)
A3C 07F9C 781138B1 (00293)          SUBL(BW1,1),JUMPP(#1)           FOR I=2,3,..LTH-2
A3D 07F9E 7ABE0002 (00294)          ADD(BW0,WS,TF,C)                OQ=...(LTH-1),CHANGE PC
A3E 07FA0 7C003A60 (00295)          JUMP(TOPSP3)                    PC->PC2
                   (00296) *
          00007F2A (00297)          G205SA= #C                      ASSIGN VALUE TO CHAIN AN
```

```
                (00298) ;                                              CHOR
07FA2           (00299)           END #A-1
                (00300) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
07FA2 00000000  (00301) G205$1  DATA 1F'0,0'
      00000080  (00302)         G205$Z=$L-G205$
07FA4           (00303)         END
```

```
AFDTSORG:   008EB (00007) (00032)
APSSMEM:    1FFC0 (00008)
CSPUSNDS:   021FC (00009) (00035)
DMYS:       00794 (00010) (00282)
DNFSA:      00083 (00205)
DNFS1:      0001F (00257)
DNFSO:      00037 (00288)
G205S:      07F24 (00034) (00216) (00222) (00302)
G205S1:     00020 (00223) (00259)
G205S3:     00039 (00268) (00290)
G205SA:     07F2A (00219) (00297)
G205SI:     07FA2 (00214) (00301)
G205SO:     00027 (00259) (00268)
G205SZ:     00080 (00218) (00302)
HS:         00001 (00012)
IADDS:      0000A (00235)
IINPTS:     0000F (00240)
INTPSI:     00017 (00249) (00251)
ISUMSS:     00006 (00230)
MSS:        00000 (00013) (00227) (00246) (00253) (00265)
NOK:        0006F (00164) (00180)
OADDS:      0002F (00277)
OINPTS:     00031 (00281)
OK:         00061 (00165)
OSUMS:      0002D (00275)
SS2:        00048 (00121) (00134)
SS3:        00049 (00124) (00135)
SS4:        0004A (00128) (00136)
SAS12:      0039A (00015) (00240)
SAS13:      0039C (00016) (00241)
SAS79:      00420 (00017) (00230)
SAS80:      00422 (00018) (00225) (00275)
SAS82:      00426 (00019) (00272)
SAS83:      00428 (00020) (00273)
SAS84:      0042A (00021) (00235)
SCAS:       07F12 (00033) (00049)
SCASSA:     00000 (00047) (00053) (00207)
SCASSZ:     00085 (00048) (00207) (00208)
SCLRS:      00002 (00216) (00225)
SINVS:      00026 (00095)
SIZFS:      000FF (00023) (00232) (00237) (00248) (00270) (00278) (00285)
SSUMS:      00038 (00115)
START:      07F10 (00022) (00041)
SVTS:       003B2 (00014) (00015) (00016) (00017) (00018) (00019) (00020) (00021)
```

```
TMP1S:      00800 (00024) (00284)
TMP2S:      00A00 (00025) (00231) (00236) (00242) (00243) (00269) (00277)
TOPSP1:     00021 (00260) (00266)
TOPSP3:     0003A (00291) (00295)
VADDS:      00006 (00059) (00071)
VADDS2:     0005C (00160) (00172) (00181)
VADDS2E:    00070 (00173) (00182)
VADDSDN:    00021 (00076) (00079) (00083) (00088)
VDNS1:      0001C (00063) (00082) (00086)
VDNS2:      00014 (00066) (00073)
VINPTS:     00071 (00179) (00185)
WS:         00002 (00026) (00228) (00233) (00238) (00245) (00249) (00250) (00254) (00261) (00271)
                  (00279) (00286) (00292) (00294)
WWS:        00004 (00027)
ZS:         00003 (00028)
ZRO:        0006A (00169) (00175)
ZROLP:      00069 (00174) (00176)
ZZMS:       00007 (00029)

LINES WITH ERRORS:     0  (MAP VERSION 800101.10)    F=     0
```

```
                   (00001) * FCB 255... "MPCDBA(Y,U,V)"          COMPLETE DCT BIT ASSIGNMENT
                   (00002) *                                 ORIGINATED:25-JUN-79
                   (00003) *                                 UPDATED:22-MAY-80
                   (00004) * DEFINE GLOBAL SYMBOLS
                   (00005)        OPADD EXP, (1 .LS. 14) + (12 .LS. 8) + $1E
                   (00006)        OPADD DOF, (3 .LS. 10) + (12 .LS. 5) + $1B
          00000BEB (00007)        AFDTSORG=$BEB
          0001FFC0 (00008)        APSSMEM=$1FFC0
          000021FC (00009)        CSPUSNOS=$21FC
          00000794 (00010)        DMYS=$794
          00000828 (00011)        IBITS=D'2088'
          00000003 (00012)        #M=3
          00000001 (00013)        HS=1
          00000000 (00014)        MSS=0
          00000382 (00015)        SVTS=$0382
          000003CF (00016)        SAS38=SVTS+2*D'38'
          00000420 (00017)        SAS79=SVTS+2*D'79'
          0000042A (00018)        SAS84=SVTS+2*D'84'
          000000FF (00019)        SIZES=D'256'-D'1'
          00008100 (00020)        START=$8100
          00000800 (00021)        TMP1S=D'2048'
          00000002 (00022)        WS=2
          00000004 (00023)        WWS=4
                   (00024) *
                   (00025) * EXPAND ARRAY FUNCTION DISPATCH TABLE
          00000BE2 (00026)        #L=AFDTSORG+3*2*(255-128)                     FCB 255
00BE2 001E8102 (00027)        ADDR CDBAS(R7,1)
00BE4 001E816C (00028)        ADDR G102S(R7,1)
00BE6 001021FC (00029)        ADDR CSPUSNOS(,1,0)
                   (00030)        EJECT
```

```
          (00031) *
          (00032) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
          (00033) *
00008100  (00034)          #L=START
          (00035)          EVEN
          (00036)          EJECT
```

```
                       (00037) *
                       (00038) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P U <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                       (00039) *
                       (00040)          EVEN
    08100 0000         (00041)          DATA CDBASSA                             START ADDRESS
    08101 0031         (00042)          DATA CDBASSZ                             MODULE SIZE
                       (00043) CDBAS    BEGIN APU(CDBA)
          00000000     (00044)          #A=0
A00 08102 08F10000     (00045) CDBASSA  MOV(IOA,A1)\NOP                          IBITSL
A01 08104 08F008F7     (00046) TSTS     MOV(IOA,A0)\MOV(IOA,A7)                  IBIT(I)
A02 08106 4820089C     (00047)          SUB(A1,A0)\MOV(R,00)                     IBITSL-IBIT(I);00=IBIT(I
                       (00048) ;                                                 -1)
A03 08108 089102F0     (00049)          MOV(R,A1)\R(A7)                          IBITSL=IBITSL-IBIT(I);R=
                       (00050) ;                                                 IBIT(I)
A04 0810A 911F0009     (00051)          JUMPS(ADJS,T1)                           IF T1=1 NO BITS LEFT
A05 0810C 90160001     (00052)          JUMPC(TSTS,FWI)                          FOR ALL I,I=1,2...LTH
A06 0810E 0000089C     (00053)          NOP\MOV(R,00)                            IBIT(LTH)=IBIT(LTH)
A07 08110 901C0007     (00054) #1       JUMPC(#1,FO)                             WAIT FOR OUTPUT TO CLEAR
A08 08112 10370010     (00055)          JUMP(CLMPS,WI),CLEAR                     BITS LEFT! RELEASE APS
                       (00056) *
A09 08114 40200000     (00057) ADJS     ADD(A1,A0)\NOP                           IBIT(I)-OV
A0A 08116 089C0000     (00058)          MOV(R,00)\NOP                            00=IBIT(I)=IBIT(I)-OV
A0B 08118 1049000D     (00059)          JUMP(FWISC,AF1),SET                      ENTER @ TEST,SET EOT
A0C 0811A 08F0081C     (00060) CLRS     MOV(IOA,NULL)\MOV(ZERO,00)               IBIT(I+1);IBIT(I+1)=0
A0D 0811C 9016000C     (00061) FWISC    JUMPC(CLRS,FWI)                          FOR ALL I,I=2,3...LTH
A0E 0811E 901C000E     (00062) #1       JUMPC(#1,FO)                             WAIT FOR OUTPUT TO CLEAR
A0F 08120 10370029     (00063)          JUMP(FIXS,WI),CLEAR                      "IFIX" VALUES. RELEASE A
                       (00064) ;                                                 PS
                       (00065) *
A10 08122 068008F0     (00066) CLMPS    K(-1,0)\MOV(IOA,A0)                      R=-1;A0=5
A11 08124 08921680     (00067)          MOV(R,A2)\K(+1,0)                        A2=-1;R=1
A12 08126 00000892     (00068)          NOP\MOV(R,A2)                            A2=1
A13 08128 02200280     (00069) LIMITS   R(A1)\R(A4)                              R=IBITSL;R=?,IBIT(I-1)
A14 0812A 089108D3     (00070) TSTS5    MOV(R,A1)\MOV(IO,A3)                     A1=IBITSL;A3=IBIT(I)
A15 0812C 414048IC     (00071)          ADD(A2,A1)\MOV(00),SUB(A0,A3)            IBITSL-1;00=?,IBIT(I-1)
                       (00072) ;                                                 THEN 5-IBIT(I)
A16 0812E 000008F4     (00073)          NOP\MOV(IOA,A4)                          A4=IBIT(I)
A17 08130 9116001C     (00074)          JUMPS(LST1SA,FWI)                        WI=1,RAF=0 SIGNALS END
A18 08132 911E0022     (00075)          JUMPS(MTSA,T1)                           IF T1=1 NO BITS LEFT
A19 08134 911F0013     (00076)          JUMPS(LIMITS,T2)                         IF T2=1 IBIT(I).GE.5
A1A 08136 00004280     (00077)          NOP\ADD(A4,A2)                           T1=T2=0 SO IBIT(I)+1
A1B 08138 10000014     (00078)          JUMP(TSTS5)                              FOR ALL I,I=1,2...LTH
A1C 0813A 911E0022     (00079) LST1SA   JUMPS(MTSA,T1)                           IF T1=1 NO BITS LEFT
A1D 0813C 901F001F     (00080)          JUMPC(LIMS1,T2)                          IF T2=1 IBIT(LTH).GE.5
```

```
A1E 0813E 00000812 (00081)         NOP\MOV(ZERO,A2)            INCREMENT=0
A1F 08140 00004280 (00082) LIMS1   NOP\ADD(A4,A2)              IBIT(LTH)+2
A20 08142 0000089C (00083)         NOP\MOV(R,00)               IBIT(LTH)=IBIT(LTH)+2
A21 08144 10000027 (00084)         JUMP(MTSOUT)                INPUT CLEAR,CHECK OUTPUT
A22 08146 00000280 (00085) MTSA    NOP\R(A4)                   R=IBIT(2)
A23 08148 0000089C (00086)         NOP\MOV(R,00)               IBIT(2)=IBIT(2)
A24 0814A 10490026 (00087)         JUMP(MTSTST,AF1),SET        ENTER "MT QUEUE" @ TEST
A25 0814C 08FC0000 (00088) MTSIN   MOV(IOA,00)\NOP             IBIT(I)=IBIT(I)
A26 0814E 90160025 (00089) MTSTST  JUMPC(MTSIN,FWI)            WAIT FOR INPUT TO COMPLE
                   (00090) ;                                   TE
A27 08150 90180027 (00091) MTSOUT  JUMPC(MTSOUT,OOE)           WAIT FOR OUTPUT TO CLEAR
A28 08152 20372037 (00092)         CLEAR(WI)                   RELEASE APS INPUT
                   (00093) *
A29 08154 08FF08FF (00094) FIXS    MOV(IOA,M7)                 M7=SA
A2A 08156 08F80000 (00095) #1      MOV(IOA,M0)\NOP             M0=UE
A2B 08158 000008F8 (00096)         NOP\MOV(IOA,M0)             M0=U0
A2C 0815A 84718471 (00097)         MOV(A1),MUL(M0,M7)          A0=P',P=SA*U
A2D 0815C 3A3C3A3C (00098)         MOV(00),ALIGN(A1)           00=R'',R''="IFIX"(P)
A2E 0815E 901D002A (00099)         JUMPC(#1,FI)                FOR ALL I=0,1,2...LTH-1
                   (00100) *
A2F 08160 20320000 (00101) DNESA   CLEAR(RA)\NOP               APU DONE
A30 08162 00000000 (00102)         NOP
                   (00103) *
          00000031 (00104)         CDBASSZ=#A-CDBASSA
    08164          (00105)         END CDBASSZ
                   (00106)         EJECT
```

```
                      (00107) *
                      (00108) *>>>>>>>>>>>>>>>>>>>>>>>>>>>> A P S <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                      (00109) *
                      (00110)          EVEN
    08164 000081F6    (00111)          ADDR G102SI                  POINTER TO CONSTRUCTED I
                      (00112) ;                                     NSTRUCTION BLOCK
    08166 00008170    (00113)          ADDR G102S+2*INSS            POINTER TO SCALAR BLOCK
    08168 0000        (00114)          DATA 0                       NUMBER OF SCALARS
    08169 007C        (00115)          DATA G102SZ                  MODULE SIZE
    0816A 000081D6    (00116)          ADDR G102SA                  POINTER TO CHAIN ANCHOR
                      (00117)          EVEN                         END ON WORD BOUNDARY
                      (00118) *
                      (00119) G102S    BEGIN APS(G102)
A00 0816C 00201F40    (00120)          JSN(G102SO,P2)               SET OUTPUT PC(P2)
A01 0816E 02300030    (00121)          SET(RO)                      ENABLE OUTPUT ADDRESSING
A02 08170 04F20420    (00122) INSS     LOAD(BR2,SAS79(1),TF)        IO=BTLTH
A03 08172 064607FF    (00123)          LOAD(BR0,TMP1S-2(3))         TMP1(.) BA-2
A04 08174 085000FF    (00124)          LOAD(BR1,SIZES)              TMP1(.) SIZE-1
A05 08176 0A8A0002    (00125) INST     ADD(BR0,WS,TF)               IO=TMP1(I)
A06 08178 0C8A0002    (00126)          ADD(BR0,WS,TF)               IO=TMP1(I+1)
A07 0817A 0E1905B2    (00127)          SUBL(BR1,2),JUMPP(INST)      FOR I=0,1,2...LTH-1
A08 0817C 10300037    (00128)          SET(WI)                      STALL APS INPUT
A09 0817E 12000020    (00129)          NOP                          RELEASED BY APU
A0A 08180 14300030    (00130)          SET(RO)                      RELEASE APS OUTPUT
A0B 08182 160017F9    (00131)          JUMPS(IFIXSS,AF1)            AF1=1 THEN "IFIX" NOW
                      (00132) *
A0C 08184 18F2042A    (00133) CLMPSI   LOAD(BR2,SAS84(1),TF)        IO=THRESHOLD=5.0
A0D 08186 1A7000FF    (00134)          LOAD(BR3,SIZES)              TMP1(.) SIZE-1=# OF PASS
                      (00135) ;                                     ES
A0E 08188 1C4607FF    (00136) PASSSI   LOAD(BR0,TMP1S-2(3))         TMP1(.) BA-2
A0F 0818A 1E5000FF    (00137)          LOAD(BR1,SIZES)              TMP1(.) SIZE-1
A10 0818C 200014F9    (00138)          JUMPS(FINSI,AF1)             APU HAS FINISHED
A11 0818E 228A0002    (00139) INCSI    ADD(BR0,WS,TF)               IO=TMP1(I)
A12 08190 24191161    (00140)          SUBL(BR1,1),JUMPP(INCSI)     FOR I=0,1,2...LTH-1
A13 08192 26390FB1    (00141)          SUBL(BR3,1),JUMPP(PASSSI)    PASS=1,2...LTH
A14 08194 28300037    (00142) FINSI    SET(WI)                      STALL APS INPUT
A15 08196 2A000020    (00143)          NOP                          RELEASED BY APU
A16 08198 2C300030    (00144)          SET(RO)                      RELEASE APS OUTPUT
                      (00145) *
A17 0819A 2FC203CE    (00146) IFIXSS   LOAD(BR0,SAS38(1),TF)        SCALAR SA=2**(-15)
A18 0819C 304607FF    (00147)          LOAD(BR0,TMP1S-2(3))         TMP1(.) BA-2
A19 0819E 32500103    (00148)          LOAD(BR1,SIZES+4)            TMP1(.) SIZE-1+4
A1A 081A0 348A0002    (00149) IFIXSI   ADD(BR0,WS,TF)               IO=TMP1(I)
A1B 081A2 368A0002    (00150)          ADD(BR0,WS,TF)               IO=TMP1(I+1)
```

```
A1C 081A4 38191AB2 (00151)          SUBL(BR1,2),JUMPP(IFIXSI)          FOR I=1,2...LTH+3
                   (00152) *
A1D 081A6 3A200031 (00153) DNESI    CLEAR(RI)                          INPUT DONE!
A1E 081A8 3C000020 (00154)          NOP
                   (00155) *
A1F 081AA 3E300032 (00156) G102S0   SET(RA)                            ENABLE APU
A20 081AC 40F20794 (00157)          LOAD(BW2,DMYS(1),TF)               FOR OO=?
A21 081AE 424607FF (00158)          LOAD(BW0,TMP1S-2(3))               TMP1(.) BA-2
A22 081B0 445000FF (00159)          LOAD(BW1,SIZES)                    TMP1(.) SIZE-1
A23 081B2 46BA0002 (00160) OUTST    ADD(BW0,WS,TF)                     OO=TMP1(J)
A24 081B4 48BA0002 (00161)          ADD(BW0,WS,TF)                     OO=TMP1(J+1)
A25 081B6 4A1123B2 (00162)          SUBL(BW1,2),JUMPP(OUTST)           FOR J=1,2...LTH-1
A26 081B8 4C200030 (00163)          CLEAR(RO)                          STALL APS OUTPUT
A27 081BA 4E000020 (00164)          NOP                                RELEASED BY APS INPUT
A28 081BC 500032F9 (00165)          JUMPS(OFIXS,AF1)                   AF1=1 THEN "IFIX" NOW
                   (00166) *
A29 081BE 52F20794 (00167) CLMPS0   LOAD(BW2,DMYS(1),L,TF)             FOR OO=?
A2A 081C0 547000FF (00168)          LOAD(BW3,SIZES)                    TMP1(.) SIZE-1=# OF PASS
                   (00169) ;                                           ES
A2B 081C2 564607FF (00170) PASSS0   LOAD(BW0,TMP1S-2(3))               TMP1(.) BA-2
A2C 081C4 585000FF (00171)          LOAD(BW1,SIZES)                    TMP1(.) SIZE-1
A2D 081C6 5A0031F9 (0   72)         JUMPS(FINS0,AF1)                   APU HAS FINISHED
A2E 081C8 5CBA0002 (0   73) INCSJ   ADD(BW0,WS,TF)                     OO=TMP1(J)
A2F 081CA 5E112FB1 (00174)          SUBL(BW1,1),JUMPP(INCSJ)           FOR J=1,2...LTH-1
A30 081CC 60312BB1 (00175)          SUBL(BW3,1),JUMPP(PASSS0)          PASS=1,2...LTH
A31 081CE 62200030 (00176) FINS0    CLEAR(RO)                          STALL APS OUTPUT
                   (00177) *
A32 081D0 64700003 (00178) OFIXS    LOAD(BW3,3)                        3+1=4 DUMMIES
A33 081D2 66F20794 (00179) #1       LOAD(BW2,DMYS(1),TF)               OO=?
A34 081D4 683133B1 (00180)          SUBL(BW3,1),JUMPP(#1)              4 TIMES
A35 081D6 6A400000 (00181)          LOAD(BW0,[0])                      IBIT(.) BA
A36 081D8 6C500000 (00182)          LOAD(BW1,MSS)                      IBIT(.) SIZE-1
A37 081DA 6E020000 (00183)          SUB(BW0,MSS)                       IBIT(.) BA-1
A38 081DC 710A0001 (00184) OFIXSI   ADD(BW0,HS,TF)                     OO=IBIT(I)
A39 081DE 730A0001 (00185)          ADD(BW0,HS,TF)                     OO=IBIT(I+1)
A3A 081E0 74113BB2 (00186)          SUBL(BW1,2),JUMPP(OFIXSI)          FOR I=1,2...LTH-1
                   (00187) *
A3B 081E2 76200030 (00188) DNES0    CLEAR(RO)                          APU DONE
A3C 081E4 78000020 (00189)          NOP
                   (00190) *
          000081D6 (00191)          G102SA= #C                         FREEZE CHAIN ANCHOR
    081E6          (00192)          END #A-1
                   (00193) ;  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
    081E6 00000000 (00194) G102SI   DATA 1F'0,0'
```

```
        0000007C (00195)        G102$Z=#L-G102$
ORIEN            (00196)        END
```

```
ADJS:       00009 (00051) (00057)
AFDTSORG:   008FB (00007) (00026)
APSSMEM:    1FFC0 (00008)
CDBAS:      08102 (00027) (00043)
CDBASSA:    00000 (00041) (00045) (00104)
CDBASSZ:    00031 (00042) (00104) (00105)
CLMPS:      00010 (00055) (00066)
CLMPSI:     0000C (00133)
CLMPSO:     00029 (00167)
CLRS:       0000C (00060) (00061)
CSPUSNOS:   021FC (00009) (00029)
DMYS:       00794 (00010) (00157) (00167) (00179)
DNFSA:      0002F (00101)
DNFSI:      0001D (00153)
DNFSO:      0003B (00188)
FINSI:      00014 (00138) (00142)
FINSO:      00031 (00172) (00176)
FIXS:       00029 (00063) (00094)
FWISC:      0000D (00059) (00061)
G102S:      0816C (00028) (00113) (00119) (00195)
G102SA:     081D6 (00116) (00191)
G102SI:     081F6 (00111) (00194)
G102SO:     0001F (00120) (00156)
G102SZ:     0007C (00115) (00195)
HS:         00001 (00013) (00184) (00185)
IBITS:      00828 (00011)
IFIXSI:     0001A (00149) (00151)
IFIXSS:     00017 (00131) (00146)
INSS:       00002 (00113) (00122)
INST:       00005 (00125) (00127)
INCSI:      00011 (00139) (00140)
INCSJ:      0002F (00173) (00174)
LIMSI:      0001F (00080) (00082)
LIMITS:     00013 (00069) (00076)
LSTISA:     0001C (00074) (00079)
MSS:        00000 (00014) (00182) (00183)
MTSA:       00022 (00075) (00079) (00085)
MTSIN:      00025 (00088) (00089)
MTSOUT:     00027 (00084) (00091) (00091)
MTSTST:     00026 (00087) (00089)
OFIXS:      00032 (00165) (00178)
OFIXSI:     0003B (00184) (00186)
OUTST:      00023 (00160) (00162)
PASSSI:     0000F (00136) (00141)
```

```
PAGE     9:       FCB 255... "MPCDHA(Y,U,V)"              COMPLETE DCT BIT ASSIGNMENT

  PASSS0:     0002B (00170) (00175)
  SAS38:      003CF (00016) (00146)
  SAS79:      00420 (00017) (00122)
  SAS84:      0042A (00018) (00133)
  SIZES:      000FF (00019) (00124) (00134) (00137) (00148) (00159) (00168) (00171)
  START:      08100 (00020) (00034)
  SVTS:       003B2 (00015) (00016) (00017) (00018)
  TMP1S:      00800 (00021) (00123) (00136) (00147) (00158) (00170)
  TSTS:       00001 (00046) (00052)
  TSTS5:      00014 (00070) (00078)
  WS:         00002 (00022) (00125) (00126) (00139) (00149) (00150) (00160) (00161) (00173)
  WWS:        00004 (00023)

LINES WITH ERRORS:     0  (MAP VERSION 800101.10)    F=     0
```

T A B L E   O F   C O N T E N T S

PAGE 1: LINE1.TXT ORIGINATED:18-APR-79

(00001) * LINE1.TXT ORIGINATED:18-APR-79

```
                    (00002) '                   MODIFIED:04-JUN-80
                    (00003) ;  TOS2 PROGRAM TO TRANSMIT AND RECEIVE DATA
                    (00004) ;  & GENERATE INTERRUPTS AT BUFFER BOUNDARIES
                    (00005) ;  THE INTERRUPTS ARE TRAPPED AT THE CSPU WITH
                    (00006) ;  INTEGER SCALAR TABLE VALUES MODIFIED TO RE-
                    (00007) ;  FLECT THE STATUS OF THE I/O PORTS AS WELL
                    (00008) ;  AS THE BUFFER THEMSELVES. SEE "GTEIO.TXT"
                    (00009) ;  FOR A DEFINITION OF SCALAR VALUES.
                    (00010) ;
                    (00011) ;  REGISTER AND SYMBOL DEFINITIONS
          0000FCC4  (00012)         CNTRL=$FCC4
          00000794  (00013)         DMYS=$794
          00000793  (00014)         DMYSM1=DMYS-1
          00000001  (00015)         INCS=1
          00000000  (00016)         ZROS=0
                    (00017) ;
                    (00018) ;  BUFFER DEFINITIONS
                    (00019) ;
          00000171  (00020)         LSEN=D'369'
          000000F6  (00021)         BLTH=D'246'
          00000170  (00022)         LSENM1=LSEN-1
          000000B8  (00023)         HB=LSENM1/2
          000011F8  (00024)         RECV1=D'4600'
          00001369  (00025)         RECV2=D'4969'
          000014DC  (00026)         SEN1=D'5340'
          0000164E  (00027)         SEN2=D'5710'
          000014DB  (00028)         SEN1M1=SEN1-1
          0000164D  (00029)         SEN2M1=SEN2-1
          0000164C  (00030)         SEN1PBSM1=SEN1+LSENM1
          00001594  (00031)         SEN1PHB=SEN1+HB
          000017BE  (00032)         SEN2PBSM1=SEN2+LSENM1
          000011F7  (00033)         RECV1M1=RECV1-1
          00001368  (00034)         RECV2M1=RECV2-1
          00001368  (00035)         RECV1PBSM1=RECV1+LSENM1
          000014D9  (00036)         RECV2PBSM1=RECV2+LSENM1
                    (00037) ;
          00000000  (00038)         R1=0
                    (00039)         EVEN
                    (00040)         BEGIN TOS2(TEST+P)
A00 00000 01C0FCC4  (00041) TPS1    LOAD(R3,CNTRL)          R3=ICW CAN BE UPDATED BY
A01 00002 62300280  (00042)         MR                      MAP READ(M -> I)
A02 00004 03E96400  (00043)         ADDL(R3,ZROS,IE)        C(R3)+0 -> I(EMADD,15:0)
A03 00006 0430CC00  (00044)         JTEC(SETSE,P1)          P1=0 NO NEED FOR DUMMY
A05 0000A 06300200  (00045)         MW                      MAP WRITE(I -> M)
```

```
A06 0000C 07820794 (00046)          LOAD(R2,DPYS(1),L)           SET INPUT REG. @ DUMMY
A07 0000E 081A0000 (00047)          ADDL(R2,ZROS,TM)             GEN ADDR TO CLEAR P1
A08 00010 09820793 (00048) SETSF    LOAD(R2,DPYSM1(1),L)         SET R2 FOR FIRST INPUT
A09 00012 0A041594 (00049) SETS0    LOAD(R0,SEN1PBB(2),L)        SET OUTPUT REG. @ BASE-1
A0A 00014 0B1411F7 (00050) SETS1    LOAD(R1,RECV1M1(2),L)        SET INPUT REG. @ BASE-1
A0B 00016 0C300180 (00051)          CF1                          RESET "R0 BUMPED" FLAG
A0C 00018 0D300780 (00052)          CF2                          RESET "R1 BUMPED" FLAG
A0D 0001A 0E300600 (00053) CHKS0    JIFC(CHKS1,F1)               R0 BUMPED?
A0F 0001E 1028163C (00054)          JNE(CHKS01,R0,SEN1PBSM1)     YES,CHECK INDEX @ MIDDLE
A11 00022 12302000 (00055)          INT1                         MIDDLE! RELEASE BID0,ATT
                   (00056) ;                                     ACH BID2
A12 00024 13041640 (00057)          LOAD(R0,SEN2M1(2),L)         RESET BASE JUST BELOW 2
                   (00058) ;                                     ND BUFFER
A13 00026 1A300180 (00059)          JUMP(CHKS1),CF1              RESET "R0 BUMPED" FLAG
A14 00028 1628176E (00060) CHKS01   JNE(CHKS1,R0,SEN2PBSM1)      YES,CHECK INDEX @ END AD
A15 0002A 00000000
                   (00061) ;                                     DRESS
A17 0002E 18302000 (00062)          INT1                         END! RELEASE BID2,ATTACH
A18 00030 19300580 (00063)          CF1                          RESET "R0 BUMPED" FLAG
A19 00032 1A041308 (00064) RESETS0  LOAD(R0,SEN1M1(2),L)         RESET OUTPUT REG. @ BASE
                   (00065) ;                                     -1
A1A 00034 1C301800 (00066) CHKS1    JIFC(CHKSP,F2)               R1 BUMPED?
A1B 00036 00000000
A1D 0003A 1E681368 (00067)          JNE(CHKS11,R1,RECV1PBSM1)    YES,CHECK INDEX @ MIDDLE
A1F 0003E 20304000 (00068)          INT2                         MIDDLE! RELEASE BID1,ATT
                   (00069) ;                                     ACH BID3
A20 00040 21441368 (00070)          LOAD(R1,RECV2M1(2),L)        RESET BASE JUST BELOW 2
                   (00071) ;                                     ND BUFFER
A21 00042 28400480 (00072)          JUMP(CHKSP),CF2              RESET "R1 BUMPED" FLAG
A22 00044 24681109 (00073) CHKS11   JNE(CHKSP,R1,RECV2PBSM1)     YES,CHECK INDEX @ END AD
A23 00046 00000000
                   (00074) ;                                     DRESS
A25 0004A 26304000 (00075)          INT2                         END! RELEASE BID3,ATTACH
A26 0004C 274411F7 (00076) RESETS1  LOAD(R1,RECV1M1(2),L)        RESET INPUT REG. @ BASE-
                   (00077) ;                                     1
A27 0004E 28300480 (00078)          CF2                          RESET "R1 BUMPED" FLAG
A28 00050 2A301C00 (00079) CHKSP    JIFC(CHKSP1,P2)              CHECK P2 FOR DATA REQUES
A29 00052 00000000
                   (00080) ;                                     T.
A2B 00056 2C300280 (00081)          KP                           MAP READ(M -> I)
A2C 00058 2D3A0101 (00082)          ADDL(R0,INCS,IM)             GEN ADDRESS & RESET P2
A2D 0005A 2E300100 (00083)          SF1                          SET "R0 BUMPED" FLAG
A2E 0005C 30400C00 (00084) CHKSP1   JIFC(CHKS0,P1)               CHECK P1 FOR DATA REQUES
A2F 0005E 00000000
```

```
                         (00085) ;                                           T.
 A31 00062 32300200 (00086)              SV
 A32 00064 33300700 (00087)              SF2                                 SET "R1 BUMPED" FLAG
 A33 00066 34AC0293 (00088)              JEQ(FIRSTS,R2,DMYSM1)               WASTE FIRST INPUT
 A35 0006A 367A0001 (00089)              ADDL(R1,INCS,T4)                    GEN ADDRESS & RESET P1
 A36 0006C 0E300400 (00090)              JUMP(CHKS0)                         TOP OF LOOP
 A37 0006E 388A0001 (00091) FIRSTS       ADDL(R2,INCS,14)                    GEN ADDRESS & RESET P1
 A38 00070 0E300400 (00092)              JUMP(CHKS0)                         TOP OF LOOP
 A39 00072 35306000 (00093) DNFS         STOP                                CSPU SHUTS IT OFF!
     00074          (00094)              END
```

```
PAGE     5:        LIPF1.TXT        ORIGINATED:18-APR-79
                                    MODIFIED:04-JUN-80

  BLTN:      000E6 (00021)
  CHKS0:     0000E (00053) (00084) (00090) (00092)
  CHKS01:    00014 (00054) (00060)
  CHKS1:     0001A (00053) (00059) (00060) (00066)
  CHKS11:    00022 (00067) (00073)
  CHKSP:     00028 (00066) (00072) (00073) (00079)
  CHKSP1:    0002E (00079) (00084)
  CNTRL:     0FCC4 (00012) (00041)
  DMYS:      00794 (00013) (00014) (00046)
  DMYSM1:    00794 (00013) (00048) (00088)
  DNES:      00038 (00093)
  FIRSTS:    00037 (00088) (00091)
  HB:        00008 (00023) (00031)
  INCS:      00001 (00015) (00082) (00089) (00091)
  LSEN:      00171 (00020) (00022)
  LSENM1:    00170 (00022) (00023) (00030) (00032) (00035) (00036)
  RECV1:     011E8 (00024) (00033) (00035)
  RECV1M1:   011E7 (00033) (00050) (00076)
  RECV1PRS:  01368 (00035) (00067)
  RECV2:     01369 (00025) (00034) (00036)
  RECV2M1:   01368 (00034) (00070)
  RECV2PRS:  01409 (00036) (00073)
  RESETS0:   00019 (00064)
  RESETS1:   00026 (00076)
  SEN1:      0140C (00026) (00028) (00030) (00031)
  SEN1M1:    0140B (00028) (00064)
  SEN1PRSP:  0164C (00030) (00054)
  SEN1PRB:   01594 (00031) (00049)
  SEN2:      0164F (00027) (00029) (00032)
  SEN2M1:    0164D (00029) (00057)
  SEN2PRSP:  017EE (00032) (00060)
  SETS0:     00009 (00049)
  SETS1:     0000A (00050)
  SETSF:     00008 (00044) (00048)
  TPS1:      00000 (00041)
  ZROS:      00000 (00016) (00043) (00047)

LINES WITH ERRORS:    0   (MAP VERSION R00101,10)    E=    0
```

3.6.2 Executive Programs

The modules contained in this section are:

GTE 300

IOS

FF2R

APDONE

CSPUIS

SNAP-II MAP-300 ARITH. MODULES - PROG. #830101.03 MAY 7, 1980

T A B L E   O F   C O N T E N T S

```
         *  SNAP-II MAP-300 ARITH. MODULES - PROG. #830101.03 MAY 7, 1980
         *
         *  MODIFIED FOR GTE SYLVANIA     BY S. TERRACE
         *
         *       MODULES INCLUDED:
         *
         *            NAME           FCR #
         *            ----           -----
         *
         *            VELT            136
         *            VMOV            143
         *            VSMA1           144
         *            VCOS            186
         *
         *            FFTN            204
         *            FFTNB           205
         *            FFTIN           206
         *            FFTINB          207
         *
         *
         ! DEFINE SYMBOLS FOR ARRAY FUNCTION ASSEMBLIES
         *
         *
         * FROM THE SNAP-II EXECUTIVE  ---  REL. 3.05  --- 5/22/79
         *
         *
000008F8           AFDTSORG = $8F8
00000245           APSASSS = $245
00000EFA           APSHNDR = $EFA
00000F63           APSHNDRO = $F63
00000F2D           APSHNDR1 = $F2D
0000024D           APSBSL = $24D
00000248           APSCSSC = $248
00000FB2           APSDUNE = $FB2              MODIFIED FOR NEW START ADDRESS
0000000C           APSG0 = $C
0000000D           APSG1 = $D
00000010           APSPBFF = $10
0000000A           APSSAID = $A
0000000F           APSSSCLR = $F
00000009           APSSSS = $9
00000004           APSSHMO = $4
0001FFC0           APSR = $1FFC0
         *
00000604           HCTSAD = $604
```

```
00000686 (00045)          BCTSAT = $686
00000582 (00046)          BCTSHA = $582
00000000 (00047)          BITSG0 = $0
         (00048) *
00000792 (00049)          CLRSGOG1 = $792
000021FC (00050)          CSPUSNOS = $21FC
         (00051) *
00000794 (00052)          DMYS = $794
         (00053) *
00001AFA (00054)          ERRORS = $1AFA
         (00055) *
0000079A (00056)          FETSCHSZ = $79A
00000000 (00057)          FLGSCLR = $0
00000004 (00058)          FLGSG0 = $4
00000005 (00059)          FLGSG1 = $5
00000011 (00060)          FLGSRI = $11
00000020 (00061)          FLGSSET = $20
         (00062) *
00000001 (00063)          HS = 1
         (00064) *
00001B7E (00065)          LOADSAP = $1B7E
00001B8F (00066)          LOADSAP1 = $1B8F
         (00067) *
0000FF00 (00068)          MSKSLBYT = $FF00
000000FF (00069)          MSKSRBYT = $00FF
00000000 (00070)          MSS = 0
         (00071) *
000007AF (00072)          SHFTLSR5 = $7AF
000003B2 (00073)          SVTS = $3B2
000003CE (00074)          SVTSUN1 = $3CE
0001FFCE (00075)          SYSSFLGS = $1FFCE
         (00076) *
000007B4 (00077)          TEMS0 = $7B4
000021FE (00078)          TOFS = $21FE
000002B8 (00079)          TOFSPTR = $2B8
         (00080) *
00000002 (00081)          WS = $2
         (00082) *
000007BA (00083)          ZERO = $7BA
         (00084) *
         (00085) *
         (00086)          EJECT
```

```
                    (00087) *
          00000288  (00088) #L = TOESPTR
                    (00089) *
    00288 00104678  (00090)        ADDR    TOESCUR(,1)          UPDATE TOP OF EXEC POINTER
                    (00091) *
                    (00092) *
                    (00093) *  DEFINE SYMBOLS FOR THIS MODULE
                    (00094) *
          00000003  (00095)        #M = 3
          00004000  (00096)        START = $4000                DEFINE START LOCATION FOR MODULE
                    (00097) *
                    (00098) *
                    (00099)        EJECT
```

```
* SPECIAL NOTE: THESE MODULES ARE THE SAME AS RELEASE 2.1 EXCEPT
* THE SPECIAL BINDING IN THE FFT'S AND SPECIAL FUNCTIONS HAVE BEEN
* MODIFIED FOR RELEASE 3.0
*
*
!  ARITHMETIC FUNCTIONS DISPATCH TABLE
*
*
* EVERY ARRAY FUNCTION OCCUPIES ONE NODE IN THE TABLE.
* THE NODE FOR ARRAY FCB #N WILL BE FOUND AT LOCATION
*       AFDTS - 3 * WS * (#N-128)
*
* EACH NODE CONSISTS OF 3 POINTERS AS FOLLOWS:
*       WORD 1 = ADDR  APU$MODULE(R7,1)
*       WORD 2 = ADDR  APS$MODULE(R7,1)
*       WORD 3 = ADDR CSPU$MODULE(,1,SSSV)
*           IF BIT 0 OF THE FCB CONTROL BITS = 0
*             NORMAL BINDING OCCURS AND CSPU$MODULE
*             IS EXECUTED FOR SPECIAL AP-DONE SUPPORT
*           ELSE CSPU$MODULE IS FOR SPECIAL BINDING
*           SSSV = SPECIAL SUPPORT SEMAPHORE VALUE
*
*
*
            EJECT
```

```
*  STANDARD FORMAT FOR ALL ARRAY FCB'S
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*        !                         !                         !
*   WORD !        LEFT BYTE        !       RIGHT BYTE        !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    0   !    POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    1   !   FCB NUMBER            !  CONTROL BITS           !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    2   !     Y [0] AND [8]       !          SA             !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    3   !     U [1] AND [9]       !          SB             !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    4   !     V [2] AND [10]      !          SC             !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*        !                         !                         !
*    5   !     W [3] AND [11]      !          SD             !
*        !                         !                         !
*   -----!-------------------------!-------------------------!
*
*
*
*
          EJECT
```

```
          000008E8 (00165)        *L = AFDTSORG           SET PC TO DISPATCH TABLE
                   (00166) *
                   (00167) *
          00000000 (00168)        APU$NULL = 0            NULL VALUES FOR UNIMPLEMENTED FCB
          00000000 (00169)        APS$NULL = 0
                   (00170) *
                   (00171) *
U   008E8 001F0000 E-00000        ADDR    SADDS(R7,1)     128 = SCALAR ADD
U   008EA 001F0000 E-00172        ADDR    S0021S(R7,1)
    008EC 001021FC (00174)        ADDR    CSPU$NOS(,1,0)
                   (00175) *
U   008EE 001F0000 E-00173        ADDR    SSUBS(R7,1)     129 = SCALAR SUBTRACT
U   008F0 001F0000 E-00176        ADDR    S0021S(R7,1)
    008F2 001021FC (00178)        ADDR    CSPU$NOS(,1,0)
                   (00179) *
U   008F4 001F0000 E-00177        ADDR    SMULS(R7,1)     130 = SCALAR MULTIPLY
U   008F6 001F0000 E-00180        ADDR    S0021S(R7,1)
    008F8 001021FC (00182)        ADDR    CSPU$NOS(,1,0)
                   (00183) *
U   008FA 001F0000 E-00181        ADDR    SDIVS(R7,1)     131 = SCALAR DIVIDE
U   008FC 001F0000 E-00184        ADDR    S0021S(R7,1)
    008FE 001021FC (00186)        ADDR    CSPU$NOS(,1,0)
                   (00187) *
    00900 001F0000 (00188)        ADDR    APU$NULL(R7,1)  132 = UNASSIGNED
    00902 001F0000 (00189)        ADDR    APS$NULL(R7,1)
    00904 001021FC (00190)        ADDR    CSPU$NOS(,1,0)
                   (00191) *
    00906 001F0000 (00192)        ADDR    APU$NULL(R7,1)  133 = UNASSIGNED
    00908 001F0000 (00193)        ADDR    APS$NULL(R7,1)
    0090A 001021FC (00194)        ADDR    CSPU$NOS(,1,0)
                   (00195) *
    0090C 001F0000 (00196)        ADDR    APU$NULL(R7,1)  134 = UNASSIGNED
    0090E 001F0000 (00197)        ADDR    APS$NULL(R7,1)
    00910 001021FC (00198)        ADDR    CSPU$NOS(,1,0)
                   (00199) *
    00912 001F0000 (00200)        ADDR    APU$NULL(R7,1)  135 = UNASSIGNED
    00914 001F0000 (00201)        ADDR    APS$NULL(R7,1)
    00916 001021FC (00202)        ADDR    CSPU$NOS(,1,0)
                   (00203) *
    00918 001F40F6 (00204)        ADDR    VFLTS(R7,1)     136 = VECTOR FLOAT
    0091A 001F41BC (00205)        ADDR    V1124AS(R7,1)
    0091C 001021FC (00206)        ADDR    CSPU$NOS(,1,0)
                   (00207) *
U   0091E 001F0000 E-00185        ADDR    VFIXS(R7,1)     137 = VECTOR FIX
```

```
U     00920 001F0000 F-00208        ADDR    V1124BS(R7,1)
      00922 001021FC (00210)        ADDR    CSPUSNOS(,1,0)
                     (00211) *
      00924 001F0000 (00212)        ADDR    APUSNULL(R7,1)       138 = VECTOR FLOAT (8-BIT)
U     00926 001F0000 F-00209        ADDR    V1124CS(R7,1)
      00928 001021FC (00214)        ADDR    CSPUSNOS(,1,0)
                     (00215) *
U     0092A 001F0000 F-00213        ADDR    VFIXS(R7,1)          139 = VECTOR FIX (8-BIT)
U     0092C 001F0000 F-00216        ADDR    V1124DS(R7,1)
      0092E 001021FC (00218)        ADDR    CSPUSNOS(,1,0)
                     (00219) *
      00930 001F0000 (00220)        ADDR    APUSNULL(R7,1)       140 = UNASSIGNED
      00932 001F0000 (00221)        ADDR    APSSNULL(R7,1)
      00934 001021FC (00222)        ADDR    CSPUSNOS(,1,0)
                     (00223) *
      00936 001F0000 (00224)        ADDR    APUSNULL(R7,1)       141 = UNASSIGNED
      00938 001F0000 (00225)        ADDR    APSSNULL(R7,1)
      0093A 001021FC (00226)        ADDR    CSPUSNOS(,1,0)
                     (00227) *
      0093C 001F0000 (00228)        ADDR    APUSNULL(R7,1)       142 = UNASSIGNED
      0093E 001F0000 (00229)        ADDR    APSSNULL(R7,1)
      00940 001021FC (00230)        ADDR    CSPUSNOS(,1,0)
                     (00231) *
      00942 001F0000 (00232)        ADDR    APUSNULL(R7,1)       143 = UNASSIGNED
      00944 001F0000 (00233)        ADDR    APSSNULL(R7,1)
      00946 001021FC (00234)        ADDR    CSPUSNOS(,1,0)
                     (00235) *
      00948 001F4002 (00236)        ADDR    VSMA1S(R7,1)         144 = VECTOR SCALAR MULT-ADD(1 VECTOR)
      0094A 001E4122 (00237)        ADDR    V1124S(R7,1)
      0094C 001021FC (00238)        ADDR    CSPUSNOS(,1,0)
                     (00239) *
U     0094E 001F0000 F-00217        ADDR    VSMA2S(R7,1)         145 = VECTOR SCALAR MULT-ADD(2 VECTOR)
U     00950 001E0000 F-00240        ADDR    V2134AS(R7,1)
      00952 001021FC (00242)        ADDR    CSPUSNOS(,1,0)
                     (00243) *
U     00954 001E0000 F-00241        ADDR    VMULS(R7,1)          146 = VECTOR MULTIPL
U     00956 001F0000 F-00244        ADDR    V2134AS(R7,1)
      00958 001021FC (00246)        ADDR    CSPUSNOS(,1,0)
                     (00247) *
U     0095A 001F0000 F-00245        ADDR    VRCPS(R7,1)          147 = VECTOR RECPROCAL
      0095C 001E4122 (00249)        ADDR    V1124S(R7,1)
      0095E 001021FC (00250)        ADDR    CSPUSNOS(,1,0)
                     (00251) *
      00960 001E0000 (00252)        ADDR    APUSNULL(R7,1)       148 = VECTOR DIVIDE
```

```
  00962 001F0000 (00253)    ADDR   APS$NULL(R7,1)     CURRENTLY NOT IMPLEMENTED
  00964 001021FC (00254)    ADDR   CSPU$NOS(,1,0)
                 (00255) *
  00966 001F0000 (00256)    ADDR   APU$NULL(R7,1)     149 = UNASSIGNED
  00968 001E0000 (00257)    ADDR   APS$NULL(R7,1)
  0096A 001021FC (00258)    ADDR   CSPU$NOS(,1,0)
                 (00259) *
  0096C 001F0000 (00260)    ADDR   APU$NULL(R7,1)     150 = UNASSIGNED
  0096E 001F0000 (00261)    ADDR   APS$NULL(R7,1)
  00970 001021FC (00262)    ADDR   CSPU$NOS(,1,0)
                 (00263) *
  00972 001E0000 (00264)    ADDR   APU$NULL(R7,1)     151 = UNASSIGNED
  00974 001F0000 (00265)    ADDR   APS$NULL(R7,1)
  00976 001021FC (00266)    ADDR   CSPU$NOS(,1,0)
                 (00267) *
U 00978 001E0000 F-00248    ADDR   VSABS$(R7,1)       152 = VECTOR ABSOLUTE
  0097A 001F4122 (00269)    ADDR   V1124$(R7,1)
  0097C 001021FC (00270)    ADDR   CSPU$NOS(,1,0)
                 (00271) *
U 0097E 001F0000 F-00268    ADDR   VSABSQ$(R7,1)      153 = VECTOR ABSOLUTE SQUARED
  00980 001E4122 (00273)    ADDR   V1124$(R7,1)
  00982 001021FC (00274)    ADDR   CSPU$NOS(,1,0)
                 (00275) *
U 00984 001E0000 E-00272    ADDR   VSQ$(R7,1)         154 = VECTOR SQUARE
  00986 001F4122 (00277)    ADDR   V1124$(R7,1)
  00988 001021FC (00278)    ADDR   CSPU$NOS(,1,0)
                 (00279) *
U 0098A 001F0000 F-00276    ADDR   VMAGSQ$(R7,1)      155 = VECTOR MAGNITUDE SQUARED
U 0098C 001E0000 E-00280    ADDR   V2134A$(R7,1)
  0098E 001021FC (00282)    ADDR   CSPU$NOS(,1,0)
                 (00283) *
U 00990 001E0000 E-00281    ADDR   VMAG$(R7,1)        156 = VECTOR MAGNITUDE
U 00992 001E0000 E-00284    ADDR   V2134A$(R7,1)
  00994 001021FC (00286)    ADDR   CSPU$NOS(,1,0)
                 (00287) *
U 00996 001F0000 F-00285    ADDR   VLOG16$(R7,1)      157 = VECTOR LOG
U 00998 001E0000 E-00288    ADDR   V2124A$(R7,1)
  0099A 001021FC (00290)    ADDR   CSPU$NOS(,1,0)
                 (00291) *
  0099C 001F0000 (00292)    ADDR   APU$NULL(R7,1)     158 = UNASSIGNED
  0099E 001E0000 (00293)    ADDR   APS$NULL(R7,1)
  009A0 001021FC (00294)    ADDR   CSPU$NOS(,1,0)
                 (00295) *
  009A2 001F0000 (00296)    ADDR   APU$NULL(R7,1)     159 = UNASSIGNED
```

```
     009A4 001F0000 (00297)        ADDR    APS$NULL(R7,1)
     009A6 001021FC (00298)        ADDR    CSPU$NOS(,1,0)
                    (00299) *
U    009A8 001F0000 F-00289        ADDR    VCLIPS(R7,1)        160 = VECTOR CLIP
     009AA 001F4122 (00301)        ADDR    V1124S(R7,1)
     009AC 001021FC (00302)        ADDR    CSPU$NOS(,1,0)
                    (00303) *
U    009AE 001F0000 F-00300        ADDR    VTHRSHS(R7,1)       161 = VECTOR THRESHOLD
     009B0 001F4122 (00305)        ADDR    V1124S(R7,1)
     009B2 001021FC (00306)        ADDR    CSPU$NOS(,1,0)
                    (00307) *
U    009B4 001F0000 F-00304        ADDR    VLIMITS(R7,1)       162 = VECTOR LIMIT
     009B6 001F4122 (00309)        ADDR    V1124S(R7,1)
     009B8 001021FC (00310)        ADDR    CSPU$NOS(,1,0)
                    (00311) *
U    009BA 001F0000 F-00308        ADDR    VCOMPS(R7,1)        163 = VECTOR COMPARE
U    009BC 001F0000 F-00312        ADDR    V2134AS(R7,1)
     009BE 001021FC (00314)        ADDR    CSPU$NOS(,1,0)
                    (00315) *
U    009C0 001F0000 F-00313        ADDR    VTAGS(R7,1)         164 = VECTOR TAG
     009C2 001F4122 (00317)        ADDR    V1124S(R7,1)
     009C4 001021FC (00318)        ADDR    CSPU$NOS(,1,0)
                    (00319) *
     009C6 001F0000 (00320)        ADDR    APU$NULL(R7,1)      165 = UNASSIGNED
     009C8 001F0000 (00321)        ADDR    APS$NULL(R7,1)
     009CA 001021FC (00322)        ADDR    CSPU$NOS(,1,0)
                    (00323) *
     009CC 001F0000 (00324)        ADDR    APU$NULL(R7,1)      166 = UNASSIGNED
     009CE 001F0000 (00325)        ADDR    APS$NULL(R7,1)
     009D0 001021FC (00326)        ADDR    CSPU$NOS(,1,0)
                    (00327) *
     009D2 001F0000 (00328)        ADDR    APU$NULL(R7,1)      167 = UNASSIGNED
     009D4 001F0000 (00329)        ADDR    APS$NULL(R7,1)
     009D6 001021FC (00330)        ADDR    CSPU$NOS(,1,0)
                    (00331) *
U    009D8 001F0000 F-00316        ADDR    SSUMS(R7,1)         168 = SCALAR SUM
U    009DA 001F0000 F-00332        ADDR    S1011S(R7,1)
     009DC 001021FC (00334)        ADDR    CSPU$NOS(,1,0)
                    (00335) *
U    009DE 001F0000 F-00333        ADDR    SSUMABS(R7,1)       169 = SCALAR SUM OF ABSOLUTES
U    009E0 001F0000 F-00336        ADDR    S1011S(R7,1)
     009E2 001021FC (00338)        ADDR    CSPU$NOS(,1,0)
                    (00339) *
U    009E4 001F0000 F-00337        ADDR    SSUMSQS(R7,1)       170 = SCALAR SUM OF SQUARES
```

```
U     009F6 001F0000 F-00340         ADDR     S1011$(R7,1)
      009F8 001021FC (00342)         ADDR     CSPU$NOS(,1,0)
                     (00343) *
U     009FA 001E0000 F-00341         ADDR     SMAX$(R7,1)           171 = SCALAR MAXIMUM
U     009FC 001F0000 F-00344         ADDR     S1002$(R7,1)
      009FE 001021FC (00346)         ADDR     CSPU$NOS(,1,0)
                     (00347) *
U     009F0 001F0000 F-00345         ADDR     SMAXAB$(R7,1)         172 = SCALAR MAXIMUM ABSOLUTE
U     009F2 001F0000 F-00348         ADDR     S1002$(R7,1)
      009F4 001021FC (00350)         ADDR     CSPU$NOS(,1,0)
                     (00351) *
U     009F6 001E0000 F-00349         ADDR     SDOT$(R7,1)           173 = SCALAR DOT PRODUCT
U     009F8 001F0000 F-00352         ADDR     S2011$(R7,1)
      009FA 001021FC (00354)         ADDR     CSPU$NOS(,1,0)
                     (00355) *
      009FC 001F0000 (00356)         ADDR     APU$NULL(R7,1)        174 = UNASSIGNED
      009FE 001F0000 (00357)         ADDR     APS$NULL(R7,1)
      00A00 001021FC (00358)         ADDR     CSPU$NOS(,1,0)
                     (00359) *
      00A02 001F0000 (00360)         ADDR     APU$NULL(R7,1)        175 = UNASSIGNED
      00A04 001F0000 (00361)         ADDR     APS$NULL(R7,1)
      00A06 001021FC (00362)         ADDR     CSPU$NOS(,1,0)
                     (00363) *
U     00A08 001E0000 F-00353         ADDR     CVMUL$(R7,1)          176 = COMPLEX VECTOR MULTIPLY
U     00A0A 001E0000 E-00364         ADDR     C2124A$(R7,1)
      00A0C 001021FC (00366)         ADDR     CSPU$NOS(,1,0)
                     (00367) *
U     00A0E 001E0000 F-00365         ADDR     CCVMUL$(R7,1)         177 = COMPLEX CONJ. VECTOR MULTIPLY
U     00A10 001F0000 F-00368         ADDR     C2124A$(R7,1)
      00A12 001021FC (00370)         ADDR     CSPU$NOS(,1,0)
                     (00371) *
U     00A14 001E0000 F-00369         ADDR     CVRCP$(R7,1)          178 = COMPLEX VECTOR RECIPRICAL
U     00A16 001E0000 E-00372         ADDR     C1124A$(R7,1)
      00A18 001021FC (00374)         ADDR     CSPU$NOS(,1,0)
                     (00375) *
U     00A1A 001E0000 F-00373         ADDR     CPOLAR$(R7,1)         179 = COMPLEX POLAR CONVERSION
U     00A1C 001F0000 F-00376         ADDR     C1124B$(R7,1)
      00A1E 001021FC (00378)         ADDR     CSPU$NOS(,1,0)
                     (00379) *
      00A20 001E0000 (00380)         ADDR     APU$NULL(R7,1)        180 = COMPLEX RECTANGULAR CONVERSION
      00A22 001F0000 (00381)         ADDR     APS$NULL(R7,1)          CURRENTLY NOT IMPLEMENTED
      00A24 001021FC (00382)         ADDR     CSPU$NOS(,1,0)
                     (00383) *
      00A26 001F0000 (00384)         ADDR     APU$NULL(R7,1)        181 = UNASSIGNED
```

```
      00A28 001F0000 (00385)          ADDR     APSSNULL(R7,1)
      00A2A 001021FC (00386)          ADDR     CSPUSNOS(,1,0)
                     (00387) *
U     00A2C 001F0000 F-00377          ADDR     CXMULS(R7,1)         182 = COMPLEX EXPONENTIAL MULTIPLY
U     00A2E 001F0000 F-00388          ADDR     C2120S(R7,1)
      00A30 001021FC (00390)          ADDR     CSPUSNOS(,1,0)
                     (00391) *
U     00A32 001F0000 F-00389          ADDR     CXMULRS(R7,1)        183 = COMPLEX EXPON. MULTIPLY BY REAL
U     00A34 001E0000 F-00392          ADDR     C1120S(R7,1)
      00A36 001021FC (00394)          ADDR     CSPUSNOS(,1,0)
                     (00395) *
U     00A38 001F0000 F-00393          ADDR     VPOLYS(R7,1)         184 = VECTOR POLYNOMIAL
U     00A3A 001F0000 F-00396          ADDR     V2116AS(R7,1)
U     00A3C 00100000 F-00397          ADDR     SHMSVPLY(,1,0)
                     (00399) *
U     00A3E 001E0000 F-00398          ADDR     VRAMPS(R7,1)         185 = VECTOR RAMP
      00A40 001F4122 (00401)          ADDR     V1124S(R7,1)
      00A42 001021FC (00402)          ADDR     CSPUSNOS(,1,0)
                     (00403) *
      00A44 001F4020 (00404)          ADDR     VCOSS(R7,1)          186 = VECTOR COSINE
      00A46 001E41F6 (00405)          ADDR     V2134BS(R7,1)
      00A48 001021FC (00406)          ADDR     CSPUSNOS(,1,0)
                     (00407) *
U     00A4A 001F0000 F-00400          ADDR     VPOWS(R7,1)          187 = VECTOR POWER
U     00A4C 001F0000 F-00408          ADDR     V3112AS(R7,1)
      00A4E 001021FC (00410)          ADDR     CSPUSNOS(,1,0)
                     (00411) *
      00A50 001E0000 (00412)          ADDR     APUSNULL(R7,1)       188 = UNASSIGNED
      00A52 001E0000 (00413)          ADDR     APSSNULL(R7,1)
      00A54 001021FC (00414)          ADDR     CSPUSNOS(,1,0)
                     (00415) *
      00A56 001E0000 (00416)          ADDR     APUSNULL(R7,1)       189 = UNASSIGNED
      00A58 001E0000 (00417)          ADDR     APSSNULL(R7,1)
      00A5A 001021FC (00418)          ADDR     CSPUSNOS(,1,0)
                     (00419) *
      00A5C 001E0000 (00420)          ADDR     APUSNULL(R7,1)       190 = UNASSIGNED
      00A5E 001E0000 (00421)          ADDR     APSSNULL(R7,1)
      00A60 001021FC (00422)          ADDR     CSPUSNOS(,1,0)
                     (00423) *
      00A62 001F0000 (00424)          ADDR     APUSNULL(R7,1)       191 = UNASSIGNED
      00A64 001F0000 (00425)          ADDR     APSSNULL(R7,1)
      00A66 001021FC (00426)          ADDR     CSPUSNOS(,1,0)
                     (00427) *
U     00A68 001E0000 F-00409          ADDR     DDIFFS(R7,1)         192 = DISCRETE DIFFERENTIATION
```

```
U   00A6A 001F0000 F-00428          ADDR   D1113$(R7,1)
    00A6C 001021FC (00430)          ADDR   CSPU$NOS(,1,0)
                   (00431) *
U   00A6E 001F0000 F-00429          ADDR   BINTG$(R7,1)        193 = DISCRETE INTEGRATION
U   00A70 001F0000 F-00432          ADDR   D1113$(R7,1)
    00A72 001021FC (00434)          ADDR   CSPU$NOS(,1,0)
                   (00435) *
U   00A74 001F0000 F-00433          ADDR   DFIL22$(R7,1)       194 = DISCRETE FILTER(2-POLES, 2-ZEROS)
U   00A76 001F0000 F-00436          ADDR   D1186$(R7,1)
    00A78 001021FC (00438)          ADDR   CSPU$NOS(,1,0)
                   (00439) *
U   00A7A 001F0000 F-00437          ADDR   DCVM$(R7,1)         195 = DISCRETE CONVOLUTION MULTIPLY
U   00A7C 001F0000 F-00440          ADDR   D2116$(R7,1)
U   00A7E 00100000 F-00441          ADDR   SHM$DCVM(,1,0)
                   (00443) *
    00A80 001F0000 (00444)          ADDR   APU$NULL(R7,1)      196 = UNASSIGNED
    00A82 001F0000 (00445)          ADDR   APS$NULL(R7,1)
    00A84 001021FC (00446)          ADDR   CSPU$NOS(,1,0)
                   (00447) *
    00A86 001F0000 (00448)          ADDR   APU$NULL(R7,1)      197 = UNASSIGNED
    00A88 001F0000 (00449)          ADDR   APS$NULL(R7,1)
    00A8A 001021FC (00450)          ADDR   CSPU$NOS(,1,0)
                   (00451) *
    00A8C 001F0000 (00452)          ADDR   APU$NULL(R7,1)      198 = UNASSIGNED
    00A8E 001F0000 (00453)          ADDR   APS$NULL(R7,1)
    00A90 001021FC (00454)          ADDR   CSPU$NOS(,1,0)
                   (00455) *
    00A92 001F0000 (00456)          ADDR   APU$NULL(R7,1)      199 = UNASSIGNED
    00A94 001F0000 (00457)          ADDR   APS$NULL(R7,1)
    00A96 001021FC (00458)          ADDR   CSPU$NOS(,1,0)
                   (00459) *
U   00A98 001F0000 F-00442 FSCRSDSP ADDR   FSCR$(R7,1)         200 = FOURIER SCRAMBLE
U   00A9A 001F0000 F-00460          ADDR   F11C$(R7,1)
    00A9C 001021FC (00462)          ADDR   CSPU$NOS(,1,0)
                   (00463) *
U   00A9E 001F0000 F-00461 FRX4SDSP ADDR   FRX4$(R7,1)         201 = FOURIER RADIX-4 TRANSFORM
U   00AA0 001F0000 F-00464          ADDR   F21B$(R7,1)
    00AA2 001021FC (00466)          ADDR   CSPU$NOS(,1,0)
                   (00467) *
U   00AA4 001F0000 F-00465 FEOSSDSP ADDR   FEOS$(R7,1)         202 = FOURIER EVEN-ODD SEPERATE
U   00AA6 001F0000 F-00468          ADDR   F11A$(R7,1)
    00AA8 001021FC (00470)          ADDR   CSPU$NOS(,1,0)
                   (00471) *
U   00AAA 001F0000 F-00469 FSC4SDSP ADDR   FSC4$(R7,1)         203 = FOURIER SCARAMBLE - EVEN PWRS OF 2
```

```
U      00AAC 001F0000 F-00472        ADDR     F11CS(R7,1)
       00AAE 001021FC (00474)        ADDR     CSPUSNOS(,1,0)
                      (00475) *
       00AB0 001F0000 (00476)        ADDR     APUSNULL(R7,1)       204 = UNASSIGNED
       00AB2 001F0000 (00477)        ADDR     APSSNULL(R7,1)
       00AB4 001021FC (00478)        ADDR     CSPUSNOS(,1,0)
                      (00479) *
       00AB6 001F0000 (00480)        ADDR     APUSNULL(R7,1)       205 = UNASSIGNED
       00AB8 001E0000 (00481)        ADDR     APSSNULL(R7,1)
       00ABA 001021FC (00482)        ADDR     CSPUSNOS(,1,0)
                      (00483) *
       00ABC 001F0000 (00484)        ADDR     APUSNULL(R7,1)       206 = UNASSIGNED
       00ABE 001F0000 (00485)        ADDR     APSSNULL(R7,1)
       00AC0 001021FC (00486)        ADDR     CSPUSNOS(,1,0)
                      (00487) *
       00AC2 001E0000 (00488)        ADDR     APUSNULL(R7,1)       207 = UNASSIGNED
       00AC4 001F0000 (00489)        ADDR     APSSNULL(R7,1)
       00AC6 001021FC (00490)        ADDR     CSPUSNOS(,1,0)
                      (00491) *
       00AC8 001F0000 (00492)        ADDR     APUSNULL(R7,1)       208 = FOWARD FFT(COMPLEX-COMPLEX)
       00ACA 001E0000 (00493)        ADDR     APSSNULL(R7,1)
U      00ACC 00100000 F-00473        ADDR     FF3SSSM(,1,0)
                      (00495) *
       00ACE 001F0000 (00496)        ADDR     APUSNULL(R7,1)       209 = INVERSE FFT(COMPLEX-COMPLEX)
       00AD0 001E0000 (00497)        ADDR     APSSNULL(R7,1)
U      00AD2 00100000 F-00494        ADDR     IFF3SSSM(,1,0)
                      (00499) *
       00AD4 001F0000 (00500)        ADDR     APUSNULL(R7,1)       210 = FOWARD FFT(REAL-COMPLEX)
       00AD6 001F0000 (00501)        ADDR     APSSNULL(R7,1)
U      00AD8 00100000 F-00498        ADDR     RF3SSSM(,1,0)
                      (00503) *
       00ADA 001F0000 (00504)        ADDR     APUSNULL(R7,1)       211 = INVERSE FFT(REAL-COMPLEX)
       00ADC 001E0000 (00505)        ADDR     APSSNULL(R7,1)
U      00ADE 00100000 E-00502        ADDR     IRF3SSSM(,1,0)
                      (00507) *
       00AE0 001E0000 (00508)        ADDR     APUSNULL(R7,1)       212 = UNASSIGNED
       00AE2 001F0000 (00509)        ADDR     APSSNULL(R7,1)
       00AE4 001021FC (00510)        ADDR     CSPUSNOS(,1,0)
                      (00511) *
       00AE6 001E0000 (00512)        ADDR     APUSNULL(R7,1)       213 = UNASSIGNED
       00AE8 001F0000 (00513)        ADDR     APSSNULL(R7,1)
       00AEA 001021FC (00514)        ADDR     CSPUSNOS(,1,0)
                      (00515) *
       00AEC 001E0000 (00516)        ADDR     APUSNULL(R7,1)       214 = UNASSIGNED
```

```
00AEE 001F0000 (00517)          ADDR     APSSNULL(R7,1)
00AF0 001021FC (00518)          ADDR     CSPUSNOS(,1,0)
               (00519) *
00AF2 001F0000 (00520)          ADDR     APUSNULL(R7,1)       215 = UNASSIGNED
00AF4 001F0000 (00521)          ADDR     APSSNULL(R7,1)
00AF6 001021FC (00522)          ADDR     CSPUSNOS(,1,0)
               (00523) *
00AF8 001F0000 (00524)          ADDR     APUSNULL(R7,1)       216 = UNASSIGNED
00AFA 001F0000 (00525)          ADDR     APSSNULL(R7,1)
00AFC 001021FC (00526)          ADDR     CSPUSNOS(,1,0)
               (00527) *
00AFE 001F0000 (00528)          ADDR     APUSNULL(R7,1)       217 = UNASSIGNED
00B00 001F0000 (00529)          ADDR     APSSNULL(R7,1)
00B02 001021FC (00530)          ADDR     CSPUSNOS(,1,0)
               (00531) *
00B04 001F0000 (00532)          ADDR     APUSNULL(R7,1)       218 = UNASSIGNED
00B06 001F0000 (00533)          ADDR     APSSNULL(R7,1)
00B08 001021FC (00534)          ADDR     CSPUSNOS(,1,0)
               (00535) *
00B0A 001F0000 (00536)          ADDR     APUSNULL(R7,1)       219 = UNASSIGNED
00B0C 001F0000 (00537)          ADDR     APSSNULL(R7,1)
00B0E 001021FC (00538)          ADDR     CSPUSNOS(,1,0)
               (00539) *
00B10 001F0000 (00540)          ADDR     APUSNULL(R7,1)       220 = UNASSIGNED
00B12 001F0000 (00541)          ADDR     APSSNULL(R7,1)
00B14 001021FC (00542)          ADDR     CSPUSNOS(,1,0)
               (00543) *
00B16 001F0000 (00544)          ADDR     APUSNULL(R7,1)       221 = UNASSIGNED
00B18 001E0000 (00545)          ADDR     APSSNULL(R7,1)
00B1A 001021FC (00546)          ADDR     CSPUSNOS(,1,0)
               (00547) *
00B1C 001F0000 (00548)          ADDR     APUSNULL(R7,1)       222 = UNASSIGNED
00B1E 001F0000 (00549)          ADDR     APSSNULL(R7,1)
00B20 001021FC (00550)          ADDR     CSPUSNOS(,1,0)
               (00551) *
00B22 001E0000 (00552)          ADDR     APUSNULL(R7,1)       223 = UNASSIGNED
00B24 001E0000 (00553)          ADDR     APSSNULL(R7,1)
00B26 001021FC (00554)          ADDR     CSPUSNOS(,1,0)
               (00555) *
00B28 001F0000 (00556)          ADDR     APUSNULL(R7,1)       224 = UNASSIGNED
00B2A 001E0000 (00557)          ADDR     APSSNULL(R7,1)
00B2C 001021FC (00558)          ADDR     CSPUSNOS(,1,0)
               (00559) *
00B2E 001F0000 (00560)          ADDR     APUSNULL(R7,1)       225 = UNASSIGNED
```

```
00B30 001F0000 (00561)         ADDR     APS$NULL(R7,1)
00B32 001021FC (00562)         ADDR     CSPU$NOS(,1,0)
               (00563) *
00B34 001F0000 (00564)         ADDR     APU$NULL(R7,1)       226 = UNASSIGNED
00B36 001F0000 (00565)         ADDR     APS$NULL(R7,1)
00B38 001021FC (00566)         ADDR     CSPU$NOS(,1,0)
               (00567) *
00B3A 001F0000 (00568)         ADDR     APU$NULL(R7,1)       227 = UNASSIGNED
00B3C 001F0000 (00569)         ADDR     APS$NULL(R7,1)
00B3E 001021FC (00570)         ADDR     CSPU$NOS(,1,0)
               (00571) *
00B40 001F0000 (00572)         ADDR     APU$NULL(R7,1)       228 = UNASSIGNED
00B42 001F0000 (00573)         ADDR     APS$NULL(R7,1)
00B44 001021FC (00574)         ADDR     CSPU$NOS(,1,0)
               (00575) *
00B46 001F0000 (00576)         ADDR     APU$NULL(R7,1)       229 = UNASSIGNED
00B48 001F0000 (00577)         ADDR     APS$NULL(R7,1)
00B4A 001021FC (00578)         ADDR     CSPU$NOS(,1,0)
               (00579) *
00B4C 001F0000 (00580)         ADDR     APU$NULL(R7,1)       230 = UNASSIGNED
00B4E 001F0000 (00581)         ADDR     APS$NULL(R7,1)
00B50 001021FC (00582)         ADDR     CSPU$NOS(,1,0)
               (00583) *
00B52 001F0000 (00584)         ADDR     APU$NULL(R7,1)       231 = UNASSIGNED
00B54 001F0000 (00585)         ADDR     APS$NULL(R7,1)
00B56 001021FC (00586)         ADDR     CSPU$NOS(,1,0)
               (00587) *
00B58 001F0000 (00588)         ADDR     APU$NULL(R7,1)       232 = UNASSIGNED
00B5A 001F0000 (00589)         ADDR     APS$NULL(R7,1)
00B5C 001021FC (00590)         ADDR     CSPU$NOS(,1,0)
               (00591) *
00B5E 001F0000 (00592)         ADDR     APU$NULL(R7,1)       233 = UNASSIGNED
00B60 001F0000 (00593)         ADDR     APS$NULL(R7,1)
00B62 001021FC (00594)         ADDR     CSPU$NOS(,1,0)
               (00595) *
00B64 001F0000 (00596)         ADDR     APU$NULL(R7,1)       234 = UNASSIGNED
00B66 001F0000 (00597)         ADDR     APS$NULL(R7,1)
00B68 001021FC (00598)         ADDR     CSPU$NOS(,1,0)
               (00599) *
00B6A 001F0000 (00600)         ADDR     APU$NULL(R7,1)       235 = UNASSIGNED
00B6C 001F0000 (00601)         ADDR     APS$NULL(R7,1)
00B6E 001021FC (00602)         ADDR     CSPU$NOS(,1,0)
               (00603) *
00B70 001F0000 (00604)         ADDR     APU$NULL(R7,1)       236 = UNASSIGNED
```

```
00B72 001F0000 (00605)          ADDR     APS$NULL(R7,1)
00B74 001021FC (00606)          ADDR     CSPU$NOS(,1,0)
               (00607) *
00B76 001F0000 (00608)          ADDR     APU$NULL(R7,1)       237 = UNASSIGNED
00B78 001F0000 (00609)          ADDR     APS$NULL(R7,1)
00B7A 001021FC (00610)          ADDR     CSPU$NOS(,1,0)
               (00611) *
00B7C 001F0000 (00612)          ADDR     APU$NULL(R7,1)       238 = UNASSIGNED
00B7E 001F0000 (00613)          ADDR     APS$NULL(R7,1)
00B80 001021FC (00614)          ADDR     CSPU$NOS(,1,0)
               (00615) *
00B82 001F0000 (00616)          ADDR     APU$NULL(R7,1)       239 = UNASSIGNED
00B84 001F0000 (00617)          ADDR     APS$NULL(R7,1)
00B86 001021FC (00618)          ADDR     CSPU$NOS(,1,0)
               (00619) *
00B88 001F0000 (00620)          ADDR     APU$NULL(R7,1)       240 = UNASSIGNED
00B8A 001F0000 (00621)          ADDR     APS$NULL(R7,1)
00B8C 001021FC (00622)          ADDR     CSPU$NOS(,1,0)
               (00623) *
00B8E 001F0000 (00624)          ADDR     APU$NULL(R7,1)       241 = UNASSIGNED
00B90 001F0000 (00625)          ADDR     APS$NULL(R7,1)
00B92 001021FC (00626)          ADDR     CSPU$NOS(,1,0)
               (00627) *
00B94 001F0000 (00628)          ADDR     APU$NULL(R7,1)       242 = UNASSIGNED
00B96 001F0000 (00629)          ADDR     APS$NULL(R7,1)
00B98 001021FC (00630)          ADDR     CSPU$NOS(,1,0)
               (00631) *
00B9A 001F0000 (00632)          ADDR     APU$NULL(R7,1)       243 = UNASSIGNED
00B9C 001F0000 (00633)          ADDR     APS$NULL(R7,1)
00B9E 001021FC (00634)          ADDR     CSPU$NOS(,1,0)
               (00635) *
00BA0 001F0000 (00636)          ADDR     APU$NULL(R7,1)       244 = UNASSIGNED
00BA2 001F0000 (00637)          ADDR     APS$NULL(R7,1)
00BA4 001021FC (00638)          ADDR     CSPU$NOS(,1,0)
               (00639) *
00BA6 001F0000 (00640)          ADDR     APU$NULL(R7,1)       245 = UNASSIGNED
00BA8 001F0000 (00641)          ADDR     APS$NULL(R7,1)
00BAA 001021FC (00642)          ADDR     CSPU$NOS(,1,0)
               (00643) *
00BAC 001F0000 (00644)          ADDR     APU$NULL(R7,1)       246 = UNASSIGNED
00BAE 001F0000 (00645)          ADDR     APS$NULL(R7,1)
00BB0 001021FC (00646)          ADDR     CSPU$NOS(,1,0)
               (00647) *
00BB2 001F0000 (00648)          ADDR     APU$NULL(R7,1)       247 = UNASSIGNED
```

```
00BB4 001F0000 (00649)          ADDR     APSSNULL(R7,1)
00BB6 001021FC (00650)          ADDR     CSPUSNOS(,1,0)
               (00651) *
00BB8 001F0000 (00652)          ADDR     APUSNULL(R7,1)       248 = UNASSIGNED
00BBA 001F0000 (00653)          ADDR     APSSNULL(R7,1)
00BBC 001021FC (00654)          ADDR     CSPUSNOS(,1,0)
               (00655) *
00BBE 001F0000 (00656)          ADDR     APUSNULL(R7,1)       249 = UNASSIGNED
00BC0 001F0000 (00657)          ADDR     APSSNULL(R7,1)
00BC2 001021FC (00658)          ADDR     CSPUSNOS(,1,0)
               (00659) *
00BC4 001F0000 (00660)          ADDR     APUSNULL(R7,1)       250 = UNASSIGNED
00BC6 001F0000 (00661)          ADDR     APSSNULL(R7,1)
00BC8 001021FC (00662)          ADDR     CSPUSNOS(,1,0)
               (00663) *
00BCA 001F0000 (00664)          ADDR     APUSNULL(R7,1)       251 = UNASSIGNED
00BCC 001F0000 (00665)          ADDR     APSSNULL(R7,1)
00BCE 001021FC (00666)          ADDR     CSPUSNOS(,1,0)
               (00667) *
00BD0 001F0000 (00668)          ADDR     APUSNULL(R7,1)       252 = UNASSIGNED
00BD2 001F0000 (00669)          ADDR     APSSNULL(R7,1)
00BD4 001021FC (00670)          ADDR     CSPUSNOS(,1,0)
               (00671) *
00BD6 001F0000 (00672)          ADDR     APUSNULL(R7,1)       253 = UNASSIGNED
00BD8 001F0000 (00673)          ADDR     APSSNULL(R7,1)
00BDA 001021FC (00674)          ADDR     CSPUSNOS(,1,0)
               (00675) *
00BDC 001F0000 (00676)          ADDR     APUSNULL(R7,1)       254 = UNASSIGNED
00BDE 001F0000 (00677)          ADDR     APSSNULL(R7,1)
00BE0 001021FC (00678)          ADDR     CSPUSNOS(,1,0)
               (00679) *
00BE2 001F0000 (00680)          ADDR     APUSNULL(R7,1)       255 = UNASSIGNED
00BE4 001F0000 (00681)          ADDR     APSSNULL(R7,1)
00BE6 001021FC (00682)          ADDR     CSPUSNOS(,1,0)
               (00683) *
      00000942 (00684) #L=AFDTSORG+3*2*(143-128)                FCB #143
00942 001E4666 (00685)            ADDR VMOVS(R7,1)
00944 001E4122 (00686)            ADDR V1124S(R7,1)
00946 001021FC (00687)            ADDR CSPUSNOS(,1,0)
               (00688) *
```

```
                 (00689) 'DISPATCH TABLE ENTRIES FOR FFTN, FFTNB, FFTNI, FFTNIB
                 (00690) ;
                 (00691) ; FFTN AND FFTNI HAVE FULL BINDING
                 (00692) ; ''TNB AND FFTNIB ONLY BIND YBASE AND UBASE
                 (00693) ;
        000000CC (00694)          FFTN$=204
        000000CD (00695)          FFTNB$=205
        000000CE (00696)          FFTNI$=206
        000000CF (00697)          FFTNIB$=207
                 (00698) ;
                 (00699) ;
                 (00700) ;
        00000AB0 (00701) #L=AFDTSORG+3*WS*(FFTN$-128)            FCB #204
00AB0 001F43A6 (00702)          ADDR CSM2$(R7, 1)
00AB2 001F4564 (00703)          ADDR CSM$(R7, 1)
00AB4 00104276 (00704)          ADDR FFTSSET(, 1, 0)
                 (00705) ;
        00000AB6 (00706) #L=AFDTSORG+3*WS*(FFTNB$-128)           FCB #205
00AB6 001F43A6 (00707)          ADDR CSM2$(R7, 1)
00AB8 001F4564 (00708)          ADDR CSM$(R7, 1)
00ABA 00104352 (00709)          ADDR SBMSFFT(, 1, 0)
                 (00710) ;
        00000ABC (00711) #L=AFDTSORG+3*WS*(FFTNI$-128)           FCB #206
00ABC 001E4482 (00712)          ADDR ISM2$(R7, 1)
00ABE 001E4564 (00713)          ADDR CSM$(R7, 1)
00AC0 00104276 (00714)          ADDR FFTSSET(, 1, 0)
                 (00715) ;
        00000AC2 (00716) #L=AFDTSORG+3*WS*(FFTNIB$-128)          FCB #207
00AC2 001E4482 (00717)          ADDR ISM2$(R7, 1)
00AC4 001E4564 (00718)          ADDR CSM$(R7, 1)
00AC6 00104352 (00719)          ADDR SBMSFFT(, 1, 0)
                 (00720) *
                 (00721) *
        00004000 (00722)          #L = START                     SET PC TO START LOCATION FOR MODULE
```

```
                        (00723) *       APU3-VSMA1       VECTOR SCALAR MULTIPLY-ADD
                        (00724) *                                  1 VECTOR BINDS TO APS3-V1124
                        (00725) *
                        (00726) *
                        (00727) *
                        (00728) *
                        (00729) *       EQ. Y=A*U+B
                        (00730) *
                        (00731)         EVEN                       START ON WORD BOUNDARY
    04000 0000          (00732)         DATA     SMA1SSA           START ADDRESS
    04001 000F          (00733)         DATA     SMA1SSZ           SIZE
                        (00734) *
                        (00735) VSMA1S BEGIN APU(VSMA1)            START OF APU MODULE
          00000000      (00736)         *A=0                       SET START ADDRESS
                        (00737) *
A00 04002 0BEF0BEF      (00738) SMA1SSA MOV(IQA,M7)                M7=A
A01 04004 0BF70BF7      (00739)         MOV(IQA,A7)                A7=B
                        (00740) *
          00000002      (00741) *1=*A
A02 04006 0BFB0000      (00742)         MOV(IQA,M0) \ NOP          M0=UE
A03 04008 00000BFB      (00743)         NOP \ MOV(IQA,M0)
A04 0400A 84718471      (00744)         MOV(A1),MUL(M0,M7)         A1=PO', PE=UE*A
A05 0400C 473C473C      (00745)         MOV(OO),ADD(A1,A7)         OUT=RE', RO'=PO'+B
A06 0400E 0BF90000      (00746)         MOV(IQA,M1) \ NOP          M1=UO
A07 04010 00000BF9      (00747)         NOP \ MOV(IQA,M1)
A08 04012 84F084F0      (00748)         MOV(AO),MUL(M1,M7)         AO=PE,PO=UO*A
A09 04014 471C471C      (00749)         MOV(OO),ADD(AO,A7)         OUT=RO',RE=PE+B
A0A 04016 901C0002      (00750)         JUMPC(*1,EO)
                        (00751) *
A0B 04018 20320000      (00752)         CLEAR(RA) \ NOP         HALT
                        (00753) *
A0C 0401A 00000000      (00754)         NOP
A0D 0401C 10000000      (00755)        JUMP(0)
                        (00756) *
          0000000E      (00757)         SMA1SSZ=*A-SMA1SSA
                        (00758) *
    0401E               (00759)         END SMA1SSZ
                        (00760)         EVEN
```

```
                         (00761) *      APU3-VCOS        VECTOR COSINE
                         (00762) *
                         (00763) *      BINDS TO APS3-V2134B
                         (00764) *
                         (00765) *      EQ. Y(K) = U(K) * COS(A*V(K)+B*K+C)
                         (00766) *
                         (00767)        EVEN                          START ON WORD BOUNDARY
        0401F 0000       (00768)        DATA VCOSSSA                  START ADDRESS
        0401F 0065       (00769)        DATA VCOSSSZ                  SIZE
                         (00770) *
                         (00771) VCOS$  BEGIN APU(VCOS)               START OF APU MODULE
              00000000   (00772)        #A=0                          SET START ADDRESS
                         (00773) *
                         (00774) *
                         (00775) *      INITIALIZATION PLACES B+C WITHIN RANGE ZERO TO ONE.
                         (00776) *
A00 04020 16801680 (00777) VCOSSSA K(1)                        B=1
A01 04022 16751675 (00778)      MOV(A5),K(.5)                  A5=1,B=1/2
A02 04024 16541654 (00779)      MOV(A4),K(.25)                 A4=1/2,B=1/4
A03 04026 08F708F7 (00780)      MOV(IQA,A7)                    A7=B
A04 04028 02F302F3 (00781)      MOV(A3),B(A7)                  A3=1/4,B=B
                   (00782) *
                   (00783) *  DECREMENTATION LOOP ON B
                   (00784) *
A05 0402A 08970897 (00785) #1   MOV(B,A7)                      A7=B
A06 0402C 911E0009 (00786)      JUMPS(VC$2,T1)                 GOTO #2 IF B<0
                   (00787) *
A07 0402E 40E040E0 (00788)      SUB(A7,A5)                     B=B-1
A08 04030 10000005 (00789)      JUMP(#1)                       REPEAT DECR PROCESS
                   (00790) *
                   (00791) *  INCREMENTATION LOOP ON B
                   (00792) *
A09 04032 08900890 (00793) VC$2 MOV(B,A0)                      A0=B(NEG)
A0A 04034 45E745E7 (00794)      MOV(A7),ADD(A7,A5)             A7=B(NEG),B=B+1
A0B 04036 08800880 (00795)      MOV(B,NULL)                    COMPLETE T1 FLAG UPDATE
A0C 04038 911E0009 (00796)      JUMPS(VC$2,T1)                 REPEAT IF B+1 NEGATIVE
                   (00797) *
                   (00798) *  CONTINUE FOR A7=B, A0=B,WHERE -1<B<0
                   (00799) *
A0D 0403A 47E047E0 (00800)      ADD(A7,A7)                     B=-2<2B<0
                   (00801) *
A0E 0403C 45E745E7 (00802) VCSB   MOV(A7),ADD(A7,A5)           A7=-2<2B<0;B=-1<2B<1
A0F 0403F 08800880 (00803)        MOV(B,NULL)                  COMPLETE T1 UPDATE
A10 04040 911F000E (00804)        JUMPS(VCSB,T1)               REPEAT IF B=-1<2B<0
```

```
                                  *
                                  *CONTINUE WITH A7=2B WHERE -1<2B<0
                                  *
A11 04042 08F608F6                     MOV(IOA,A6)              A6=C
A12 04044 02C002C0                     R(A6)                    R=C
                                  *
                                  *  DECREMENTATION LOOP ON C
                                  *
A13 04046 08960896                VCS3 MOV(R,A6)                A6=C
A14 04048 911E0017                     JUMPS(VCS4,T1)           GO TO #4 OF C<0
                                  *
A15 0404A 4DC04DC0                     SUB(A6,A5)               R=C-1
A16 0404C 10000013                     JUMP(VCS3)               REPEAT DECR PROCESS
                                  *
                                  *  INCREMENTATION LOOP ON C
                                  *
A17 0404E 45D645D6                VCS4 MOV(A6),ADD(A6,A5)       A6=C, R=C+1
A18 04050 08800880                     MOV(R,NULL)              COMPLETE T1 FLAG UPDATE
A19 04052 911E0017                     JUMPS(VCS4,T1)           REPEAT IF C+1 NEGATIVE
                                  *
                                  *  CONTINUE FOR A6=C, WHERE -1<C<0
                                  *
A1A 04054 4FC04FC0                VCS5 SUB(A6,A7)\SUB(A6,A0)    R=-1<C-2B<1;R=-1<C-B<1
A1B 04056 5DD65DD6                     MOV(A6),ADDLT(A6,A5)     A6=C-2B,C-B;R=0<C-2B<1,R=0<C-B<1
A1C 04058 08960896                     MOV(R,A6)                A6=0<C-2B<1,0<C-B<1;
                                  *
                                  *
                                  * END OF INITIALIZATION*
                                  *
                                  *
                                  *  INPUT/OUTPUT PARAMETERS AND COMPUTE ARGUMENT OF COSINE
                                  *    ENTER LOOP WITH K=0
                                  *
A1D 0405A 47C847C8                VCS6 MOV(M0),ADD(A6,A7)       M0=R23'; R1=(4K+1)B+C
A1E 0405C 84328432                     MOV(A2),MUL(M0,M5)       A2=P12'; P13'=R23'*U(4K-4), R23'*U(4K-3)
A1F 0405E 08FF08FF                     MOV(IOA,M7)              M7=A
A20 04060 5D105D10                     MOV(A0),ADDLT(A0,A5)     A0=R1, R2=R1+T(1)=0<-((4KK)B+C),((4K+1)B+C)<1
A21 04062 08EB0000                     MOV(IOA,M3) \ NOP         M3=V(2K)
A22 04064 000008EB                     NOP \ MOV(IOA,M3)         M3=V(4K+1)
A23 04066 42304230                     MOV(A0),ADD(A1,A2)       A0=R2,R24'=C0+P12'
A24 04068 85ED85ED                     MOV(M5),MUL(M3,M7)       M5=P13'; P1=V(4K)*A,V(4K+1)*A
A25 0406A 470C470C                     MOV(M4),ADD(A0,A7)       M4=R24', R3=R2+B(=(4K+2)B+C),(=(4K+3)B+C)
A26 0406C 5DD65DD6                     MOV(A6),ADDLT(A6,A5)     A6=R3, R4=R3+T(1)=0<-((4K+2)B+C),((4K+3)B+C)<1
A27 0406E 84918491                     MOV(A1),MUL(M1,M4)       A1=P1; P14'=U(4K+2)*R24'U(4K+3)*R24'
```

```
A28 04070 4C164C16 (00849)      MOV(A6),SUB(A0,A4)      A6=R4,R5=R2-1/2 (-1/2<R5<1/2)
A29 04072 08F80000 (00850)      MOV(IOA,M3) \ NOP       M3=V(2K+1)
A2A 04074 000008F8 (00851)      NOP \ MOV(IOA,M3)       M3=V(4K+3)
A2B 04076 41104110 (00852)      MOV(A0),ADD(A0,A1)      A0=R5, R6=R2+P1
A2C 04078 85F985F9 (00853)      MOV(M1),MUL(M3,M7)      M1=P14'; P2=V(4K+2)*A,V(4K+3)*A
A2D 0407A 5D105D10 (00854)      MOV(A0),ADDLT(A0,A5)    A0=R6, R7=R6+T(1)
A2E 0407C 4C104C10 (00855)      MOV(A0),SUB(A0,A4)      A0=R7, R8=R7-1/2 (=CARG(4K)),(=CARG(4K+1))
A2F 0407E 85318531 (00856)      MOV(A1),MUL(M2,M5)      A1=P2;P15'=Y(4K+4),Y(4K-3)
A30 04080 12101210 (00857)      MOV(A0),ABS(A0)         A0=R3, R=ABS(R8) 0<CARG(2K)21/2
                   (00858) *                                0<CARG(4K+1)<1/2
A31 04082 48704870 (00859)      MOV(A0),SUB(A3,A0)      A0=R9,R10=1/4,-R9, -1/4<SARG(4K)<1/4
                   (00860) *                                -1/4<SARG(4K+1)<1/4
A32 04084 84DC84DC (00861)      MOV(OQ),MUL(M1,M6)      OQ=Y(4K-4),Y(4K-3);P6=Y(4K-2),Y(4K-1)
A33 04086 08RA08RA (00862)      MOV(R,M2)               M2=SARG(4K)(=X(4K)),SARG(4K+1)(=X(4K+1))
A34 04088 4CCF4CCF (00863)      MOV(M7),SUB(A6,A4)      M7=X(4K),X(4K+1);,R11=R4-1/2
A35 0408A 41104110 (00864)      MOV(A0),ADD(A0,A1)      A0=R11, R12=R11+P2
A36 0408C 857C857C (00865)      MOV(OQ),MUL(M2,M7)      OQ=Y(4K-2),Y(4K-1);P3=X(4K)^2,X(4K+1)^2
A37 0408E 911C0061 (00866)      JUMPS(VCS7,E0)          EXIT
                   (00867) *
                   (00868) *
                   (00869) *  FETCH COEFFIECIENTS AND EVALUATE POLYNOMINAL APPROX.
                   (00870) *  VIA HORNER METHOD
                   (00871) *
                   (00872) *
A38 04090 5D105D10 (00873)      MOV(A0),ADDLT(A0,A5)    A0=R12, R13=R12+T(1)
A39 04092 08C808C8 (00874)      MOV(IQ,M0)              M0=C4
A3A 04094 4C104C10 (00875)      MOV(A0),SUB(A0,A4)      A0=R13, R14=R13-1/2 (=CARG(4K+2)
                   (00876) *                            CARG(4K+3)
A3B 04096 08AF08AF (00877)      MOV(P,M7)               M7=X(4K)^2,X(4K+1)^2
A3C 04098 84608460 (00878)      MUL(M0,M7)              P4=C4*X(4K)^2,C4*X(4K+1)^2
A3D 0409A 12101210 (00879)      MOV(A0),ABS(A0)         A0=R14, R15=ABS(R14) (0<CARG(4K+2)
                   (00880) *                            CARG(4K+3)<1/2)
A3E 0409C 08FC08FC (00881)      MOV(IOA,M4)             M4=C4
A3F 0409E 48704870 (00882)      MOV(A0),SUB(A3,A0)      A0=R15,R16=1/4-R15 (-1/4<SARG(4K+2)
                   (00883) *                            SARG(4K+3)<1/4)
A40 040A0 088F088E (00884)      MOV(R,M6)               M6=SARG(4K+2)(=X(4K+2))
                   (00885) *                            SARG(4K+3)(=X(4K+3))
A41 040A2 088B088B (00886)      MOV(R,M3)               M3=X(4K+2),X(4K+3)
A42 040A4 85D285D2 (00887)      MOV(A2),MUL(M3,M6)      A2=C4*X(4K)^2,C4*X(4K+1)^2,P5=X(4K+2)^2
                   (00888) *                            X(4K+3)^2
A43 040A6 08F008F0 (00889)      MOV(IOA,A0)             A0=C3
A44 040A8 40404040 (00890)      ADD(A2,A0)              R17=C3+C4*X(4K)^2,C3+C4*X(4K+1)^2
A45 040AA 08AB08AB (00891)      MOV(P,M3)               M3=X(4K+2)^2,X(4K+3)^2
A46 040AC 85808580 (00892)      MUL(M3,M4)              P6=C4*X(4K+2)^2,X(4K+3)^2
```

```
A47 040AF 08880888 (00893)        MOV(R,M0)              M0=R7
A48 040B0 08F108F1 (00894)        MOV(IOA,A1)            A1=C2
A49 040B2 84728472 (00895)        MOV(A2),MUL(M0,M7)     A2=P6;P7=R11*X(4K)^2,R11*X(4K+1)^2
A4A 040B4 40404040 (00896)        ADD(A2,A0)             R18=C3+C4*X(4K+2)^2,C3+C4*X(4K+3)^2
A4B 040B6 088C088C (00897)        MOV(R,M4)              M4=R18
A4C 040B8 85928592 (00898)        MOV(A2),MUL(M3,M4)     A2=P7;P8=X(4K+2)^2*R18,X(4K+3)^2*R18
A4D 040BA 41404140 (00899)        ADD(A2,A1)             R19=P7+C2
A4E 040BC 08F008F0 (00900)        MOV(IOA,A0)            A0=C1
A4F 040BE 08880888 (00901)        MOV(R,M0)              M0=R19
A50 040C0 84728472 (00902)        MOV(A2),MUL(M0,M7)     A2=P8;P9=R19*X(4K)^2,R19*X(4K+1)^2
A51 040C2 41404140 (00903)        ADD(A2,A1)             R20=P8+C2
A52 040C4 088C088C (00904)        MOV(R,M4)              M4=R20
A53 040C6 85928592 (00905)        MOV(A2),MUL(M3,M4)     A2=P9;P10=X(4K+2)^2*R20,X(4K+3)^2*R20
A54 040C8 40404040 (00906)        ADD(A2,A0)             R21=P9+C1
A55 040CA 08F108F1 (00907)        MOV(IOA,A1)            A1=C0
A56 040CC 08880888 (00908)        MOV(R,M0)              M0=R21
A57 040CE 84728472 (00909)        MOV(A2),MUL(M0,M7)     A2=P10;P11=R21*X(4K)^2,R21*X(4K+1)^2
A58 040D0 40404040 (00910)        ADD(A2,A0)             R22=P10+C1
A59 040D2 08ED0000 (00911)        MOV(IOA,M5) \ NOP      M5=U(4K)
A5A 040D4 000008ED (00912)        NOP \ MOV(IOA,M5)      M5=U(4K+1)
A5B 040D6 08E90000 (00913)        MOV(IOA,M1) \ NOP      M1=U(4K+1)
A5C 040D8 000008E9 (00914)        NOP \ MOV(IOA,M1)      M1=U(4K+3)
A5D 040DA 088C088C (00915)        MOV(R,M4)              M4=R22
A5E 040DC 85928592 (00916)        MOV(A2),MUL(M3,M4)     A2=P11;P12=X(4K+2)^2*R22,X(4K+3)^2*R22
A5F 040DE 41404140 (00917)        ADD(A2,A1)             R23=P11+C0(=SIN(4K)),C=SIN((4K+1))
A60 040E0 1000001D (00918)        JUMP(VCS6)             GO TO TOP OF LOOP
                   (00919) *
                   (00920) *   EXIT
                   (00921) *
A61 040E2 088C088C (00922) VCS7   MOV(P,00)
A62 040E4 20322032 (00923)        CLEAR(RA)
A63 040E6 00000000 (00924)        NOP
A64 040E8 10000000 (00925)        JUMP(0)
                   (00926) *
          00000065 (00927)        VCOS$SZ=#A-VCOS$SA
                   (00928) *
    040EA          (00929)        END VCOS$SZ
                   (00930)        EVEN
                   (00931) *
                   (00932) *  SPECIAL CONSTANTS TABLE
                   (00933) *
    040EA 13DAFF42 (00934) VCOSCS:      DATA  39.71091766     C4
    040EC A6499942 (00935)              DATA  -76.57499214    C3
    040EE 28CD15C2 (00936)              DATA  81.60223119     C2
```

```
040F0 94ABBC42 (00937)              DATA -41.34167750      C1
040F2 3243F6C1 (00938)              DATA  6.283185272      C0
               (00939) *
```

```
                          '       APU3-VFLT
                          *
                          *  VECTOR VECTOR FLOAT(VFLT, VFLTB)
                          *
                          *  BINDS TO APS3-V1124A FOR VFLT AND V1124C FOR VFLTB
                          *
                          *  EQ.  Y=NORM(U)+B
                          *
                                EVEN                     START ON WORD BOUNDARY
    040F4 0000                  DATA VFLTSSA             START ADDRESS
    040F5 0012                  DATA VFLTSSZ             SIZE
                          *
                          VFLTS BEGIN APU(VFLT)          START OF APU MODULE
          00000000              #A=0
                          *
A00 040F6 08FF08FF        VFLTSSA  MOV(IOA,M7)           M7=A
A01 040F8 08F708F7              MOV(IOA,A7)              A7=B
                          *
          00000002              #1=#A
A02 040FA 08F20000              MOV(IOA,A2) \ NOP        A2=UEE/
A03 040FC 000008F2              NOP \ MOV(IOA,A2)               A2=UEE
A04 040FE 325C325C              MOV(OQ),NORM(A2)         OUT=RE',R=NORM(UEF)=UE
A05 04100 08880888              MOV(R,M0)                M0=UE
A06 04102 84718471              MOV(A1),MUL(M0,M7)       A1=PO',PE=UE*A
A07 04104 47204720              ADD(A1,A7)               RO'=PO'+B
A08 04106 08F30000              MOV(IOA,A3) \ NOP        A3=UOO/
A09 04108 000008F3              NOP \ MOV(IOA,A3)               A3=UOO
A0A 0410A 327C327C              MOV(OQ),NORM(A3)         OUT=RO',R=NORM(UOO)=UO
A0B 0410C 08890889              MOV(R,M1)                M1=UO
A0C 0410E 84F084F0              MOV(A0),MUL(M1,M7)       A0=PE,PO=UO*A
A0D 04110 47004700              ADD(A0,A7)               RE=PE+B
A0E 04112 901C0002              JUMPC(#1,EO)
                          *
A0F 04114 20320000              CLEAR(RA) \ NOP          HALT
                          *
A10 04116 00000000              NOP
A11 04118 10000000              JUMP(0)
                          *
          00000012              VFLTSSZ=#A-VFLTSSA
                          *
    0411A                       END VFLTSSZ
                                EVEN
```

```
                    (00982) '      APS3-V1124
                    (00983) *
                    (00984) *
                    (00985) * INPUT STREAM - SA, SB;UK; FI
                    (00986) * OUTPUT STREAM - D1,D2,D3,D4; YK; EO
                    (00987) *
                    (00988) *
                    (00989) * START OF HEADER BLOCK
                    (00990) *
                    (00991)              EVEN                        START ON WORD BOUNDRY
    0411A 0000416C  (00992)              ADDR V1124S1                PTR TO CONSTR INSTR BLOCK
    0411C 00004126  (00993)              ADDR V1124S+2*V1124SS       PTR TO SCALAR BLOCK
    0411E 0002      (00994)              DATA 2                      NUMBER OF SCALARS
    0411F 0062      (00995)              DATA V1124SZ                MODULE SIZE
    04120 00004164  (00996)              ADDR V1124SA                PTR TO CHAIN ANCHOR
                    (00997)              EVEN                        END OF WORD BOUNDRY
                    (00998) *
                    (00999) *
                    (01000) V1124S       BEGIN APS(V1124)
                    (01001) *
                    (01002) *
                    (01003) * INPUT PROGRAM
                    (01004) *
                    (01005) *
A00 04122 00201240  (01006) #1           JSN(V1124S3,P2)             SET OUTPUT PC
A01 04124 02300030  (01007)              SET(R0)                     TURN ON APU
                    (01008) *
A02 04126 04C20000  (01009) V1124SS      LOAD(BR0,MSS(1),TF)         GEN SA ADDR
A03 04128 06C20000  (01010)              LOAD(BR0,MSS(1),TF)         GEN SB ADDR
A04 0412A 08401000  (01011)              LOAD(BR0,[1])               LOAD VECTOR-U BASE ADDR
A05 0412C 0A600000  (01012)              LOAD(BR2,MSS)               LOAD VECTOR-U SIZE-1
A06 0412E 0C020000  (01013)               SUB(BR0,MSS)               SET BR0 TO U BASE-SPACING
A07 04130 0E290C79  (01014)              ADDL(BR2,1),JUMP(V1124S5)   ENTER LOOP AT TEST
                    (01015) *
                    (01016) * INPUT ADDRESS GENERATION LOOP
                    (01017) *
A08 04132 108A9008  (01018) #2           ADD(BR0,[9],TF)             GEN U-ELEMENT ADDRESS
A09 04134 128A9002  (01019)              ADD(BR0,[9],TF)
A0A 04136 148A9002  (01020)              ADD(BR0,[9],TF)
A0B 04138 168A9002  (01021)              ADD(BR0,[9],TF)
                    (01022) *
A0C 0413A 18290884  (01023) V1124S5      SUBL(BR2,4),JUMPP(#2)       LOOP UNTIL CNTR NEG
                    (01024) *
A0D 0413C 1A2910E8  (01025)              ADDL(BR2,3),JUMPN(V1124S7)  SET UP CLEAN-UP LOOP
```

```
                          (01026) *
 A0F 0413F 1CHA9006 (01027) #6          ADD(BRO,[9],TF)                  GEN LAST 1-3 U-ADDR
 A0F 04140 1F290FB1 (01028)             SUBL(BR2,1),JUMPP(#6)            LOOP UNTIL FINISHED
                          (01029) *
 A10 04142 20200031 (01030) V1124$7     CLEAR(RI)                        HALT INPUT
 A11 04144 22000020 (01031)             NOP(0)
                          (01032) *
                          (01033) *
                          (01034) * OUTPUT PROGRAM
                          (01035) *
 A12 04146 24300032 (01036) V1124$3     SET(RA)                          TURN-ON APU
 A13 04148 26C20794 (01037)             LOAD(BWO,DMYS(1),TF)             GEN DUMMY-1 ADDRESS
 A14 0414A 28C20794 (01038)             LOAD(BWO,DMYS(1),TF)
 A15 0414C 2AC20794 (01039)             LOAD(BWO,DMYS(1),TF)
 A16 0414E 2CC20794 (01040)             LOAD(BWO,DMYS(1),TF)             GEN DUMMY-4 ADDR
 A17 04150 2E400012 (01041)             LOAD(BWO,[0])                    LOAD VECTOR-Y BASE ADDR
 A18 04152 30500000 (01042)             LOAD(BW1,M$S)                    LOAD VECTOR-Y SIZE-1
 A19 04154 32020000 (01043)             SUB(BWO,M$S)                     SET BWO TO Y BASE-SPACING
 A1A 04156 34111F79 (01044)             ADDL(BW1,1),JUMP(V1124$8)        ENTER LOOP AT TEST
                          (01045) *
                          (01046) * OUTPUT ADDRESS GENERATION LOOP
                          (01047) *
 A1B 04158 36BA8008 (01048) #4          ADD(BWO,[8],TF)                  GEN Y-ELEMENT ADDR
 A1C 0415A 38BA8002 (01049)             ADD(BWO,[8],TF)
 A1D 0415C 3ABA8002 (01050)             ADD(BWO,[8],TF)
 A1E 0415E 3CBA8002 (01051)             ADD(BWO,[8],TF)
                          (01052) *
 A1F 04160 3E111BB4 (01053) V1124$8     SUBL(BW1,4),JUMPP(#4)            LOOP UNTIL NEG
                          (01054) *
 A20 04162 401123FB (01055)             ADDL(BW1,3),JUMPN(V1124$0)       SET UP CLEAN-UP LOOP
                          (01056) *
 A21 04164 42BA8006 (01057) #9            ADD(BWO,[8],TF)                GEN LAST 1-3 Y-ADDR
 A22 04166 441121B1 (01058)               SUBL(BW1,1),JUMPP(#9)          LOOP UNTIL FINISHED
                          (01059) *
 A23 04168 46200030 (01060) V1124$0     CLEAR(RO)                        HALT OUTPUT
 A24 0416A 48000020 (01061)             NOP(0)
                          (01062) *
                          (01063) *
           00004164 (01064) V1124$A=#C                                 ASSIGN VALUE TO CHAIN ANCHOR
                          (01065) *
     0416C          (01066)               END #A-1                     END OF MODULE
                          (01067) *
                          (01068) *
                          (01069) * STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
```

```
                  (01070) *
                  (01071) *
0416C 00000000 (01072) V1124SI DATA     12F'0.0'
 ...
                  (01073) *
                  (01074) *
      00000062 (01075) V1124SZ=*L-V1124S                    COMPUTE MODULE SIZE
```

```
*..-... ARITH. MODULES - PROG. #830101.03 MAY 7, 1980
..V1124A        APS PGM FOR VELT

                       (01076) *        APS3-V1124A        APS PGM FOR VELT
                       (01077) *
                       (01078) *
                       (01079) * INPUT STREAM  - SA, SB; UK FIXED LONG; EI
                       (01080) * OUTPUT STREAM - D1,D2,D3,D4; VK; EO
                       (01081) *
                       (01082) *
                       (01083) *
                       (01084) *
                       (01085) * START OF HEADER BLOCK
                       (01086) *
                       (01087)            EVEN                           START ON WORD BOUNDARY
    04184 00004106 (01088)            ADDR V1124AS1                  PTR TO CONSTR INSTR BLOCK
    04186 00004190 (01089)            ADDR V1124AS+2*V1124ASS        PTR TO SCALAR BLOCK
    04188 0002     (01090)            DATA 2                         NUMBER OF SCALARS
    04189 0062     (01091)            DATA V1124ASZ                  MODULE SIZE
    0418A 000041CF (01092)            ADDR V1124ASA                  PTR TO CHAIN ANCHOR
                       (01093)            EVEN                           END OF WORD BOUNDRY
                       (01094) *
                       (01095) *
                       (01096) V1124AS    BEGIN APS(V1124A)
                       (01097) *
                       (01098) *
                       (01099) * INPUT PROGRAM
                       (01100) *
A00 0418C 00201240 (01101) #1         JSN(V1124AS3,P2)               SET OUTPUT PC
A01 0418E 02300030 (01102)            SET(PO)                        TURN ON OUTPUT
                       (01103) *
A02 04190 04C20000 (01104) V1124ASS   LOAD(BR0,MSS(1),TF)            GEN SA ADDRESS - COEF A
A03 04192 06F20000 (01105)            LOAD(BR3,MSS(1),TF)            GEN SB ADDR, SAVE
A04 04194 08401000 (01106)            LOAD(BR0,[1])                  LOAD VECTOR-U BASE ADDR-LONG
A05 04196 0A600000 (01107)            LOAD(BR2,MSS)                  LOAD VECTOR-U SIZE-1
A06 04198 0C020000 (01108)            SUB(BR0,MSS)                   SET BR0 TO U BASE-SPACING
A07 0419A 0F290C79 (01109)            ADDL(BR2,1),JUMP(V1124AS5)     ENTER LOOP AT TEST
                       (01110) *
                       (01111) * INPUT ADDRESS GENERATION LOOP
                       (01112) *
A08 0419C 110A9008 (01113) #2         ADD(BR0,[9],TF)                GEN U-ELEMENT ADDR FIXED-LONG
A09 0419E 130A9002 (01114)            ADD(BR0,[9],TF)
A0A 041A0 150A9002 (01115)            ADD(BR0,[9],TF)
A0B 041A2 170A9002 (01116)            ADD(BR0,[9],TF)
                       (01117) *
A0C 041A4 18290884 (01118) V1124AS5   SUBL(BR2,4),JUMPP(#2)          LOOP UNTIL CNTR NEG
                       (01119) *
```

```
A0D 041A6 1A2910E6 (01120)             ADDL(BR2,3),JUMPN(V1124AS7)  SET UP CLEAN-UP LOOP
                   (01121) *
A0E 041A8 1D0A9006 (01122) #6          ADD(BR0,[9],TF)              GEN LAST 1-3 D-ADDR
A0F 041AA 1F290E81 (01123)             SUBL(BR2,1),JUMPP(#6)        LOOP UNTIL FINISHED
                   (01124) *
A10 041AC 20200031 (01125) V1124AS7    CLEAR(RI)                    HALT INPUT
A11 041AE 22000020 (01126)             NOP(0)
                   (01127) *
                   (01128) * OUTPUT PROGRAM
                   (01129) *
A12 041B0 24300032 (01130) V1124AS3    SET(RA)                      TURN-ON APU
A13 041B2 26C20794 (01131)             LOAD(BW0,DMYS(1),TF)         GEN DUMMY-1 ADDR
A14 041B4 28C20794 (01132)             LOAD(BW0,DMYS(1),TF)         GEN DUMMY-2 ADDR
A15 041B6 2AC20794 (01133)             LOAD(BW0,DMYS(1),TF)
A16 041B8 2CC20794 (01134)             LOAD(BW0,DMYS(1),TF)         GEN DUMMY-4 ADDR
A17 041BA 2E400012 (01135)             LOAD(BW0,[0])                LOAD VECTOR-Y BASE ADDR
A18 041BC 30500000 (01136)             LOAD(BW1,MSS)                LOAD VECTOR-Y SIZE-1
A19 041BE 32020000 (01137)             SUB(BW0,MSS)                 SET BW0,TO YBASE-SPACING
A1A 041C0 34111E79 (01138)             ADDL(BW1,1),JUMP(V1124AS8)   ENTER LOOP AT TEST
                   (01139) *
                   (01140) *
                   (01141) *  OUTPUT GENERATION LOOP
                   (01142) *
A1B 041C2 368AB008 (01143) #4          ADD(BW0,[8],TF)              GEN Y-ELEMENT ADDR
A1C 041C4 388AB002 (01144)             ADD(BW0,[8],TF)
A1D 041C6 3A8AB002 (01145)             ADD(BW0,[8],TF)
A1E 041C8 3C8AB002 (01146)             ADD(BW0,[8],TF)
                   (01147) *
A1F 041CA 3E111884 (01148) V1124AS8    SUBL(BW1,4),JUMPP(#4)        LOOP UNTIL CNTR NEG
                   (01149) *
A20 041CC 401123EB (01150)             ADDL(BW1,3),JUMPN(V1124AS0)  SET UP CLEAN-UP LOOP
                   (01151) *
A21 041CE 428AB006 (01152) #9          ADD(BW0,[8],TF)              GEN LAST 1-3 Y-ADDR
A22 041D0 441121B1 (01153)             SUBL(BW1,1),JUMPP(#9)        LOOP UNTIL FINISHED
                   (01154) *
A23 041D2 46200030 (01155) V1124AS0    CLEAR(RO)                    HALT OUTPUT
A24 041D4 48000020 (01156)             NOP(0)
                   (01157) *
                   (01158) *
                   (01159) *
          000041CE (01160) V1124ASA=#C                              ASSIGN VALUE TO CHAIN ANCHOR
                   (01161) *
    041D6          (01162)             END #A-?                     END OF MODULE
                   (01163) *
```

```
                  (01164) * STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                  (01165) *
                  (01166) *
041D6 00000000 (01167) V1124AS1 DATA     12F'0.0'
 ...
                  (01168) *
                  (01169) *
       00000062 (01170) V1124ASZ=*1-V1124AS                    COMPUTE MODULE SIZE
```

```
                  (01171) *       APS3-V2134B        APS PGM FOR VCOS
                  (01172) *
                  (01173) *
                  (01174) * INPUT STREAM - SB,SC; SA,V(4K),...V(4K+3),C4...C0,U(4K),...U(4K+3);FI
                  (01175) * OUTPUT STREAM - D1,D2,D3,D4; YK; FO
                  (01176) *
                  (01177) *
                  (01178) * START OF HEADER BLOCK
                  (01179) *
                  (01180)           EVEN                         START ON WORD BOUNDRY
    041EE 00004256 (01181)          ADDR V2134BSI                PTR TO CONSTR INSTR BLOCK
    041F0 000041FA (01182)          ADDR V2134BS+2*V2134BSS      PTR TO SCALAR BLOCK
    041F2 0003     (01183)          DATA 3                       NUMBER OF SCALARS
    041F3 0080     (01184)          DATA V2134BSZ                MODULE SIZE
    041F4 0000424F (01185)          ADDR V2134BSA                PTR TO CHAIN ANCHOR
                  (01186)           EVEN                         END OF WORD BOUNDRY
                  (01187) *
                  (01188) *
                  (01189) V2134BS    BEGIN APS(V2134B)
                  (01190) *
                  (01191) *
                  (01192) * INPUT PROGRAM
                  (01193) *
                  (01194) *
A00 041F6 00201040 (01195) #1       JSN(V2134BS3,P2)             SET OUTPUT PC
A01 041F8 02300030 (01196)          SET(R0)                      TURN ON APU
                  (01197) *
A02 041FA 04720000 (01198) V2134BSS LOAD(BR3,MSS(1))             GET SA ADDR SAVE
A03 041FC 06C20000 (01199)          LOAD(BR0,MSS(1),TF)          GEN SB ADDR
A04 041FE 08C20000 (01200)          LOAD(BR0,MSS(1),TF)          GEN SC ADDR
A05 04200 0A310017 (01201)          MOVB(BW3,BR3)                SA ADDR IN BW3
A06 04202 0C401000 (01202)          LOAD(BR0,[1])                LOAD VECTOR-U BASE ADDR
A07 04204 0E600000 (01203)          LOAD(BR2,MSS)                LOAD VECTOR-U SIZE-1
A08 04206 10020000 (01204)          SUB(BR0,MSS)                 SET BR0 TO U BASE-SPACING
A09 04208 12502006 (01205)          LOAD(BR1,[2])                LOAD VECTOR-V BASE ADDR
A0A 0420A 14600000 (01206)           LOAD(BR2,MSS)               LOAD(VECTOR-V SIZE-1
A0B 0420C 16120000 (01207)           SUB(BR1,MSS)                SET BR1 TO V BASE-SPACING
                  (01208) *
                  (01209) *
                  (01210) * INPUT ADDRESS GENERATION LOOP
                  (01211) *
A0C 0420E 18890016 (01212) #2       MOVB(BR3,BW3,TF)             GEN SA ADDR
A0D 04210 1A9AA008 (01213)          ADD(BR1,[10],TF)             GEN V-ELEMENT ADDRESS
A0E 04212 1C9AA002 (01214)          ADD(BR1,[10],TF)
```

```
A0F 04214 1E9AA002 (01215)              ADD(BR1,[10],TF)                 GEN V-ELEMENT ADDR
A10 04216 209AA002 (01216)              ADD(BR1,[10],TF)
A11 04218 22E240FA (01217)              LOAD(BR3,VCOSCS(1),TF)           GEN VCSC4 ADDR
A12 0421A 2489003A (01218)              ADDL(BR3,2,TF)
A13 0421C 2689003A (01219)              ADDL(BR3,2,TF)
A14 0421E 2889003A (01220)              ADDL(BR3,2,TF)
A15 04220 2A89003A (01221)              ADDL(BR3,2,TF)                   GEN VCSCO ADDRESS
A16 04222 2CRA900C (01222)              ADD(BR0,[9],TF)                  GEN U-ELEMENT ADDR
A17 04224 2EBA9002 (01223)              ADD(BR0,[9],TF)
A18 04226 30BA9002 (01224)              ADD(BR0,[9],TF)
A19 04228 32BA9002 (01225)              ADD(BR0,[9],TF)
A1A 0422A 34290C84 (01226)              SUBL(BR2,4),JUMPP(#2)            LOOP UNTIL FINISHED
                   (01227) *
A1B 0422C 36200031 (01228)              CLEAR(R1)
A1C 0422E 38000020 (01229)              NOP(0)
                   (01230) *
                   (01231) * OUTPUT PROGRAM
                   (01232) *
A1D 04230 3A300032 (01233) V2134BS3  SET(RA)                          TURN-ON APU
A1E 04232 3CC20794 (01234)              LOAD(BW0,DMYS(1),TF)             GEN DUMMY-1 ADDR
A1F 04234 3EC20794 (01235)              LOAD(BW0,DMYS(1),TF)
A20 04236 40C20794 (01236)              LOAD(BW0,DMYS(1),TF)
A21 04238 42C20794 (01237)              LOAD(BW0,DMYS(1),TF)
A22 0423A 44400012 (01238)              LOAD(BW0,[0])                    LOAD VECTOR-Y BASE ADDR
A23 0423C 46500000 (01239)              LOAD(BW1,MSS)                    LOAD(VECTOR-Y SIZE-1
A24 0423E 48020000 (01240)              SUB(BW0,MSS)                     SET BW0 TO Y BASE-SPACING
A25 04240 4A112A79 (01241)              ADDL(BW1,1),JUMP(V2134BS8)       ENTER LOOP AT TEST
                   (01242) *
                   (01243) *
                   (01244) *OUTPUT ADDRESS GENERATION LOOP
                   (01245) *
A26 04242 4CBAB00B (01246) #4           ADD(BW0,[8],TF)                  GEN Y-ELEMENT ADDRESS
A27 04244 4EBAB002 (01247)              ADD(BW0,[8],TF)
A28 04246 50BAB002 (01248)              ADD(BW0,[8],TF)
A29 04248 52BAB002 (01249)              ADD(BW0,[8],TF)
                   (01250) *
A2A 0424A 541126B4 (01251) V2134BS8  SUBL(BW1,4),JUMPP(#4)            LOOP UNTIL CNTR NEG
                   (01252) *
A2B 0424C 56112FFB (01253)              ADDL(BW1,3),JUMPN(V2134BS0)      SET UP CLEAN-UP LOOP
                   (01254) *
A2C 0424E 58BAB006 (01255) #9           ADD(BW0,[8],TF)                  GEN LAST 1-3 Y-ADDR
A2D 04250 5A112CB1 (01256)              SUBL(BW1,1),JUMPP(#9)            LOOP UNTIL FINISHED
                   (01257) *
A2E 04252 5C200030 (01258) V2134BS0  CLEAR(R0)                        HALT OUTPUT
```

```
 A2F 04254 5F000020 (01259)           NOP(0)
                    (01260) *
                    (01261) *
                    (01262) *
           0000424F (01263) V2134B$A=#C                             ASSIGN VALUE TO CHAIN ANCHOR
                    (01264) *
     04256          (01265)             END #A-1                    END OF MODULE
                    (01266) *
                    (01267) *
                    (01268) * STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                    (01269) *
                    (01270) *
     04256 00000000 (01271) V2134BS1 DATA     16F'0,0'
      ...
                    (01272) *
                    (01273) *
           00000080 (01274) V2134BSZ=#L-V2134B$                     COMPUTE MODULE SIZE
```

```
(01275) 'MAP-300 EXTENDED ARRAY FUNCTIONS - PROG. #R30102.03 - MAY 7, 1980
(01276) *
(01277) *
(01278) *
```

```
'NOT-IN-PLACE FFT FOR ONE AND TWO DIMENSIONS
*
*
!FAST FOURIER TRANSFORM AND INVERSE TRANSFORM ALGORITHMS
;
; PERFORMS AN N POINT FFT OR IFFT ON DATA STORED IN
; THE U BUFFER USING THE V BUFFER AS THE
; COSINE TABLE AND THE W BUFFER AS A
; WORKING BUFFER AND LEAVES THE RESULT IN THE
; Y BUFFER.  THIS ROUTINE CANNOT BE DONE IN
; PLACE (I.E. W AND U CAN'T BE THE SAME BUF-
; FER).  THE NUMBER OF POINTS IN THE FFT,
; N, MUST BE A POWER OF TWO AND NOT GREATER
; THAN 1024.
;
;
; THE BUFFER DESCRIPTIONS ARE:
;   Y BUFFER (10-39)  COMPLEX 32-BIT FLOATING POINT
;   U BUFFER (10-39)  COMPLEX 32-BIT FLOATING POINT
;   V BUFFER (10-39)  REAL 32-BIT FLOATING POINT
;   W BUFFER (10-39)  COMPACT, COMPLEX 32-BIT FLOATING POINT
;
; THE COSINE TABLE ENTRIES ARE:
;   CT(K)=COS(2*PI*K/CSIZE)
; WHERE CSIZE IS A MULTIPLE OF N
;
;!
**'ENTRY TO SPECIAL BINDING MODULE FOR 2-D FFT SETUP
**;
**; ENTER WITH R1 POINTING TO FCB NUMBER
**;             R2 POINTING TO DISPATCH TABLE
**;
***
***************** BEGIN REPEAT FFT CODE INSERTION ***************
***
***
**RPTFT01 MOVRR   R5,R1                    ; COPY FCB BLOCK POINTER
**        EVEN
**        MOVLM   2,FFTSTYP                ; TAG REPEATED FFT CALL
**        MOVMR   R7,2*HS(R5)              ; GET LOG2(FFT SIZE)
**        ANDIR   R7,$F                    ; FORCE TO BE <= 15
**        MOVRM   R7,LOG2SZ                ; SAVE FOR LATER PERMANENT STORAGE
**        MOVIR   R7,SMPSPC                ; POINT TO PARAMETER STORAGE AREA
**        MOVIR   R6,3                     ; SET LOOP COUNTER FOR 4 TRANSFERS
```

```
**
**RPTFT02 POPXI    R5,R4                  ; LOAD PASSED PARAMETER INTO R4
**        EVEN
**        ANDIR    R4,SF                  ; FORCE ALL TO BE <= 15
**        MOVMR    R3,RPTFT03             ; GET COPY OF SHIFT INSTRUCTION
**        IORRR    R4,R3,SFFF0            ; OR IN SHIFT COUNT
**        MOVRM    R4,RPTFT03             ; STORE SHIFT INSTRUCTION
**        MOVIR    R4,1                   ; INIT R4 TO 2**0
**RPTFT03 LLS      R4,0                   ; COMPUTE 2**N
**        PUSHXI R7,R4                    ; SAVE IN PARAMETER AREA
**        DJP      R6,RPTFT02             ; LOOP OVER ALL 4 PARAMETERS
**        PUSHXI R7,R4                    ; NEED 2 COPIES OF FFT SPACING
**        EVEN
**
**;        COMPUTE MAXSMP = (MAXIMUM SAMPLE INDEX) - 1
**;
**;        MAXSMP = (SAMPLE_SPACING)*(FFT_SIZE - 1) +
**;                        (NUMBER_OF_FFT'S - 1)*(FFT_SPACING)
**;               = SA*(SB-1) + (SC-1)*SD
**;
**        MOVMR    R5,FFTSIZ              ; GET SIZE
**        MOVMR    R7,NFFT                ; GET NUMBER OF FFT'S
**        DECR     R5,1                   ; CORRECT TO (FFTSIZE-1)
**        DECR     R7,1                   ; CORRECT TO (NFFT-1)
**        MOVRM    R7,NFFT                ; SAVE DECREMENTED COUNT FOR LOOPING
**        MULMR    R5,SMPSPC              ; FORM 1ST PRODUCT
**        MULMR    R7,INSPC               ; FORM 2ND PRODUCT
**        ADDRR    R5,R7                  ; FORM SUM
**        LRS      R5,1                   ; COMMON CORRECTION FOR DOUBLING BY MU
**        MOVRM    R5,MAXSMP              ; SAVE FOR LATER
**        HOP      RPTFT04                ; HOP INTO BINDING CODE
**        EVEN
*
*************** END REPEAT FFT CODE INSERTION ***************
*
```

```
                   'ENTRY TO SPECIAL BINDING MODULE FOR VECTOR FFT SETUP
                   ;
                   ; DOES ALL BINDING NOT NEEDED EVERY TIME
                   ; AN FFT IS DONE.
                   ;
                   ; ENTER WITH R1 POINTING TO FCB NUMBER
                   ;            R2 POINTING TO DISPATCH TABLE
                   ;
                   ;
         00004276  FFTSSET = #L
                   **      MOVZM   FFTSTYP                  ; TAG 'MUXED' FFT
04276 F0420002     RPTFT04 MOVMR R4, 2*HS(R1)            GET U BUFFER ID
04278 9A40FF00             ANDIR R4, MSKSLBY1            MASK OUT RIGHT HALF
0427A 3C47                 LRS R4, 7                     SET FOR FULL WORD INDEX
0427B 0800                 EVEN                          SKIP TO EVEN BOUNDARY
0427C F0580687             MOVMR R5, BCTSAT+HS(R4)       R5= U BUFFER ATTRIBUTES
0427F F050439D             MOVRM R5, PWR2S               STORE FOR LATER REFERENCE
                   ;
                   ;BIND ALL USIZE-1'S
                   ;
04280 C0580604             MOVMRL R5, BCTSAD(R4)         R5<=USEP, R6<=USIZE-1
                   *
                   *************** BEGIN REPEAT FFT CODE INSERTION ***************
                   *
                   **      MOVMR   R7,FFTSTYP               ; GET FFT TYPE TAG INDEX
                   **      JMP     @RPTFT05(R7)             ; JUMP TO NEXT SECTION OF CODE
                   ;
                   ** RPTFT05 ADDR    RPTFT07 ( , 1)           ;    'MUXED' FFT
                   **      ADDR    RPTFT06 ( , 1)           ;    REPEATED FFT
                   ;
                   ** RPTFT06 MOVMR   R7,LOG2SZ                ; GET LOG2(FFT SIZE)
                   **      MOVRM   R7,PWR2S                 ; STORE FOR BINDING CODE USAGE
                   **      CMPMR   R6,MAXSMP                ; CHECK INPUT BUFFER SIZE
                   **      JMP     ERRSS,LT                 ; JUMP IF BUFFER TOO SMALL
                   **      MOVMR   R7,INSPC                 ; GET FFT INPUT VECTOR SPACING
                   **      MULRR   R7,R5                    ; SCALE BY BUFFER SAMPLE SPACING
                   **      LRS     R7,1                     ; CORRECT FOR FACTOR OF 2 BY MUL
                   **      MOVRM   R7,INSPC                 ; SAVE FOR USE IN POST-SUPPORT MODULE
                   **      MULMR   R5,SMPSPC                ; COMPUTE INPUT SAMPLE SPACING
                   **      LRS     R5,1                     ; CORRECT FACTOR OF 2 FROM MUL
                   **      EVEN
                   **      MOVMR   R6,FFTSIZ                ; GET CORRECT FFT SIZE
                   **      MOVRR   R7,R6                    ; GET 2ND COPY
                   **      DECR    R7,1                     ; NEED TO BIND (FFT_SIZE - 1) 6 PLACES
```

```
                  (01402) **       MOVRM   R7,CSMS + WS*CR4A03 + HS
                  (01403) **       MOVRM   R7,CSMS + WS*CSML0S + HS
                  (01404) **       MOVRM   R7,CSMS + WS*CR4A3S + HS
                  (01405) **       MOVRM   R7,CSMS + WS*CR4ARS + HS
                  (01406) **       MOVRM   R7,CSMS + WS*CR4A9S + HS
                  (01407) **       MOVRM   R7,CSMS + WS*CR4A13S + HS
                  (01408) **       HOP     RPTFTOR
                  (01409) **       EVEN
                  (01410) *
                  (01411) *************** END REPEAT FFT CODE INSERTION ***************
                  (01412) *
042R2 F06045F7 (01413) RPTFT07 MOVRM R6, CSMS+WS*CR4A03+HS   STORE ALL USIZE-1'S
04284 F0604569 (01414)         MOVRM R6, CSMS+WS*CSML0S+HS   ...
04286 F06045D7 (01415)         MOVRM R6, CSMS+WS*CR4A3S+HS   ...
04288 F0604631 (01416)         MOVRM R6, CSMS+WS*CR4ARS+HS   ...
                  (01417) ;
                  (01418) ;
                  (01419) ;SCALAR A ID IS REALLY THE NUMBER OF MINI-BUFFERS
                  (01420) ;WITHIN THE U BUFFER.
                  (01421) ;FOR EXAMPLE:
                  (01422) ;  IF U BUFFER HAS 1024 COMPLEX POINTS BUT IS
                  (01423) ;  REALLY 4 GROUPS OF 256 POINTS THEN SA ID
                  (01424) ;  SHOULD BE SET TO 4.
                  (01425) ;  IF U BUFFER HAS 1024 POINTS AND IS REALLY
                  (01426) ;  16 GROUPS OF 64 POINTS THEN SA ID SHOULD BE
                  (01427) ;  16.
                  (01428) ;  NORMAL OPERATION WHERE THE U BUFFER CONTAINS
                  (01429) ;  ONE BUFFER TO BE FFT'ED MEANS SA ID SHOULD
                  (01430) ;  BE 1.
                  (01431) ;  SA ID MUST BE A POWER OF 4. (I.E. 1, 4,
                  (01432) ;  16, 64, 256)
                  (01433) ;
0428A F0420001 (01434)         MOVMR R4, HS(R1)              POINT TO SCALAR A ID (SAID)
0428C 9A4000FF (01435)         ANDIR R4, MSKSRBYT            MASK IT OUT
0428E 407C     (01436)         MOVRR R7, R6                  PUT USIZE-1 IN R7
0428F 2808     (01437) XX1S    SRBC 0, R4                    SKIP SHIFTING NEEDED
04290 2003     (01438)         HOP XX2S
04291 3C71     (01439)         LRS     R7,1                    ; DIVIDE BY 2
04292 3C41     (01440)         LRS R4, 1                     AND SHIFT TILL BIT APPEARS
04293 2105     (01441)         HOP XX1S
                  (01442)         EVEN
04294 F0704649 (01443) XX2S    MOVRM R7, CSMS+WS*CR4A9S+HS   STORE FFTSIZE-1
04296 F070460F (01444)         MOVRM R7, CSMS+WS*CR4A13S+HS  ...
                  (01445)
```

```
04298 2661     (01446)          INCR R6, 1                R6<=N(USIZE)
04299 403A     (01447) RPTFTOR  MOVRR R3, R5              R3<=USEP
0429A 485C     (01448)          MULRR R5, R6              R5<=2(USEP)(USIZE)
0429B 3C52     (01449)          LRS R5, 2                 R5<=0.5(USEP)(USIZE)
               (01450) ;
               (01451) ;BIND ALL HU'S
               (01452) ;
0429C 9040439D (01453)          MOVIR   R4,HUSTBL-1          ; POINT TO TABLE OF HU BINDING LOC'S
0429E 304E     (01454) HUSLP    POPXI   R4,R7                ; GET NEXT BINDING ADDR
0429F 0270     (01455)          TEST    R7                   ; CHECK FOR END OF TABLE
042A0 801042AA (01456)          JMP     HUSDN,EQ             ; ZERO MARKS END OF TABLE
042A2 D00F0000 (01457)          CMB     0,0(R7)              ; RESET MSB OF DELTA FIELD
042A4 F05E0001 (01458)          MOVRM   R5,1(R7)             ; STORE LOW 16 BITS
042A6 1810     (01459)          SKPL    NE                   ; IF NOT 0, DELTA IS OK
042A7 D20F0000 (01460)          SMB     0,0(R7)              ; PRODUCT WAS 2**18 BEFORE CORRECTION
042A9 210C     (01461)          HOP     HUSLP                ; LOOP FOR ALL HU BINDING
               (01462) ;
042AA 0250     (01463) HUSDN    TEST    R5                   ; CHECK PRODUCT
042AB 1810     (01464)          SKPL    NE                   ; IF NOT 0, DELTA IS OK
042AC FC510000 (01465)          MOVAR   R5,$10000            ; SET UP CORRECT DELTA FOR 2**18
               (01466)
042AE 3C51     (01467)          LRS R5, 1                 R5<=HU/2 (QU)
042AF 0800     (01468)          EVEN                      TO TO EVEN BOUNDARY
               (01469) ;
               (01470) ;BIND ALL QU'S
               (01471) ;
042B0 F0504577 (01472)          MOVRM R5, CSMS+WS*CSML3S+HS   STORE QU'S
042B2 4E56     (01473)          SUBRR R5, R3              R5<=QU-USEP
042B3 0800     (01474)          EVEN
042B4 F050456F (01475)          MOVRM R5, CSMS+WS*CSML1S+HS   STORE QU-USEP
               (01476) ;
               (01477)          EJECT
```

```
042B6 405C     (01478)          MOVRR R5, R6                 R5<=N
042B7 0800     (01479)          EVEN
042B8 98500003 (01480)          MULIR R5, 3                  R5<=4*(3N/2)
               (01481) ;
               (01482) ;BIND D0-D7'S
               (01483) ;
042BA F0504585 (01484)          MOVRM R5, CSMS+WS*SCRM0+HS   STORE ALL D0'S
042BC F0504589 (01485)          MOVRM R5, CSMS+WS*SCRM2S+HS  ...
042BE F050458D (01486)          MOVRM R5, CSMS+WS*SCRM4S+HS  ...
042C0 F0504591 (01487)          MOVRM R5, CSMS+WS*SCRM6S+HS  ...
042C2 3C51     (01488)          LRS R5, 1                    R5<=D0/2 (D1)
042C3 0800     (01489)          EVEN
042C4 F0504587 (01490)          MOVRM R5, CSMS+WS*SCRM1S+HS  STORE ALL D1'S
042C6 F050458F (01491)          MOVRM R5, CSMS+WS*SCRM5S+HS  ...
042C8 3C51     (01492)          LRS R5, 1                    R5<=D1/2 (D2)
042C9 0800     (01493)          EVEN
042CA F050458B (01494)          MOVRM R5, CSMS+WS*SCRM3S+HS  STORE D2
042CC 3C51     (01495)          LRS R5, 1                    R5<=D2/2 (D3)
042CD 0800     (01496)          EVEN
042CF F0504595 (01497)          MOVRM R5, CSMS+WS*SCRM7S+HS  STORE D3
042D0 3C51     (01498)          LRS R5, 1                    R5<=D3/2 (D4)
042D1 0800     (01499)          EVEN
042D2 F050459B (01500)          MOVRM R5, CSMS+WS*SCRM8S+HS  STORE D4
042D4 3C51     (01501)          LRS R5, 1                    R5<=D4/2 (D5)
042D5 0800     (01502)          EVEN
042D6 F05045A1 (01503)          MOVRM R5, CSMS+WS*SCRM9S+HS  STORE D5
042D8 3C51     (01504)          LRS R5, 1                    R5<=D5/2 (D6)
042D9 0800     (01505)          EVEN
042DA F05045A7 (01506)          MOVRM R5, CSMS+WS*SCRM10S+HS  STORE D6
042DC 3C51     (01507)          LRS R5, 1                    R5<=D6/2 (D7)
042DD 0800     (01508)          EVEN
042DF F05045AB (01509)          MOVRM R5, CSMS+WS*SCRM7+HS   STORE D7
               (01510) ;
               (01511)
               (01512)          EJECT
```

ENTRY TO SPECIAL BINDING MODULE FOR VECTOR FFT SETUP

```
042E0 405C     (01513)        MOVRR R5, R6                  R5<=N (USIZE)
042E1 3A52     (01514)        LLS R5, 2                     R5<=4*N
042E2 2654     (01515)        INCR R5, 4                    R5<=4*N+4
042E3 0800     (01516)        EVEN
               (01517) ;
               (01518)        EJECT
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

5 OF 9

AD
A091663

```
                     ;
                     ;BIND ALL 4N+4 AND 4N+12'S
                     ;
042E4 F05045F9               MOVRM R5, CSMS+WS*CR4A2S+HS    STORE 4N+4'S
042E6 F050462F               MOVPM R5, CSMS+WS*CR4A3+HS     ...
042E8 2658                   INCR R5, 8                     R5<=4*N+12
042F9 0800                   EVEN
042FA F05045B5               MOVRM R5, CSMS+WS*CSMUL+HS     STORE 4N+12
042FC F0420001               MOVMR R4, HS(R1)               LOAD Y BUFFER ID
042FF 9A40FF00               ANDIR R4, MSKSLBYT             MASK LEFT HALF
042F0 3C47                   LRS R4, 7                      CREATE FULL WORD INDEX
042F1 0800                   EVEN
                     ;
                     ;BIND YSEP
                     ;
042F2 C0580604               MOVMRL R5, BCTSAD(R4)          R5<=YSEP, R6<=YSIZE-1
                     *
                     *************** BEGIN REPEAT FFT CODE INSERTION ***************
                     *
                     **      MOVMR   R7,FFTSTYP                ; GET FFT TYPE TAG
                     **      JMP     @RPTFT09(R7)              ; JUMP TO NEXT SECTION OF CODE
                     ;
                     ** RPTFT09 ADDR    RPTFT11( , 1)             ;    'MUXED' FFT
                     **      ADDR    RPTFT10( , 1)             ;    REPEATED FFT
                     ;
                     ** RPTFT10 MOVMR   R7,OUTSPC                 ; GET FFT OUTPUT VECTOR SPACING
                     **      MULRR   R7,R5                     ; SCALE BY BUFFER SAMPLE SPACING
                     **      LRS     R7,1                      ; CORRECT FOR FACTOR OF 2 BY MUL
                     **      MOVRM   R7,OUTSPC                 ; SAVE FOR USE IN POST-SUPPORT
                     **      CMPMR   R6,MAXSMP                 ; CHECK IF OUTPUT VECTOR IS LARGE ENOUGH
                     **      JMP     ERRSS,LT                  ; ERROR IF TOO SMALL
                     **      MULMR   R5,SMPSPC                 ; COMPUTE OUTPUT SAMPLE SPACING
                     **      LRS     R5,1                      ; CORRECT FOR FACTOR OF 2 BY MUL
                     **      EVEN
                     **      MOVRM   R5,CSMS + WS*CR4A12S + HS       ; STORE YSEP
                     **      MOVMR   R6,FFTSIZ                 ; GET SIZE OF OUTPUT VECTOR FOR 1 PASS
                     **      HOP     RPTFT12                   ; CONTINUE BINDING
                     **      EVEN
                     ;
                     **         ERRSIBCR = 29                  ; IMPROPER BUFFER ERROR FLAG
                     ** ERRS6   MOVIR   R7,ERRSIBCR               ; SET IMPROPER BUFFER ERROR
                     **      JMP     ERRORS                    ; GO TO EXEC ERROR PROCESSING
                     *
                     *************** END REPEAT FFT CODE INSERTION ***************
```

ENTRY TO SPECIAL BINDING MODULE FOR VECTOR FFT SETUP

```
                    (01563) *
042F4 F05045F1      (01564) RPTFT11 MOVRM R5, CSMS+WS*CR4A12S+HS   STORE YSEP
042F6 2661          (01565)         INCR R6, 1                     R6<=YSIZE
042F7 485C          (01566) RPTFT12 MULRR R5, R6                   R5<=2*YSIZE*YSEP
042F8 0250          (01567)         TEST    R5                      ;SET CONDITON CODES FROM MULTIPLY
042F9 3C51          (01568)         LRS R5, 1                      R5<=YSIZE*YSEP
                    (01569)         EVEN
                    (01570) ;
                    (01571)         EJECT
```

```
                 (01572) ;
                 (01573) ;BIND YSEP*YSIZE
                 (01574) ;
042FA D00045DF (01575)         CMB     0,CSMS + WS*CR4AOF       ; RESET MSB OF DELTA FILED
042FC 1B30     (01576)         SKPL    GE                       ; LEAVE MSB CLEAR WHEN PRODUCT POS
042FD D20045DF (01577)         SMB     0,CSMS + WS*CR4AOF       ; MSB SHOULD BE SET
042FF 81104306 (01578)         JMP     YSPSZ,NF                 ; IF NOT 0, ALL IS WELL
04301 CC0045DF (01579)         MOVZM   CSMS + WS*CR4AOF + HS    ; PRODUCT IS 2**18 -- SET DELTA = 0
04303 9050B000 (01580)         MOVIR   R5,$8000                 ; YSEP*YSIZE/4 = 2**16
04305 2004     (01581)         HOP     YSPSZ4                   ; GO CONTINUE BINDING
04306 F05045DF (01582) YSPSZ   MOVRM R5, CSMS+WS*CR4AOF+HS   STORE YSEP*YSIZE
04308 3C52     (01583)         LRS R5, 2                     R5<=YSIZE*YSEP/4
04309 0800     (01584)         EVEN
                 (01585) ;
                 (01586) ;BIND YSEP*YSIZE/4
                 (01587) ;
0430A F05045DD (01588) YSPSZ4  MOVRM R5, CSMS+WS*CR4A11S+HS  STORE YSEP*YSIZE/4
0430C F0420003 (01589)         MOVMR R4, 3*HS(R1)            LOAD C BUFFER ID
0430E 9A40FF00 (01590)         ANDIR R4, MSKSLBYT
04310 3C47     (01591)         LRS R4, 7                     CREATE FULL WORD INDEX
04311 0800     (01592)         EVEN
                 (01593) ;
                 (01594)         EJECT
```

```
                 (01595) ;
                 (01596) ;BIND CBASE
                 (01597) ;
04312 C0680582  (01598)          MOVMRL R6, BCTSBA(R4)          LOAD C BASE ADDRESS
04314 9050461E  (01599)          MOVIR R5, CSMS+WS*CR4A4S       POINT TO LOAD CBASE INST.
04316 9A60000F  (01600)          ANDIR R6, SF                   AND OUT LOW FOUR BITS
04318 766AFFF0  (01601)          IORWK R6, R5, SFFF0            OR IN NEW FOUR BITS
0431A 846A0000  (01602)          MOVRML R6, 0(R5)               AND STORE BACK IN INST.
0431C C0580604  (01603)          MOVMRL R5, BCTSAD(R4)          R5<=CSEP, R6<=CSIZE-1
0431E 2661      (01604)          INCR R6, 1                     R6<=CSIZE
0431F 485C      (01605)          MULRR R5, R6                   R5<=2*(CSEP)(CSIZE)
04320 3C53      (01606)          LRS R5, 3                      R5<=0.25*(CSEP)(CSIZE) (HPI)
04321 0800      (01607)          EVEN
                 (01608) ;
                 (01609) ;BIND ALL HPI'S
                 (01610) ;
04322 F0504625  (01611)          MOVRM R5, CSMS+WS*CR4A5S+HS    STORE ALL HPI'S
04324 E0504629  (01612)          MOVRM R5, CSMS+WS*CR4A6S+HS    ...
04326 E050462D  (01613)          MOVRM R5, CSMS+WS*CR4A7S+HS    ...
                 (01614) ;
                 (01615) ;S1ANG=HPI/SSI, RIGHT NOW ONLY RADIX 2 AND RADIX 4
                 (01616) ;HAVE BEEN IMPLEMENTED.  SO ALL THAT IS NECESSARY
                 (01617) ;TO CALCULATE SSI IS TO KNOW IF USIZE IS AN EVEN OR
                 (01618) ;ODD POWER OF TWO.  THIS IS DONE BY CHECKING THE
                 (01619) ;POWER OF TWO ENTRY FOR THE U BUFFER IN THE BCTSAT
                 (01620) ;TABLE.  IF BIT 0 IS ON, THEN IT'S AN ODD POWER.
                 (01621) ;
04328 F030024D  (01622)          MOVMR R3, APSBSL               FETCH POINTER
0432A D800439D  (01623)          SMBC 0, PWR2S                  SKIP IF EVEN POWER OF TWO
0432C 2006      (01624)          HOP SBMS1                      ... ODD POWER
0432D F80845C7  (01625)          MOVLM 4, CSMS+WS*CR4AD1+HS     SET AND BIND SSI = 4
0432F F80A439C  (01626)          MOVLM FLGSG1, GSFLGS+HS         RESET G1
04331 3C52      (01627)          LRS R5, 2                      S1ANG=HPI/4
04332 2005      (01628)          HOP SBMS2
04333 F80445C7  (01629) SBMS1    MOVLM 2, CSMS+WS*CR4AD1+HS     SET AND BIND SSI = 2
04335 F84A439C  (01630)          MOVLM FLGSG1+FLGSSET, GSFLGS+HS     SET G1
04337 3C51      (01631)          LRS R5, 1                      S1ANG=HPI/2
                 (01632)          EVEN
                 (01633) SBMS2
                 (01634) ;
                 (01635) ;BIND S1ANG TO S7ANG'S
                 (01636) ;
04338 F0504657  (01637)          MOVRM R5, CSMS+WS*ANGL1S+HS    STORE S1ANG
0433A 3C52      (01638)          LRS R5, 2
```

ENTRY TO SPECIAL BINDING MODULE FOR VECTOR FFT SETUP

```
04338 0800      (01639)        EVEN
0433C F0504659 (01640)        MOVRM R5, CSMS+WS*ANGL2S+HS    STORE S2ANG
0433E 3C52      (01641)        LRS R5, 2
0433F 0800      (01642)        EVEN
04340 F050465B (01643)        MOVRM R5, CSMS+WS*ANGL3S+HS    STORE S3ANG
04342 3C52      (01644)        LRS R5, 2
04343 0800      (01645)        EVEN
04344 F050465D (01646)        MOVRM R5, CSMS+WS*ANGL4S+HS    STORE S4ANG
04346 3C52      (01647)        LRS R5, 2
04347 0800      (01648)        EVEN
04348 F050465F (01649)        MOVRM R5, CSMS+WS*ANGL5S+HS    STORE S5ANG
0434A 3C52      (01650)        LRS R5, 2
0434B 0800      (01651)        EVEN
0434C F0504661 (01652)        MOVRM R5, CSMS+WS*ANGL6S+HS    STORE S6ANG
0434E 3C52      (01653)        LRS R5, 2
0434F 0800      (01654)        EVEN
04350 F0504663 (01655)        MOVRM R5, CSMS+WS*ANGL7S+HS    STORE S7ANG
```

```
                  (01656) '          SPECIAL BINDING FOR WBASE, YBASE, AND UBASE
                  (01657) ;
                  (01658) ; THIS SECTION DOES THE FAST BINDING FOR
                  (01659) ; CHANGES IN WBASE, YBASE, AND UBASE ONLY.
                  (01660) ;
                  (01661) SBWSFFT EVEN                         START ON EVEN BOUNDARY
04352 F0420004    (01662)        MOVMR R4, 4*HS(R1)            GET W BUFFER ID
04354 9A40FF00    (01663)        ANDIR R4, MSKSLBYT            MASK THE LEFT HALF
04356 3C47        (01664)        LRS R4, 7                     CREATE FULL WORD INDEX
04357 0800        (01665)        EVEN
                  (01666) ;
                  (01667) ;BIND ALL WBASE'S
                  (01668) ;
0435B C06B05B2    (01669)        MOVMRL R6, BCTSBA(R4)         LOAD BASE ADDRESS IN R6, R7
0435A 905045B0    (01670)        MOVIR R5, CSMS+WS*CSM01$      POINT TO LOAD INSTRUCTION
0435C 9A60000F    (01671)        ANDIR R6, $F                  ONLY LOW FOUR BITS
0435E 766AFFF0    (01672)        IORWK R6, R5, $FFF0           'OR'ED INTO LOAD INST.
04360 846A0000    (01673)        MOVRML R6, 0(R5)              STORE ALL WBASE'S
04362 9A60000F    (01674)        ANDIR R6, $F                  MASK OUT OLD LOAD INST.
04364 905045CC    (01675)        MOVIR R5, CSMS+WS*CR4A1$      POINT TO NEXT LOAD WBASE
04366 766AFFF0    (01676)        IORWK R6, R5, $FFF0           OR WBASE INTO IT
04368 846A0000    (01677)        MOVRML R6, 0(R5)              STORE WBASE
                  (01678) ;
                  (01679) ;BIND UBASE
                  (01680) ;
0436A F0420002    (01681)        MOVMR R4, 2*HS(R1)            LOAD U BUFFER ID
0436C 9A40FF00    (01682)        ANDIR R4, MSKSLBYT            MASK THE LEFT HALF
0436E 3C47        (01683)        LRS R4, 7                     CREATE FULL WORD INDEX
0436F 0800        (01684)        EVEN
04370 C06B05B2    (01685)        MOVMRL R6, BCTSBA(R4)         LOAD UBASE ADDRESS IN R6, R7
04372 90504566    (01686)        MOVIR R5, CSMS+WS*CSML6$      POINT TO LOAD UBASE INST.
04374 9A60000F    (01687)        ANDIR R6, $F                  ONLY LOW ORDER FOUR BITS
04376 766AFFF0    (01688)        IORWK R6, R5, $FFF0           'OR' INTO INST.
04378 846A0000    (01689)        MOVRML R6, 0(R5)              STORE UBASE
                  (01690) *
                  (01691) *************** BEGIN REPEAT FFT CODE INSERTION ***************
                  (01692) *
                  (01693) **       ADDMR   R7,INSPC              ; BUMP TO NEXT INPUT VECTOR
                  (01694) **       ADC     R6                    ; COMPLETE THE BUMP
                  (01695) **       EVEN
                  (01696) **       MOVRML  R6,LDSUBS             ; SAVE NEXT <LOAD UBASE> INSTRUCTION
                  (01697) *
                  (01698) *************** END REPEAT FFT CODE INSERTION ***************
                  (01699) *
```

SPECIAL BINDING FOR WBASE, YBASE, AND UBASE

```
                   (01700) ;
                   (01701) ;BIND YBASE
                   (01702) ;
0437A F0420001     (01703)           MOVMR R4, HS(R1)              LOAD Y BUFFER ID
0437C 9A40FF00     (01704)           ANDIR R4, MSK$LBYT            MASK LEFT HALF
0437E 3C47         (01705)           LRS R4, 7                     CREATE FULL WORD INDEX
0437F 0800         (01706)           EVEN
04380 C06805B2     (01707)           MOVMRL R6, BCT$BA(R4)         LOAD YBASE ADDRESS IN R6, R7
04382 9050450A     (01708)           MOVIR R5, CSMS+WS*CR4A10S     POINT TO LOAD INST.
04384 9A60000F     (01709)           ANDIR R6, $F                  MASK LOW FOUR BITS
04386 766AFFF0     (01710)           IORWK R6, R5, $FFF0           OR INTO INST.
04388 846A0000     (01711)           MOVRML,R6, 0(R5)              STORE YBASE
                   (01712) *
                   (01713) *************** BEGIN REPEAT FFT CODE INSERTION ***************
                   (01714) *
                   (01715) **        ADDMR    R7,OUTSPC              ; BUMP TO NEXT OUTPUT VECTOR
                   (01716) **        ADC      R6                     ; COMPLETE THE BUMP
                   (01717) **        EVEN
                   (01718) **        MOVRML   R6,LD$YBS              ; SAVE NEXT <LOAD YBASE> INSTRUCTION
                   (01719) *
                   (01720) *************** END REPEAT FFT CODE INSERTION ***************
                   (01721) *
```

```
                    '         SET PENDING SLOTS TO PROPER VALUES
                    ;
0438A C6340BFB               PUSHMIL R3, AFDTSORG(R2)      STORE APU MODULE BUS ORIGIN

0438C 9071FFFF               MOVIR R7, -WS
0438F C6H7FFFF               PUSHMIL R3, @-WS(R3)          STORE APU MODULE START AND SIZE
04390 C6340BFA               PUSHMIL R3, AFDTSORG+WS(R2)   STORE APS MODULE BUS ORIGIN
                    *
                    *************** BEGIN REPEAT FFT CODE INSERTION ***************
                    *
                    **       MOVMR    R4,FFTSTYP                  ; GET FFT TYPE TAG
                    **     . XCT      RPTFT20(R4)                 ; PUSH CSPU POST-SUPPORT MODULE ADDR
                    **         PUSHMIL R3, AFDTSORG+2*WS(R2)
                    *
                    *************** END REPEAT FFT CODE INSERTION ***************
                    *
04392 2632                   INCR   R3, 2
04393 2771                   DECR R7, 1
                             EVEN
04394 C4H7FFFC               PUSHMI R3, @-2*WS(R3)         STORE APS MODULE SIZE
                    **       MOVIR R4, -3*HS(R3)           GET SPECIAL SUPPORT SEMAPHORE
                    **       MOVWK R4, R4, MSKSLBYT
                    **       LRS R4, 8
                    **       PUSHXI R3, R4                 STORE SPECIAL SUPPORT SEMAPHORE
                    **       MOVRR R5, R1
                    **       INCR R5, HS                   POINT TO SCALAR A ID
                    **       MOVWK R4, R5, MSKSRBYT
                    **       PUSHXIL R3, R4                STORE SCALAR A IDENTIFIER
04396 2634                    INCR   R3, 4
04397 C430439C               PUSHMI  R3, GSFLGS+HS         STORE FLAG G1
                    **       INCR R3, 4
                    **       PUSHMI R3, ZERO               SET FOR NO PBF
                    *
                    *
04399 80000F63               JMP    APSHNDRO               AND EXIT
                    **'DATA STORAGE FOR SPECIAL FFT BINDING
                    ***
                    ***************** BEGIN REPEAT FFT CODE INSERTION ***************
                    ***
                    **;
                    **;        EXECUTION LIST FOR POST-SUPPORT ADDR QUEUEING
                    **;
                    **         EVEN
                    **RPTFT20 PUSHMIL R3,AFDTSORG+2*WS(R2)    ; CSPU PRE-SUPPORT ADDR FOR 'MUXED' FF
```

```
                (01766) **         PUSHMIL R3,RPTFT21            ; POST-SUPPORT ADDR FOR REPEAT FFT
                (01767) **;
                (01768) **RPTFT21 ADDR    RPTFT31( ,1,SF)        ; POST-SUPPORT ENTRY ADDR
                (01769) **;
                (01770) **;
                (01771) **FFTSTYP DATA    MSS                    ; FFT TYPE TAG
                (01772) **LOG2SZ  DATA    MSS                    ; TEMPORARY STORAGE OF LOG2(FFT_SIZE)
                (01773) **MAXSMP  DATA    MSS                    ; MAXIMUM SAMPLE INDEX - 1
                (01774) **;
                (01775) **        EVEN                           ; MAKE THIS BLOCK EVEN FOR LATER TRANS
                (01776) **SMPSPC  DATA    MSS                    ; INPUT SAMPLE SPACING
                (01777) **FFTSIZ  DATA    MSS                    ; SINGLE FFT SIZE
                (01778) **NFFT    DATA    MSS                    ; NUMBER OF FFT'S TO DO
                (01779) **INSPC   DATA    MSS                    ; INPUT FFT VECTOR SPACING
                (01780) **OUTSPC  DATA    MSS                    ; OUTPUT FFT VECTOR SPACING
                (01781) **;
                (01782) **        DATA    MSS                    ; NEED FULL WORD BOUNDARY
                (01783) **;
                (01784) **LDSUBS  DATA    F'0,0'                 ; BOUND <LOAD UBASE> INSTR
                (01785) **LDSYBS  DATA    F'0,0'                 ; BOUND <LOAD YBASE> INSTR
                (01786) *
                (01787) *************** END REPEAT FFT CODE INSERTION ***************
                (01788) *
0439B 0004      (01789) GSFLGS  DATA    $4, $25                ; CODES FOR CONTROLLING FLAGS G0 & G1
0439C 0025
0439D 0000      (01790) PWR2S   DATA MSS                         STORAGE FOR POWER OF TWO
                (01791) ;
0439E 456C      (01792) BUSTBL  DATA    CSMS+WS*CSML           ; TABLE OF ADDRS FOR BINDING HU
0439F 4570      (01793)         DATA    CSMS+WS*CSME
043A0 4574      (01794)         DATA    CSMS+WS*CSML2S
043A1 4578      (01795)         DATA    CSMS+WS*CSML4S
043A2 457E      (01796)         DATA    CSMS+WS*CSML5S
043A3 0000      (01797)         DATA    0
                (01798) **'       REPEAT FFT POST-SUPPORT MODULE
                (01799) **;
                (01800) **        EVEN
                (01801) **RPTFT31 MOVIR   R7,NFFT-2              ; POINT TO NEEDED STEPPING INFO
                (01802) **        MOVIR   R6,3                   ; FOUR FULL WORDS TO TRANSFER
                (01803) **        BMOVEL  R7, R6, RPTSPRM        ; COPY THE BLOCK FOR UPDATING APS CODE
                (01804) **        MOVZM   APSASSS                ; ALLOW BINDING TO CONTINUE
                (01805) **        HOP     RPTFT33                ; JUMP TO BINDING UPDATE
                (01806) **;
                (01807) **RPTFT30 RET                            ; RETURN FROM INTERRUPT
                (01808) **        EVEN                           ; GET TO FULL WORD BOUNDARY
```

```
(01809) **RPTFT33 MOVIR   R7,RPTSUBS-2              ; POINT BEFORE <LOAD UBASE> INSTRUCTIO
(01810) **        MOVIR   R6,1                      ; 2 FULL WORDS TO TRANSFER TO APS
(01811) **        LPROCL  R7,R6,APSR                ; WRITE TO APS
(01812) **        MOVLM   FLG$SET+FLG$RI,SYS$FLGS   ; SET RI -- TURN ON AP
(01813) **        DECM    RPTSCNT                   ; COUNT THIS FFT
(01814) **        SKP     LE                        ; SKIP THE HOP IF DONE
(01815) **        HOP     RPTFT32                   ; GO COMPUTE NEXT SET OF BASE ADDR'S
(01816) **;
(01817) **        RET                               ; RELEASE THE CSPU
(01818) **        EVEN                              ; GET TO FULL WORD BOUNDARY
(01819) **        JMP     APSDONE                   ; ALL DONE -- LET EXEC CONTINUE
(01820) **;
(01821) **RPTFT32 MOVMRL  R4,RPTSYBS                ; GET <LOAD YBASE> INSTRUCTION
(01822) **        MOVMRL  R6,RPTSUBS                ; GET <LOAD UBASE> INSTRUCTION
(01823) **        ADDMR   R5,RPTSOSP                ; ADD SPACING TO NEXT VECTOR BASE
(01824) **        ADC     R4                        ; COMPLETE THE COMPUTATION
(01825) **        ADDMR   R7,RPTSISP                ; ADD SPACING TO NEXT VECTOR BASE
(01826) **        ADC     R6                        ; COMPLETE THE COMPUTATION
(01827) **        MOVRML  R4,RPTSYBS                ; STORE THE INSTRUCTION BACK
(01828) **        MOVRML  R6,RPTSUBS                ; STORE THE INSTRUCTION BACK
(01829) **        HOP     RPTFT30                   ; GO BACK TO LOOP START FOR RET
(01830) **;
(01831) **        EVEN
(01832) **;
(01833) **RPTSPRM                                   ; DATA TABLE FOR UPDATING APS CODE
(01834) **RPTSCNT DATA    M$$                       ; CURRENT COUNT FOR FFT
(01835) **RPTSISP DATA    M$$                       ; CURRENT SPACING TO NEXT INPUT VECTOR
(01836) **RPTSOSP DATA    M$$                       ; CURRENT SPACING TO NEXT OUTPUT VECTO
(01837) **        DATA    M$$                       ; POSITION HOLDER
(01838) **RPTSUBS DATA    F'0.0'                    ; CURRENT INPUT VECTOR BASE ADDR
(01839) **RPTSYBS DATA    F'0.0'                    ; CURRENT OUTPUT VECTOR BASE ADDR
```

```
                    (01840) 'FFT - AP PROGRAMS
                    (01841)         OPADD FWI, (3 .LS. 10)+(12 .LS. 5)+X'16'
                    (01842) !   APU PROGRAMS
                    (01843) ;
                    (01844) ;BINDING SECTION FOR EXEC
                    (01845) ;
                    (01846)         EVEN
    043A4 0000      (01847)         DATA CSM2SSA
    043A5 006D      (01848)         DATA CR4SSZ
                    (01849)         EVEN
                    (01850) ;
                    (01851) ;
                    (01852) !          SCRAMBLE AND FIRST RADIX-2 STAGE, FORWARD
                    (01853) ; 03/08/78
                    (01854) ; SCRAMBLE AND FIRST STAGE OF FFT,G1 SET
                    (01855) ;
                    (01856) ; FUNCTION
                    (01857) ; LETTING EIGHT SUCESSIVE INPUTS BEING DESIGNATED BY
                    (01858) ; RU0,RU1,IU1,IU0,IU2,IU3,RU3,RU2
                    (01859) ; THE EIGHT OUTPUTS ARE PROVIDED
                    (01860) ; RU0+RU1,RU0-RU1,IU0+IU1,IU0-IU1
                    (01861) ;  IU2+IU3,IU2-IU3,RU2+RU3,RU2-RU3
                    (01862) ; FOR RELATIONSHIP OF INPUTS TO U(K), AND OUTPUTS TO Y(K),
                    (01863) ; SEE SUPPORTING APS PROGRAM, CSM
                    (01864) ;
                    (01865) CSM2S   BEGIN APU(CSM2)
           00000000 (01866)         #A=00
                    (01867) ;
A00 043A6 9005000D (01868) CSM2SSA JUMPC(CSM4F, G1)
A01 043A8 10000003 (01869)         JUMP(CSM2E)
                    (01870) ;
A02 043AA 089C089C (01871) CSM2L   MOV(R,00)                                  00=IY0\IY1
                    (01872) ;
A03 043AC 08F008F0 (01873) CSM2F   MOV(IQA,A0)                                A0=RU0
A04 043AE 08F108F1 (01874)         MOV(IQA,A1)                                A1=RU1
A05 043B0 41004900 (01875)         ADD(A0,A1)\SUB(A0,A1)
                    (01876) ;
A06 043B2 08F308F3 (01877)         MOV(IQA,A3)                                A3=IU1
A07 043B4 08F208F2 (01878)         MOV(IQA,A2)                                A2=IU0
A08 043B6 435C485C (01879)         MOV(00),ADD(A2,A3)\MOV(00),SUB(A2,A3)      00=RY0\RY1
                    (01880) ;
A09 043B8 90160002 (01881)         JUMPC(CSM2L,FWI)
                    (01882) ;
A0A 043BA 089C089C (01883)         MOV(R,00)                                  00=IY0\IY1
```

```
                      (01884) ;
 A0B 043BC 00000000 (01885)         NOP
 A0C 043BE 1000002B (01886)         JUMP(CR4FS)
                      (01887) ;
```

```
                       (01888) '          SCRAMBLE AND FIRST RADIX-4 STAGE, FORWARD
                       (01889) ; SCRAMBLE AND FORWARD RADIX 4 STAGE OF FFT,G1 CLEAR
                       (01890) ;
                       (01891) ; FUNCTION
                       (01892) ; THE EIGHT SUCESSIVE INPUTS ARE PROVIDED TO LOOP
                       (01893) ; RU0,RU1,IU1,IU0,IU2,IU3,RU3,RU2
                       (01894) ; THE INTERMEDIATE RADIX TWO RESULTS ARE CALCULATED
                       (01895) ; S0=U0+U1,S1=U0-U1,S2=U2+U3,S3=U2-U3
                       (01896) ; THE OUTPUTS THEN BEING GIVEN BY
                       (01897) ; Y0=S0+S2,Y1=S1-JS3,Y2=S0-S2,Y3=S1+JS3
                       (01898) ; THE ACTUAL OUTPUT SEQUENCE BEING
                       (01899) ; RY0,RY1,IY0,IY1,IY2,IY3,RY2,RY3
                       (01900) ;
                       (01901) ; APU INITIALIZATION
                       (01902) ;
                       (01903) ;
A0D 043C0 08F008F0 (01904) CSM4F   MOV(IOA,A0)                                A0=RU0\RU0
A0E 043C2 08F108F1 (01905)         MOV(IOA,A1)                                A1=RU1\RU1
A0F 043C4 08F308F3 (01906)         MOV(IOA,A3)                                A3=IU1\IU1
A10 043C6 41004900 (01907)         ADD(A0,A1)\SUB(A0,A1)
A11 043C8 08F208F2 (01908)         MOV(IOA,A2)                                A2=IU0\IU0
A12 043CA 1000001B (01909)         JUMP(CSM4FS)
                   (01910) ;
                   (01911) ;
                   (01912) ; CSM4F, APU INNER LOOP
                   (01913)
A13 043CC 46BC4FBC (01914) #1      MOV(OO),ADD(A5,A6)\MOV(OO),SUB(A5,A7)      OO=RY0\RY1
A14 043CE 08F008F0 (01915)         MOV(IOA,A0)                                A0=RU0
                   (01916)
A15 043D0 4FBC47BC (01917)         MOV(OO),SUB(A5,A6)\MOV(OO),ADD(A5,A7)      OO=IY0\IY1
A16 043D2 08F108F1 (01918)         MOV(IOA,A1)                                A1=RU1
                   (01919)
A17 043D4 4F9C4F9C (01920)         MOV(OO),SUB(A4,A7)\MOV(OO),SUB(A4,A6)      OO=IY2\IY3
A18 043D6 08F308F3 (01921)         MOV(IOA,A3)                                A3=IU1
                   (01922)
A19 043D8 411C491C (01923)         MOV(OO),ADD(A0,A1)\MOV(OO),SUB(A0,A1)      OO=RY2\RY3
A1A 043DA 08F208F2 (01924)         MOV(IOA,A2)                                A2=IU0
                   (01925)
A1B 043DC 43544B54 (01926) CSM4FS  MOV(A4),ADD(A2,A3)\MOV(A4),SUB(A2,A3)      A4=RS0\RS1
A1C 043DE 08F008F0 (01927)         MOV(IOA,A0)                                A0=IU2
A1D 043E0 08F108F1 (01928)         MOV(IOA,A1)                                A1=IU3
                   (01929)
A1E 043E2 41154915 (01930)         MOV(A5),ADD(A0,A1)\MOV(A5),SUB(A0,A1)      A5=IS0\IS1
A1F 043E4 08F308F3 (01931)         MOV(IOA,A3)                                A3=RU3
```

```
A20 043E6 08F208F2 (01932)      MOV(IQA,A2)                                A2=RU2
                   (01933)
A21 043E8 43564B56 (01934)      MOV(A6),ADD(A2,A3)\MOV(A6),SUB(A2,A3)      A6=IS2\IS3
                   (01935)
A22 043EA 47974697 (01936)      MOV(A7),ADD(A4,A7)\MOV(A7),ADD(A4,A6)      A7=RS2\RS3
                   (01937)
A23 043EC 90160013 (01938)      JUMPC(#1,FWI)
                   (01939)
A24 043EE 46BC4EBC (01940)      MOV(OQ),ADD(A5,A6)\MOV(OQ),SUB(A5,A7)      OQ=RY0\RY1
A25 043F0 4EBC47BC (01941)      MOV(OQ),SUB(A5,A6)\MOV(OQ),ADD(A5,A7)      OQ=IY0\IY1
A26 043F2 4F9C4F9C (01942)      MOV(OQ),SUB(A4,A7)\MOV(OQ),SUB(A4,A6)      OQ=IY2\IY3
A27 043F4 089C089C (01943)      MOV(R,OQ)                                  OQ=RY2\RY3
```

```
'          SUCCESSIVE RADIX-4 STAGES, FORWARD
;
;USES APS PROGRAM CR4A
;
;MATHEMATICS
;
; W=A+BE+CE2+DE3=P+Q
; X=A-JBE-CE2+JDE3=R-S
; Y=A-BE+CE2-DE3=P-Q
; Z=A+JBE-CE2-JDE3=R+S

; P=A+CE2,R=A-CE2
; Q=E(B+DE2),S=JE(B-DE2)

; PR=AR+CRCOS2X+CISIN2X
; PI=AI+CICOS2X-CRSIN2X
; RR=AR-CRCOS2X-CISIN2X
; RI=AI-CICOS2X+CRSIN2X
; QR=BRCOSX+BISINX+DRCOS3X+DISIN3X
; QI=BICOSX-BRSINX+DICOS3X-DRSIN3X
; SR=BRSINX-BICOSX+DICOS3X-DRSIN3X
; SI=BRCOSX+BISINX-DRCOS3X-DISIN3X
;
; SINX STORED IN M1\M5
; SIN2X STORED IN M6\M2
; SIN3X STORED IN M7\M3
; COSX STORED IN M5\M1
; COS2X STORED IN M2\M6
; COS3X STORED IN M3\M7

        EJECT
```

```
                   (01975) ;CR4F-PIPELINE STARTUP
                   (01976)
                   (01977)
A28 043F6 202A202A (01978) CR4FS   CLEAR(AF2)
A29 043F8 20292029 (01979) CR4FSA  CLEAR(AF1)
A2A 043FA 20282028 (01980)         CLEAR(AF0)
                   (01981)
A2B 043FC 16801680 (01982)         K(1)
A2C 043FE 080908F0 (01983)         MOV(ZERO,M1)\MOV(IOA,A0)                 A0=DI\DR
A2D 04400 08F0080D (01984)         MOV(IOA,A0)\MOV(ZERO,M5)                 SINX=0=M1\M5
                   (01985)
                   (01986)
A2E 04402 088D0889 (01987)         MOV(R,M5)\MOV(R,M1)                      COSX=1=M5\M1
A2F 04404 088A088F (01988)         MOV(R,M2)\MOV(R,M6)                      COS2X=1=M2\M6
A30 04406 088B088F (01989)         MOV(R,M3)\MOV(R,M7)                      COS3X=1=M3\M7
                   (01990)
A31 04408 08F1080A (01991)         MOV(IOA,A1)\MOV(ZERO,M2)                 A1=BI\BR
A32 0440A 080E08F1 (01992)         MOV(ZERO,M6)\MOV(IOA,A1)                 SIN2X=0=M6\M2
A33 0440C 080F080B (01993)         MOV(ZERO,M7)\MOV(ZERO,M3)                SIN3X=0=M7\M3
                   (01994)
A34 0440E 08F808F8 (01995)         MOV(IOA,M0)                              M0=CR
A35 04410 84408440 (01996)         MUL(M0,M6)                               P=CRSIN2X\CRCOS2X
A36 04412 0A200220 (01997)         NEG(A1)\R(A1)                            R=-BI\BR
                   (01998)
A37 04414 10000053 (01999)         JUMP(CR4FF)
                   (02000)
                   (02001) ;CR4F;COSINE ENTRY
                   (02002) ;
A38 04416 08ED08E9 (02003) CR4FC   MOV(IOA,M5)\MOV(IOA,M1)                  COSX TO M5\M1
A39 04418 08E908ED (02004)         MOV(IOA,M1)\MOV(IOA,M5)                  SINX TO M1\M5
A3A 0441A 08A008A0 (02005)         MOV(P,NULL)
A3B 0441C 08EE08EA (02006)         MOV(IOA,M6)\MOV(IOA,M2)                  SIN2X TO M6\M2
A3C 0441E 08EA08EE (02007)         MOV(IOA,M2)\MOV(IOA,M6)                  COS2X TO M2\M6
A3D 04420 08EB08EF (02008)         MOV(IOA,M3)\MOV(IOA,M7)                  COS3X TO M3\M7
A3E 04422 08EF08EB (02009)         MOV(IOA,M7)\MOV(IOA,M3)                  SIN3X TO M7\M3
                   (02010) ;
                   (02011) ;CR4F-BUTTERFLY
                   (02012)
A3F 04424 08E808E8 (02013) CR4FB   MOV(IOA,M0)                              DR TO M0
A40 04426 A474A474 (02014)         MOV(A4),MULN(M0,M7)\MOV(A4),MUL(M0,M7)   A4=CICOS2X\CISIN2X
A41 04428 4A954455 (02015)         MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)    A5=QI\QR
A42 0442A 08EC08EC (02016)         MOV(IOA,M4)                              DI TO M4
A43 0442C 4A724A72 (02017)         MOV(A2),SUB(A3,A2)                       A2=CICOS2X-CRSIN2X
                   (02018) ;                                                   \CISIN2X+CRCOS2X
```

```
                         (02019)
                         (02020)
A44 0442E 8590A590 (02021)         MOV(A0),MUL(M3,M4)\MOV(A0),MULN(M3,M4)    A0=DRSIN3X\DRCOS3X
A45 04430 4278427B (02022)         MOV(EX0),ADD(A3,A2)                       EX=RI\RR
A46 04432 08560856 (02023)         MOV(EXI,A6)                               A6=RR\RI
A47 04434 08F808FB (02024)         MOV(IOA,M0)                               BI TO M0
A48 04436 47D447D4 (02025)         MOV(A4),ADD(A6,A7)                        A4=PI\PR
                         (02026)
                         (02027)
A49 04438 8431A431 (02028)         MOV(A1),MUL(M0,M5)\MOV(A1),MULN(M0,M5)    A1=DICOS3X\DISIN3X
A4A 0443A 483C403C (02029)         MOV(OO),SUB(A1,A0)\MOV(OO),ADD(A1,A0)     OO=ZR\ZI
A4B 0443C 08EC08EC (02030)         MOV(IOA,M4)                               BR TO M4
A4C 0443E 4D904D90 (02031)         MOV(A0),SUB(A4,A5)                        A0=DICOS3X-DRSIN3X
                         (02032) ;                                              \DISIN3X+DRCOS3X
                         (02033)
                         (02034)
                         (02035)
A4D 04440 A4918491 (02036)         MOV(A1),MULN(M1,M4)\MOV(A1),MUL(M1,M4)    A1=BICOSX\BISINX
A4E 04442 4FDC4FDC (02037)         MOV(OO),SUB(A6,A7)                        OO=YI\YR
A4F 04444 08F808EB (02038)         MOV(IOA,M0)                               CR TO M0
A50 04446 459C459C (02039)         MOV(OO),ADD(A4,A5)                        OO=XR\XI
                         (02040)
                         (02041)
A51 04448 A4528452 (02042)         MOV(A2),MULN(M0,M6)\MOV(A2),MUL(M0,M6)    A2=BRSINX\BRCOSX
A52 0444A 495C423C (02043)         MOV(OO),SUB(A2,A1)\MOV(OO),ADD(A1,A2)     OO=WR\WI
A53 0444C 08FC08FC (02044) CR4FF   MOV(IOA,M4)                               CI TO M4
A54 0444E 41114831 (02045)         MOV(A1),ADD(A0,A1)\MOV(A1),SUB(A1,A0)     A1=BRSINX-BICOSX
                         (02046) ;                                              \BRCOSX+BISINX
                         (02047)
                         (02048)
A55 04450 8512A512 (02049)         MOV(A2),MUL(M2,M4)\MOV(A2),MULN(M2,M4)    A2=CRSIN2X\CRCOS2X
A56 04452 08F34037 (02050)         MOV(IOA,A3)\MOV(A7),ADD(A1,A0)            A3=AI \ A7=SI
A57 04454 491708F3 (02051)         MOV(A7),SUB(A0,A1)\MOV(IOA,A3)            A7=SR \ A3=AR
A58 04456 9028003F (02052)         JUMPC(CR4FB,AF0),CLEAR                    AF1;STAGE DONE
A59 04458 90290038 (02053)         JUMPC(CR4FC,AF1),CLEAR                    AF0;COSINES USED
                         (02054)
                         (02055)
                         (02056) ; CR4F-PIPELINE CLEANUP
A5A 0445A 08B408B4 (02057)         MOV(P,A4)                                 A4=CICOS2X\CISIN2X
                         (02058)
                         (02059)
A5B 0445C 4A954455 (02060)         MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)     A5=QI\QR
A5C 0445E 4A724A72 (02061)         MOV(A2),SUB(A3,A2)                        A2=CICOS2X-CRSIN2X
                         (02062) ;                                              \CISIN2X+CRCOS2X
```

```
                         (02063)
A5D 04460 427842278 (02064)      MOV(EXO),ADD(A3,A2)           EX=RI\RR
A5E 04462 08560856  (02065)      MOV(EXI,A6)                   A6=RR\RI
                         (02066)
A5F 04464 47D447D4  (02067)      MOV(A4),ADD(A6,A7)            A4=PI\PR
A60 04466 4D9C0000  (02068)      MOV(OO),SUB(A4,A5)\NOP        OO=ZR
A61 04468 00004D9C  (02069)      NOP\MOV(OO),SUB(A4,A5)        OO=ZI
A62 0446A 4FDC0000  (02070)      MOV(OO),SUB(A6,A7)\NOP        OO=YI
A63 0446C 00004FDC  (02071)      NOP\MOV(OO),SUB(A6,A7)        OO=YR
A64 0446E 204A204A  (02072)      SET(AF2)                      APS/OTPT STAGE DN
A65 04470 459C0000  (02073)      MOV(OO),ADD(A4,A5)\NOP        OO=XR
A66 04472 0000459C  (02074)      NOP\MOV(OO),ADD(A4,A5)        OO=XI
A67 04474 089C0000  (02075)      MOV(R,OO)\NOP                 OO=WR
A68 04476 0000089C  (02076)      NOP\MOV(R,OO)                 OO=WI
                         (02077)
A69 04478 901D0029  (02078)      JUMPC(CR4$SA,FI)
                         (02079)
A6A 0447A 20322032  (02080)      CLEAR(RA)
A6B 0447C 00000000  (02081)      NOP
A6C 0447E 10000000  (02082)      JUMP(CSM2$SA)
                         (02083) ;
          0000006D  (02084)      CR4$SZ = #A-CSM2$SA
    04480                (02085)      END
                         (02086) ;
                         (02087)
                         (02088)
                         (02089) ;
```

```
                            '          SCRAMBLE AND FIRST RADIX-2 STAGE, REVERSE
                            ; 03/08/78
                            ; SCRAMBLE AND FIRST STAGE OF IFFT,G1 SET
                            ;
                            ; FUNCTION
                            ; LETTING EIGHT SUCESSIVE INPUTS BEING DESIGNATED BY
                            ;        RU0,RU1,IU1,IU0,IU2,IU3,RU3,RU2
                            ; THE EIGHT OUTPUTS ARE PROVIDED
                            ;        RU0+RU1,RU0-RU1,IU0+IU1,IU0-IU1
                            ;           IU2+IU3,IU2-IU3,RU2+RU3,RU2-RU3
                            ; FOR RELATIONSHIP OF INPUTS TO U(K), AND OUTPUTS TO Y(K),
                            ; SEE SUPPORTING APS PROGRAM, CSM
                            ;
                            ;
                            ;BINDING SECTION FOR EXEC
                            ;
                                    EVEN
    04480 0000                      DATA ISM2$SA
    04481 006D                      DATA CR41$SZ
                                    EVEN
                            ;
                            ;
                            ISM2$   BEGIN APU(ISM2)
          00000000                  $A=00
                            ;
A00 04482 9005000D          ISM2$SA JUMPC(CSM41, G1)
A01 04484 10000003                  JUMP(ISM2E)
                            ;
A02 04486 089C089C          ISM2L   MOV(R,OO)                                  OO=IY0\IY1
                            ;
A03 04488 08F008F0          ISM2E   MOV(IOA,A0)                                A0=RU0
A04 0448A 08F108F1                  MOV(IOA,A1)                                A1=RU1
A05 0448C 41004900                  ADD(A0,A1)\SUB(A0,A1)
                            ;
A06 0448E 08F308F3                  MOV(IOA,A3)                                A3=IU1
A07 04490 08F208F2                  MOV(IOA,A2)                                A2=IU0
A08 04492 435C485C                  MOV(OO),ADD(A2,A3)\MOV(OO),SUB(A2,A3)      OO=RY0\RY1
                            ;
A09 04494 90160002                  JUMPC(ISM2L,EWT)
                            ;
A0A 04496 089C089C                  MOV(R,OO)                                  OO=IY0\IY1
                            ;
A0B 04498 00000000                  NOP
A0C 0449A 10000028                  JUMP(CR41S)
```

```
                   (02134) '        SCRAMBLE AND FIRST RADIX-4 STAGE, REVERSE
                   (02135) ; SCRAMBLE AND RADIX 4 STAGE OF IFFT,G1 CLEAR
                   (02136) ;
                   (02137) ; FUNCTION
                   (02138) ; THE EIGHT SUCESSIVE INPUTS ARE PROVIDED TO LOOP
                   (02139) ; RU0,RU1,IU1,IU0,IU2,IU3,RU3,RU2
                   (02140) ; THE INTERMEDIATE RADIX TWO RESULTS ARE CALCULATED
                   (02141) ; S0=U0+U1,S1=U0-U1,S2=U2+U3,S3=U2-U3
                   (02142) ; THE OUTPUTS THEN BEING GIVEN BY
                   (02143) ; Y0=S0+S2,Y1=S1+JS3,Y2=S0-S2,Y3=S1-JS3
                   (02144) ; THE ACTUAL OUTPUT SEQUENCE BEING
                   (02145) ; RY0,RY1,IY0,IY1,IY2,IY3,RY2,RY3
                   (02146) ;
                   (02147) ; APU INITIALIZATION
                   (02148) ;
                   (02149) ;
A0D 0449C 0BF00BF0 (02150) CSM4I   MOV(IOA,A0)                                  A0=RU0\RU0
A0E 0449E 0BF10BF1 (02151)         MOV(IOA,A1)                                  A1=RU1\RU1
A0F 044A0 0BF30BF3 (02152)         MOV(IOA,A3)                                  A3=IU1\IU1
A10 044A2 41004900 (02153)         ADD(A0,A1)\SUB(A0,A1)
A11 044A4 0BF20BF2 (02154)         MOV(IOA,A2)                                  A2=IU0\IU0
A12 044A6 1000001B (02155)         JUMP(CSM4I$)
                   (02156) ;
                   (02157) ;
                   (02158) ; CSM4I, APU INNER LOOP
                   (02159)
A13 044A8 46BC47BC (02160) #1      MOV(OQ),ADD(A5,A6)\MOV(OQ),ADD(A5,A7)        OQ=RY0\RY1
A14 044AA 0BF00BF0 (02161)         MOV(IOA,A0)                                  A0=RU0
                   (02162)
A15 044AC 4EBC4FBC (02163)         MOV(OQ),SUB(A5,A6)\MOV(OQ),SUB(A5,A7)        OQ=IY0\IY1
A16 044AE 0BF10BF1 (02164)         MOV(IOA,A1)                                  A1=RU1
                   (02165)
A17 044B0 4E9C469C (02166)         MOV(OQ),SUB(A4,A7)\MOV(OQ),ADD(A4,A6)        OQ=IY2\IY3
A18 044B2 0BF30BF3 (02167)         MOV(IOA,A3)                                  A3=IU1
                   (02168)
A19 044B4 411C491C (02169)         MOV(OQ),ADD(A0,A1)\MOV(OQ),SUB(A0,A1)        OQ=RY2\RY3
A1A 044B6 0BF20BF2 (02170)         MOV(IOA,A2)                                  A2=IU0
                   (02171)
A1B 044B8 43544B54 (02172) CSM4I$  MOV(A4),ADD(A2,A3)\MOV(A4),SUB(A2,A3)        A4=RS0\RS1
A1C 044BA 0BF00BF0 (02173)         MOV(IOA,A0)                                  A0=IU2
A1D 044BC 0BF10BF1 (02174)         MOV(IOA,A1)                                  A1=IU3
                   (02175)
A1E 044BE 41154915 (02176)         MOV(A5),ADD(A0,A1)\MOV(A5),SUB(A0,A1)        A5=IS0\IS1
A1F 044C0 0BF30BF3 (02177)         MOV(IOA,A3)                                  A3=RU3
```

```
A20 044C2 08F208F2 (02178)     MOV(IOA,A2)                                A2=RU2
                   (02179)
A21 044C4 43564856 (02180)     MOV(A6),ADD(A2,A3)\MOV(A6),SUB(A2,A3)      A6=IS2\IS3
                   (02181)
A22 044C6 47974F97 (02182)     MOV(A7),ADD(A4,A7)\MOV(A7),SUB(A4,A6)      A7=RS2\RS3
                   (02183)
A23 044C8 90160013 (02184)     JUMPC(#1,FWI)
                   (02185)
A24 044CA 46BC47BC (02186)     MOV(OQ),ADD(A5,A6)\MOV(OQ),ADD(A5,A7)      OQ=RY0\RY1
A25 044CC 4EBC4FBC (02187)     MOV(OQ),SUB(A5,A6)\MOV(OQ),SUB(A5,A7)      OQ=IY0\IY1
A26 044CE 4F9C469C (02188)     MOV(OQ),SUB(A4,A7)\MOV(OQ),ADD(A4,A6)      OQ=IY2\IY3
A27 044D0 089C089C (02189)     MOV(R,OQ)                                  OQ=RY2\RY3
```

```
'          SUCCESSIVE RADIX-4 STAGES, REVERSE
;
;USES APS PROGRAM CR4A
;
;MATHEMATICS
;
; W=A+BE+CE2+DE3=P+Q
; X=A+JBE-CE2-JDE3=R+S
; Y=A-BE+CE2-DE3=P-Q
; Z=A-JBE-CE2+JDE3=R-S

; P=A+CE2,R=A-CE2
; Q=E(B+DE2),S=JE(B-DE2)

; PR=AR+CRCOS2X-CISIN2X
; PI=AI+CICOS2X+CRSIN2X
; RR=AR-CRCOS2X+CISIN2X
; RI=AI-CICOS2X-CRSIN2X
; QR=BRCOSX-BISINX+DRCOS3X-DISIN3X
; QI=BICOSX+BRSINX+DICOS3X+DRSIN3X
; SR=BICOSX-BRSINX+DICOS3X-DRSIN3X
; SI=BRCOSX-BISINX-DRCOS3X+DISIN3X
;
; SINX STORED IN M1\M5
; SIN2X STORED IN M6\M2
; SIN3X STORED IN M7\M3
; COSX STORED IN M5\M1
; COS2X STORED IN M2\M6
; COS3X STORED IN M3\M7

         EJECT
```

```
                    (02221) ;CR4I-PIPELINE STARTUP
                    (02222)
                    (02223)
A28 044D2 202A202A (02224) CR4IS    CLEAR(AF2)
A29 044D4 20292029 (02225) CR4ISA   CLEAR(AF1)
A2A 044D6 20282028 (02226)          CLEAR(AF0)
                    (02227)
A2B 044D8 16801680 (02228)          K(1)
A2C 044DA 0B090BF0 (02229)          MOV(ZERO,M1)\MOV(IOA,A0)          A0=DI\DR
A2D 044DC 0BF00B0D (02230)          MOV(IOA,A0)\MOV(ZERO,M5)          SINX=0=M1\M5
                    (02231)
                    (02232)
A2E 044DE 0B8D0B89 (02233)          MOV(R,M5)\MOV(R,M1)               COSX=1=M5\M1
A2F 044E0 0B8A0B8E (02234)          MOV(R,M2)\MOV(R,M6)               COS2X=1=M2\M6
A30 044E2 0B8B0B8F (02235)          MOV(R,M3)\MOV(R,M7)               COS3X=1=M3\M7
                    (02236)
A31 044E4 0BF10B0A (02237)          MOV(IOA,A1)\MOV(ZERO,M2)          A1=BI\BR
A32 044E6 0B0E0BF1 (02238)          MOV(ZERO,M6)\MOV(IOA,A1)          SIN2X=0=M6\M2
A33 044E8 0B0F0B0B (02239)          MOV(ZERO,M7)\MOV(ZERO,M3)         SIN3X=0=M7\M3
                    (02240)
A34 044EA 0BFB0BF8 (02241)          MOV(IOA,M0)                       M0=CR
A35 044EC 84408440 (02242)          MUL(M0,M6)                        P=CRSIN2X\CRCOS2X
A36 044EE 0A200220 (02243)          NEG(A1)\R(A1)                     R=-BI\BR
                    (02244)
A37 044F0 10000053 (02245)          JUMP(CR4IE)
                    (02246)
                    (02247) ;CR4I-COSINE ENTRY
                    (02248) ;
A38 044F2 0BFD0BF9 (02249) CR4IC    MOV(IOA,M5)\MOV(IOA,M1)           COSX TO M5\M1
A39 044F4 0BF90BFD (02250)          MOV(IOA,M1)\MOV(IOA,M5)           SINX TO M1\M5
A3A 044F6 0BA00BA0 (02251)          MOV(P,NULL)
A3B 044F8 0BFE0BFA (02252)          MOV(IOA,M6)\MOV(IOA,M2)           SIN2X TO M6\M2
A3C 044FA 0BFA0BFE (02253)          MOV(IOA,M2)\MOV(IOA,M6)           COS2X TO M2\M6
A3D 044FC 0BFB0BFF (02254)          MOV(IOA,M3)\MOV(IOA,M7)           COS3X TO M3\M7
A3E 044FE 0BFF0BFB (02255)          MOV(IOA,M7)\MOV(IOA,M3)           SIN3X TO M7\M3
                    (02256) ;
                    (02257) ;CR4I-BUTTERFLY
                    (02258)
A3F 04500 0BF80BF8 (02259) CR4IB    MOV(IOA,M0)                       DR TO M0
A40 04502 84748474 (02260)          MOV(A4),MUL(M0,M7)\MOV(A4),MUL(M0,M7)   A4=CICOS2X\-CISIN2X
A41 04504 4A954455 (02261)          MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)   A5=DI\DR
A42 04506 0BFC0BFC (02262)          MOV(IOA,M4)                       DI TO M4
A43 04508 4A724A72 (02263)          MOV(A2),SUB(A3,A2)                A2=CICOS2X+CRSIN2X
                    (02264) ;                                            \-CISIN2X+CRCOS2X
```

```
                         (02265)
                         (02266)
A44 0450A 85908590 (02267)          MOV(A0),MUL(M3,M4)\MOV(A0),MUL(M3,M4)    A0=-DRSIN3X\DRCOS3X
A45 0450C 427B427B (02268)          MOV(EXO),ADD(A3,A2)                      EX=RI\RR
A46 0450E 08560856 (02269)          MOV(EXI,A6)                              A6=RR\RI
A47 04510 0BE80BE8 (02270)          MOV(IOA,M0)                              RI TO M0
A48 04512 4F044FD4 (02271)          MOV(A4),SUB(A6,A7)                       A4=PI\PR
                         (02272)
                         (02273)
A49 04514 84318431 (02274)          MOV(A1),MUL(M0,M5)\MOV(A1),MUL(M0,M5)    A1=DICOS3X\-DISIN3X
A4A 04516 4B3C403C (02275)          MOV(OO),SUB(A1,A0)\MOV(OO),ADD(A1,A0)    OO=ZR\ZI
A4B 04518 0BEC0BEC (02276)          MOV(IOA,M4)                              RR TO M4
A4C 0451A 4D904D90 (02277)          MOV(A0),SUB(A4,A5)                       A0=DICOS3X+DRSIN3X
                         (02278) ;                                              \-DISIN3X+DRCOS3X
                         (02279)
                         (02280)
                         (02281)
A4D 0451C 84918491 (02282)          MOV(A1),MUL(M1,M4)\MOV(A1),MUL(M1,M4)    A1=BICOSX\-BISINX
A4E 0451E 47DC47DC (02283)          MOV(OO),ADD(A6,A7)                       OO=YI\YR
A4F 04520 0BE80BE8 (02284)          MOV(IOA,M0)                              CR TO M0
A50 04522 459C459C (02285)          MOV(OO),ADD(A4,A5)                       OO=XR\XI
                         (02286)
                         (02287)
A51 04524 84528452 (02288)          MOV(A2),MUL(M0,M6)\MOV(A2),MUL(M0,M6)    A2=-BRSINX\BRCOSX
A52 04526 495C423C (02289)          MOV(OO),SUB(A2,A1)\MOV(OO),ADD(A1,A2)    OO=WR\WI
A53 04528 0BEC0BEC (02290) CR4IF    MOV(IOA,M4)                              CI TO M4
A54 0452A 41114831 (02291)          MOV(A1),ADD(A0,A1)\MOV(A1),SUB(A1,A0)    A1=-BRSINX-BICOSX
                         (02292) ;                                              \BRCOSX-BISINX
                         (02293)
                         (02294)
A55 0452C 85128512 (02295)          MOV(A2),MUL(M2,M4)\MOV(A2),MUL(M2,M4)    A2=-CRSIN2X\CRCOS2X
A56 0452E 0BE34037 (02296)          MOV(IOA,A3)\MOV(A7),ADD(A1,A0)           A3=AI \ A7=SI
A57 04530 49170BE3 (02297)          MOV(A7),SUB(A0,A1)\MOV(IOA,A3)           A7=SR \ A3=AR
A58 04532 9028003F (02298)          JUMPC(CR4IB,AF0),CLEAR                   AF1;STAGE DONE
A59 04534 90290038 (02299)          JUMPC(CR4IC,AF1),CLEAR                   AF0;COSINES USED
                         (02300)
                         (02301)
                         (02302) ; CR4I-PIPELINE CLEANUP
A5A 04536 0BB40BB4 (02303)          MOV(P,A4)                                A4=CICOS2X\-CISIN2X
                         (02304)
                         (02305)
A5B 04538 4A954455 (02306)          MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)    A5=QI\QR
A5C 0453A 4A724A72 (02307)          MOV(A2),SUB(A3,A2)                       A2=CICOS2X+CRSIN2X
                         (02308) ;                                              \-CISIN2X+CRCOS2X
```

```
A5D 0453C 42784278          MOV(FXO),ADD(A3,A2)            FX=RI\RR
A5E 0453E 08560856          MOV(FXI,A6)                    A6=RR\RI

A5F 04540 4FD44FD4          MOV(A4),SUB(A6,A7)             A4=PI\PR
A60 04542 409C0000          MOV(OO),SUB(A4,A5)\NOP         OQ=ZR
A61 04544 0000409C          NOP\MOV(OQ),SUB(A4,A5)         OQ=ZI
A62 04546 470C0000          MOV(OQ),ADD(A6,A7)\NOP         OQ=YI
A63 04548 0000470C          NOP\MOV(OQ),ADD(A6,A7)         OQ=YR
A64 0454A 204A204A          SET(AF2)                       APS/OTPT STAGE DN
A65 0454C 459C0000          MOV(OQ),ADD(A4,A5)\NOP         OQ=XR
A66 0454E 0000459C          NOP\MOV(OQ),ADD(A4,A5)         OQ=XI
A67 04550 089C0000          MOV(R,OQ)\NOP                  OQ=WR
A68 04552 0000089C          NOP\MOV(R,OQ)                  OQ=WI

A69 04554 90100029          JUMPC(CR4ISA,FI)

A6A 04556 20322032          CLEAR(RA)
A6B 04558 00000000          NOP
A6C 0455A 10000000          JUMP(ISM2SSA)
                    ;
          0000006D          CR4ISSZ = #A-ISM2SSA
    0455C                   END
                    ;
                    ;
```

```
                (02340) '   APS PROGRAMS
                (02341) ;
                (02342) ;
                (02343) ; START OF HEADER BLOCK
                (02344) ;
                (02345)         EVEN                          START ON WORD BOUNDARY
0455C 00004664  (02346)         ADDR CSMSI                    PTR TO CONSTR INSTR BLOCK(NONE)
0455F 00004564  (02347)         ADDR CSMS+2*CSMSS             PTR TO SCALAR BLOCK (NONE)
04560 0000      (02348)         DATA 0                        NO SCALARS
04561 0100      (02349)         DATA CR4ASZ                   MODULE SIZE
04562 00000000  (02350)         ADDR CR4ASA                   PTR TO CHAIN ANCHOR
                (02351)         EVEN                          END OF BOUNDARY
                (02352) ;
                (02353) ;
```

```
                      (02354) '         CSM - COMPLEX FFT SCRAMBLE (P0 - INPUT)
                      (02355) ; APS PROGRAM PROVIDING SCRAMBLE FOR COMPLEX FFT
                      (02356) ; MAY BE USED WITH THE MAP-300 APU PROGRAMS
                      (02357) ; CSM2(Y,U), CSM4F(Y,U), CSM4I(Y,U)
                      (02358) ;
                      (02359) ; RESTRICTIONS
                      (02360) ; IN PLACE OPERATION NOT PERMITTED
                      (02361) ; BUFFER SIZES MUST BE 1024 OR LESS
                      (02362) ; Y BUFFER MUST BE COMPACT 32 BIT FLOATING
                      (02363) ;
                      (02364) ;
                      (02365) ; BINDING PARAMETERS
                      (02366) ;
                      (02367) ;         N=# OF POINTS IN FFT (USIZE)
                      (02368) ;         HU=USEP*N/2
                      (02369) ;         QU=HU/2
                      (02370) ; APS-INPUT ADDRESS SEQUENCE
                      (02371) ; RU(K), RU(K+N/2), IU(K+N/2), IU(K), IU(K+N/4)
                      (02372) ; IU(K+3N/4), RU(K+3N/4), RU(K+N/4)], FOR 0<=K<N/4
                      (02373) ; THUS PROVIDING REVERSAL OF TWO BITS
                      (02374) ;
                      (02375) ; APS-OUTPUT ADDRESS SEQUENCE
                      (02376) ; RW(J), RW(J+1), IW(J), IW(J+1), RW(J+2),
                      (02377) ; RW(J+3), IW(J+2), IW(J+3)
                      (02378) ; WHERE J= BIT REVERSAL OF K
                      (02379) ; THE DIFFERENCE, D(K) = J(K+1)-J(K) IS
                      (02380) ; PROVIDED BY THE P3 SUBROUTINE
                      (02381) ;
                      (02382) ;
                      (02383) CSMS    BEGIN APS(CSM)
                      (02384) ;
A00 04564 00202540    (02385) CSMSS   JSN(CSMU,P2)
                      (02386) ;
A01 04566 02C00000    (02387) CSML6S  LOAD(BRO,MSS,L,TF)          INPT RU(0) [UBASE BOUND]
A02 0456B 04500000    (02388) CSML0S  LOAD(BR1,MSS)               [USIZE-1 BOUND]
A03 0456A 06300672    (02389)         JUMP(CSMF,RA),SET           TURN ON APU
                      (02390) ;
A04 0456C 0BB20000    (02391) CSML    SUB(BRO,MSS,TF)             INPT RU(K+N/4) [HU BOUND]
                      (02392) ;
A05 0456E 0AB20000    (02393) CSML1S  SUB(BRO,MSS,TF)             INPT RU(K) [QU-USEP BOUND]
A06 04570 0CHA0000    (02394) CSMF    ADD(BRO,MSS,TF)             INPT RU(K+N/2) [HU BOUND]
A07 04572 0FBA0002    (02395)         ADD(BRO,2,TF)               INPT IU(K+N/2)
A0B 04574 10B20000    (02396) CSML2S  SUB(BRO,MSS,TF)             INPT IU(K) [HU BOUND]
                      (02397) ;
```

CSM - COMPLEX FFT SCRAMBLE (PO - INPUT)

```
A09 04576 128A0000 (02398) CSML3S  ADD(BRO,MSS,TF)          INPT IU(K+N/4) (QU BOUND)
A0A 04578 148A0000 (02399) CSML4S  ADD(BRO,MSS,TF)          INPT IU(K+3N/4) (HU BOUND)
A0B 0457A 16820002 (02400)         SUB(BRO,2,TF)            INPT RU(K+3N/4)
                   (02401) ;
A0C 0457C 18190484 (02402)         SUBL(BR1,4),JUMPP(CSML)
                   (02403) ;
A0D 0457E 1A820000 (02404) CSML5S  SUB(BRO,MSS,TF)          INPT RU(LAST) (HU BOUND)
A0E 04580 1C305977 (02405)         JUMP(CR4AF,WI),SET
                   (02406) ;
```

```
                       (02407) '         CSM-SCRAMBLE SUBROUTINE (P3 - OUTPUT)
                       (02408) ;
                       (02409) ; DATA POINT ADDRESS AT ENTRY 4*[J(K)-N]+WBASE
                       (02410) ; DATA OUTPUT ADDRESS GENERATED [4J(K+1)]+WBASE
                       (02411) ;
                       (02412) ; ITERATION EQUATION, J=J(K)=K,BIT REVERSED
                       (02413) ; 4J(K+1)=4*[J(K)-N]+DM
                       (02414) ; WHERE M=NUMBER OF TRAILING 1'S IN (K)
                       (02415) ;
                       (02416) ; BINDING CONSTANTS
                       (02417) ;
                       (02418) ;          D0=4*(3*N/2)
                       (02419) ;          D1=D0/2
                       (02420) ;          D2=D1/2
                       (02421) ;          D3=D2/2
                       (02422) ;          D4=D3/2
                       (02423) ;          D5=D4/2
                       (02424) ;          D6=D5/2
                       (02425) ;          D7=D6/2
                       (02426) ;;
                       (02427) ;
A0F 04582 1F107840 (02428) SCRM    JSN(ANGLE,P1)               LOADS P1 FOR CR4A
                       (02429) ;
A10 04584 20RE0000 (02430) SCRM0   ADD(RW0,MSS,TF,C)           [D0 BOUND]
A11 04586 22RE0000 (02431) SCRM1S  ADD(RW0,MSS,TF,C)           [D1 BOUND]
A12 04588 24RE0000 (02432) SCRM2S  ADD(RW0,MSS,TF,C)           [D0 BOUND]
A13 0458A 26RE0000 (02433) SCRM3S  ADD(RW0,MSS,TF,C)           [D2 BOUND]
A14 0458C 28RE0000 (02434) SCRM4S  ADD(RW0,MSS,TF,C)           [D0 BOUND]
A15 0458E 2ARE0000 (02435) SCRM5S  ADD(RW0,MSS,TF,C)           [D1 BOUND]
A16 04590 2CRE0000 (02436) SCRM6S  ADD(RW0,MSS,TF,C)           [D0 BOUND]
                       (02437) ;
A17 04592 2E201AE8 (02438)         JUMPS(SCRM4,AF0),CLEAR
A18 04594 30RE0000 (02439) SCRM7S  ADD(RW0,MSS,TF,C)           [D3 BOUND]
A19 04596 32301068 (02440)         JUMP(SCRM0,AF0),SET
                       (02441) ;
A1A 04598 34201DE9 (02442) SCRM4   JUMPS(SCRM5,AF1),CLEAR
A1B 0459A 36RE0000 (02443) SCRM8S  ADD(RW0,MSS,TF,C)           [D4 BOUND]
A1C 0459C 38301069 (02444)         JUMP(SCRM0,AF1),SET
                       (02445) ;
                       (02446) ;
A1D 0459E 3A2020EA (02447) SCRM5   JUMPS(SCRM6,AF2),CLEAR
A1E 045A0 3CRE0000 (02448) SCRM9S  ADD(RW0,MSS,TF,C)           [D5 BOUND]
A1F 045A2 3E30106A (02449)         JUMP(SCRM0,AF2),SET
                       (02450) ;
```

```
 A20 045A4 402023EB (02451) SCRM6    JUMPS(SCRM7,AF3),CLEAR
 A21 045A6 428F0000 (02452) SCRM10$ ADD(HWO,MSS,TF,C)              [D6 BOUND]
 A22 045A8 4430106B (02453)          JUMP(SCRM0,AF3),SET
                    (02454) ;
 A23 045AA 468F0000 (02455) SCRM7    ADD(HWO,MSS,TF,C)              [D7 BOUND]
 A24 045AC 48001060 (02456)          JUMP(SCRM0)
                    (02457) ;
```

```
                      (02458) '         CSM-SCRAMBLE SUBROUTINE (OUTPUT - P2)
                      (02459) ;
                      (02460) ;
A25 045AE 4A300F40    (02461) CSMO     JSN(SCRM,P3)
A26 045B0 4CC00000    (02462) CSM01S   LOAD(BW0,MSS,L,TF)              OTPT RW(0) (WBASE BOUND)
A27 045B2 4F112953    (02463)          MOVB(BW1,BR1),JUMP(CSMOE)       BW1=WSIZE-1
                      (02464) ;
A28 045B4 50060000    (02465) CSMOL    SUB(BW0,MSS,C)                  (4N+12 BOUND)
                      (02466) ;
                      (02467) ; PRIOR TO JUMP TO P3,BW0=4J(K)-4N
                      (02468) ; SCRAMBLE SUBROUTINE OUTPUTS RW(J)
                      (02469) ; LEAVING BW0=4J(K+1)
                      (02470)
A29 045B6 52BA0004    (02471) CSMOE    ADD(BW0,4,TF)                   OTPT RW(J+1)
A2A 045B8 54B20002    (02472)          SUB(BW0,2,TF)                   OTPT IW(J)
A2B 045BA 56BA0004    (02473)          ADD(BW0,4,TF)                   OTPT IW(J+1)
                      (02474) ;
A2C 045BC 58BA0004    (02475)          ADD(BW0,4,TF)                   OTPT IW(J+2)
A2D 045BE 5ABA0004    (02476)          ADD(BW0,4,TF)                   OTPT IW(J+3)
A2E 045C0 5C820006    (02477)          SUB(BW0,6,TF)                   OTPT RW(J+2)
A2F 045C2 5EBA0004    (02478)          ADD(BW0,4,TF)                   OTPT RW(J+3)
                      (02479) ;
A30 045C4 60112BB4    (02480)          SUBL(BW1,4),JUMPP(CSMOL)
                      (02481) ;
```

```
'         SUCCESSIVE RADIX-4 STAGES (OUTPUT - P2)
;         PROGRAM ASSUMES SCRAMBLE COMPLETED


; 12/14/77
; SCRAMBLE LEAVES APS AS FOLLOWS
; INPUT - LAST COMMAND,JUMP(CR4AF,WI),SET
; INPUT-P1 LOCATED AT ANGLE
; OUTPUT - ACTIVE, IN P2
;
; SS=STAGE SEPARATION, DATA SEPARATION A B C D
; STARTING VALUES(/4)ARE
; SSI=1, SCRAMBLE ONLY PRECEEDING        N=4^M
; SSI=2, SCRAMBLE PLUS RADIX2 PRECEEDING N=2*4^M
; SSI=3, SCRAMBLE PLUS RADIX3 PRECEEDING N=3*4^M
; SSI=4, SCRAMBLE PLUS RADIX4 PRECEEDING N=4*4^M
; SSI=6, SCRAMBLE PLUS RADIX6 PRECEEDING N=6*4^M
;
;
; COSINE TABLE IS A REAL BUFFER WITH CONTENTS:
;     CT(K)=COS(2*PI*K/CSIZE)
; RESTRICTION.
; CSIZE MUST BE MULTIPLE OF N, FFT SIZE
;
; COSINE TABLE BINDING PARAMETERS
;
;         HPI=CSEP*(CSIZE/4)            90 DEG SEPARATION
;
; REGISTER USAGE
;         BRO/BW0                       DATA ADDRESSES
;         BR1/BW1                       # OF USES OF A COSINE, COUNTER
;         BW2                           # OF COSINES USED, COUNTER
;         BW3                           STAGE SEPARATION
;         BR2                           SEP BETWEEN COSINES WITHIN A SET
;         BR3                           SEP BETWEEN SETS OF COSINES
        EJECT
```

```
                        (02518)
 A31 045C6 62700000 (02519) CR4A01 LOAD(BW3,MSS)                    [SSI BOUND]
                        (02520) ;
                        (02521) ;  CR4A-APS OUTPUT PROGRAM-STAGE INITIALIZATION
                        (02522) ;
                        (02523) ;
 A32 045C8 6431002F (02524) CR4A02  ADDB(BW3,BW3)
 A33 045CA 6631002F (02525)         ADDB(BW3,BW3)              SET SEPARATION FOR STAGE
 A34 045CC 68400000 (02526) CR4A1S  LOAD(BW0,MSS)              SET BW0=WB-4 [WBASE BOUND]
 A35 045CE 6A010034 (02527)         SUBL(BW0,4)                SET WBASE-4
 A36 045D0 6C210016 (02528)         MOVB(BW2,BW3)
 A37 045D2 6E090010 (02529)         MOVB(BR0,BW0)              SET UP INPUT BASE
 A38 045D4 70200037 (02530)         CLEAR(W1)                  START UP INPUT
                        (02531) ;
                        (02532) ;TEST FOR LAST STAGE
                        (02533) ; IF SO, CHANGE OUTPUT TO Y BUFFER INSTEAD
                        (02534) ; OF THE W BUFFER
                        (02535) ;
 A39 045D6 72500000 (02536) CR4A3S  LOAD(BW1, MSS)             [USIZE-1 BOUND]
 A3A 045D8 741149A6 (02537)         SUBB(BW1, BW3), JUMPP(CR4A03) JUMP IF NOT LAST STAGE
                        (02538) ;
                        (02539) ;LAST STAGE OUTPUT ADDRESSES
                        (02540) ;
 A3B 045DA 76400000 (02541) CR4A10S LOAD(BW0, MSS)             [YBASE BOUND]
 A3C 045DC 78500000 (02542) CR4A11S LOAD(BW1, MSS)             [YSEP*YSIZE/4 BOUND]
                        (02543) ;
 A3D 045DE 7A0A0000 (02544) CR4A0F  ADD(BW0, MSS)              [YSEP*YSIZE BOUND]
                        (02545) ;
 A3E 045E0 7C810022 (02546)         SUBB(BW0,BW1,TF)           ZR
 A3F 045E2 7E8A0002 (02547)         ADD(BW0,2,TF)              ZI
 A40 045E4 80810022 (02548)         SUBB(BW0,BW1,TF)           YI
 A41 045E6 82820002 (02549)         SUB(BW0,2,TF)              YR
 A42 045E8 84810022 (02550)         SUBB(BW0,BW1,TF)           XR
 A43 045EA 868A0002 (02551)         ADD(BW0,2,TF)              XI
 A44 045EC 88810022 (02552)         SUBB(BW0,BW1,TF)           WI
 A45 045EE 8A820002 (02553)         SUB(BW0,2,TF)              WR
                        (02554) ;
                        (02555) ;TEST FFT DONE
                        (02556) ;
 A46 045F0 8C0A0000 (02557) CR4A12S ADD(BW0,MSS)               [YSEP BOUND]
 A47 045F2 8E203DAA (02558)         JUMPC(CR4A0F,AF2),CLEAR
 A48 045F4 90203470 (02559)         JUMP(CR4A1S,R0),CLEAR      HALT APS OUTPUT
                        (02560)
 A49 045F6 92500000 (02561) CR4A03  LOAD(BW1,MSS)              SET BUTTERFLY COUNT [USIZE-1 BOUND]
```

```
A4A 045F8 940A0000 (02562) CR4A2$  ADD(BW0,MSS)                SET STARTING ADDRESS [4N+4 BOUND]
                   (02563)
                   (02564)
A4B 045FA 96810026 (02565) CR4A04  SUBB(BW0,BW3,TF)            ZR
A4C 045FC 98BA0002 (02566)         ADD(BW0,2,TF)               ZI
A4D 045FE 9A810026 (02567)         SUBB(BW0,BW3,TF)            YI
A4E 04600 9C820002 (02568)         SUB(BW0,2,TF)               YR
A4F 04602 9E810026 (02569)         SUBB(BW0,BW3,TF)            XR
A50 04604 A0BA0002 (02570)         ADD(BW0,2,TF)               XI
A51 04606 A2810026 (02571)         SUBB(BW0,BW3,TF)            WI
A52 04608 A4820002 (02572)         SUB(BW0,2,TF)               WR
                   (02573) ;TEST FOR NEW COSINES, ELSE OUTPUT NEXT 4 POINTS
                   (02574) ;
A53 0460A A61146A6 (02575)         SUBB(BW1,BW3),JUMPP(CR4A04)
                   (02576) ;TEST STAGE DONE, ELSE GET NEW COSINES
                   (02577) ;
A54 0460C A82049AA (02578)         JUMPC(CR4A03,AF2),CLEAR
                   (02579) ;
                   (02580) ;STOPPING SHORT CHECK IS HERE
                   (02581) ;
A55 0460E AA500000 (02582) CR4A13$ LOAD(BW1, MSS)              [FFTSIZE-1 BOUND]
A56 04610 AC1132A6 (02583)         SUBB(BW1, BW3), JUMPP(CR4A02)
A57 04612 AE203470 (02584)         JUMP(CR4A1$, R0), CLEAR     HALT APS OUTPUT
                   (02585) ;
                   (02586)
```

```
                     (02587) '          SUCCESSIVE RADIX-4 STAGES (INPUT - P0)
                     (02588)
A58 04614 80300037 (02589) CR4A1    SET(W1)
                     (02590) ;STAGE INITIALIZATION
                     (02591) ;
A59 04616 82600000 (02592) CR4AF    LOAD(BR2,0)                     SET COSINE VALUE=0
A5A 04618 843D0015 (02593)          MOVB(BR3,BR2,C)
                     (02594) ;
                     (02595) ; ANGLE SUBROUTINE,P1,INSERTS COSINE SEPARATION
                     (02596) ; FOR STAGE INTO BR3
                     (02597)
                     (02598)
A5B 0461A 86216571 (02599)          SUBL(BW2,1),JUMP(CR4A3)         BW2=#OF COSINES-1
                     (02600)
A5C 0461C 88890032 (02601) CR4A2    SUBL(BR0,2,TF)                  AR
                     (02602) ;COSINE ENTRY
                     (02603) ;
A5D 0461E 8A500000 (02604) CR4A4S   LOAD(BR1,MSS,L)                 (CHASE BOUND)
A5E 04620 8C29002F (02605)          ADDB(BR2,BR3)                   BR2=NEXT ANGLE
                     (02606)
A5F 04622 8E99002D (02607)          ADDB(BR1,BR2,TF)                COSX
A60 04624 C09A0000 (02608) CR4A5S   ADD(BR1,MSS,TF)                 SINX [HPI BOUND]
A61 04626 C299002D (02609)          ADDB(BR1,BR2,TF)                SIN2X
A62 04628 C4920000 (02610) CR4A6S   SUB(BR1,MSS,TF)                 COS2X [HPI BOUND]
A63 0462A C699002D (02611)          ADDB(BR1,BR2,TF)                COS3X
A64 0462C C89A0000 (02612) CR4A7S   ADD(BR1,MSS,TF)                 SIN3X [HPI BOUND]
A65 0462E CA0A0000 (02613) CR4A3    ADD(BR0,MSS)                    SET BR0 TO 4N+COS# [4N+4 BOUND]
A66 04630 CC500000 (02614) CR4A8S   LOAD(BR1,MSS)                   SET COSINE USE COUNTER [USIZE-1 BOUND]
                     (02615)
                     (02616) ; CR4A-APS BUTTERFLY INPUT
                     (02617) ;
                     (02618)
A67 04632 CE890026 (02619) CR4A4    SUBB(BR0,BW3,TF)                DR
A68 04634 D089003A (02620)          ADDL(BR0,2,TF)                  DI
A69 04636 D2890026 (02621)          SUBB(BR0,BW3,TF)                BI
A6A 04638 D4890032 (02622)          SUBL(BR0,2,TF)                  BR
A6B 0463A D6890026 (02623)          SUBB(BR0,BW3,TF)                CR
A6C 0463C D889003A (02624)          ADDL(BR0,2,TF)                  CI
A6D 0463E DA890026 (02625)          SUBB(BR0,BW3,TF)                AI
                     (02626)
                     (02627) ; TEST IF NEW COSINE NEEDED, ELSE INPUT NEXT 4 POINTS
                     (02628) ;
A6E 04640 DC1976A6 (02629)          SUBB(BR1,BW3),JUMPP(CR4A5)
A6F 04642 DE30002B (02630)          SET(AF0)                        TELL APU COSINES COMING
```

```
                     (02631) ; TEST IF STAGE DONE, ELSE GET NEW COSINES
                     (02632)
A70 04644 F0215CB4   (02633)         SUBL(BW2,4),JUMPP(CR4A2)
A71 04646 F2300029   (02634)         SET(AF1)                       TELL APU STAGE DONE
                     (02635) ;********* CHANGE VALUE HERE TO STOP SHORT ***********
                     (02636) ;TO STOP SHORT 1 STAGE, BIND USIZE/4-1 HERE
A72 0464B F4500000   (02637) CR4A9S  LOAD(BR1,MSS)                  (FFTSIZE-1 BOUND)
A73 0464A F6B90032   (02638)         SUBL(BR0,2,TF)                 LAST DATA POINT
                     (02639)
                     (02640) ; TEST IF FFT DONE, ELSE START NEXT STAGE
                     (02641)
A74 0464C FB195BA6   (02642)         SUBB(BR1,BW3),JUMPP(CR4A1)     GO BACK
A75 0464E FA205971   (02643)         JUMP(CR4AE,RI),CLEAR           HALT APS INPUT
                     (02644)
A76 04650 FCB90032   (02645) CR4A5   SUBL(BR0,2,TF)                 AR
A77 04652 FE006760   (02646)         JUMP(CR4A4)                    BACK TO BUTTERFLY
                     (02647)
```

```
                        (02648) '          ANGULAR SEPARATION SUBROUTINE (INPUT - P1)
                        (02649) ; S1ANG=HPI/SSI
                        (02650) ; S2ANG=S1ANG/4
                        (02651) ; S3ANG=S2ANG/4
                        (02652) ; S4ANG=S3ANG/4
                        (02653) ; S5ANG=S4ANG/4
                        (02654) ; S6ANG=S5ANG/4
                        (02655) ; S7ANG=S6ANG/4
                        (02656)
                        (02657)
A78 04654 F0300030 (02658) ANGLE    SET(R0)
A79 04656 F23F0000 (02659) ANGL1$   ADD(BR3,M$S,C)             [S1ANG BOUND]
A7A 04658 F43F0000 (02660) ANGL2$   ADD(BR3,M$S,C)             [S2ANG BOUND]
A7B 0465A F63F0000 (02661) ANGL3$   ADD(BR3,M$S,C)             [S3ANG BOUND]
A7C 0465C F83F0000 (02662) ANGL4$   ADD(BR3,M$S,C)             [S4ANG BOUND]
A7D 0465E FA3F0000 (02663) ANGL5$   ADD(BR3,M$S,C)             [S5ANG BOUND]
A7E 04660 FC3F0000 (02664) ANGL6$   ADD(BR3,M$S,C)             [S6ANG BOUND]
A7F 04662 FE3F0000 (02665) ANGL7$   ADD(BR3,M$S,C)             [S7ANG BOUND]
                        (02666) ;
          00000000 (02667)  CR4ASA=#C                          ASSIGN VALUE TO CHAIN
    04664              (02668)        END                      #A-1  END OF MODULE
                        (02669) ;
          00004664 (02670) CSMST    #L = #L + 2*0
                        (02671) ;
          00000100 (02672)  CR4ASZ = #L - CSMS
                        (02673) ;
                        (02674) ;
```

```
                    (02675) *        VMOV
                    (02676) *
                    (02677) *        AP ROUTINES FOR VMOV
                    (02678) *
                    (02679) *
                    (02680) *
                    (02681) * APU3-VMOV
                    (02682) *
                    (02683) *        BINDS TO APS3-V1124
                    (02684) *
                    (02685) *        EQ.  Y=U
                    (02686) *
                    (02687)          EVEN                     START ON WORD BOUNDRY
    04664 0000      (02688)          DATA VMOVSSA             START ADDRESS
    04665 0009      (02689)          DATA VMOVSSZ             SIZE
                    (02690) *
                    (02691) VMOVS    BEGIN APU(VMOV)          START OF APU MODULE
          00000000  (02692) #A=0
                    (02693) *
A00 04666 08DC0000  (02694) VMOVSSA  MOV(IQ,OQ)\NOP           DUMMY SCALAR 1 TO DUMMY OUT 1
A01 04668 08DC0000  (02695)          MOV(IQ,OQ)\NOP           DUMMY SCALAR 1 TO DUMMY OUT 2
A02 0466A 08FC0000  (02696)          MOV(IQA,OQ)\NOP          DUMMY SCALAR 1 TO DUMMY OUT 3
A03 0466C 08FC0000  (02697)          MOV(IQA,OQ)\NOP          DUMMY SCALAR 2 TO DUMMY OUT 4
                    (02698) *
          00000004  (02699) #1=#A
A04 0466E 08FC0000  (02700)          MOV(IQA,OQ)\NOP          LOOP TO MOVE U INTO Y
A05 04670 901C0004  (02701)          JUMPC(#1,FO)
A06 04672 20322032  (02702)          CLEAR(RA)                HALT
                    (02703) *
A07 04674 00000000  (02704)          NOP
A08 04676 10000000  (02705)          JUMP(VMOVSSA)            IF RESTART WITHOUT RELOAD
          00000009  (02706) VMOVSSZ=#A-VMOVSSA
    04678           (02707)          END
```

```
DEFINE TOP OF MODULE

                 (02708) 'DEFINE TOP OF MODULE
                 (02709) *
        00004678 (02710) TOESCUR=#L              DEFINE TOP LOCATION OF MODULE
                 (02711) *
                 (02712) *
    04678        (02713)         END
```

DEFINE TOP OF MODULE

```
AFDTSORG:   00BEB (00027) (00165) (00684) (00701) (00706) (00711) (00716) (01724) (01728)
ANGL1S:     00079 (01637) (02659)
ANGL2S:     0007A (01640) (02660)
ANGL3S:     0007B (01643) (02661)
ANGL4S:     0007C (01646) (02662)
ANGL5S:     0007D (01649) (02663)
ANGL6S:     0007E (01652) (02664)
ANGL7S:     0007F (01655) (02665)
ANGLE:      00078 (02428) (02658)
APSASSS:    00245 (00028)
APSBNDR:    00EEA (00029)
APSBNDRO:   00E63 (00030) (01756)
APSBNDR1:   00E2D (00031)
APSBSL:     0024D (00032) (01622)
APSCSSC:    0024B (00033)
APSDONE:    00EB2 (00034)
APSGO:      0000C (00035)
APSG1:      0000D (00036)
APSPBFF:    00010 (00037)
APSSAID:    0000A (00038)
APSSSCLR:   0000F (00039)
APSSSS:     00009 (00040)
APSSBMO:    00004 (00041)
APSSNULL:   00000 (00169) (00189) (00193) (00197) (00201) (00221) (00225) (00229) (00233) (00253)
                  (00257) (00261) (00265) (00293) (00297) (00321) (00325) (00329) (00357) (00361)
                  (00381) (00385) (00413) (00417) (00421) (00425) (00445) (00449) (00453) (00457)
                  (00477) (00481) (00485) (00489) (00493) (00497) (00501) (00505) (00509) (00513)
                  (00517) (00521) (00525) (00529) (00533) (00537) (00541) (00545) (00549) (00553)
                  (00557) (00561) (00565) (00569) (00573) (00577) (00581) (00585) (00589) (00593)
                  (00597) (00601) (00605) (00609) (00613) (00617) (00621) (00625) (00629) (00633)
                  (00637) (00641) (00645) (00649) (00653) (00657) (00661) (00665) (00669) (00673)
                  (00677) (00681)
APSR:       1EEC0 (00042)
APUSNULL:   00000 (00168) (00188) (00192) (00196) (00200) (00212) (00220) (00224) (00228) (00232)
                  (00252) (00256) (00260) (00264) (00292) (00296) (00320) (00324) (00328) (00356)
                  (00360) (00380) (00384) (00412) (00416) (00420) (00424) (00444) (00448) (00452)
                  (00456) (00476) (00480) (00484) (00488) (00492) (00496) (00500) (00504) (00508)
                  (00512) (00516) (00520) (00524) (00528) (00532) (00536) (00540) (00544) (00548)
                  (00552) (00556) (00560) (00564) (00568) (00572) (00576) (00580) (00584) (00588)
                  (00592) (00596) (00600) (00604) (00608) (00612) (00616) (00620) (00624) (00628)
                  (00632) (00636) (00640) (00644) (00648) (00652) (00656) (00660) (00664) (00668)
                  (00672) (00676) (00680)
BCTSAD:     00604 (00044) (01378) (01534) (01603)
BCTSAT:     00686 (00045) (01373)
```

DEFINE TOP OF MODULE

```
  BCTSBA:     00582 (00046) (01598) (01669) (01685) (01707)
  BITSG0:     00000 (00047)
U C1120$:     00000 (00393)
U C1124AS:    00000 (00373)
U C1124BS:    00000 (00377)
U C2120$:     00000 (00389)
U C2124AS:    00000 (00365) (00369)
U CCVMULS:    00000 (00368)
  CLRSG0G1:   00792 (00049)
U CPOLARS:    00000 (00376)
  CR4SSZ:     0006D (01848) (02084)
  CR4ASA:     00000 (02350) (02667)
  CR4ASZ:     00100 (02349) (02672)
  CR4A1:      0005B (02589) (02642)
  CR4A1S:     00034 (01675) (02526) (02559) (02584)
  CR4A10$:    0003B (01708) (02541)
  CR4A11S:    0003C (01588) (02542)
  CR4A12S:    00046 (01564) (02557)
  CR4A13$:    00055 (01444) (02582)
  CR4A2:      0005C (02601) (02633)
  CR4A2S:     0004A (01522) (02562)
  CR4A3:      00065 (01523) (02599) (02613)
  CR4A3$:     00039 (01415) (02536)
  CR4A4:      00067 (02619) (02646)
  CR4A4S:     0005D (01599) (02604)
  CR4A5:      00076 (02629) (02645)
  CR4A5S:     00060 (01611) (02608)
  CR4A6S:     00062 (01612) (02610)
  CR4A7S:     00064 (01613) (02612)
  CR4A8S:     00066 (01416) (02614)
  CR4A9$:     00072 (01443) (02637)
  CR4AE:      00059 (02405) (02592) (02643)
  CR4AD1:     00031 (01625) (01629) (02519)
  CR4AD2:     00032 (02524) (02583)
  CR4AD3:     00049 (01413) (02537) (02561) (02578)
  CR4AD4:     0004B (02565) (02575)
  CR4ADE:     0003D (01575) (01577) (01579) (01582) (02544) (02558)
  CR4FB:      0003F (02013) (02052)
  CR4FC:      0003B (02003) (02053)
  CR4FE:      00053 (01999) (02044)
  CR4FS:      0002B (01886) (01978)
  CR4FSA:     00029 (01979) (02078)
  CR4ISSZ:    0006D (02108) (02330)
  CR4IB:      0003F (02259) (02298)
```

DEFINE TOP OF MODULE

```
CR4IC:     0003B (02249) (02299)
CR4IF:     00053 (02245) (02290)
CR4IS:     00028 (02133) (02224)
CR4ISA:    00029 (02225) (02324)
CSMS:      04564 (00703) (00708) (00713) (00718) (01413) (01414) (01415) (01416) (01443) (01444)
                 (01472) (01475) (01484) (01485) (01486) (01487) (01490) (01491) (01494) (01497)
                 (01500) (01503) (01506) (01509) (01522) (01523) (01526) (01564) (01575) (01577)
                 (01579) (01582) (01588) (01599) (01611) (01612) (01613) (01625) (01629) (01637)
                 (01640) (01643) (01646) (01649) (01652) (01655) (01670) (01675) (01686) (01708)
                 (01792) (01793) (01794) (01795) (01796) (02347) (02383) (02672)
CSMS1:     04664 (02346) (02670)
CSMSS:     00000 (02347) (02385)
CSM2S:     043A6 (00702) (00707) (01865)
CSM2SSA:   00000 (01847) (01868) (02082) (02084)
CSM2F:     00003 (01869) (01873)
CSM2L:     00002 (01871) (01881)
CSM4F:     0000D (01868) (01904)
CSM4FS:    0001B (01909) (01926)
CSM4I:     0000D (02115) (02150)
CSM4IS:    0001B (02155) (02172)
CSMF:      00006 (01793) (02389) (02394)
CSML:      00004 (01792) (02391) (02402)
CSML0S:    00002 (01414) (02388)
CSML1S:    00005 (01475) (02393)
CSML2S:    00008 (01794) (02396)
CSML3S:    00009 (01472) (02398)
CSML4S:    0000A (01795) (02399)
CSML5S:    0000D (01796) (02404)
CSML6S:    00001 (01686) (02387)
CSMO:      00025 (02385) (02461)
CSMO1S:    00026 (01670) (02462)
CSMOF:     00029 (02463) (02471)
CSMOL:     00028 (01526) (02465) (02480)
CSPUSNOS:  021FC (00050) (00174) (00178) (00182) (00186) (00190) (00194) (00198) (00202) (00206)
                 (00210) (00214) (00218) (00222) (00226) (00230) (00234) (00238) (00242) (00246)
                 (00250) (00254) (00258) (00262) (00266) (00270) (00274) (00278) (00282) (00286)
                 (00290) (00294) (00298) (00302) (00306) (00310) (00314) (00318) (00322) (00326)
                 (00330) (00334) (00338) (00342) (00346) (00350) (00354) (00358) (00362) (00366)
                 (00370) (00374) (00378) (00382) (00386) (00390) (00394) (00402) (00406) (00410)
                 (00414) (00418) (00422) (00426) (00430) (00434) (00438) (00446) (00450) (00454)
                 (00458) (00462) (00466) (00470) (00474) (00478) (00482) (00486) (00490) (00510)
                 (00514) (00518) (00522) (00526) (00530) (00534) (00538) (00542) (00546) (00550)
                 (00554) (00558) (00562) (00566) (00570) (00574) (00578) (00582) (00586) (00590)
                 (00594) (00598) (00602) (00606) (00610) (00614) (00618) (00622) (00626) (00630)
```

```
                DEFINE TOP OF MODULE

                     (00634) (00638) (00642) (00646) (00650) (00654) (00658) (00662) (00666) (00670)
                     (00674) (00678) (00682) (00687)
U CVMULS:     00000 (00364)
U CVRCPS:     00000 (00372)
U CXMULS:     00000 (00388)
U CXMULRS:    00000 (00392)
U D1113S:     00000 (00429) (00433)
U D1186S:     00000 (00437)
U D2116S:     00000 (00441)
U DCVMS:      00000 (00440)
U DDIFFS:     00000 (00428)
U DFIL22S:    00000 (00436)
U DINTGS:     00000 (00432)
  DMYS:       00794 (00052) (01037) (01038) (01039) (01040) (01131) (01132) (01133) (01134) (01234)
                    (01235) (01236) (01237)
  ERRORS:     01AFA (00054)
U F11AS:      00000 (00469)
U F11CS:      00000 (00461) (00473)
U F21BS:      00000 (00465)
U FEDSS:      00000 (00468)
  FEDSSDSP:   00AA4 (00468)
U FF3SSSM:    00000 (00494)
  FFTSCBSZ:   0079A (00056)
  FFTSSET:    04276 (00704) (00714) (01367)
  FFTNS:      000CC (00694) (00701)
  FFTNBS:     000CD (00695) (00706)
  FFTNIS:     000CE (00696) (00711)
  FFTNIBS:    000CF (00697) (00716)
  FLGSCLR:    00000 (00057)
  FLGSG0:     00004 (00058)
  FLGSG1:     00005 (00059) (01626) (01630)
  FLGSR1:     00011 (00060)
  FLGSSET:    00020 (00061) (01630)
U FRX4S:      00000 (00464)
  FRX4SDSP:   00A9F (00464)
U FSC4S:      00000 (00472)
  FSC4SDSP:   00AAA (00472)
U FSCRS:      00000 (00460)
  FSCRSDSP:   00A98 (00460)
  G$FLGS:     0439B (01626) (01630) (01751) (01789)
  HS:         00001 (00063) (01369) (01373) (01413) (01414) (01415) (01416) (01434) (01443) (01444)
                    (01472) (01475) (01484) (01485) (01486) (01487) (01490) (01491) (01494) (01497)
                    (01500) (01503) (01506) (01509) (01522) (01523) (01526) (01527) (01564) (01579)
                    (01582) (01588) (01589) (01611) (01612) (01613) (01625) (01626) (01629) (01630)
```

DEFINE TOP OF MODULE

```
                         (01637) (01640) (01643) (01646) (01649) (01652) (01655) (01662) (01681) (01703)
                         (01751)
  HUSDN:          042AA (01456) (01463)
  HUSLP:          0429F (01454) (01461)
  HUSTHL:         0439F (01453) (01792)
U IFF3SSSM:       00000 (00498)
U IRF3SSSM:       00000 (00506)
  ISM2S:          044B2 (00712) (00717) (02112)
  ISM2SSA:        00000 (02107) (02115) (02328) (02330)
  ISM2F:          00003 (02116) (02120)
  ISM2L:          00002 (02118) (02128)
  LOADSAP:        0187F (00065)
  LOADSAP1:       018BF (00066)
  MSS:            00000 (00703) (01009) (01010) (01012) (01013) (01042) (01043) (01104) (01105) (01107)
                         (01108) (01136) (01137) (01198) (01199) (01200) (01203) (01204) (01206) (01207)
                         (01239) (01240) (01790) (02387) (02388) (02391) (02393) (02394) (02396) (02398)
                         (02399) (02404) (02430) (02431) (02432) (02433) (02434) (02435) (02436) (02439)
                         (02443) (02448) (02452) (02455) (02462) (02465) (02519) (02526) (02536) (02541)
                         (02542) (02544) (02557) (02561) (02562) (02582) (02604) (02608) (02610) (02612)
                         (02613) (02614) (02637) (02659) (02660) (02661) (02662) (02663) (02664) (02665)
  MSKSLBYT:       0FF00 (00068) (01370) (01528) (01590) (01663) (01682) (01704)
  MSKSRBYT:       000FF (00069) (01435)
  PWR2S:          0439D (01374) (01623) (01790)
U RF3SSSM:        00000 (00502)
  RPTFT04:        04276 (01369)
  RPTFT07:        04282 (01413)
  RPTFT08:        04299 (01447)
  RPTFT11:        042E4 (01564)
  RPTFT12:        042E7 (01566)
U S0021S:         00000 (00173) (00177) (00181) (00185)
U S1002S:         00000 (00345) (00349)
U S1011S:         00000 (00333) (00337) (00341)
U S2011S:         00000 (00353)
U SADDS:          00000 (00172)
  SBMS1:          04333 (01624) (01629)
  SBMS2:          04338 (01628) (01633)
U SBMSDCVM:       00000 (00442)
  SBMSFFT:        04352 (00709) (00719) (01661)
U SBMSVPLY:       00000 (00398)
  SCRM:           0000F (02428) (02461)
  SCRM0:          00010 (01484) (02430) (02440) (02444) (02449) (02453) (02456)
  SCRM1S:         00011 (01490) (02431)
  SCRM10S:        00021 (01506) (02452)
  SCRM2S:         00012 (01485) (02432)
```

```
                 DEFINE TOP OF MODULE

  SCRM3S:     00013 (01494) (02433)
  SCRM4:      0001A (02438) (02442)
  SCRM4S:     00014 (01486) (02434)
  SCRM5:      0001D (02442) (02447)
  SCRM5S:     00015 (01491) (02435)
  SCRM6:      00020 (02447) (02451)
  SCRM6S:     00016 (01487) (02436)
  SCRM7:      00023 (01509) (02451) (02455)
  SCRM7S:     00018 (01497) (02439)
  SCRM8S:     00018 (01500) (02443)
  SCRM9S:     0001F (01503) (02448)
U SDIVS:      00000 (00184)
U SDOTS:      00000 (00352)
  SHFTLSR5:   007AF (00072)
  SMA1SSA:    00000 (00732) (00738) (00757)
  SMA1SSZ:    0000F (00733) (00757) (00759)
U SMAXS:      00000 (00344)
U SMAXABS:    00000 (00348)
U SMULS:      00000 (00180)
U SSUBS:      00000 (00176)
U SSUMS:      00000 (00332)
U SSUMABS:    00000 (00336)
U SSUMSQS:    00000 (00340)
  START:      04000 (00096) (00722)
  SVTS:       003B2 (00073)
  SVTSUN1:    003CF (00074)
  SYSSFLGS:   1FFCE (00075)
  TEMSO:      007B4 (00077)
  TOFS:       021FF (00078)
  TOFSCUR:    04678 (00090) (02710)
  TOFSPTR:    0028B (00079) (00088)
  V1124S:     04122 (00237) (00249) (00269) (00273) (00277) (00301) (00305) (00309) (00317) (00401)
                    (00686) (00993) (01000) (01075)
  V1124S0:    00023 (01055) (01060)
  V1124S3:    00012 (01006) (01036)
  V1124S5:    0000C (01014) (01023)
  V1124S7:    00010 (01025) (01030)
  V1124S8:    0001F (01044) (01053)
  V1124SA:    04164 (00996) (01064)
  V1124SI:    0416C (00992) (01072)
  V1124SS:    00002 (00993) (01009)
  V1124SZ:    00062 (00995) (01075)
  V1124AS:    0418C (00205) (01089) (01096) (01170)
  V1124AS0:   00023 (01150) (01155)
```

DEFINE TOP OF MODULE

```
  V1124A$3:  00012 (01101) (01130)
  V1124A$5:  0000C (01109) (01118)
  V1124A$7:  00010 (01120) (01125)
  V1124A$8:  0001F (01138) (01148)
  V1124A$A:  041CF (01092) (01160)
  V1124A$I:  041F6 (01088) (01167)
  V1124A$S:  00002 (01089) (01104)
  V1124A$Z:  00062 (01091) (01170)
U V1124B$:   00000 (00209)
U V1124C$:   00000 (00213)
U V1124D$:   00000 (00217)
U V2116A$:   00000 (00397)
U V2124A$:   00000 (00289)
U V2134A$:   00000 (00241) (00245) (00281) (00285) (00313)
  V2134B$:   041F6 (00405) (01182) (01189) (01274)
  V2134B$0:  0002F (01253) (01258)
  V2134B$3:  0001D (01195) (01233)
  V2134B$8:  0002A (01241) (01251)
  V2134B$A:  0424F (01185) (01263)
  V2134B$I:  04256 (01181) (01271)
  V2134B$S:  00002 (01182) (01198)
  V2134B$Z:  00080 (01184) (01274)
U V3112A$:   00000 (00409)
  VC$2:      00009 (00786) (00793) (00796)
  VC$3:      00013 (00813) (00817)
  VC$4:      00017 (00814) (00821) (00823)
  VC$5:      0001A (00827)
  VC$6:      0001D (00838) (00918)
  VC$7:      00061 (00866) (00922)
  VC$8:      0000F (00802) (00804)
U VCLIP$:    00000 (00300)
U VCOMP$:    00000 (00312)
  VCOS$:     04020 (00404) (00771)
  VCOS$SA:   00000 (00768) (00777) (00927)
  VCOS$SZ:   00065 (00769) (00927) (00929)
  VCOSC$:    040FA (00934) (01217)
U VFIX$:     00000 (00208) (00216)
  VFLT$:     040F6 (00204) (00952)
  VFLT$SA:   00000 (00949) (00955) (00978)
  VFLT$SZ:   00012 (00950) (00978) (00980)
U VLIMIT$:   00000 (00308)
U VLOG16$:   00000 (00288)
U VMAG$:     00000 (00284)
U VMAGSQ$:   00000 (00280)
```

```
                DEFINE TOP OF MODULE

  VMOVS:       04666 (00685) (02691)
  VMOVSSA:     00000 (02688) (02694) (02705) (02706)
  VMOVSSZ:     00009 (02689) (02706)
U VMULS:       00000 (00244)
U VPOLYS:      00000 (00396)
U VPOWS:       00000 (00408)
U VRAMPS:      00000 (00400)
U VRCPS:       00000 (00248)
U VSABS:       00000 (00268)
U VSABSQS:     00000 (00272)
  VSMA1S:      04002 (00236) (00735)
U VSMA2S:      00000 (00240)
U VSQS:        00000 (00276)
U VTAGS:       00000 (00316)
U VTHRSHS:     00000 (00304)
  WS:          00002 (00081) (00701) (00706) (00711) (00716) (01413) (01414) (01415) (01416) (01443)
                     (01444) (01472) (01475) (01484) (01485) (01486) (01487) (01490) (01491) (01494)
                     (01497) (01500) (01503) (01506) (01509) (01522) (01523) (01526) (01564) (01575)
                     (01577) (01579) (01582) (01588) (01599) (01611) (01612) (01613) (01625) (01629)
                     (01637) (01640) (01643) (01646) (01649) (01652) (01655) (01670) (01675) (01686)
                     (01708) (01726) (01727) (01728) (01741) (01792) (01793) (01794) (01795) (01796)
  XX1S:        0428F (01437) (01441)
  XX2S:        04294 (01438) (01443)
  YSPSZ:       04306 (01578) (01582)
  YSPSZ4:      0430A (01581) (01588)
  ZERO:        007BA (00083)

LINES WITH ERRORS:   84  (MAP VERSION 800101.10)   E-  506
```

T A B L E   O F   C O N T E N T S

T A B L E   O F   C O N T E N T S

```
          (00001) *   SNAP-II IOS PACKAGE  ---- APR 18, 1980 ----  PROGRAM # 840001.01A
          (00002) *
          (00003) *
          (00004) *   RELEASE 0.0  -- FIRST DELIVERED VERSION
          (00005) *
          (00006) *   RELEASE 1.0A
          (00007) *   1. CORRECTED READ SCROLL REGISTERS FCB. PR #4000 11/20/79
          (00008) *   2. CORRECTED SPECIAL BINDING MODULE FOR ADMRBS. PR #4001 11/20/79
          (00009) *   3. PR #4002 NOT INCLUDED, ONLY FOR REV 12 CSPU. MUST PATCH OBJECT FILE.
          (00010) *   4. FIXED START ADDRESSES FOR IOS-3. PR #4003 11/20/79.
          (00011) *   5. ADDED IOS-4 SUPPORT. PR #4008 2/13/80
          (00012) *
          (00013) *
          (00014) *
          (00015) *
          (00016) *   DEFINE SYMBOLS FOR SNAP-II EXECUTIVE REL. 3.05   5/22/79
          (00017) *
          (00018) *
000000E5  (00019) ADMRBFCB = 229                          ROTATE BUFFER FCB #
000008F8  (00020) APDTSORG = $8F8                         AP DISPATCH TABLE ORIGIN
00000EFA  (00021) APSBNDR = $EFA                          AP BINDER ROUTINE
          (00022) *
00000604  (00023) BCT$AD = $604
00000686  (00024) BCT$AT = $686
00000582  (00025) BCT$BA = $582
00000040  (00026) BCT$SIZ = 64
0000000B  (00027) BIT$BU = 11                             BUFFER UNDEFINED BIT
00000006  (00028) BIT$IORD = 6
00000009  (00029) BIT$IOWT = 9
          (00030) *
00000286  (00031) CSW$I161 = $286
00000322  (00032) CSW$IOS = $322
00000405  (00033) CSW$SIOB = $405
          (00034) *
00000014  (00035) ERR$IOS = 20
00001AFA  (00036) ERRORS = $1AFA
          (00037) *
000007F8  (00038) FDTS = $7F8
          (00039) *
00000001  (00040) HS = 1
          (00041) *
0000003F  (00042) IDEV$63 = $3F
00001B50  (00043) IDIV$ = $1B50                           INTEGER DIVIDE ROUTINE
00000001  (00044) ILVL$1 = 1
```

```
         00000002 (00045) ILVLS2 = 2
         00000003 (00046) ILVLS3 = 3
         00000502 (00047) ISVTS = $502
         00000080 (00048) ISVTSSIZ = 128
                  (00049) *
         0000FF00 (00050) MSKSLBYT = $FF00
         000000FF (00051) MSKSRBYT = $00FF
         00000000 (00052) MSS = 0
                  (00053) *
         00001C12 (00054) SETSSMB = $1C12
         00000BF8 (00055) SNAPSINL = $BF8
                  (00056) *
         000007B4 (00057) TEMS0 = $7B4                          TEMPORARY STORAGE
         000007B5 (00058) TEMS1 = $7B5
         000007B6 (00059) TEMS2 = $7B6
         00000288 (00060) TOESPTR = $288                        TOP OF EXEC POINTER
                  (00061) *
         00000002 (00062) WS = 2
                  (00063) *
                  (00064) *
                  (00065) *-----DEFINE START LOCATION FOR IOS PACKAGE
                  (00066) *
         00004900 (00067)         START = $4900                 DEFINE START LOCATION
                  (00068) *
                  (00069) *
         00000288 (00070) *L = TOESPTR
                  (00071) *
    0288 0010515A (00072)         ADDR     TOESCUR(,1)          UPDATE TOP OF EXEC POINTER
                  (00073) *
```

```
                    (00074) ! FCB DISPATCH TABLE FOR IOS ROUTINES
                    (00075) *
                    (00076) *
          0000086C  (00077) *L = FDTS + 2 * 66
                    (00078) *
0086C 00004B9C      (00079)  ADDR     LDSS                         66=LOAD IOS SCROLL
0086E 00004CE4      (00080)  ADDR     RNSS                         67=RUN IOS SCROLL
00870 00004D66      (00081)  ADDR     MPRAAS                       68=RUN ADAM AND AOM
                    (00082) *
          00000874  (00083) *L = FDTS + 2 * 70
                    (00084) *
00874 00004D9A      (00085)  ADDR     RSRS                         70=READ SCROLL REGISTERS
00876 00004DB2      (00086)  ADDR     WSRS                         71=WRITE SCROLL REGISTERS
                    (00087) *
          0000087A  (00088) *L = FDTS + 2 * 73
                    (00089) *
0087A 00004E32      (00090)  ADDR     ADAMSSSP                     73=SINGLE OR DUAL CHANNEL SAMPLING
0087C 00004F7C      (00091)  ADDR     ADAMS16S                     74=ADAM-16 ROUTINES
                    (00092) *
                    (00093) *
                    (00094) *
                    (00095) ! FCB DISPATCH TABLE ENTRY FOR ADMRB
                    (00096) *
          00000846  (00097)          *L = AFDTSORG + 6 *(ADMRBFCB - 128)
                    (00098) *
00846 001F5102      (00099)          ADDR ADMRBAPU (R7, 1)
00848 001F511E      (00100)          ADDR ADMRBAPS (R7, 1)
0084A 001050B0      (00101)          ADDR ADMRBS ( , 1, 0)
                    (00102) *
                    (00103) *
```

```
                    (00104) * IOS INTERRUPT TABLES
                    (00105) *
                    (00106) *         SYSTEM STATEWORDS FOR IOS DEVICES
                    (00107) *
                    (00108) *
                    (00109) *
           000002B6 (00110) #L = CSWSI161
                    (00111) *
                    (00112) *
002B6 49F21014 (00113)           CSW       CSWSSTDB,D16SINT1          IOS-2 DEVICE 16: LINE 1
002B8 4A001014 (00114) CSWSI162 CSW       CSWSSTDB,D16SINT2          IOS-2 DEVICE 16: LINE 2
002BA 4A0F1014 (00115) CSWSI163 CSW       CSWSSTDB,D16SINT3          IOS-2 DEVICE 16: LINE 3
002BC 4A161014 (00116) CSWSI171 CSW       CSWSSTDB,D17SINT1          IOS-2 DEVICE 17: LINE 1
002BE 4A241014 (00117) CSWSI172 CSW       CSWSSTDB,D17SINT2          IOS-2 DEVICE 17: LINE 2
002C0 4A321014 (00118) CSWSI173 CSW       CSWSSTDB,D17SINT3          IOS-2 DEVICE 17: LINE 3
002C2 4A3A1014 (00119) CSWSI181 CSW       CSWSSTDB,D18SINT1          IOS-2 DEVICE 18: LINE 1
002C4 4A481014 (00120) CSWSI182 CSW       CSWSSTDB,D18SINT2          IOS-2 DEVICE 18: LINE 2
002C6 4A561014 (00121) CSWSI183 CSW       CSWSSTDB,D18SINT3          IOS-2 DEVICE 18: LINE 3
002C8 4A5F1014 (00122) CSWSI191 CSW       CSWSSTDB,D19SINT1          IOS-2 DEVICE 19: LINE 1
002CA 4A6C1014 (00123) CSWSI192 CSW       CSWSSTDB,D19SINT2          IOS-2 DEVICE 19: LINE 2
002CC 4A7A1014 (00124) CSWSI193 CSW       CSWSSTDB,D19SINT3          IOS-2 DEVICE 19: LINE 3
002CE 4A821014 (00125) CSWSI201 CSW       CSWSSTDB,D20SINT1          IOS-2 DEVICE 20: LINE 1
002D0 4A901014 (00126) CSWSI202 CSW       CSWSSTDB,D20SINT2          IOS-2 DEVICE 20: LINE 2
002D2 4A9F1014 (00127) CSWSI203 CSW       CSWSSTDB,D20SINT3          IOS-2 DEVICE 20: LINE 3
002D4 4AA61014 (00128) CSWSI211 CSW       CSWSSTDB,D21SINT1          IOS-2 DEVICE 21: LINE 1
002D6 4AB41014 (00129) CSWSI212 CSW       CSWSSTDB,D21SINT2          IOS-2 DEVICE 21: LINE 2
002D8 4AC21014 (00130) CSWSI213 CSW       CSWSSTDB,D21SINT3          IOS-2 DEVICE 21: LINE 3
002DA 4ACA1014 (00131) CSWSI221 CSW       CSWSSTDB,D22SINT1          IOS-2 DEVICE 22: LINE 1
002DC 4AD81014 (00132) CSWSI222 CSW       CSWSSTDB,D22SINT2          IOS-2 DEVICE 22: LINE 2
002DE 4AE61014 (00133) CSWSI223 CSW       CSWSSTDB,D22SINT3          IOS-2 DEVICE 22: LINE 3
002E0 4AEE1014 (00134) CSWSI231 CSW       CSWSSTDB,D23SINT1          IOS-2 DEVICE 23: LINE 1
002E2 4AFC1014 (00135) CSWSI232 CSW       CSWSSTDB,D23SINT2          IOS-2 DEVICE 23: LINE 2
002E4 4B0A1014 (00136) CSWSI233 CSW       CSWSSTDB,D23SINT3          IOS-2 DEVICE 23: LINE 3
                    (00137) *
                    (00138) *
                    (00139) *
                    (00140)           EJECT
```

```
                   (00141) *
                   (00142) *        INITIAL VALUES FOR SYSTEM STATEWORDS FOR IOS DEVICES
                   (00143) *
                   (00144) *
          00000322 (00145) *L = CSW$IOS
                   (00146) *
                   (00147) *
00322 49F21014 (00148)          CSW      CSW$STDB,D16$INT1          IOS-2 DEVICE 16: LINE 1
00324 4A001014 (00149)          CSW      CSW$STDB,D16$INT2          IOS-2 DEVICE 16: LINE 2
00326 4A0F1014 (00150)          CSW      CSW$STDB,D16$INT3          IOS-2 DEVICE 16: LINE 3
00328 4A161014 (00151)          CSW      CSW$STDB,D17$INT1          IOS-2 DEVICE 17: LINE 1
0032A 4A241014 (00152)          CSW      CSW$STDB,D17$INT2          IOS-2 DEVICE 17: LINE 2
0032C 4A321014 (00153)          CSW      CSW$STDB,D17$INT3          IOS-2 DEVICE 17: LINE 3
0032E 4A3A1014 (00154)          CSW      CSW$STDB,D18$INT1          IOS-2 DEVICE 18: LINE 1
00330 4A481014 (00155)          CSW      CSW$STDB,D18$INT2          IOS-2 DEVICE 18: LINE 2
00332 4A561014 (00156)          CSW      CSW$STDB,D18$INT3          IOS-2 DEVICE 18: LINE 3
00334 4A5F1014 (00157)          CSW      CSW$STDB,D19$INT1          IOS-2 DEVICE 19: LINE 1
00336 4A6C1014 (00158)          CSW      CSW$STDB,D19$INT2          IOS-2 DEVICE 19: LINE 2
00338 4A7A1014 (00159)          CSW      CSW$STDB,D19$INT3          IOS-2 DEVICE 19: LINE 3
0033A 4A821014 (00160)          CSW      CSW$STDB,D20$INT1          IOS-2 DEVICE 20: LINE 1
0033C 4A901014 (00161)          CSW      CSW$STDB,D20$INT2          IOS-2 DEVICE 20: LINE 2
0033E 4A9F1014 (00162)          CSW      CSW$STDB,D20$INT3          IOS-2 DEVICE 20: LINE 3
00340 4AA61014 (00163)          CSW      CSW$STDB,D21$INT1          IOS-2 DEVICE 21: LINE 1
00342 4AB41014 (00164)          CSW      CSW$STDB,D21$INT2          IOS-2 DEVICE 21: LINE 2
00344 4AC21014 (00165)          CSW      CSW$STDB,D21$INT3          IOS-2 DEVICE 21: LINE 3
00346 4ACA1014 (00166)          CSW      CSW$STDB,D22$INT1          IOS-2 DEVICE 22: LINE 1
00348 4AD81014 (00167)          CSW      CSW$STDB,D22$INT2          IOS-2 DEVICE 22: LINE 2
0034A 4AE61014 (00168)          CSW      CSW$STDB,D22$INT3          IOS-2 DEVICE 22: LINE 3
0034C 4AEF1014 (00169)          CSW      CSW$STDB,D23$INT1          IOS-2 DEVICE 23: LINE 1
0034E 4AFC1014 (00170)          CSW      CSW$STDB,D23$INT2          IOS-2 DEVICE 23: LINE 2
00350 4B0A1014 (00171)          CSW      CSW$STDB,D23$INT3          IOS-2 DEVICE 23: LINE 3
                   (00172) *
                   (00173) *
                   (00174) *
                   (00175) *
                   (00176) *  INITIALIZE IOS BUSY FLAGS
                   (00177) *  ENTERED FROM SNAP-II SCHEDULER
                   (00178) *
          00000BEF (00179) *L = SNAPSINL + 6                      SET PC TO ENTRY IN SCHEDULER
                   (00180) *
00BEF 860049F1 (00181)          JMP      IOS$INL
                   (00182) *
                   (00183) *
                   (00184) *
```

```
00004900 (00185) *L = START                              RESET START LOCATION FOR MODULE
         (00186) *
         (00187) *
```

```
                (00188) *      LOAD SCROLL ERROR CODES
                (00189) *
                (00190) *
                (00191) *
                (00192) *          #        ERROR MEANING
                (00193) *         ---       -------------
                (00194) *
                (00195) *
                (00196) *          1        TRANSFER DIRECTION ILLEGAL
                (00197) *          2        BID 0 ILLEGAL ( OUT OF RANGE OR UNDEFINED )
                (00198) *          3        BID 1 ILLEGAL ( OUT OF RANGE OR UNDEFINED )
                (00199) *          4        BID 2 ILLEGAL ( OUT OF RANGE OR UNDEFINED )
                (00200) *          5        BID 3 ILLEGAL ( OUT OF RANGE OR UNDEFINED )
                (00201) *          6        SIZE(BID 0)  .NE. INTEGRAL MULTIPLE OF # CHANNELS
                (00202) *          7        IF DOUBLE BUFFERED MODE, AND SIZE(BID 0) .NE. SIZE(BID 2)
                (00203) *          8        1ST REGISTER # TO WRITE IS OUT OF RANGE
                (00204) *          9        # REGISTERS TO WRITE IS OUT OF RANGE
                (00205) *         10        INTEGER SCALAR ID OF OFFSET .GT. MAXIMUM INTEGER ID
                (00206) *         11        BINDING CHAIN IN ERROR
                (00207) *
                (00208) *
                (00209) *
049D0 0000      (00210) ERRSVAL  DATA     0                    SCROLL ERROR INDICATOR
                (00211) *
                (00212) *                                      BUSY FLAG PER IOS DEVICE
049D1 0000      (00213) IOSSFLG  DATA     0,0,0,0,0,0,0,0      0 => FREE, 1 => BUSY
049D2 0000
049D3 0000
049D4 0000
049D5 0000
049D6 0000
049D7 0000
049D8 0000
049D9 0000      (00214) ACQSLIST DATA     0,0,0,0,0,0,0,0      ACQUISITION MODE PER IOS DEVICE
049DA 0000
049DB 0000
049DC 0000
049DD 0000
049DE 0000
049DF 0000
049E0 0000
                (00215) *
                (00216)          EJECT
```

```
                  (00217) *
                  (00218) *  INITIALIZE IOS BUSY FLAGS
                  (00219) *
049E1 86001C12 (00220) IOS$INL  CALL    R0, SET$SMB       REPLACE INSTRUCTION IN SCHEDULER
                  (00221) *
049E3 1E200007 (00222)          LDRPT   R2, 7             SET TO CLEAR 8 WORDS
049E5 CC0449D1 (00223)          MOVZM   IOS$FLG(R2)
                  (00224) *
049E7 80000BE0 (00225)          JMP     SNAP$INL+8        RETURN TO SCHEDULER
                  (00226) *
049E9 0800     (00227)          EVEN
```

```
                      (00228) * IOS WAIT ROUTINE
                      (00229) *
                      (00230) *
                      (00231) *  ENTERED WITH R7 = SCROLL ID
                      (00232) *
                      (00233) *
049EA DC0E49C1        (00234) WAITSIOS SMBS     0, IOS$FLG-16(R7)    IF IOS FREE,
049EC 0F70            (00235)          RETURN                        RETURN
                      (00236) *
049ED 0190003F        (00237) WAITSO   WAITS    IDEVS63, ILVLS1      IF BUSY, WAIT FOR I/O DONE
                      (00238) *
049EF 2106            (00239)          HOP      WAITSIOS             REPEAT TEST
                      (00240) *
049F0 0F70            (00241)          RETURN
049F1 0800            (00242)          EVEN
```

```
                    (00243) !      INTERRUPT ROUTINES FOR IOS-SCROLL BUFFER MANAGEMENT
                    (00244) *
                    (00245) *
                    (00246) !          DEVICE NUMBER 16 - LINE 1
                    (00247) *
049F2 90700000 (00248) D16SINT1 MOVIR   R7, 0          POINT TO VALUES FOR FIRST INTERRUPT
049F4 86004B12 (00249)          CALL    R0, IOSSINT    GO TO INTERRUPT PROCESSING SUBROUTINE
049F6 0E70     (00250)          RET
049F7 0800     (00251)          EVEN
049F8 90700001 (00252)          MOVIR   R7, 1          POINT TO VALUES FOR SECOND INTERRUPT
049FA 86004B12 (00253)          CALL    R0, IOSSINT    GO TO INTERRUPT PROCESSING SUBROUTINE
049FC 0E70     (00254)          RET
049FD 0800     (00255)          EVEN
049FE 800049F2 (00256)          JMP     D16SINT1       REPEAT LOOP ON NEXT INTERRUPT
                    (00257) *
                    (00258) *
                    (00259) !          DEVICE NUMBER 16 - LINE 2
                    (00260) *
                    (00261) *
04A00 90700010 (00262) D16SINT2 MOVIR   R7, 16         POINT TO VALUES FOR FIRST INTERRUPT
04A02 86004B12 (00263)          CALL    R0, IOSSINT    GO TO INTERRUPT PROCESSING SUBROUTINE
04A04 0E70     (00264)          RET
04A05 0800     (00265)          EVEN
04A06 90700011 (00266)          MOVIR   R7, 17         POINT TO VALUES FOR SECOND INTERRUPT
04A08 86004B12 (00267)          CALL    R0, IOSSINT    GO TO INTERRUPT PROCESSING SUBROUTINE
04A0A 0E70     (00268)          RET
04A0B 0800     (00269)          EVEN
04A0C 80004A00 (00270)          JMP     D16SINT2       REPEAT LOOP ON NEXT INTERRUPT
                    (00271) *
                    (00272) *
                    (00273) !          DEVICE NUMBER 16 - LINE 3
                    (00274) *
                    (00275) *
04A0E 90700000 (00276) D16SINT3 MOVIR   R7, 0          POINT TO VALUES FOR INTERRUPT 16
04A10 86004B24 (00277)          CALL    R0, IOSSINTC   GO TO INTERRUPT COMPLETION SUBROUTINE
04A12 0E70     (00278)          RET
04A13 0800     (00279)          EVEN
04A14 80004A0E (00280)          JMP     D16SINT3
                    (00281) *
                    (00282)          EJECT
```

```
                   (00283) !            DEVICE NUMBER 17 - LINE 1
                   (00284) *
04A16 90700002 (00285) D17SINT1 MOVIR    R7, 2            POINT TO VALUES FOR FIRST INTERRUPT
04A18 86004812 (00286)          CALL     R0, IOSSINT      GO TO INTERRUPT PROCESSING SUBROUTINE
04A1A 0E70     (00287)          RET
04A1B 0800     (00288)          EVEN
04A1C 90700003 (00289)          MOVIR    R7, 3            POINT TO VALUES FOR SECOND INTERRUPT
04A1E 86004812 (00290)          CALL     R0, IOSSINT      GO TO INTERRUPT PROCESSING SUBROUTINE
04A20 0E70     (00291)          RET
04A21 0800     (00292)          EVEN
04A22 80004A16 (00293)          JMP      D17SINT1         REPEAT LOOP ON NEXT INTERRUPT
               (00294) *
               (00295) *
               (00296) !            DEVICE NUMBER 17 - LINE 2
               (00297) *
               (00298) *
04A24 90700012 (00299) D17SINT2 MOVIR    R7, 18           POINT TO VALUES FOR FIRST INTERRUPT
04A26 86004812 (00300)          CALL     R0, IOSSINT      GO TO INTERRUPT PROCESSING SUBROUTINE
04A28 0E70     (00301)          RET
04A29 0800     (00302)          EVEN
04A2A 90700013 (00303)          MOVIR    R7, 19           POINT TO VALUES FOR SECOND INTERRUPT
04A2C 86004812 (00304)          CALL     R0, IOSSINT      GO TO INTERRUPT PROCESSING SUBROUTINE
04A2E 0E70     (00305)          RET
04A2F 0800     (00306)          EVEN
04A30 80004A24 (00307)          JMP      D17SINT2         REPEAT LOOP ON NEXT INTERRUPT
               (00308) *
               (00309) *
               (00310) !            DEVICE NUMBER 17 - LINE 3
               (00311) *
               (00312) *
04A32 90700002 (00313) D17SINT3 MOVIR    R7, 2            POINT TO VALUES FOR INTERRUPT 17
04A34 86004824 (00314)          CALL     R0, IOSSINTC     GO TO INTERRUPT COMPLETION SUBROUTINE
04A36 0E70     (00315)          RET
04A37 0800     (00316)          EVEN
04A38 80004A32 (00317)          JMP      D17SINT3
               (00318) *
               (00319)          EJECT
```

```
                   (00320) !            DEVICE NUMBER 18 - LINE 1
                   (00321) *
04A3A 90700004 (00322) D18SINT1 MOVIR    R7, 4              POINT TO VALUES FOR FIRST INTERRUPT
04A3C 86004B12 (00323)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A3E 0F70     (00324)          RET
04A3F 0800     (00325)          EVEN
04A40 90700005 (00326)          MOVIR    R7, 5              POINT TO VALUES FOR SECOND INTERRUPT
04A42 86004B12 (00327)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A44 0F70     (00328)          RET
04A45 0800     (00329)          EVEN
04A46 80004A3A (00330)          JMP      D18SINT1           REPEAT LOOP ON NEXT INTERRUPT
                   (00331) *
                   (00332) *
                   (00333) !            DEVICE NUMBER 18 - LINE 2
                   (00334) *
                   (00335) *
04A48 90700014 (00336) D18SINT2 MOVIR    R7, 20             POINT TO VALUES FOR FIRST INTERRUPT
04A4A 86004B12 (00337)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A4C 0F70     (00338)          RET
04A4D 0800     (00339)          EVEN
04A4E 90700015 (00340)          MOVIR    R7, 21             POINT TO VALUES FOR SECOND INTERRUPT
04A50 86004B12 (00341)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A52 0F70     (00342)          RET
04A53 0800     (00343)          EVEN
04A54 80004A48 (00344)          JMP      D18SINT2           REPEAT LOOP ON NEXT INTERRUPT
                   (00345) *
                   (00346) *
                   (00347) !            DEVICE NUMBER 18 - LINE 3
                   (00348) *
                   (00349) *
04A56 90700004 (00350) D18SINT3 MOVIR    R7, 4              POINT TO VALUES FOR INTERRUPT 18
04A58 86004B24 (00351)          CALL     R0, IOSSINTC       GO TO INTERRUPT COMPLETION SUBROUTINE
04A5A 0F70     (00352)          RET
04A5B 0800     (00353)          EVEN
04A5C 80004A56 (00354)          JMP      D18SINT3
                   (00355) *
                   (00356)          EJECT
```

```
                   (00357) !           DEVICE NUMBER 19 - LINE 1
                   (00358) *
04A5E 90700006 (00359) D19SINT1 MOVIR    R7, 6              POINT TO VALUES FOR FIRST INTERRUPT
04A60 86004812 (00360)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A62 0F70     (00361)          RET
04A63 0800     (00362)          EVEN
04A64 90700007 (00363)          MOVIR    R7, 7              POINT TO VALUES FOR SECOND INTERRUPT
04A66 86004812 (00364)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A68 0F70     (00365)          RET
04A69 0800     (00366)          EVEN
04A6A 80004A5E (00367)          JMP      D19SINT1           REPEAT LOOP ON NEXT INTERRUPT
               (00368) *
               (00369) *
               (00370) !           DEVICE NUMBER 19 - LINE 2
               (00371) *
               (00372) *
04A6C 90700016 (00373) D19SINT2 MOVIR    R7, 22             POINT TO VALUES FOR FIRST INTERRUPT
04A6E 86004812 (00374)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A70 0F70     (00375)          RET
04A71 0800     (00376)          EVEN
04A72 90700017 (00377)          MOVIR    R7, 23             POINT TO VALUES FOR SECOND INTERRUPT
04A74 86004812 (00378)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A76 0F70     (00379)          RET
04A77 0800     (00380)          EVEN
04A78 80004A6C (00381)          JMP      D19SINT2           REPEAT LOOP ON NEXT INTERRUPT
               (00382) *
               (00383) *
               (00384) !           DEVICE NUMBER 19 - LINE 3
               (00385) *
               (00386) *
04A7A 90700006 (00387) D19SINT3 MOVIR    R7, 6              POINT TO VALUES FOR INTERRUPT 16
04A7C 86004824 (00388)          CALL     R0, IOSSINTC       GO TO INTERRUPT COMPLETION SUBROUTINE
04A7E 0F70     (00389)          RET
04A7F 0800     (00390)          EVEN
04A80 80004A7A (00391)          JMP      D19SINT3
               (00392) *
               (00393)          EJECT
```

INTERRUPT ROUTINES FOR IOS-SCROLL BUFFER MANAGEMENT

```
                   (00394) !            DEVICE NUMBER 20 - LINE 1
                   (00395) *
04AB2 90700008     (00396) D20$INT1 MOVIR    R7, 8              POINT TO VALUES FOR FIRST INTERRUPT
04AB4 86004B12     (00397)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04AB6 0F70         (00398)          RET
04AB7 0800         (00399)          EVEN
04AB8 90700009     (00400)          MOVIR    R7, 9              POINT TO VALUES FOR SECOND INTERRUPT
04ABA 86004B12     (00401)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04ABC 0F70         (00402)          RET
04ABD 0800         (00403)          EVEN
04ABE 80004AB2     (00404)          JMP      D20$INT1           REPEAT LOOP ON NEXT INTERRUPT
                   (00405) *
                   (00406) *
                   (00407) !            DEVICE NUMBER 20 - LINE 2
                   (00408) *
                   (00409) *
04A90 90700018     (00410) D20$INT2 MOVIR    R7, 24             POINT TO VALUES FOR FIRST INTERRUPT
04A92 86004B12     (00411)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A94 0F70         (00412)          RET
04A95 0800         (00413)          EVEN
04A96 90700019     (00414)          MOVIR    R7, 25             POINT TO VALUES FOR SECOND INTERRUPT
04A98 86004B12     (00415)          CALL     R0, IOSSINT        GO TO INTERRUPT PROCESSING SUBROUTINE
04A9A 0F70         (00416)          RET
04A9B 0800         (00417)          EVEN
04A9C 80004A90     (00418)          JMP      D20$INT2           REPEAT LOOP ON NEXT INTERRUPT
                   (00419) *
                   (00420) *
                   (00421) !            DEVICE NUMBER 20 - LINE 3
                   (00422) *
                   (00423) *
04A9E 90700008     (00424) D20$INT3 MOVIR    R7, 8              POINT TO VALUES FOR INTERRUPT 20
04AA0 86004B24     (00425)          CALL     R0, IOSSINTC       GO TO INTERRUPT COMPLETION SUBROUTINE
04AA2 0F70         (00426)          RET
04AA3 0800         (00427)          EVEN
04AA4 80004A9E     (00428)          JMP      D20$INT3
                   (00429) *
                   (00430)          EJECT
```

```
                  (00431) !          DEVICE NUMBER 21 - LINE 1
                  (00432) *
04AA6 9070000A (00433) D21SINT1 MOVIR    R7, 10            POINT TO VALUES FOR FIRST INTERRUPT
04AA8 86004B12 (00434)          CALL     R0, IOSSINT       GO TO INTERRUPT PROCESSING SUBROUTINE
04AAA 0F70     (00435)          RET
04AAB 0800     (00436)          EVEN
04AAC 9070000B (00437)          MOVIR    R7, 11            POINT TO VALUES FOR SECOND INTERRUPT
04AAE 86004B12 (00438)          CALL     R0, IOSSINT       GO TO INTERRUPT PROCESSING SUBROUTINE
04AB0 0F70     (00439)          RET
04AB1 0800     (00440)          EVEN
04AB2 80004AA6 (00441)          JMP      D21SINT1          REPEAT LOOP ON NEXT INTERRUPT
               (00442) *
               (00443) *
               (00444) !          DEVICE NUMBER 21 - LINE 2
               (00445) *
               (00446) *
04AB4 9070001A (00447) D21SINT2 MOVIR    R7, 26            POINT TO VALUES FOR FIRST INTERRUPT
04AB6 86004B12 (00448)          CALL     R0, IOSSINT       GO TO INTERRUPT PROCESSING SUBROUTINE
04AB8 0F70     (00449)          RET
04AB9 0800     (00450)          EVEN
04ABA 9070001B (00451)          MOVIR    R7, 27            POINT TO VALUES FOR SECOND INTERRUPT
04ABC 86004B12 (00452)          CALL     R0, IOSSINT       GO TO INTERRUPT PROCESSING SUBROUTINE
04ABE 0F70     (00453)          RET
04ABF 0800     (00454)          EVEN
04AC0 80004AB4 (00455)          JMP      D21SINT2          REPEAT LOOP ON NEXT INTERRUPT
               (00456) *
               (00457) *
               (00458) !          DEVICE NUMBER 21 - LINE 3
               (00459) *
               (00460) *
04AC2 9070000A (00461) D21SINT3 MOVIR    R7, 10            POINT TO VALUES FOR INTERRUPT 21
04AC4 86004B24 (00462)          CALL     R0, IOSSINTC      GO TO INTERRUPT COMPLETION SUBROUTINE
04AC6 0F70     (00463)          RET
04AC7 0800     (00464)          EVEN
04AC8 80004AC2 (00465)          JMP      D21SINT3
               (00466) *
               (00467)          EJECT
```

```
                  (00468) !          DEVICE NUMBER 22 - LINE 1
                  (00469) *
04ACA 9070000C    (00470) D22$INT1 MOVIR   R7, 12           POINT TO VALUES FOR FIRST INTERRUPT
04ACC 86004B12    (00471)          CALL    R0, IOS$INT      GO TO INTERRUPT PROCESSING SUBROUTINE
04ACE 0E70        (00472)          RET
04ACF 0800        (00473)          EVEN
04AD0 9070000D    (00474)          MOVIR   R7, 13           POINT TO VALUES FOR SECOND INTERRUPT
04AD2 86004B12    (00475)          CALL    R0, IOS$INT      GO TO INTERRUPT PROCESSING SUBROUTINE
04AD4 0E70        (00476)          RET
04AD5 0800        (00477)          EVEN
04AD6 80004ACA    (00478)          JMP     D22$INT1         REPEAT LOOP ON NEXT INTERRUPT
                  (00479) *
                  (00480) *
                  (00481) !          DEVICE NUMBER 22 - LINE 2
                  (00482) *
                  (00483) *
04AD8 9070001C    (00484) D22$INT2 MOVIR   R7, 28           POINT TO VALUES FOR FIRST INTERRUPT
04ADA 86004B12    (00485)          CALL    R0, IOS$INT      GO TO INTERRUPT PROCESSING SUBROUTINE
04ADC 0E70        (00486)          RET
04ADD 0800        (00487)          EVEN
04ADE 9070001D    (00488)          MOVIR   R7, 29           POINT TO VALUES FOR SECOND INTERRUPT
04AE0 86004B12    (00489)          CALL    R0, IOS$INT      GO TO INTERRUPT PROCESSING SUBROUTINE
04AE2 0E70        (00490)          RET
04AE3 0800        (00491)          EVEN
04AE4 80004AD8    (00492)          JMP     D22$INT2         REPEAT LOOP ON NEXT INTERRUPT
                  (00493) *
                  (00494) *
                  (00495) !          DEVICE NUMBER 22 - LINE 3
                  (00496) *
                  (00497) *
04AE6 9070000C    (00498) D22$INT3 MOVIR   R7, 12           POINT TO VALUES FOR INTERRUPT 22
04AE8 86004B24    (00499)          CALL    R0, IOS$INTC     GO TO INTERRUPT COMPLETION SUBROUTINE
04AEA 0E70        (00500)          RET
04AEB 0800        (00501)          EVEN
04AEC 80004AE6    (00502)          JMP     D22$INT3
                  (00503) *
                  (00504)          EJECT
```

```
                   (00505) !           DEVICE NUMBER 23 - LINE 1
                   (00506) *
04AEE 9070000E (00507) D23$INT1 MOVIR   R7, 14          POINT TO VALUES FOR FIRST INTERRUPT
04AF0 86004812 (00508)          CALL    R0, IOSSINT     GO TO INTERRUPT PROCESSING SUBROUTINE
04AF2 0E70     (00509)          RET
04AF3 0800     (00510)          EVEN
04AF4 9070000F (00511)          MOVIR   R7, 15          POINT TO VALUES FOR SECOND INTERRUPT
04AF6 86004812 (00512)          CALL    R0, IOSSINT     GO TO INTERRUPT PROCESSING SUBROUTINE
04AF8 0E70     (00513)          RET
04AF9 0800     (00514)          EVEN
04AFA 80004AEE (00515)          JMP     D23$INT1        REPEAT LOOP ON NEXT INTERRUPT
                   (00516) *
                   (00517) *
                   (00518) !           DEVICE NUMBER 23 - LINE 2
                   (00519) *
                   (00520) *
04AFC 9070001E (00521) D23$INT2 MOVIR   R7, 30          POINT TO VALUES FOR FIRST INTERRUPT
04AFE 86004812 (00522)          CALL    R0, IOSSINT     GO TO INTERRUPT PROCESSING SUBROUTINE
04B00 0E70     (00523)          RET
04B01 0800     (00524)          EVEN
04B02 9070001F (00525)          MOVIR   R7, 31          POINT TO VALUES FOR SECOND INTERRUPT
04B04 86004812 (00526)          CALL    R0, IOSSINT     GO TO INTERRUPT PROCESSING SUBROUTINE
04B06 0E70     (00527)          RET
04B07 0800     (00528)          EVEN
04B08 80004AFC (00529)          JMP     D23$INT2        REPEAT LOOP ON NEXT INTERRUPT
                   (00530) *
                   (00531) *
                   (00532) !           DEVICE NUMBER 23 - LINE 3
                   (00533) *
                   (00534) *
04B0A 9070000E (00535) D23$INT3 MOVIR   R7, 14          POINT TO VALUES FOR INTERRUPT 23
04B0C 86004824 (00536)          CALL    R0, IOSSINTC    GO TO INTERRUPT COMPLETION SUBROUTINE
04B0E 0E70     (00537)          RET
04B0F 0800     (00538)          EVEN
04B10 80004B0A (00539)          JMP     D23$INT3
                   (00540) *
                   (00541)          EJECT
```

```
                   (00542) !      GENERAL SUBROUTINES TO PROCESS IOS INTERRUPTS
                   (00543) *
                   (00544) *
04812 F05E484C     (00545) IOSSINT  MOVMR   R5, IOS$SRWI(R7)     GET READ/WRITE INDEX
04814 406F         (00546)          MOVRR   R6, R7               COMPUTE LOGICAL BUFFER INDEX
04815 3C61         (00547)          LRS     R6, 1
04816 4C7C         (00548)          ADDRR   R7, R6
                   (00549) *
04817 F06F486C     (00550)          MOVMR   R6, IOSLBI(R7)       IF BID LEGAL,
04819 1A10         (00551)          SKPL    EQZ
0481A 820A4846     (00552)          XCT     IOS$MTBL+WS(R5)      FLAG CURRENT BUFFER AVAILABLE
                   (00553) *
0481C F07F486D     (00554)          MOVMR   R7, IOSLBI+1(R7)     IF BID LEGAL,
0481E 1A10         (00555)          SKPL    EQZ
0481F 820A4844     (00556)          XCT     IOS$MTBL(R5)         FLAG NEXT BUFFER BUSY
                   (00557) *
04821 1200003F     (00558)          AINT    IOEVS63, ILVL$1      SIGNAL I/O OPERATION COMPLETE
04823 0E70         (00559)          RETURN
                   (00560)          EVEN
                   (00561) *
                   (00562) *
                   (00563) *
04824 402F         (00564) IOSSINTC MOVRR   R2, R7               SAVE INDEX FOR LATER USE
04825 F05E484C     (00565)          MOVMR   R5, IOS$SRWI(R7)     GET READ/WRITE INDEX
04827 406F         (00566)          MOVRR   R6, R7               COMPUTE INDEX TO LOGICAL BUFFERS ID'S
04828 3C61         (00567)          LRS     R6, 1
04829 4C7C         (00568)          ADDRR   R7, R6
                   (00569)          EVEN
0482A C03F4884     (00570)          MOVMRL  R3, IOSLBI+24(R7)    GET LOGICAL BUFFER IDS (1 AND 3)
0482C C06F486C     (00571)          MOVMRL  R6, IOSLBI(R7)       GET LOGICAL BUFFER IDS (0 AND 2)
                   (00572) *
0482E 0260         (00573)          TEST    R6                   IF BID LEGAL,
0482F 1A10         (00574)          SKPL    EQZ
04830 820A4846     (00575)          XCT     IOS$MTBL+WS(R5)      FLAG BUFFER 0 AVAILABLE
                   (00576) *
04832 406F         (00577)          MOVRR   R6, R7               IF BID LEGAL,
04833 1A10         (00578)          SKPL    EQZ
04834 820A4846     (00579)          XCT     IOS$MTBL+WS(R5)      FLAG BUFFER 2 AVAILABLE
                   (00580) *
04836 4066         (00581)          MOVRR   R6, R3               IF BID LEGAL,
04837 1A10         (00582)          SKPL    EQZ
04838 820A4846     (00583)          XCT     IOS$MTBL+WS(R5)      FLAG BUFFER 1 AVAILABLE
                   (00584) *
0483A 4068         (00585)          MOVRR   R6, R4               IF BID LEGAL,
```

```
04B3B 1A10      (00586)        SKPL    F0Z
04B3C 820A4B46  (00587)        XCT     IOSBMTBL+WS(R5)      FLAG BUFFER 3 AVAILABLE
                (00588) *
04B3E 3C21      (00589)        LRS     R2, 1
04B3F CC0449D1  (00590)        MOVZM   IOSSFLG(R2)          CLEAR IOS DEVICE BUSY FLAG
                (00591) *
04B41 1200003F  (00592)        AINT    IDEVS63, ILVLS1      SIGNAL I/O OPERATION COMPLETE
04B43 0F70      (00593)        RETURN
                (00594)        EVEN
                (00595) *
                (00596)        EJECT
```

```
                  (00597) * BUFFER CONTROL TABLE FOR I/O INTERRUPTS
                  (00598) *
                  (00599) *
04B44 D26F0686 (00600) IOSBMTBL SMB      BITSIORD, BCTSAT(R7) READ DEVICE
04B46 D06C0686 (00601)          CMB      BITSIORD, BCTSAT(R6)
04B48 D31F0686 (00602)          SMB      BITSIOWT, BCTSAT(R7) WRITE DEVICE
04B4A D31C0686 (00603)          CMB      BITSIOWT, BCTSAT(R6)
                  (00604) *
                  (00605) *
                  (00606) *  BIT/SET READ/WRITE INDEX FOR I/O INTERRUPTS
                  (00607) *
                  (00608) *  SET EACH HALFWORD TO 0 FOR READ, 4 FOR WRITE
                  (00609) *
                  (00610) *
                  (00611) *
04B4C 0000        (00612) IOSBSRWI DATA     0,0                  DEVICE 16 - LINE 1
04B4D 0000
04B4E 0000        (00613)          DATA     0,0                  DEVICE 17 - LINE 1
04B4F 0000
04B50 0000        (00614)          DATA     0,0                  DEVICE 18 - LINE 1
04B51 0000
04B52 0000        (00615)          DATA     0,0                  DEVICE 19 - LINE 1
04B53 0000
04B54 0000        (00616)          DATA     0,0                  DEVICE 20 - LINE 1
04B55 0000
04B56 0000        (00617)          DATA     0,0                  DEVICE 21 - LINE 1
04B57 0000
04B58 0000        (00618)          DATA     0,0                  DEVICE 22 - LINE 1
04B59 0000
04B5A 0000        (00619)          DATA     0,0                  DEVICE 23 - LINE 1
04B5B 0000
                  (00620) *
04B5C 0000        (00621)          DATA     0,0                  DEVICE 16 - LINE 2
04B5D 0000
04B5E 0000        (00622)          DATA     0,0                  DEVICE 17 - LINE 2
04B5F 0000
04B60 0000        (00623)          DATA     0,0                  DEVICE 18 - LINE 2
04B61 0000
04B62 0000        (00624)          DATA     0,0                  DEVICE 19 - LINE 2
04B63 0000
04B64 0000        (00625)          DATA     0,0                  DEVICE 20 - LINE 2
04B65 0000
04B66 0000        (00626)          DATA     0,0                  DEVICE 21 - LINE 2
04B67 0000
```

```
04868 0000     (00627)        DATA     0,0                  DEVICE 22 - LINE 2
04869 0000
0486A 0000     (00628)        DATA     0,0                  DEVICE 23 - LINE 2
0486B 0000
               (00629) *
               (00630)        EJECT
```

```
                (00631) *  LOGICAL BUFFER IOS # 2 FOR I/O INTERRUPTS
                (00632) *
                (00633) *
                (00634) *  SET THE HALFWORD TRIPLET TO BUF0, BUF2, BUF0
                (00635) *
                (00636) *
04B6C 0000      (00637) IOSLB1   DATA    0,0,0           DEVICE 16 - LINE 1
04B6D 0000
04B6E 0000
04B6F 0000      (00638)          DATA    0,0,0           DEVICE 17 - LINE 1
04B70 0000
04B71 0000
04B72 0000      (00639)          DATA    0,0,0           DEVICE 18 - LINE 1
04B73 0000
04B74 0000
04B75 0000      (00640)          DATA    0,0,0           DEVICE 19 - LINE 1
04B76 0000
04B77 0000
04B78 0000      (00641)          DATA    0,0,0           DEVICE 20 - LINE 1
04B79 0000
04B7A 0000
04B7B 0000      (00642)          DATA    0,0,0           DEVICE 21 - LINE 1
04B7C 0000
04B7D 0000
04B7E 0000      (00643)          DATA    0,0,0           DEVICE 22 - LINE 1
04B7F 0000
04B80 0000
04B81 0000      (00644)          DATA    0,0,0           DEVICE 23 - LINE 1
04B82 0000
04B83 0000
                (00645) *
                (00646) *  SET THE HALFWORD TRIPLET TO BUF1, BUF3, BUF1
                (00647) *
04B84 0000      (00648)          DATA    0,0,0           DEVICE 16 - LINE 2
04B85 0000
04B86 0000
04B87 0000      (00649)          DATA    0,0,0           DEVICE 17 - LINE 2
04B88 0000
04B89 0000
04B8A 0000      (00650)          DATA    0,0,0           DEVICE 18 - LINE 2
04B8B 0000
04B8C 0000
04B8D 0000      (00651)          DATA    0,0,0           DEVICE 19 - LINE 2
04B8E 0000
```

```
04B8F 0000
04B90 0000      (00652)          DATA      0,0,0                DEVICE 20 - LINE 2
04B91 0000
04B92 0000
04B93 0000      (00653)          DATA      0,0,0                DEVICE 21 - LINE 2
04B94 0000
04B95 0000
04B96 0000      (00654)          DATA      0,0,0                DEVICE 22 - LINE 2
04B97 0000
04B98 0000
04B99 0000      (00655)          DATA      0,0,0                DEVICE 23 - LINE 2
04B9A 0000
04B9B 0000
```

```
' LDS$      MODULE TO PROCESS THE LOAD IOS SCROLL FCB
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*            !                         !                         !
*   WORD     !        LEFT BYTE        !       RIGHT BYTE        !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    0       !    POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    1       !           66            !          BID            !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    2       !                    SCROLL IDENT                   !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    3       !                    SCROLL TYPE                    !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    4       !           0             !           0             !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*            !                         !                         !
*    5       !           0             !           0             !
*            !                         !                         !
*   ---------!-------------------------!-------------------------!
*
*
*
*
             EJECT
```

```
!             LOGICAL BUFFER FORMAT FOR IOS PROGRAM
*
*
*       HW              CONTENTS
*       --              --------
*
*       0               N = SCROLL PROGRAM SIZE IN HALFWORDS
*
*       1               BID1      :      BID0
*
*       2 - N+1         SCROLL PROGRAM IN HALFWORDS
*
*       N+2             0 IF MAP -> SCROLL TRANSFER, ELSE 2 (ADAM=2, ADM=0)
*
*       N+3             # CHANNELS       :  ACQUISITION MODE
*
*       N+4             BID3      :      BID2
*
*       N+5             M = NUMBER OF SCROLL REGISTERS TO LOAD
*
*       N+6             # OF FIRST SCROLL REGISTER TO LOAD
*
*       N+7 - N+M+6     VALUES TO LOAD INTO SCROLL REGISTERS
*
*       N+M+7           INTEGER SCALAR ID OF OFFSET
*
*       N+M+8           BUFFER 0 CHAIN ANCHOR
*
*       N+M+9           BUFFER 1 CHAIN ANCHOR
*
*       N+M+10          BUFFER 2 CHAIN ANCHOR
*
*       N+M+11          BUFFER 3 CHAIN ANCHOR
*
*-------------------------------------------------------------
*        ACQUISTION MODE:
*                                 0 => DISCRETE OR BURST
*                           .NE. 0 => CONTINUOUS
*                                      1 => DOUBLE BUFFERED
*                                      2 => CIRCULAR
*-------------------------------------------------------------
           EJECT
```

```
                   (00738) *
                   (00739) *---  REGISTER USAGE
                   (00740) *
                   (00741) *          R1       POINTER TO FCB # IN FCB BLOCK
                   (00742) *          R2       SCROLL PROGRAM SIZE = N
                   (00743) *          R3       BA = IOS LOGICAL BUFFER BASE ADDRESS
                   (00744) *          R4       SCROLL INDEX   ( 16 => 0, 17 => 2, ... , 23 => 14 )
                   (00745) *
                   (00746) *---
                   (00747) *
04B9C 707200FF (00748) LDSS     MOVWK    R7, R1, MSKSRBYT      EXTRACT LOGICAL BUFFER ID FROM FCB
04B9E 3A71     (00749)          LLS      R7, 1                 CONVERT TO WORD INDEX
04B9F 0600     (00750)          EVEN
04BA0 F03F05B3 (00751)          MOVMR    R3, BCTSBA+BS(R7)     GET LOGICAL BUFFER START ADDRESS
               (00752) *
04BA2 F0420001 (00753) LDSSADAM MOVMR    R4, 1(R1)             GET SCROLL ID FROM FCB
04BA4 4078     (00754)          MOVRR    R7, R4
04BA5 860049FA (00755)          CALL     R0, WAITSIOS          WAIT TILL SCROLL DEVICE FREE
               (00756) *
04BA7 CC0049D0 (00757)          MOVZM    ERRSVAL               INITIALIZE ERROR INDICATOR
04BA9 6026     (00758)          MOVWR    R2, R3                GET IOS PROGRAM SIZE
               (00759)          EVEN
04BAA 842007B4 (00760)          MOVRML   R2, TEMS0             SAVE BA & N
               (00761) *
04BAC 4056     (00762)          MOVRR    R5, R3                COMPUTE PNTR TO SCROLL PROGRAM END
04BAD 4C54     (00763)          ADDRR    R5, R2                ( BA+N )
               (00764) *
04BAE F06A0003 (00765)          MOVMR    R6, 3(R5)             GET ACQUISITION MODE
04BB0 9A6000FF (00766)          ANDIR    R6, MSKSRBYT
04BB2 F06B49C9 (00767)          MOVRM    R6, ACQSLIST-16(R4)   SAVE ACQUISITION MODE FOR RUN SCROLL
               (00768) *
04BB4 9C49FFF0 (00769)          ADDIR    R4, -32(R4)           CONVERT SCRLID TO SCROLL INDEX
04BB6 F05A0002 (00770)          MOVMR    R5,2(R5)              GET TRANSFER DIR. FOR BUFFER CONTROL
               (00771) *
               (00772) *   TEST IF TRANSFER DIRECTION LEGAL ( 0 OR 2 )
               (00773) *
04BB8 1910     (00774)          SKP      NEZ
04BB9 2005     (00775)          HOP      LDSSC                 0 => OK
04BBA 92500002 (00776)          CMPIR    R5, 2
04BBC 1A10     (00777)          SKPL     EO                    2 => OK
04BBD 80004CBC (00778)          JMP      IOSSER1               OTHERWISE, ERROR
               (00779) *
04BBF 3A51     (00780) LDSSC    LLS      R5, 1                 CONVERT TO 0 OR 4
04BC0 F05B4B4C (00781)          MOVRM    R5, IOSBSRWI(R4)       MOVE READ/WRITE FLAG INTO TABLE
```

```
04BC2 E05B4B4D (00782)        MOVRM    R5, IOS$SRWI+1(R4)
               (00783) *
04BC4 E0760001 (00784)        MOVMR    R7, B$(R3)            EXTRACT LOGICAL BUFFER IDENTIFIERS
04BC6 506F00FF (00785)        MOVKR    R6, R7, MSK$RBYT      (BID 0) * 2
04BC8 3A61     (00786)        LLS      R6, 1
04BC9 3C77     (00787)        LRS      R7, 7                 (BID 1) * 2
               (00788) *
04BCA 405C     (00789)        MOVRR    R5, R6                CHECK BID 0
04BCB 86604CBC (00790)        CALL     R6, BIDSCHK
04BCD 0250     (00791)        TEST     R5                    ERROR ENCOUNTERED?
04BCE 80304CBF (00792)        JMP      IOSSER2, LTZ          YES, EXIT
               (00793) *
04BD0 405F     (00794)        MOVRR    R5, R7                CHECK BID 1
04BD1 86604CBC (00795)        CALL     R6, BIDSCHK
04BD3 0250     (00796)        TEST     R5
04BD4 80304CC2 (00797)        JMP      IOSSER3, LTZ
               (00798) *
04BD6 4058     (00799)        MOVRR    R5, R4                COMPUTE INDEX INTO TRIPLET BID TABLE
04BD7 3C51     (00800)        LRS      R5, 1
04BD8 4C58     (00801)        ADDRR    R5, R4
04BD9 0800     (00802)        EVEN
04BDA E06A4B6C (00803)        MOVRM    R6, IOS$LBI(R5)       STORE BID 0 FOR INTERRUPT LINE 1
04BDC E06A4B6E (00804)        MOVRM    R6, IOS$LBI+2(R5)
               (00805) *
04BDE E07A4B84 (00806)        MOVRM    R7, IOS$LBI+24(R5)    STORE BID 1 FOR INTERRUPT LINE 2
04BE0 E07A4B86 (00807)        MOVRM    R7, IOS$LBI+26(R5)
               (00808) *
04BE2 90760004 (00809)        MOVIR    R7, 4(R3)             COMPUTE POINTER TO BUFFER 2 & 3 IDS
04BE4 4C74     (00810)        ADDRR    R7, R2                  ( BA+N+4 )
04BE5 607E     (00811)        MOVWR    R7, R7                GET BUFFER 2 & 3 IDS
               (00812) *
04BE6 506E00FF (00813)        MOVKR    R6, R7, MSK$RBYT      EXTRACT THEM
04BE8 3A61     (00814)        LLS      R6, 1                 (BID 2) * 2
04BE9 3C77     (00815)        LRS      R7, 7                 (BID 3) * 2
               (00816) *
04BEA E06A4B6D (00817)        MOVRM    R6, IOS$LBI+1(R5)     STORE BID 2 FOR INTERRUPT LINE 1
04BEC E07A4B85 (00818)        MOVRM    R7, IOS$LBI+25(R5)    STORE BID 3 FOR INTERRUPT LINE 2
               (00819) *
04BEE 405C     (00820)        MOVRR    R5, R6                CHECK BID 2
04BEF 86604CBC (00821)        CALL     R6, BIDSCHK
04BF1 0250     (00822)        TEST     R5                    ERROR ENCOUNTERED?
04BF2 80304CC5 (00823)        JMP      IOSSER4, LTZ          YES, EXIT
               (00824) *
04BF4 405F     (00825)        MOVRR    R5, R7                CHECK BID 3
```

```
04BF5 B6604CBC (00826)          CALL    R6, BIDSCHK
04BF7 0250     (00827)          TEST    R5                  ERROR ENCOUNTERED?
04BF8 80304CC8 (00828)          JMP     IOSSER5, LTZ        YES, EXIT
               (00829) *
               (00830) *  TEST IF SIZE OF BID 0 = INTEGRAL MULTIPLE OF # CHANNELS
               (00831) *
04BFA 4056     (00832)          MOVRR   R5, R3
04BFB 4C54     (00833)          ADDRR   R5, R2              ( BA+N )
               (00834) *
04BFC F06A0003 (00835)          MOVMR   R6, 3(R5)           EXTRACT # CHANNELS
04BFE 3C68     (00836)          LRS     R6, 8
               (00837) *
04BFF F0760001 (00838)          MOVMR   R7, 1(R3)           EXTRACT BID 0
04C01 9A7000FF (00839)          ANDIR   R7, MSKSRBYT
04C03 3A71     (00840)          LLS     R7, 1
               (00841) *
04C04 F07F0605 (00842)          MOVMR   R7, BCTSAD+HS(R7)   GET BUFFER SIZE-1
04C06 2671     (00843)          INCR    R7, 1
04C07 F07007B6 (00844)          MOVRM   R7, TEMS2           SAVE FOR LATER USE
               (00845) *
04C09 0260     (00846)          TEST    R6
04C0A 81204C13 (00847)          JMP     LDSSA, LEZ          SKIP TEST IF NULL
               (00848) *
               (00849) *  SET UP LOOP -- R6 = # CHANNELS, R7 = BUFFER SIZE
               (00850) *
04C0C 4E7C     (00851) #1       SUBRR   R7, R6              SUBTRACT # CHANNELS
04C0D 80104C13 (00852)          JMP     LDSSA, EQZ          DIFF. = 0 => NO ERROR
04C0F 80204C0C (00853)          JMP     #1, GTZ                   > 0 => CONTINUE
               (00854) *
               (00855) *  ERROR, SIZE NOT A MULTIPLE OF # CHANNELS
               (00856) *
04C11 80004CCB (00857)          JMP     IOSSER6
               (00858) *
               (00859) *  TEST IF SIZE(BID 0) = SIZE(BID 2) IF DOUBLE BUFFERED
               (00860) *
04C13 F06A0003 (00861) LDSSA    MOVMR   R6, 3(R5)           EXTRACT ACQUISITION MODE
04C15 9A6000FF (00862)          ANDIR   R6, MSKSRBYT
04C17 92600001 (00863)          CMPIR   R6, 1               IF DOUBLE BUFFERED MODE, DO TEST
04C19 81104C27 (00864)          JMP     LDSSB, NE           OTHERWISE, CONTINUE
               (00865) *
04C1B F07A0004 (00866)          MOVMR   R7, 4(R5)           EXTRACT BID 2
04C1D 9A7000FF (00867)          ANDIR   R7, MSKSRBYT
04C1F 3A71     (00868)          LLS     R7, 1
04C20 F07F0605 (00869)          MOVMR   R7, BCTSAD+HS(R7)   GET SIZE-1 OF BID 2
```

```
04C22 2671     (00870)        INCR    R7, 1
               (00871) *
04C23 E27007B6 (00872)        CMPWR   R7, TEMS2            = SIZE OF BID 0?
04C25 81104CCF (00873)        JMP     IOSSER7, NE          NO, ERROR
               (00874) *
04C27 E0520002 (00875) LOSSB  MOVWR   R5, 2(R1)            GET SCROLL TYPE FROM FCB
04C29 92500003 (00876)        CMPIR   R5, 3                DISPATCH ON SCROLL TYPE
04C2B 80104C9B (00877)        JMP     LOSS01, EQ           JUMP IF IOS-3
04C2D 92500004 (00878)        CMPIR   R5, 4                TEST IF IOS-4
04C2F 80104CA6 (00879)        JMP     LOSS02, EQ           JUMP IF IOS-4
               (00880) *
               (00881) *
               (00882) *        LOAD IOS-2 SCROLL
               (00883) *
               (00884) *
               (00885) *    WRITE THE SCROLL REGISTERS
               (00886) *
04C31 90560006 (00887)        MOVIR   R5, 6(R3)            COMPUTE POINTER TO SCROLL DATA - 1
04C33 4C54     (00888)        ADDRR   R5, R2               ( BA+N+6 )
04C34 606A     (00889)        MOVWR   R6, R5               GET FIRST SCROLL REGISTER ID
04C35 E07BFFFF (00890)        MOVWR   R7, -1(R5)           GET M = # OF SCROLL REGISTERS TO LOAD
04C37 80104C4A (00891)        JMP     LOSS0, EQZ           BYPASS LOAD IF NONE, ELSE
04C39 80304CD4 (00892)        JMP     IOSSER9, LTZ         # REG < 0 => ERROR
               (00893) *
               (00894) *
               (00895) *   TEST IF NUMBER AND FIRST SCROLL REGISTER TO LOAD ARE LEGAL
               (00896) *     R6 = 1ST REGISTER TO LOAD
               (00897) *     R7 = # REGISTERS TO LOAD = M
               (00898) *
04C3B 0260     (00899)        TEST    R6                   CHECK 1ST REGISTER
04C3C 80304CD1 (00900)        JMP     IOSSER8, LTZ         < 0 => ERROR
04C3E 9260000F (00901)        CMPIR   R6, 15
04C40 80204CD1 (00902)        JMP     IOSSER8, GT          > 15 => ERROR
               (00903) *
04C42 4C7C     (00904)        ADDRR   R7, R6               CHECK # + 1ST REGISTER
04C43 92700010 (00905)        CMPIR   R7, 16
04C45 80204CD4 (00906)        JMP     IOSSER9, GT          > 16 => ERROR
               (00907) *
04C47 4E7C     (00908)        SUBRR   R7, R6               RESTORE R7 = M
               (00909) *
               (00910) *
04C48 86604DBC (00911)        CALL    R6, WSRS1            LOAD THE SCROLL REGISTERS
               (00912) *
               (00913) *---  R5 ASSUMED = (BA+N+6+M )
```

```
                   (00914) *
04C4A 2721         (00915) LDSS0     DECR     R2, 1                   ADJUST COUNT
04C4B 2631         (00916)           INCR     R3, 1                   AND START ADDRESS
04C4C 82084C7C     (00917)           XCT      LDSSCRL2(R4)            EXECUTE PROPER BLOCK MOVE
                   (00918) *
                   (00919) *-----  BIND THE LOGICAL BUFFERS
                   (00920) *
                   (00921) *    SET UP FOR CALLS TO IO2SBDR ROUTINE
                   (00922) *
04C4E 0070         (00923)           CLR      R7                      SET UP NULL OFFSET VALUE
04C4F 2651         (00924)           INCR     R5, 1                   POINT INTEGER SCALAR ID OF OFFSET
04C50 606A         (00925)           MOVWR    R6, R5                  GET SCALAR ID
                   (00926) *
04C51 80304C59     (00927)           JMP      LDSSNULL, LTZ           IF < 0, => NULL
04C53 92600080     (00928)           CMPIR    R6, ISVTSSIZ
04C55 81304CD7     (00929)           JMP      IOSSER10, GE            IF > MAX => ERROR
                   (00930) *
04C57 F07C0502     (00931)           MOVMR    R7, ISVTS(R6)           LEGAL ID, GET OFFSET VALUE
04C59 F0704E30     (00932) LDSSNULL  MOVRM    R7, IOBDSOFF            STORE FOR IO2SBDR ROUTINE
                   (00933) *
04C5B C02007B4     (00934)           MOVMRL   R2,TEMS0                RESTORE R2=N, R3=BA
04C5D 4C26         (00935)           ADDRR    R2, R3                  (BA + N )
04C5F F0740004     (00936)           MOVMR    R7, 4(R2)               R7 = BID 3 : 2 FOR LATER USE
04C60 F0484C7D     (00937)           MOVMR    R4, LDSSCRL2+HS(R4)     SET UP SCROLL LOAD ADDRESS
                   (00938) *
04C62 F0260001     (00939)           MOVMR    R2, 1(R3)               GET BID 1 : 0
04C64 506400FF     (00940)           MOVKR    R6, R2, MSKSRBYT        R6 = BID 0
04C66 2651         (00941)           INCR     R5, 1                   R5 = POINTER TO BINDING CHAIN ANCHOR
04C67 2632         (00942)           INCR     R3, 2                   R3 = POINTER TO SCROLL PROGRAM
                   (00943) *---
04C68 86104DC8     (00944)           CALL     R1, IO2SBDR             BIND BUFFER 0
                   (00945) *
04C6A 2651         (00946)           INCR     R5, 1                   STEP TO NEXT CHAIN ANCHOR
04C6B 5064FF00     (00947)           MOVKR    R6, R2, MSKSLBYT        GET BID 1
04C6D 3C68         (00948)           LRS      R6, 8
                   (00949) *---
04C6E 86104DC8     (00950)           CALL     R1, IO2SBDR             BIND BUFFER 1
                   (00951) *
04C70 2651         (00952)           INCR     R5, 1                   STEP TO NEXT CHAIN ANCHOR
04C71 506F00FF     (00953)           MOVKR    R6, R7, MSKSRBYT        GET BID 2
                   (00954) *---
04C73 86104DC8     (00955)           CALL     R1, IO2SBDR             BIND BUFFER 2
                   (00956) *
04C75 2651         (00957)           INCR     R5, 1                   STEP TO NEXT CHAIN ANCHOR
```

```
04C76 506FFF00 (00958)          MOVKR    R6, R7, MSKSLBYT     GET BID 3
04C78 3C68     (00959)          LRS      R6, 8
               (00960) *---
04C79 86104DCB (00961)          CALL     R1, IO2$BDR          BIND BUFFER 3
               (00962) *
               (00963) *
04C7B 0F70     (00964)          RETURN
               (00965)          EVEN
               (00966) *
               (00967) *
               (00968) *  IOS-2 LOAD INSTRUCTIONS
               (00969) *
               (00970) *
04C7C D535F000 (00971) LDSSCRL2 RMOVE    R3, R2, $1F000       LOAD SCROLL 16
04C7E D535F200 (00972)          RMOVE    R3, R2, $1F200       LOAD SCROLL 17
04C80 D535F400 (00973)          RMOVE    R3, R2, $1F400       LOAD SCROLL 18
04C82 D535F600 (00974)          RMOVE    R3, R2, $1F600       LOAD SCROLL 19
04C84 D535F800 (00975)          RMOVE    R3, R2, $1F800       LOAD SCROLL 20
04C86 D535FA00 (00976)          RMOVE    R3, R2, $1FA00       LOAD SCROLL 21
04C88 D535FC00 (00977)          RMOVE    R3, R2, $1FC00       LOAD SCROLL 22
04C8A D535FE00 (00978)          RMOVE    R3, R2, $1FE00       LOAD SCROLL 23
               (00979) *
               (00980) *
               (00981)          EJECT
```

```
                    (00982) *
                    (00983) *  SUBROUTINE TO CHECK BID ARGUMENT
                    (00984) *            WITHIN LIMITS AND IF DEFINED
                    (00985) *
                    (00986) *      LIMITS:
                    (00987) *        ( 0 .LE. BID*2 .LE. BCTSSIZ*2 )
                    (00988) *
                    (00989) *        ENTER WITH R5 = BID * 2
                    (00990) *
                    (00991) *
                    (00992) *        EXIT WITH R5 = 0 => NO ERROR
                    (00993) *                     =-1 => ERROR
                    (00994) *
                    (00995) *        ROUTINE USES R6 AND R7 AS SCRATCH REGISTERS
                    (00996) *
                    (00997) *
04C8C 0070          (00998) BIDSCHK  CLR      R7                    INITIALIZE ERROR INDICATOR
04C8D 0250          (00999)          TEST     R5                    BID
04C8F 80104C9A      (01000)          JMP      BIDSRET, EQZ          IF NULL, RETURN
04C90 1820          (01001)          SKP      GTZ                   IF .LT. 0
04C91 0C70          (01002)          CCR      R7                    FLAG ERROR
                    (01003) *
04C92 92500080      (01004)          CMPIR    R5, BCTSSIZ*WS        IF .GE. MAX
04C94 1830          (01005)          SKP      LT
04C95 0C70          (01006)          CCR      R7                    FLAG ERROR
                    (01007) *
04C96 D93A0686      (01008)          SMBC     BITSBU, BCTSAT(R5)    IF BUFFER UNDEFINED,
04C98 0C70          (01009)          CCR      R7                    FLAG ERROR
                    (01010) *
04C99 405F          (01011)          MOVRR    R5, R7                MOVE INTO R5
                    (01012) *
04C9A 0F70          (01013) BIDSRET  RETURN
                    (01014) *
                    (01015) *
                    (01016)          EJECT
```

```
      PACKAGE  ---- APR 18, 1980 ----  PROGRAM # 840001.01A
    *  ... TO PROCESS THE LOAD IOS SCROLL FCB

                    (01017) !            LOAD IOS-3 SCROLL
                    (01018) *
                    (01019) *
. .   . .           (01020) LOS$01   LRS     R2, 1              ADJUST REPEAT COUNT
. .     .           (01021)          DECR    R2, RS
. .   .. ..         (01022)          XCT     LOSSCRL3(R4)       EXECUTE PROPER BLOCK MOVE
. ..  . ..          (01023)          RETURN
                    (01024)          EVEN
                    (01025) *
                    (01026) *
                    (01027) *  IOS-3 LOAD INSTRUCTIONS
                    (01028) *
                    (01029) *
04CA0 D735F880      (01030) LOSSCRL3 BMOVEL  R3, R2, $1F880     LOAD SCROLL 16
04CA2 D735F900      (01031)          BMOVEL  R3, R2, $1F900     LOAD SCROLL 17
04CA4 D735F980      (01032)          BMOVEL  R3, R2, $1F980     LOAD SCROLL 18
                    (01033) *
                    (01034) *
                    (01035)          EJECT
```

```
04CA6 3C21      (01036) LDS$02   LRS     R2, 1                ADJUST REPEAT COUNT
04CA7 2721      (01037)          DECR    R2, R$
04CA8 8208 4CAC (01038)          XCT     LDSSCRL4(R4)         EXECUTE PROPER LPROCL
04CAA 0E70      (01039)          RETURN
04CAB 0800      (01040)          EVEN
                (01041) *
04CAC D635FF1E  (01042) LDSSCRL4 LPROCL  R3, R2, $1FF1E       LOAD SCROLL 16
04CAE D635FF3E  (01043)          LPROCL  R3, R2, $1FF3E       LOAD SCROLL 17
04CB0 D635FF5E  (01044)          LPROCL  R3, R2, $1FF5E       LOAD SCROLL 18
04CB2 D635FF7E  (01045)          LPROCL  R3, R2, $1FF7E       LOAD SCROLL 19
04CB4 D635FF9E  (01046)          LPROCL  R3, R2, $1FF9E       LOAD SCROLL 20
04CB6 D635FFBE  (01047)          LPROCL  R3, R2, $1FFBE       LOAD SCROLL 21
04CB8 D635FFDE  (01048)          LPROCL  R3, R2, $1FFDE       LOAD SCROLL 22
04CBA D635FFFE  (01049)          LPROCL  R3, R2, $1FFFE       LOAD SCROLL 23
                (01050) *
                (01051)          EJECT
```

```
                  (01052) *
                  (01053) *  ERROR INDICATORS
                  (01054) *
04CBC 90700001 (01055) IOSSER1  MOVIR    R7, 1
04CBE 201F     (01056)          HOP      IOS$ERR
04CBF 90700002 (01057) IOSSER2  MOVIR    R7, 2
04CC1 201B     (01058)          HOP      IOS$ERR
04CC2 90700003 (01059) IOSSER3  MOVIR    R7, 3
04CC4 2018     (01060)          HOP      IOS$ERR
04CC5 90700004 (01061) IOSSER4  MOVIR    R7, 4
04CC7 2015     (01062)          HOP      IOS$ERR
04CC8 90700005 (01063) IOSSER5  MOVIR    R7, 5
04CCA 2012     (01064)          HOP      IOS$ERR
04CCB 90700006 (01065) IOSSER6  MOVIR    R7, 6
04CCD 200F     (01066)          HOP      IOS$ERR
04CCE 90700007 (01067) IOSSER7  MOVIR    R7, 7
04CD0 200C     (01068)          HOP      IOS$ERR
04CD1 90700008 (01069) IOSSER8  MOVIR    R7, 8
04CD3 2009     (01070)          HOP      IOS$ERR
04CD4 90700009 (01071) IOSSER9  MOVIR    R7, 9
04CD6 2006     (01072)          HOP      IOS$ERR
04CD7 9070000A (01073) IOSSER10 MOVIR    R7, 10
04CD9 2003     (01074)          HOP      IOS$ERR
04CDA 9070000B (01075) IOSSER11 MOVIR    R7, 11
04CDC 2000     (01076)          HOP      IOS$ERR
                  (01077) *
04CDD E07049D0 (01078) IOSSERR  MOVRM    R7, ERRSVAL        STORE ERROR INDICATOR
04CDF 90700014 (01079)          MOVIR    R7, ERR$IOS        FLAG IOS ERROR
04CE1 80001AFA (01080)          JMP      ERRORS             EXIT
                  (01081) *
04CE3 0800     (01082)          EVEN
```

```
' RNSS      MODULE TO PROCESS THE RUN IOS SCROLL FCB
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*        !                         !                         !
*  WORD  !        LEFT BYTE        !       RIGHT BYTE        !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    0   !  POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    1   !           67            !            0            !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    2   !                SCROLL IDENT                       !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    3   !                SCROLL TYPE                        !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    4   !           SCROLL START ADDRESS                    !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*        !                         !                         !
*    5   !            0            !            0            !
*        !                         !                         !
*  -----!-------------------------!-------------------------!
*
*
*
*
          EJECT
```

```
04CF4 F0720001 (01123) RNSS    MOVMR   R7, 1(R1)            GET SCROLL ID
04CF6 B60049FA (01124)         CALL    R0, WAITSIOS         WAIT TILL SCROLL DEVICE FREE
04CF8 B6104D06 (01125)         CALL    R1, RNSSINT          SET UP BUFFERS AND INTERRUPTS
               (01126) *
04CFA 9C7FFFE0 (01127)         ADDIR   R7, -32(R7)          CONVERT TO SCROLL INDEX
04CFC 0D60     (01128)         CLR     R6                   SET UP REGISTER FOR START SCROLL
04CFD 0800     (01129)         EVEN
04CFE C0420002 (01130)         MOVMRL   R4, 2(R1)           R4 = IOS TYPE ; R5 = START LITERAL
04CF0 92400003 (01131)         CMPIR   R4, 3                DISPATCH ON SCROLL TYPE
04CF2 80104D40 (01132)         JMP     RNSS01, EQ           JUMP IF IOS-3
               (01133) *
04CF4 92400004 (01134)         CMPIR   R4, 4                TEST FOR IOS-4
04CF6 80104D4A (01135)         JMP     RNSS02, EQ           JUMP IF IOS-4
               (01136) *
               (01137) * START IOS-2 SCROLL
               (01138) *
04CF8 3A58     (01139)         LLS     R5, 8                MOVE START ADDRESS TO LEFT BYTE
04CF9 F0DF4030 (01140)         MOVRM   R5, @RNSSIOS2(R7)    START THE SCROLL
04CFB 3C71     (01141)         LRS     R7, 1
04CFC F02F4909 (01142)         MOVMR   R2, ACQSLIST(R7)     GET ACQUISITION MODE FLAG
               (01143) *
04CFE 1810     (01144)         SKP     EOZ                  SKIP IF DISCRETE SAMPLING
04CFF 0F70     (01145)         RETURN                       ELSE, EXIT
               (01146)         EVEN
               (01147) *
04D00 90600009 (01148)         MOVIR   R6, 9                POINT TO SYNCSTOP REGISTER
04D02 3A71     (01149)         LLS     R7, 1
04D03 F0AF4030 (01150)         MOVRM   R2, @RNSSIOS2(R7)    THEN REFERENCE IT TO SET SYNCSTOP
04D05 0F70     (01151)         RETURN
               (01152)         EVEN
               (01153) *
               (01154)         EJECT
```

```
04D06 F0704D61 (01155) RNSSINT  MOVRM   R7, RNSSFINT+HS      SET-UP, THEN
04D08 82004D60 (01156)          XCT     RNSSFINT             ENABLE SCROLL INTERRUPT, LEVEL 1
04D0A F0704D63 (01157)          MOVRM   R7, RNSSFINT+3*HS    DITTO FOR LEVEL 2
04D0C 82004D62 (01158)          XCT     RNSSFINT+WS
04D0E F0704D65 (01159)          MOVRM   R7, RNSSFINT+5*HS    DITTO FOR LEVEL 3
04D10 82004D64 (01160)          XCT     RNSSFINT+2*WS
               (01161) *
04D12 90200001 (01162)          MOVIR   R2, 1
04D14 F02F49C1 (01163)          MOVRM   R2, IOSSFLG-16(R7)   FLAG SCROLL BUSY
               (01164) *
04D16 9C7FFFE0 (01165)          ADDIR   R7, -32(R7)          CONVERT TO INDEX
               (01166) *
04D18 2671     (01167)          INCR    R7, 1                FORCE INTERRUPT POINTERS AND BUFFER FLAGS
04D19 0800     (01168)          EVEN                         TO CORRECT STATE
04D1A 86704B12 (01169)          CALL    R7, IOSSINT
04D1C 9C700010 (01170)          ADDIR   R7, 16               POINT TO BUFFERS 1 AND 3
04D1E 86704B12 (01171)          CALL    R7, IOSSINT
               (01172) *
04D20 1000003F (01173)          CINT    IOEVS63, ILVLS1      CLEAR I/O DONE
               (01174) *
04D22 9F700011 (01175)          SUBIR   R7, 17
04D24 406F     (01176)          MOVRR   R6, R7               COMPUTE POINTER TO CSW WORD
04D25 3A71     (01177)          LLS     R7, 1
04D26 4C7C     (01178)          ADDRR   R7, R6
04D27 906F0320 (01179)          MOVIR   R6, CSWSIOS-WS(R7)
04D29 CA6F02B6 (01180)          POPMIL  R6, CSWSI161(R7)
04D2B CA6F02B8 (01181)          POPMIL  R6, CSWSI162(R7)
04D2D CA6F02BA (01182)          POPMIL  R6, CSWSI163(R7)
               (01183) *
04D2F 0F70     (01184)          RETURN
               (01185)          EVEN
               (01186) *
               (01187)          EJECT
```

```
                  (01188) *  START ADDRESS TABLE FOR IOS2
                  (01189) *
04D30 001DF1FF (01190) RNSSIOS2 ADDR    $1F1FF(R6,1)        START SCROLL 16
04D32 001DF3FF (01191)          ADDR    $1F3FF(R6,1)        START SCROLL 17
04D34 001DF5FF (01192)          ADDR    $1F5FF(R6,1)        START SCROLL 18
04D36 001DF7FF (01193)          ADDR    $1F7FF(R6,1)        START SCROLL 19
04D38 001DF9FF (01194)          ADDR    $1F9FF(R6,1)        START SCROLL 20
04D3A 001DFBFF (01195)          ADDR    $1FBFF(R6,1)        START SCROLL 21
04D3C 001DFDFF (01196)          ADDR    $1FDFF(R6,1)        START SCROLL 22
04D3E 001DFFFF (01197)          ADDR    $1FFFF(R6,1)        START SCROLL 23
                  (01198) *
                  (01199) *  START IOS-3 TYPE SCROLL
                  (01200) *
04D40 F0DF1D44 (01201) RNSS01   MOVRM   R5, @RNSSIOS3(R7)   START THE SCROLL
04D42 0F70     (01202)          RETURN
04D43 0800     (01203)          EVEN
                  (01204) *
                  (01205) *  START ADDRESSES FOR IOS-3
                  (01206) *
04D44 001DFFDC (01207) RNSSIOS3 ADDR    $1FFDC(R6,1)        START SCROLL 16
04D46 001DFFEC (01208)          ADDR    $1FFEC(R6,1)        START SCROLL 17
04D48 001DFFFC (01209)          ADDR    $1FFFC(R6,1)        START SCROLL 18
                  (01210) *
                  (01211) *  START IOS-4 TYPE SCROLL
                  (01212) *
04D4A 3A5C     (01213) RNSS02   LLS     R5, 12              MOVE START ADDR TO LEFT MOST DIGIT
04D4B 0800     (01214)          EVEN
04D4C F0DF4D50 (01215)          MOVRM   R5, @RNSSIOS4(R7)   START THE SCROLL
04D4E 0F70     (01216)          RETURN
04D4F 0800     (01217)          EVEN
                  (01218) *
                  (01219) *  START ADDRESSES FOR IOS-4
                  (01220) *
04D50 001DFF1A (01221) RNSSIOS4 ADDR    $1FF1A(R6,1)        START SCROLL 16
04D52 001DFF3A (01222)          ADDR    $1FF3A(R6,1)        START SCROLL 17
04D54 001DFF5A (01223)          ADDR    $1FF5A(R6,1)        START SCROLL 18
04D56 001DFF7A (01224)          ADDR    $1FF7A(R6,1)        START SCROLL 19
04D58 001DFF9A (01225)          ADDR    $1FF9A(R6,1)        START SCROLL 20
04D5A 001DFFBA (01226)          ADDR    $1FFBA(R6,1)        START SCROLL 21
04D5C 001DFFDA (01227)          ADDR    $1FFDA(R6,1)        START SCROLL 22
04D5E 001DFFFA (01228)          ADDR    $1FFFA(R6,1)        START SCROLL 23
                  (01229) *
                  (01230)          EJECT
```

```
                (01231) *
                (01232) *
                (01233) *  INTERRUPT ENABLE INSTRUCTIONS
                (01234) *
04D60 12900000  (01235) RNSSEINT EINT      MSS, ILVLS1
04D62 12A00000  (01236)          EINT      MSS, ILVLS2
04D64 12B00000  (01237)          EINT      MSS, ILVLS3
                (01238) *
                (01239)          EVEN
```

```
* RAAS      MODULE TO PROCESS THE RUN ADAM AND AOM FCB
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*           !                          !                          !
*   WORD    !        LEFT BYTE         !        RIGHT BYTE        !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    0      !   POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    1      !           68             !          ADAMID          !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    2      !         ADAMSA           !          AOMID           !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    3      !          AOMSA           !            0             !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    4      !            0             !            0             !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*           !                          !                          !
*    5      !            0             !            0             !
*           !                          !                          !
*   -----!-------------------------!-------------------------!
*
*
*
*
            EJECT
```

```
                  (01280) *
                  (01281) *         REGISTER USAGE
                  (01282) *
                  (01283) *         R1 POINTS TO FCB#
                  (01284) *         R2 POINTS TO FCB#
                  (01285) *         R3 = ADAMID
                  (01286) *         R4 = ADAMSA
                  (01287) *         R5 = AOMID
                  (01288) *         R6 = AOMSA
                  (01289) *
                  (01290) *
                  (01291) *
04D66 4022        (01292) MPRAAS    MOVRR    R2,R1               ;COPY R1 TO R2
04D67 703400FF    (01293)           MOVWK    R3,R2,MSKSRBYT      ;(R3) <---- ADAMID
04D69 2621        (01294)           INCR     R2,1                ;R2 POINTS TO NEXT WORD IN FCB
04D6A 7044FF00    (01295)           MOVWK    R4,R2,MSKSLBYT      ;(R4) <----- ADAMSA
04D6C 705400FF    (01296)           MOVWK    R5,R2,MSKSRBYT      ;(R5) <----- AOMID
04D6E 2621        (01297)           INCR     R2,1                ;R2 POINTS TO NEXT WORD OF FCB
04D6F 7064FF00    (01298)           MOVWK    R6,R2,MSKSLBYT      ;(R6) <------ AOMSA
                  (01299) *
04D71 4076        (01300)           MOVRR    R7, R3
04D72 860049FA    (01301)           CALL     R0, WAITSIOS        WAIT TILL ADAM FREE
04D74 86104D06    (01302)           CALL     R1, RNSSINT         SET UP BUFFERS AND INTERRUPTS
                  (01303) *
04D76 407A        (01304)           MOVRR    R7, R5
04D77 860049FA    (01305)           CALL     R0, WAITSIOS        WAIT TILL AOM FREE
04D79 86104D06    (01306)           CALL     R1, RNSSINT         SET UP BUFFERS AND INTERRUPTS
                  (01307) *
04D7B 9C37FFE0    (01308)           ADDIR    R3, -32(R3)         CONVERT TO SCROLL INDEX
04D7D 9C58FFE0    (01309)           ADDIR    R5, -32(R5)         CONVERT TO SCROLL INDEX
                  (01310) *
04D7F 07E0        (01311)           SCS      14                  ;MASTER DISABLE THE INTERRUPTS
04D80 F0C64D30    (01312)           MOVRM    R4,@RNSSIOS2(R3)    ;START ADAM
04D82 F0EA4D30    (01313)           MOVRM    R6,@RNSSIOS2(R5)    ;START AOM
04D84 03E0        (01314)           CCS      14                  ;MASTER ENABLE THE INTERRUPTS
                  (01315) *
04D85 3C31        (01316)           LRS      R3, 1
04D86 F02649D9    (01317)           MOVMR    R2, ACQSLIST(R3)    GET ACQUISITION MODE FOR ADAM
04D88 81104D8F    (01318)           JMP      MPRAAS0, NEZ
                  (01319) *
04D8A 90600009    (01320)           MOVIR    R6, 9               IF = 0,
04D8C 3A31        (01321)           LLS      R3, 1
04D8D F0A64D30    (01322)           MOVRM    R2, @RNSSIOS2(R3)   SET SYNCSTOP REGISTER
                  (01323) *
```

```
04D8F 3C51     (01324) MPRAAS0  LRS     R5, 1
04D90 F02A49D9 (01325)          MOVMR   R2, ACQSLIST(R5)     GET ACQUISITION MODE FOR AOM
04D92 81104D99 (01326)          JMP     MPRAAS1, NEZ
               (01327) *
04D94 90600009 (01328)          MOVIR   R6, 9                IF = 0,
04D96 3A51     (01329)          LLS     R5, 1
04D97 F0AA4D30 (01330)          MOVRM   R2, @RNSSIOS2(R5)    SET SYNCSTOP REGISTER
               (01331) *
04D99 0F70     (01332) MPRAAS1  RETURN
               (01333)          EVEN
```

```
* RSRS      MODULE TO PROCESS THE READ FROM SCROLL REGISTERS FCB
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*          !                         !                        !
*   WORD   !        LEFT BYTE        !       RIGHT BYTE       !
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     0    !   POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     1    !          70             !INTEGER SCALAR IDENTIFIER!
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     2    !              SCROLL IDENTIFIER                   !
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     3    !                SCROLL TYPE                       !
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     4    !              STARTING REGISTER                   !
*          !                         !                        !
*   -------!-------------------------!------------------------!
*          !                         !                        !
*     5    !                  COUNT                           !
*          !                         !                        !
*   -------!-------------------------!------------------------!
*
*
*
*
          EJECT
```

```
                  (01374) *
                  (01375) *  MODULE ASSUMES TOS-2 ONLY
                  (01376) *
04D9A 3072OOFF    (01377) RSRS     MOVWK    R7, R1, MSKSRBYT          EXTRACT INTEGER TABLE IDENTIFIER
04D9C 9C700502    (01378)          ADDIR    R7, ISVTS                 CONVERT TO MAP ADDRESS
04D9E F0704DB1    (01379)          MOVRM    R7, RSRSBMOV+HS           INSTALL IN BLOCK MOVE INST.
04DA0 F0420001    (01380)          MOVMR    R4, 1(R1)                 GET SCROLL IDENT
04DA2 9C49FFE0    (01381)          ADDIR    R4, -32(R4)          CONVERT SCROLL ID TO WORD INDEX
                  (01382) *
04DA4 C0520003    (01383)          MOVMRL   R5, 3(R1)                 GET STARTING REGISTER AND COUNT
04DA6 F0784D31    (01384)          MOVMR    R7, RNSSIOS2+HS(R4)       CONVERT REGISTER ID TO PSUDO MEM. A
04DA8 2671        (01385)          INCR     R7, 1                   ADJUST POINTER FOR BMOVE
04DA9 3A71        (01386)          LLS      R7, 1                     FORCE BIT 18 TO ZERO
04DAA 3C71        (01387)          LRS      R7, 1
04DAB 4C7A        (01388)          ADDRR    R7, R5         POINT TO REGISTER TO START THE READ
04DAC 2761        (01389)          DECR     R6, 1                     ADJUST REPEAT COUNT
04DAD 82004DB0    (01390)          XCT      RSRSBMOV                  READ THE REGISTERS
04DAF 0F70        (01391)          RETURN                             RETURN FOR NEXT FCB
                  (01392) *
                  (01393) *
04DB0 D57C0000    (01394) RSRSBMOV BMOVE    R7, R6, MSS               BLOCK MOVE INST. FOR READ
                  (01395)          EVEN
```

```
* WSRS      MODULE TO PROCESS THE WRITE INTO SCROLL REGISTERS FCB
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*        !                      !                      !
*  WORD  !       LEFT BYTE      !      RIGHT BYTE      !
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   0    !  POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   1    !          71          !INTEGER SCALAR IDENTIFIER!
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   2    !            SCROLL IDENTIFIER                !
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   3    !               SCROLL TYPE                   !
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   4    !             STARTING REGISTER               !
*        !                      !                      !
*  -----!----------------------!----------------------!
*        !                      !                      !
*   5    !                  COUNT                      !
*        !                      !                      !
*  -----!----------------------!----------------------!
*
*
*
*
            EJECT
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

AD
A091663

```
                   (01436) *
                   (01437) *  MODULE ASSUMES IOS-2 ONLY
                   (01438) *
04DB2 F0420001     (01439) WSR$     MOVMR   R4, 1(R1)              GET SCROLL ID
04DB4 9C49FFE0     (01440)          ADDIR   R4, -32(R4)            CONVERT SCROLL ID TO WORD INDEX
                   (01441) *
04DB6 705200FF     (01442)          MOVWK   R5, R1, MSK$BBYT       GET INTEGER TABLE ID
04DB8 9C500501     (01443)          ADDIR   R5, ISVTS-1            CONVERT TO MAP ADDRESS-1
04DBA C0620003     (01444)          MOVMRL  R6, 3(R1)              GET START REG. AND COUNT
                   (01445) *
                   (01446) *   ENTER HERE FROM LDS$ WITH REG 4-7 SET UP PROPERLY
                   (01447) *
                   (01448) *        R4      SCROLL INDEX
                   (01449) *        R5      POINTER - 1 TO DATA TO LOAD SCROLL REGISTER
                   (01450) *        R6      1ST SCROLL REGISTER
                   (01451) *        R7      # SCROLL REGISTERS TO LOAD
                   (01452) *
04DBC FC684D31     (01453) WSR$1    ADDMR   R6, RNS$IOS2+H$(R4)    CONVERT REG. ID TO PSUEDO-MEM. ADDR.
04DBE 2662         (01454)          INCR    R6, 2
04DBF F0604DC7     (01455)          MOVRM   R6, WSR$BMOV+H$        STORE IN BLOCK MOVE INSTRUCTION
04DC1 2771         (01456)          DECR    R7, 1                  CONVERT COUNT TO NO. OF REPEATS
04DC2 82004DC6     (01457)          XCT     WSR$BMOV               WRITE INTO THE REGISTERS.
04DC4 0E70         (01458)          RETURN
04DC5 0800         (01459)          EVEN
                   (01460) *
04DC6 D55F0000     (01461) WSR$BMOV BMOVE   R5, R7, M$$+$10000     BLOCK MOVE INSTRUCTION FOR WRITE
                   (01462)          EVEN
```

```
                  *      IO2SBDR    IOS SCROLL PROGRAM BINDING SUBROUTINE
                  *
                  *  THIS MODULE BINDS THE SPECIFIED ATTRIBUTES OF A LOGICAL BUFFER
                  *  TO THE IOS-2 PROGRAM. THE UNBOUND SCROLL PROGRAM MUST HAVE BEEN
                  *  LOADED PRIOR TO CALLING THIS PROGRAM.
                  *  THE CHAIN INFORMATION IS CONTAINED IN THE RIGHTMOST 15 BITS OF A
                  *  32 BIT IOS-2 INSTRUCTION.
                  *
                  *  ROUTINE ASSUMES WL = 0 => LONG (16 BIT)
                  *
                  *  THE FORMAT OF THE CHAIN DESCRIPTOR IS:
                  *
                  *  BITS:   0-7   BACKWARD (WORD) POINTER TO THE NEXT CHAIN ENTRY
                  *  BITS:   8-14  ATTRIBUTE SPECIFIER
                  *          VALUE             ATTRIBUTE
                  *            0               BASE ADDRESS
                  *            1               BASE ADDRESS - SEPARATION
                  *            2               BUFFER SEPERATION
                  *            3               BUFFER EXTENT
                  *            4               BUFFER SIZE
                  *            5               BUFFER SIZE - OFFSET
                  *            6               OFFSET
                  *
                  *
                  *  THIS MODULE IS CALLED WITH:
                  *
                  *    R3 = BASE ADDRESS OF SCROLL PROGRAM ON BUS 1
                  *    R4 = SCROLL PROGRAM LOAD ADDRESS (LEAST SIGNIFICANT 16 BITS)
                  *    R5 = POINTER TO BINDING CHAIN ANCHOR
                  *    R6 = LOGICAL BUFFER ID
                  *
                  *    IOBDSOFF CONTAINS INTEGER OFFSET
                  *
                  *---------
                  *
                  *        BID .LE. 0 OR
                  *        BINDING CHAIN ANCHOR .LT. 0
                  *
                  *            => NO BINDING DONE
                  *
                  *---------
                  *
04DC8 605A        IO2SBDR  MOVWR    R5, R5               GET BINDING CHAIN ANCHOR
04DC9 1930                 SKP      GEZ
```

```
04DCA 0F70      (01507)          RETURN                          EXIT IF NULL
                (01508) *
04DCB 4C6C      (01509)          ADDRR    R6, R6                 CONVERT BUFFER ID TO WORD INDEX
04DCC 1B20      (01510)          SKP      GTZ
04DCD 0F70      (01511)          RETURN                          EXIT IF NULL
                (01512) *
                (01513) *---  REGISTER USAGE:
                (01514) *
                (01515) *        R1       INSTRUCTION TO BE BOUND
                (01516) *        R2       ATTRIBUTE & THEN RIGHT HALF OF INSTRUCTION
                (01517) *        R7       DISPLACEMENT TO NEXT INSTRUCTION
                (01518) *
04DCF 3A51      (01519)          LLS      R5, 1                  ADJUST BINDING POINTER
04DCF 0800      (01520)          EVEN
04DD0 E0304F21  (01521)          MOVRM    R3, IOBDSPPT+H$        SET POINTER TO IOS PROGRAM ON BUS 1
04DD2 E0404F1F  (01522)          MOVRM    R4, IOBDSLPT+H$        SET SCROLL ADDRESS
                (01523) *
                (01524) *   GET NEXT INSTRUCTION TO BE BOUND
                (01525) *   DISPATCH TO PROPER BINDING ROUTINE
                (01526) *
04DD4 C0904F20  (01527) IO2SBDRN MOVRML   R1, @IOBDSPPT          GET INSTRUCTION TO BE BOUND
04DD6 507400FF  (01528)          MOVKR    R7, R2, MSK$RBYT       EXTRACT DISPLACEMENT TO NEXT INSTRUCT.
04DD8 3A71      (01529)          LLS      R7, 1                  ADJUST DISPLACEMENT TO HALF WORD
04DD9 3C27      (01530)          LRS      R2, 7                  POSITION ATTRIBUTE SPECIFIER AS WORD INDE
04DDA 9A2000FF  (01531)          ANDIR    R2, $FF                RESET LEAST SIGNIFICANT BIT
04DDC 80844F22  (01532)          JMP      @IO2SBDRD(R2)          DISPATCH TO PROPER BINDING MODULE
                (01533)          EVEN
                (01534) *
                (01535) *--- BASE ADDRESS BINDING ---
                (01536) *
04DDE C03C0582  (01537) IO2SBDR0 MOVRML   R3, HCTSBA(R6)         GET BASE ADDRESS ATTRIBUTES OF BUFFER
04DE0 9A11FFF8  (01538)          ANDIR    R1, $1FFF8             CLEAR OUT BUS AND BIT 16
04DE2 56160007  (01539)          IORKR    R1, R3, $7             MERGE BUS AND BIT 16 FROM BUFFER
                (01540)          EVEN
                (01541) *
                (01542) *---  PATH FOR BINDING INTO IOS MEMORY ---
                (01543) *---  AND FOR LOOPING TO NEXT INSTRUCTION
                (01544) *
04DE4 4028      (01545) #0       MOVRR    R2, R4                 GET LOW ORDER BITS
                (01546) *
04DE5 84904F1F  (01547) #1       MOVRML   R1, @IOBDSLPT          MOVE BOUND INSTRUCTION TO IOS2 MEM.
04DE7 0270      (01548)          TEST     R7                     ANOTHER INSTRUCTION IN CHAIN?
04DE8 1910      (01549)          SKP      NEZ
04DE9 0F70      (01550)          RETURN                          NO, EXIT
```

```
                    (01551) *
04DFA 4F5F          (01552)          SUBRR    R5, R7              YES, COMPUTE DISPLACEMENT TO NEXT INST.
04DFB B0304CDA      (01553)          JMP      IOS$ER11, LTZ       BINDING CHAIN ERROR, EXIT
04DFD 211A          (01554)          HOP      IO2$BDRN            OK, GET NEXT INST TO BE BOUND
                    (01555) *
                    (01556)          EVEN
                    (01557) *
                    (01558) *--- BASE ADDRESS - SEPARATION BINDING ---
                    (01559) *
04DFF C03C0582      (01560) IO2$BDR1 MOVMRL   R3, BCT$BA(R6)      GET BASE ADDRESS ATTRIBUTES OF BUFFER
04DF0 9A11FFF8      (01561)          ANDIR    R1, $1FFF8          CLEAR OUT BUS AND BIT 16
04DF2 56160006      (01562)          IORRR    R1, R3, $6          MERGE BUS ONLY
04DF4 FF4C0604      (01563)          SUBMR    R4, BCTSAD(R6)      SUBTRACT SEPARATION
04DF6 0630          (01564)          DAC      R3
04DF7 56160001      (01565)          IORRR    R1, R3, 1           MERGE HIGH ORDER ADDRESS BIT
04DF9 2116          (01566)          HOP      #0                  GO MOVE INST TO SCROLL MEMORY
                    (01567)          EVEN
                    (01568) *
                    (01569) *--- BUFFER SEPARATION BINDING ---
                    (01570) *
04DFA F03C0604      (01571) IO2$BDR2 MOVMR    R3, BCTSAD(R6)      GET BUFFER SEPARATION
04DFC 50267FFF      (01572)          MOVKR    R2, R3, $7FFF       MERGE 15 BIT SEPARATION
04DFF 211A          (01573)          HOP      #1                  GO MOVE INST TO SCROLL MEMORY
04DFF 0800          (01574)          EVEN
                    (01575) *
                    (01576) *--- BUFFER EXTENT BINDING ---
                    (01577) *
04F00 C03C0604      (01578) IO2$BDR3 MOVMRL   R3, BCTSAD(R6)      GET BUFFER SEPARATION AND SIZE
04F02 483B          (01579)          MULRR    R3, R4              COMPUTE HALF-WORD DISPLACEMENT
04F03 FC4C0583      (01580)          ADDMR    R4, BCT$BA+HS(R6)   ADD BUFFER BASE ADDRESS
04F05 50287FFF      (01581)          MOVKR    R2, R4, $7FFF        MERGE 15 BIT LLIT FIELD
04F07 2123          (01582)          HOP      #1                  GO MOVE INST. TO SCROLL MEMORY
                    (01583)          EVEN
                    (01584) *
                    (01585) *--- BUFFER SIZE BINDING ---
                    (01586) *
04F08 F04C0605      (01587) IO2$BDR4 MOVMR    R4, BCTSAD+HS(R6)   GET BUFFER SIZE-1
04F0A 2641          (01588)          INCR     R4, 1
04F0B 50287FFF      (01589)          MOVKR    R2, R4, $7FFF       MERGE 15 BIT LLIT FIELD
04F0D 2129          (01590)          HOP      #1                  GO MOVE INSTRUCTION TO SCROLL MEMORY
                    (01591)          EVEN
                    (01592) *
                    (01593) *--- BUFFER SIZE - OFFSET BINDING ---
                    (01594) *
```

```
04F0F F0304F30 (01595) IO2SBDR5 MOVMR    R3, IOBDSOFF        GET OFFSET
04F10 F04C0605 (01596)          MOVMR    R4, BCTSAD+HS(R6)   GET BUFFER SIZE-1
04F12 2641     (01597)          INCR     R4, 1
04F13 4F46     (01598)          SUBRR    R4, R3              SUBTRACT OFFSET
04F14 50287FFF (01599)          MOVKR    R2, R4, $7FFF       MERGE 15 BIT LLIT
04F16 2132     (01600)          HOP      #1
04F17 0800     (01601)          EVEN
               (01602) *
               (01603) *--- OFFSET BINDING ---
               (01604) *
04F18 F0304F30 (01605) IO2SBDR6 MOVMR    R3, IOBDSOFF        GET OFFSET
04F1A 50267FFF (01606)          MOVKR    R2, R3, $7FFF       MERGE 15 BIT LLIT
04F1C 2138     (01607)          HOP      #1                  GO MOVE INSTRUCTION TO SCROLL MEMORY
04F1D 0800     (01608)          EVEN
               (01609) *
               (01610)          EJECT
```

```
04F1E 000B0000 (01611) IOBDSLPT ADDR     $10000(R5)        POINTER TO STORE INST IN SCROLL
04F20 000A0000 (01612) IOBDSPPT ADDR     MSS(R5)           POINTER TO GET INST FROM SCROLL PROGRAM
               (01613) *
04F22 00004DDF (01614) IO2SBDRD ADDR     IO2SBDR0          BINDING DISPATCH TABLE
04F24 00004DFF (01615)          ADDR     IO2SBDR1
04F26 00004DFA (01616)          ADDR     IO2SBDR2
04F28 00004E00 (01617)          ADDR     IO2SBDR3
04F2A 00004E08 (01618)          ADDR     IO2SBDR4
04F2C 00004E0F (01619)          ADDR     IO2SBDR5
04F2E 00004E18 (01620)          ADDR     IO2SBDR6
               (01621) *
04F30 0000     (01622) IOBDSOFF DATA     MSS              OFFSET VALUE
               (01623) *
04F31 0800     (01624)          EVEN
```

```
' ADAMSSSP   MODULE TO PROCESS THE ADAM SIMPLIFIED SAMPLING PLAN
*
*
*
*  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
*
*
*         !                        !                        !
*   WORD  !       LEFT BYTE        !       RIGHT BYTE       !
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    0    !  POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)!
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    1    !          73            !      ADAM NUMBER       !
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    2    !   LOGICAL BUFFER #1    !   LOGICAL BUFFER #0    !
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    3    !       CHANNEL #1       !       CHANNEL #0       !
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    4    !                 COUNTER VALUE                   !
*         !                        !                        !
*   -----!------------------------!------------------------!
*         !                        !                        !
*    5    !   ACQUISITION MODE     ! TRIG. & CLOCK PARAMETERS!
*         !                        !                        !
*   -----!------------------------!------------------------!
*
*
*
*
          EJECT
```

```
04E32 F0720002 (01665) ADAMSSSP MOVMR    R7, WS(R1)         FIRST SET SINGLE OR DUAL SAMPLING
04E34 9A70FF00 (01666)          ANDIR    R7, MSKSLBYT
04E36 1A10     (01667)          SKPL     EOZ                SKIP IF SINGLE CHANNEL
04E37 90700002 (01668)          MOVIR    R7, 2              ELSE, SET DUAL CHANNEL FLAG
               (01669) *
04E39 0800     (01670)          EVEN
04E3A F0220002 (01671)          MOVMR    R2, WS(R1)         GET MUX ADDRESSES
04E3C 9A2000FF (01672)          ANDIR    R2, MSKSRBYT       FIRST MUX ADDRESS TO R3
04E3E 96208000 (01673)          IORIR    R2, $8000          SET ACQUIRE BIT
04E40 F06F4FD0 (01674)          MOVMR    R6, ADMSFMA(R7)    GET POINTER TO FMA IN IOS2 CODE
04E42 F0AE4FCC (01675)          MOVRM    R2, @ADMSSPLN(R7)  STORE FIRST MUX ADDRESS IN IOS2 PROG
04E44 0050     (01676)          CLR      R5                 SET COUNTER FOR TWO PASSES
04E45 0800     (01677)          EVEN
               (01678) *
04E46 F0220001 (01679) ADAMSSS1 MOVMR    R2, HS(R1)         GET LOGICAL BUFFER ID
04E48 0250     (01680)          TEST     R5
04E49 1810     (01681)          SKP      EOZ
04E4A 3C28     (01682)          LRS      R2, 8
               (01683) *
04E4B 4C5E     (01684)          ADDRR    R5, R7             CONSTRUCT POINTER FOR BUFFER 0 OR 1 INFO
04E4C 9A2000FF (01685)          ANDIR    R2, MSKSRBYT
04E4E 4C24     (01686)          ADDRR    R2, R2             CONVERT TO WORD INDEX
04E4F 0800     (01687)          EVEN
04E50 C03405B2 (01688)          MOVMRL   R3, BCTSBA(R2)     GET BUFFER START ADDRESS
04E52 9A300007 (01689)          ANDIR    R3, $7             CLEAR OUT UNWANTED BITS
04E54 F06A4FD4 (01690)          MOVMR    R6, ADMSBA(R5)     CONSTRUCT POINTER TO BASE ADDR LOAD INST
04E56 90FE4FCC (01691)          MOVIR    R6, @ADMSSPLN(R7)  GET PNTR TO BASE ADDRESS LOC
04E58 763CFFF0 (01692)          IORWK    R3, R6, $FFF0      CONSTRUCT PROPER BASE ADDRESS INSTRUCTION
04E5A 843C0000 (01693)          MOVRML   R3, 0(R6)          STORE IT IN IOS PROG
04E5C C0340604 (01694)          MOVMRL   R3, BCTSAD(R2)     GET BUFFER SEPARATION AND SIZE
04E5E 2663     (01695)          INCR     R6, WS+HS          STEP TO FIRST SEPARATION LOC
04E5F 0800     (01696)          EVEN
04E60 F03C0000 (01697)          MOVRM    R3, 0(R6)
04E62 F06A4FD8 (01698)          MOVMR    R6, ADMSSEP(R5)    GET POINTER TO SEPERATION INFO
04E64 F0BE4FCC (01699)          MOVRM    R3, @ADMSSPLN(R7)  STORE IT
04E66 0270     (01700)          TEST     R7
04E67 1910     (01701)          SKP      NEZ                SKIP IF DUAL CHANNEL SAMPLING
04E68 2003     (01702)          HOP      ADAMSSS2
               (01703) *
04E69 2666     (01704)          INCR     R6, 3*WS           POINT TO NEXT LOC FOR SEPERATION
04E6A F0BE4FCC (01705)          MOVRM    R3, @ADMSSPLN(R7)  AND STORE IT
               (01706) *
04E6C 4838     (01707) ADAMSSS2 MULRR    R3, R4             COMPUTE DISPLACEMENT TO BUFFER END
04E6D 3C31     (01708)          LRS      R3, 1
```

```
04F6E FC3405B3 (01709)           ADDMR    R3, BCTSBA+HS(R2)    ADD IN BUFFER BASE ADDRESS
04F70 F06A4FDC (01710)           MOVMR    R6, ADM$BEND(R5)     GET POINTER TO BUF END LOC
04F72 F0BF4FCC (01711)           MOVRM    R3, @ADM$SPLN(R7)    STORE IT IN THE SAMPLING PLAN
04F74 2761     (01712)           DECR     R6, HS               POINT TO FIRST HW OF INST
04F75 0800     (01713)           EVEN
04F76 D0BF4FCC (01714)           CMB      0, @ADM$SPLN(R7)
04F78 DA0405B2 (01715)           SMBCL    0, BCTSBA(R2)
04F7A D2BF4FCC (01716)           SMB      0, @ADM$SPLN(R7)
04F7C 4F5F     (01717)           SUBRR    R5, R7               RESTORE PASS COUNTER
04F7D 0270     (01718)           TEST     R7                   IF DUAL SAMPLING PLAN
04F7E 80104F92 (01719)           JMP      ADAM$SS3, EQZ              (NOT)
               (01720) *
04F80 F06A4FE0 (01721)           MOVMR    R6, ADM$SMA(R5)      SET UP SECOND MUX ADDRESS
04F82 F0220002 (01722)           MOVMR    R2, W$(R1)           GET THE TWO MUX ADDRESSES
04F84 503400FF (01723)           MOVKR    R3, R2, MSK$RBYT     FIRST MUX ADDRESS TO R3
04F86 9A20FF00 (01724)           ANDIR    R2, MSK$LBYT         SECOND MOX ADDRESS TO R2
04F88 3C28     (01725)           LRS      R2, 8
04F89 4F26     (01726)           SUBRR    R2, R3               WANT DISPLACEMENT TO 2ND MUX ADDR
04F8A F0AF4FCC (01727)           MOVRM    R2, @ADM$SPLN(R7)    STORE IN IOS PROG
04F8C 2664     (01728)           INCR     R6, 2*W$             POINT TO NEXT MUX STORAGE LOCATION
04F8D 0820     (01729)           NEG      R2
04F8E 9C200010 (01730)           ADDIR    R2, $10              SET FMA COMP WITH S/H RELEASE
04F90 F0AF4FCC (01731)           MOVRM    R2, @ADM$SPLN(R7)    PUT IN IOS PROGRAM
               (01732) *
04F92 0250     (01733) ADAM$SS3 TEST     R5                   HAS BOTH BUFFER ID'S BEEN PROCESSED
04F93 1810     (01734)           SKPL     NEZ                      (YES)
04F94 88504F46 (01735)           IJP      R5, ADAM$SS1         NO, GO REPEAT FOR SECOND BUFFER
               (01736) *
               (01737) *
               (01738) *  DISTRIBUTE FCB PARAMETERS
               (01739) *
04F96 3602     (01740)           PUSHXIL  R0, R1               SAVE FCB POINTER
04F97 2711     (01741)           DECR     R1, HS               PREPARE FOR POP OPERATIONS
04F98 3014     (01742)           POPXI    R1, R2               GET ADAM NUMBER
04F99 0C60     (01743)           CCR      R6                   PREPARE FOR LOGICAL BUFFER DISPATCH
               (01744)           EVEN
04F9A 9A2000FF (01745)           ANDIR    R2, MSK$RBYT         EXTRACT SCROLL IDENTIFIER
04F9C F0204FF4 (01746)           MOVRM    R2, ADM$DFCB+W$      STORE IN DUMMY FCB
               (01747) *
04F9E F0320001 (01748)           MOVMR    R3, 1(R1)            EXTRACT BID 0
04FA0 504600FF (01749)           MOVKR    R4, R3, MSK$RBYT
04FA2 F0CF4FCC (01750)           MOVRM    R4, @ADM$SPLN(R7)    PUT INTO SCROLL HEADER
               (01751) *
04FA4 3C38     (01752)           LRS      R3, 8                EXTRACT BID 1
```

```
04FA5 2761     (01753)      DECR    R6, 1               POINT TO IOS SIZE
04FA6 F0FF4FCC (01754)      MOVKR   R6, @ADMSSPLN(R7)   GET IOS SIZE
04FA8 2662     (01755)      INCR    R6, 2               POINT TO SLOT IN LDS BUFFER FOR ACQ. MODE
04FA9 F0BF3FCC (01756)      MOVRM   R3, @ADMSSPLN(R7)   STORE BID 1 AS 'BID 2' FOR LDS
               (01757) *
04FAB 2612     (01758)      INCR    R1, 2
04FAC 90600002 (01759)      MOVIR   R6, 2               POINT TO 1ST ADAM CONTROL REGISTER
04FAE 9C25FFE0 (01760)      ADDIR   R2, -32(R2)         CONVERT ADAM DISPLACEMENT TO INDEX
04FB0 C8944D30 (01761)      POPMI   R1, @RNSSIOS2(R2)   STORE ADAM COUNTER IN SCROLL REGISTER
04FB2 3018     (01762)      POPXI   R1, R4              GET ACQUIS. MODE: TRIG/CLK PAR
04FB3 503800FF (01763)      MOVKR   R3, R4, MSKSRBYT    EXTRACT TRIG/CLK SETTINGS
04FB5 2662     (01764)      INCR    R6, 2
04FB6 F0B44D30 (01765)      MOVRM   R3, @RNSSIOS2(R2)   STORE IN SCROLL REG 2
04FB8 0D60     (01766)      CLR     R6                  SET-UP FOR LATER ADDRESS FETCH
04FB9 0800     (01767)      EVEN
               (01768) *
04FBA 3C48     (01769)      LRS     R4, 8               EXTRACT ACQUISTION MODE
04FBB 90104FF3 (01770)      MOVIR   R1, ADMSDFCB+HS     POINT TO START OF DUMMY FCB
04FBD FCBF4FCC (01771)      MOVAR   R3, @ADMSSPLN(R7)   POINT TO START OF SCROLL PROGRAM
04FBF 9E300002 (01772)      SUBIR   R3, WS              LOAD SCROLL WANTS POINTER TO SIZE
               (01773) *
04FC1 6026     (01774)      MOVWR   R2, R3              PUT ACQUISITION MODE
04FC2 4C26     (01775)      ADDRR   R2, R3              IN TRAILER ( BA+N+3 )
04FC3 F0440003 (01776)      MOVRM   R4, 3(R2)
04FC5 86004BA2 (01777)      CALL    R0, LDSSADAM        LOAD THE ADAM SCROLL
04FC7 86004CF4 (01778)      CALL    R0, RNSS            EXECUTE SCROLL PROGRAM
               (01779) *
04FC9 3302     (01780)      POPXDL  R0, R1              RESTORE SAVED FCB POINTER
04FCA 0F70     (01781)      RETURN                      AND EXIT
04FCB 0800     (01782)      EVEN
               (01783) *
```

```
                    (01784) *              TABLES FOR ADAM CODE SET-UP
                    (01785) *
                    (01786) *
                    (01787)          EVEN
                    (01788) *
                    (01789) *
04FCC 001C4FEA      (01790) ADM$SPLN ADDR    ADAM$SCS(R6,1)          POINTER TO SINGLE CHANNEL PLAN
04FCE 001C4F30      (01791)          ADDR    ADAM$DCS(R6,1)          POINTER TO DUAL CHANNEL PLAN
                    (01792) *
04FD0 0003          (01793) ADM$SFMA DATA    ASC$FMA*WS + HS, 0
04FD1 0000
04FD2 0003          (01794)          DATA    ADC$FMA*WS + HS, 0
04FD3 0000
                    (01795) *
04FD4 0008          (01796) ADM$HA   DATA    ASC$H1AD*WS
04FD5 001E          (01797)          DATA    ASC$H2AD*WS
04FD6 0008          (01798)          DATA    ADC$H1AD*WS
04FD7 0022          (01799)          DATA    ADC$H2AD*WS
                    (01800) *
04FD8 0011          (01801) ADM$SEP  DATA    ASC$H1SP*WS +HS
04FD9 0027          (01802)          DATA    ASC$H2SP*WS +HS
04FDA 000F          (01803)          DATA    ADC$H1SP*WS +HS
04FDB 0029          (01804)          DATA    ADC$H2SP*WS +HS
                    (01805) *
04FDC 0013          (01806) ADM$HEND DATA    ASC$H1ND*WS + HS
04FDD 0029          (01807)          DATA    ASC$H2ND*WS + HS
04FDE 0017          (01808)          DATA    ADC$H1ND*WS + HS
04FDF 0031          (01809)          DATA    ADC$H2ND*WS + HS
                    (01810) *
04FE0 0006          (01811) ADM$SMA  DATA    ADC$SMA1*WS + HS
04FE1 0027          (01812)          DATA    ADC$SMA2*WS + HS
                    (01813) *
04FE2 0000          (01814) ADM$DFCB DATA    0                       DUMMY FCB FOR LOAD/RUN SCROLL CALLS
04FE3 0000          (01815)          DATA    0                       DUMMY FCB NUMBER
04FE4 0000          (01816)          DATA    0                       SCROLL NUMBER
04FE5 0002          (01817)          DATA    2                       SCROLL TYPE = IOS2
04FE6 0000          (01818)          DATA    0                       LITERAL START VALUE
04FE7 0000          (01819)          DATA    0
```

```
                        (01820) *            ADAM PROGRAM TO SUPPORT SINGLE CHANNEL SAMPLING
                        (01821) *
                        (01822) *
                        (01823)         EVEN
                        (01824) *
                        (01825) *
     04FF8 003A         (01826)         DATA     ASCSSIZE*WS, 0        SCROLL SIZE IN HW : LOGICAL BID 1:0
     04FF9 0000
                        (01827) *
                        (01828) *
                        (01829) ADAMSSCS BEGIN IOS2(ADAMSSCS)          ADAM PROGRAM FOR SINGLE CHANNEL SAMPLING
                        (01830) *
                        (01831)         OPADD    SYNCSTOP, (6 .LS. 10) + (26 .LS. 5) + 4
                        (01832) *
                        (01833) *
A00 04FEA 01300600 (01834)         PR                              SET SCROLL TO READ FROM PERIPHERAL
A01 04FEC 02400000 (01835) ASCSFMA  LOAD(R1, MSS)                   R1 = FIRST MUX ADDRESS
A02 04FEE 03590000 (01836)         ADD(R1, 0, TP)                  SEND INITIAL MUX ADDRESS
A03 04FF0 04300100 (01837)         SET                             START ACQUISITION
                        (01838) *
A04 04FF2 05000000 (01839) ASCSB1AD LOAD(R0, MSS)                   R0 = BUFFER ONE START ADDRESS-SEP
A05 04FF4 06100000 (01840)         SUB(R0, MSS)
                        (01841) *
A06 04FF6 07590010 (01842) #1       ADD(R1,$10,TP)                  RELEASE SAMPLE HOLDS
A07 04FF8 08500010 (01843)         SUB(R1, $10)
A08 04FFA 091A0000 (01844) ASCSB1SP ADD(R0, MSS, TM)                ACQUIRE SAMPLE
A09 04FFC 0A280000 (01845) ASCSB1ND JNE(#1, R0, MSS)                IF END OF BLOCK
                        (01846) *
A0B 04F00 0C302000 (01847)         INT1                            INTERRUPT CSPU
A0C 04F02 0F341000 (01848)         SKIPC(SYNCSTOP)                 STOP PROGRAM AT CSPU REQUEST
A0D 04F04 00000000
A0F 04F06 18300400 (01849)         JUMP(ASCSSTOP)
                        (01850) *
A0F 04F08 10000000 (01851) ASCSB2AD LOAD(R0, MSS)                   R0 = BUFFER TWO START ADDRESS-SEP
A10 04F0A 11100000 (01852)         SUB(R0, MSS)
                        (01853) *
A11 04F0C 12590010 (01854) #2       ADD(R1, $10, TP)                SEND NEXT MUX ADDRESS
A12 04F0F 13500010 (01855)         SUB(R1, $10)
A13 04F10 141A0000 (01856) ASCSB2SP ADD(R0, MSS, TM)                ACQUIRE SAMPLE
A14 04F12 16280000 (01857) ASCSB2ND JNE(#2, R0, MSS)                IF END OF BLOCK
A15 04F14 00000000
                        (01858) *
A17 04F18 18302000 (01859)         INT1                            INTERRUPT CSPU
A18 04F1A 1A301000 (01860)         SKIPS(SYNCSTOP)                 STOP PROGRAM AT CSPU REQUEST
```

```
 A19 04F1C 00000000
 A1A 04F1E 04300000 (01861)          JUMP(ASC$B1AD)              ELSE, CONTINUE SAMPLING
                    (01862) *
 A1B 04F20 1C300180 (01863) ASC$STOP CE1                         TURN ACQUISITION OFF
 A1C 04F22 1D306400 (01864)          STOP                        THEN STOP THE SCROLL
                    (01865) *
           0000001D (01866) ASC$SIZE = *A
                    (01867) *
     04F24          (01868)          END
                    (01869) *
                    (01870) *  SET UP TRAILER INFORMATION
                    (01871) *
     04F24 0002     (01872)          DATA     2                  SET TRANSFER DIRECTION TO WRITE
     04F25 0000     (01873)          DATA     0                  # CHANNELS : ACQUISITION MODE
     04F26 0000     (01874)          DATA     0                  BID 3 : 2
     04F27 0000     (01875)          DATA     0                  # SCROLL REGISTERS TO LOAD
     04F28 0000     (01876)          DATA     0                  1ST SCROLL REGISTER TO LOAD
     04F29 FFFF     (01877)          DATA     -1                 INTEGER SCALAR ID OF OFFSET = NULL
     04F2A FFFF     (01878)          DATA     -1,-1,-1,-1        BINDING CHAIN ANCHORS = NULL
     04F2B FFFF
     04F2C FFFF
     04F2D FFFF
                    (01879)          EVEN
```

```
                     (01880) *           ADAM PROGRAM TO SUPPORT DUAL CHANNEL SAMPLING
                     (01881) *
                     (01882) *
    04F2E 0042       (01883)          DATA    ADCSSIZE*WS,  0          MODULE SIZE IN HW : LOGICAL RID 1:0
    04F2F 0000
                     (01884) *
                     (01885) ADAMSDCS BEGIN IOS2(ADAMSDCS)         ADAM PROGRAM FOR DUAL CHANNEL SAMPLING
                     (01886) *
                     (01887) *
A00 04F30 01300600   (01888)          PR                           SET TO READ FROM PERIPHERAL
A01 04F32 02400000   (01889) ADCSFMA  LOAD(R1, MSS)                R1 = FIRST MUX ADDRESS
A02 04F34 03590000   (01890)          ADD(R1, 0, TP)               SEND INITIAL MUX ADDRESS
A03 04F36 04300100   (01891)          SF1                          START ACQUISITION
                     (01892) *
A04 04F38 05000000   (01893) ADCSB1AD LOAD(R0, MSS)                R0 = BUFFER ONE START ADDRESS-SEP
A05 04F3A 06100000   (01894)          SUB(R0, MSS)
                     (01895) *
                     (01896) #1
A06 04F3C 07590000   (01897) ADCSSMA1 ADD(R1, MSS, TP)             SEND SECOND CHANNEL ADDRESS
A07 04F3E 081A0000   (01898) ADCSB1SP ADD(R0, MSS, TM)             ACQUIRE SAMPLE
A08 04F40 09590000   (01899)          ADD(R1, MSS, TP)             SEND FIRST MUX ADDR : RELEASE S/H
A09 04F42 0A500010   (01900)          SUB(R1, $10)
A0A 04F44 0B1A0000   (01901)          ADD(R0, MSS, TM)             ACQUIRE SAMPLE
A0B 04F46 0C280000   (01902) ADCSB1ND JNE(#1, R0, MSS)             IF END OF BLOCK
                     (01903) *
A0D 04F4A 0F302000   (01904)          INT1                         INTERRUPT THE CSPU
A0E 04F4C 10341000   (01905)          SKIPC(SYNCSTOP)              STOP PROGRAM AT CSPU REQUEST
A0F 04F4F 00000000
A10 04F50 1F300400   (01906)          JUMP(ADCSSTOP)
                     (01907) *
A11 04F52 12000000   (01908) ADCSB2AD LOAD(R0, MSS)                R0 = BUFFER TWO START ADDRESS-SEP
A12 04F54 13100000   (01909)          SUB(R0, MSS)
                     (01910) *
                     (01911) #2
A13 04F56 14590000   (01912) ADCSSMA2 ADD(R1, MSS, TP)             SEND SECOND CHANNEL ADDRESS
A14 04F58 151A0000   (01913) ADCSB2SP ADD(R0, MSS, TM)             ACQUIRE SAMPLE
A15 04F5A 16590000   (01914)          ADD(R1, MSS, TP)             SEND FIRST MUX ADDR : RELEASE S/H
A16 04F5C 17500010   (01915)          SUB(R1, $10)
A17 04F5E 181A0000   (01916)          ADD(R0, MSS, TM)             ACQUIRE SAMPLE
A18 04F60 1A280000   (01917) ADCSB2ND JNE(#2, R0, MSS)             IF END OF BLOCK
A19 04F62 00000000
                     (01918) *
A1B 04F66 1C302000   (01919)          INT1                         INTERRUPT CSPU
A1C 04F68 1F301000   (01920)          SKIPS(SYNCSTOP)              STOP PROGRAM AT CSPU REQUEST
```

```
 A1D 04F6A 00000000
 A1E 04F6C 04300000 (01921)         JUMP(ADCSBIAD)                ELSE, CONTINUE SAMPLING
                    (01922) *
 A1F 04F6E 20300180 (01923) ADCSSTOP   CFI                        FIRST TURN OFF ACQUISITION
 A20 04F70 21306400 (01924)            STOP                       THEN STOP THE SCROLL
           00000021 (01925) ADCSSIZE = *A
                    (01926) *
     04F72          (01927)         END
                    (01928) *
                    (01929) *  SET UP TRAILER INFORMATION
                    (01930) *
     04F72 0002     (01931)         DATA     2                    SET TRANSFER DIRECTION TO WRITE
     04F73 0000     (01932)         DATA     0                    # CHANNELS : ACQUISITION MODE
     04F74 0000     (01933)         DATA     0                    BID 3 : 2
     04F75 0000     (01934)         DATA     0                    # SCROLL REGISTERS TO LOAD
     04F76 0000     (01935)         DATA     0                    1ST SCROLL REGISTER TO LOAD
     04F77 FFFF     (01936)         DATA     -1                   INTEGER SCALAR ID OF OFFSET = NULL
     04F78 FFFF     (01937)         DATA     -1,-1,-1,-1          BINDING CHAIN ANCHORS = NULL
     04F79 FFFF
     04F7A FFFF
     04F7B FFFF
                    (01938)         EVEN
```

```
* ADAMS16S - MODULE TO PROCESS THE 16 CHANNEL ADAM SIMP, SAMP, FCB
*
*   FCB FORMAT  (16 BIT WORD FORMAT SHOWN)
*
*
*          !                        !                         !
*    WORD  !       LEFT BYTE        !       RIGHT BYTE        !
*          !                        !                         !
*   -------!------------------------!-------------------------!
*          !                                                  !
*     0    ! POINTER TO NEXT FCB AND FUNCTION LIST FLAG (LSB) !
*          !                                                  !
*   -------!------------------------!-------------------------!
*          !                        !                         !
*     1    !          74            !       ADAM NUMBER       !
*          !                        !                         !
*   -------!------------------------!-------------------------!
*          !                        !                         !
*     2    !   LOGICAL BUFFER #1    !    LOGICAL BUFFER #0    !
*          !                        !                         !
*   -------!------------------------!-------------------------!
*          !                        !                         !
*     3    !          0             !           0             !
*          !                        !                         !
*   -------!------------------------!-------------------------!
*          !                                                  !
*     4    !                  COUNTER VALUE                   !
*          !                                                  !
*   -------!------------------------!-------------------------!
*          !                        !                         !
*     5    !   ACQUISITION MODE     ! TRIG. & CLOCK PARAMETER !
*          !                        !                         !
*   -------!------------------------!-------------------------!
*
*
*
*
          EJECT
```

```
04F7C F0220001 (01977) ADAMS16S MOVMR  R2, HS(R1)           GET BUFFER IDS
04F7E 9A2000FF (01978)          ANDIR  R2, MSKSRBYT          GET BUFFER ID#0
04F80 4C24     (01979)          ADDRR  R2, R2                CONVERT TO BCT INDEX
04F81 0800     (01980)          EVEN
04F82 C0340582 (01981)          MOVMRL R3, BCTSBA(R2)        GET BUFFER BUS NUMBER AND BASE ADDRESS
04F84 9A300006 (01982)          ANDIR  R3, $6                CLEAR OUT UNWANTED BITS
04F86 90604FEB (01983)          MOVIR  R6, A16SB1D           GET LOC. OF PNTR TO BASE ADDR INST.
04F88 763CFFF0 (01984)          IORWK  R3, R6, $FFF0         CONSTRUCT PROPER FORMAT TO BASE ADDR INST.
04F8A 843C0000 (01985)          MOVRML R3, 0(R6)             STORE IN IOS PROG.
04F8C C0340604 (01986)          MOVMRL R3, BCTSAD(R2)        GET BUFFER SEPERATION AND SIZE
04F8E 2663     (01987)          INCR   R6, $3                STEP TO SEPERATION LOC IN IOS PROG.
04F8F 0800     (01988)          EVEN
04F90 F03C0000 (01989)          MOVRM  R3, 0(R6)             STORE IN IOS PROG.
04F92 4838     (01990)          MULRR  R3, R4                COMPUTE DISPLACEMENT TO BUFFER END LOC.
04F93 3C31     (01991)          LRS    R3, $1                GET BASE ADDR. IN PROPER IOS FMT.
04F94 FC340583 (01992)          ADDMR  R3, BCTSBA+HS(R2)     ADD IN BASE ADDRESS TO YIELD BUFFER END
04F96 F030503F (01993)          MOVRM  R3, A16SB1F+HS        STORE IN IOS PROG.
               (01994) *
04F98 F0220001 (01995)          MOVMR  R2, HS(R1)            GET BUFFER IDS
04F9A 9A20FF00 (01996)          ANDIR  R2, MSKSLBYT          GET BUFFER ID #1
04F9C 0220     (01997)          TEST   R2                    SEE IF BUFFER IS USED
04F9D 80104FBB (01998)          JMP    ADMSC0, EQZ
               (01999) *
04F9F 3C28     (02000)          LRS    R2, $8                GET BUFFER ID IN PROPER FORMAT
04FA0 4C24     (02001)          ADDRR  R2, R2                IT IS, SO CONVERT TO BCT INDEX
04FA1 0800     (02002)          EVEN
04FA2 C0340582 (02003)          MOVMRL R3, BCTSBA(R2)        GET BUFFER BUS NUMBER AND BASE ADDR.
04FA4 9A300006 (02004)          ANDIR  R3, $6                CLEAR OUT UNWANTED BITS
04FA6 9060504C (02005)          MOVIR  R6, A16SB2D           GET LOC. OF POINTER TO BASE ADDR. INST.
04FA8 763CFFF0 (02006)          IORWK  R3, R6, $FFF0         CONSTRUCT PROPER FORMAT FOR BASE ADDR.
04FAA 843C0000 (02007)          MOVRML R3, 0(R6)             STORE IN IOS PROG.
04FAC C0340604 (02008)          MOVMRL R3, BCTSAD(R2)        GET BUFFER SEPERATION AND SIZE
04FAE 2663     (02009)          INCR   R6, $3                STEP TO SEPERATION LOC IN IOS PROG.
04FAF 0800     (02010)          EVEN
04FB0 F03C0000 (02011)          MOVRM  R3, 0(R6)             STORE IN IOS PROG.
04FB2 4838     (02012)          MULRR  R3, R4                COMPUTE DISPLACEMENT TO BUFFER END.
04FB3 3C31     (02013)          LRS    R3, $1                PUT BASE ADDR. IN PROPER IOS FMT.
04FB4 FC340583 (02014)          ADDMR  R3, BCTSBA+HS(R2)     ADD IN BASE ADDRESS TO YIELD BUFFER END
04FB6 F0305093 (02015)          MOVRM  R3, A16SB2F+HS        STORE IN IOS PROG.
               (02016) *
               (02017) *
               (02018) * DISTRIBUTE FCB PARAMETERS
               (02019) *
               (02020) *
```

```
04FB8 3602     (02021) ADMSCO  PUSHXIL R0, R1           SAVE FCB POINTER
04FB9 2711     (02022)         DECR    R1, HS           PREPARE FOR POP OPERATIONS
04FBA 3014     (02023)         POPXI   R1, R2           GET ADAM NUMBER
04FBB 0800     (02024)         EVEN
04FBC 9A2000FF (02025)         ANDIR   R2, MSKSRBYT     EXTRACT SCROLL IDENTIFIER
04FBE F0204FF4 (02026)         MOVRM   R2, ADMSDFCB+WS  STORE IN DUMMY FCB
               (02027) *
04FC0 F0320001 (02028)         MOVMR    R3, 1(R1)          GET BID 0
04FC2 504600FF (02029)         MOVKR    R4, R3, MSKSRBYT
04FC4 F0404FFD (02030)         MOVRM    R4, ADM$16C-HS     STORE IN IOS HEADER
               (02031) *
04FC6 3C38     (02032)         LRS      R3, 8              GET BID 1
04FC7 90404FFF (02033)         MOVIR    R4, ADM$16C        COMPUTE POINTER TO 'BID 2'
04FC9 FC404FFC (02034)         ADDMR    R4, ADM$16C-2      IN SCROLL BUFFER AREA
04FCB F0380002 (02035)         MOVRM    R3, 2(R4)
               (02036) *
04FCD 2612     (02037)         INCR    R1, 2
04FCE 90600002 (02038)         MOVIR    R6, 2               POINT TO 1ST ADAM CONTROL REGISTER
04FD0 9C25FFF0 (02039)         ADDIR   R2, -32(R2)      CONVERT ADAM DISPLACEMENT TO INDEX
04FD2 C8944D30 (02040)         POPMI   R1, @RNS$IOS2(R2) STORE ADAM COUNTER IN SCROLL REGISTER 1
               (02041) *
04FD4 301B     (02042)         POPXI   R1, R4           GET ACQUIS. MODE : TRIG/CLK PARA.
04FD5 503B00FF (02043)         MOVKR   R3, R4, MSKSRBYT EXTRACT TRIG. & CLOCK SETTINGS
04FD7 2662     (02044)         INCR    R6, 2
04FDB F0B44D30 (02045)         MOVRM   R3, @RNS$IOS2(R2) STORE IN SCROLL REG 2
               (02046)         EVEN
04FDA 3C4B     (02047)         LRS      R4, 8              EXTRACT ACQUISITION MODE
04FDB 90104FF3 (02048)         MOVIR   R1, ADMSDFCB+HS  POINT TO START OF DUMMY FCB
04FDD FC304FFF (02049)         MOVAR   R3, ADM$16C      POINT TO START OF SCROLL PROG.
04FDF 9F300002 (02050)         SUBIR   R3, WS           LOAD SCROLL  (R3 = START OF SCROLL BUFFER)
               (02051) *
04FE1 6026     (02052)         MOVWR    R2, R3             PUT ACQUISITION MODE IN SCROLL BUFFER
04FE2 4C26     (02053)         ADDRR    R2, R3             ( BA+N+3 )
04FE3 F0440003 (02054)         MOVRM    R4, 3(R2)
               (02055) *
04FE5 B6004BA2 (02056)         CALL    R0, LDSSADAM     LOAD THE ADAM WITH THE SCROLL PROGRAM
04FE7 B6004CF4 (02057)         CALL     R0, RNS$           EXECUTE SCROLL PROGRAM
               (02058) *
04FE9 3302     (02059)         POPXIL  R0, R1           RESTORE SAVED FCB POINTER
04FEA 0F70     (02060)         RETURN                   AND EXIT
04FEB 0800     (02061)         EVEN
```

```
                        (02062) *            ADAM PROGRAM TO SUPPORT 16 CHANNEL SAMPLING
                        (02063) *
                        (02064) *
    04FFC 0088          (02065)           DATA     A16SSIZ*WS, 0    MODULE SIZE IN HALF-WORDS : LOGICAL BID'S
    04FFD 0000
                        (02066) *
                        (02067) ADMS16C BEGIN   IOS2(ADMS16C)    ADAM PROGRAM FOR 16 CHANNEL SAMPLING
                        (02068) *
                        (02069)           OPADD    SYNCSTOP, (6 .LS. 10) + (26 .LS. 5) + 4
                        (02070) *
A00 04FFF 01300600 (02071)           FR                        SET TO READ FROM ADAM
          00004FF0 (02072)           A16SFMA=#L
A01 04FF0 02408000 (02073)           LOAD     (R1, $8000)      SET ADAM REGISTER #1 TO READ FROM CHANNEL #00
A02 04FF2 03590000 (02074)           ADD      (R1, 0, TP)      ACTIVATE CHANNEL #00
A03 04FF4 04300100 (02075)           SE1                       ENABLE ADAM DATA ACQUISITION
A04 04FF6 05300700 (02076)           SE2
                        (02077) *
          00004FF8 (02078)           A16S81D=#L
A05 04FF8 06000000 (02079) A16S81A LOAD     (R0, MSS)        SET R0 TO CONTAIN THE MAP BUFFER MEMORY ADDR.,
A06 04FFA 07100000 (02080)           SUB      (R0, MSS)        AND SEPARATION.
                        (02081) *
                        (02082) #1
A07 04FFC 08590001 (02083)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #01
A08 04FFE 091A0001 (02084)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #00 INTO MAP MEMORY
A09 05000 0A590001 (02085)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #02
A0A 05002 0B1A0001 (02086)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #01 INTO MAP MEMORY
A0B 05004 0C590001 (02087)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #03
A0C 05006 0D1A0001 (02088)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #02 INTO MAP MEMORY
A0D 05008 0E590001 (02089)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #04
A0E 0500A 0F1A0001 (02090)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #03 INTO MAP MEMORY
A0F 0500C 10590001 (02091)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #05
A10 0500E 111A0001 (02092)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #04 INTO MAP MEMORY
A11 05010 12590001 (02093)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #06
A12 05012 131A0001 (02094)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #05 INTO MAP MEMORY
A13 05014 14590001 (02095)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #07
A14 05016 151A0001 (02096)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #06 INTO MAP MEMORY
A15 05018 16590001 (02097)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #08
A16 0501A 171A0001 (02098)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #07 INTO MAP MEMORY
A17 0501C 18590001 (02099)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #09
A18 0501E 191A0001 (02100)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #08 INTO MAP MEMORY
A19 05020 1A590001 (02101)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #10
A1A 05022 1B1A0001 (02102)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #09 INTO MAP MEMORY
A1B 05024 1C590001 (02103)           ADD      (R1, 1, TP)      ACTIVATE CHANNEL #11
A1C 05026 1D1A0001 (02104)           ADD      (R0, 1, TM)      PUT DATA FROM CHANNEL #10 INTO MAP MEMORY
```

```
A1D 05028 1E590001 (02105)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #12
A1E 0502A 1F1A0001 (02106)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #11 INTO MAP MEMORY
A1F 0502C 20590001 (02107)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #13
A20 0502E 211A0001 (02108)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #12 INTO MAP MEMORY
A21 05030 22590001 (02109)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #14
A22 05032 231A0001 (02110)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #13 INTO MAP MEMORY
A23 05034 24590001 (02111)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #15
A24 05036 251A0001 (02112)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #14 INTO MAP MEMORY
A25 05038 26590021 (02113)          ADD     (R1, S21, TP)    SET CHANNEL # TO 00 AND ACTIVATE
A26 0503A 271A0001 (02114)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #15 INTO MAP MEMORY
                   (02115) *
A27 0503C 28500030 (02116)          SUB     (R1, S30)        RESET S/H
                   (02117) *
          0000503E (02118)          A16SH1E=#L
A28 0503E 2A280000 (02119)          JNE     (#1, R0, MSS)    LOOP AGAIN IF BUFFER NOT FILLED
A29 05040 00000000
                   (02120) *
A2B 05044 2C302000 (02121)          INT1                     OTHERWIZE, INTERRUPT THE CSPU
A2C 05046 2E341000 (02122)          SKIPC   (SYNCSTOP)       AND STOP THE PROGRAM AT THE CSPU'S REQUEST
A2D 05048 00000000
A2E 0504A 59300400 (02123)          JUMP    (A16SSTOP)
                   (02124) *
                   (02125) * OTHERWISE CONTINUE SAMPLING INTO THE SECOND BUFFER
                   (02126) *
          0000504C (02127)          A16SH2D=#L
A2F 0504C 30000000 (02128)          LOAD    (R0, MSS)        SET R0 TO CONTAIN THE SECOND MEMORY ADDR.,
A30 0504E 31100000 (02129)          SUB     (R0, MSS)        AND SEPARATION.
                   (02130) *
                   (02131) #2
A31 05050 32590001 (02132)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #01
A32 05052 331A0001 (02133)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #00 INTO MAP MEMORY
A33 05054 34590001 (02134)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #02
A34 05056 351A0001 (02135)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #01 INTO MAP MEMORY
A35 05058 36590001 (02136)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #03
A36 0505A 371A0001 (02137)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #02 INTO MAP MEMORY
A37 0505C 38590001 (02138)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #04
A38 0505E 391A0001 (02139)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #03 INTO MAP MEMORY
A39 05060 3A590001 (02140)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #05
A3A 05062 3B1A0001 (02141)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #04 INTO MAP MEMORY
A3B 05064 3C590001 (02142)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #06
A3C 05066 3D1A0001 (02143)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #05 INTO MAP MEMORY
A3D 05068 3E590001 (02144)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #07
A3E 0506A 3F1A0001 (02145)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #06 INTO MAP MEMORY
A3F 0506C 40590001 (02146)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #08
```

```
 A40 0506E 411A0001 (02147)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #07 INTO MAP MEMORY
 A41 05070 42590001 (02148)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #09
 A42 05072 431A0001 (02149)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #08 INTO MAP MEMORY
 A43 05074 44590001 (02150)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #10
 A44 05076 451A0001 (02151)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #09 INTO MAP MEMORY
 A45 05078 46590001 (02152)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #11
 A46 0507A 471A0001 (02153)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #10 INTO MAP MEMORY
 A47 0507C 48590001 (02154)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #12
 A48 0507E 491A0001 (02155)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #11 INTO MAP MEMORY
 A49 05080 4A590001 (02156)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #13
 A4A 05082 4B1A0001 (02157)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #12 INTO MAP MEMORY
 A4B 05084 4C590001 (02158)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #14
 A4C 05086 4D1A0001 (02159)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #13 INTO MAP MEMORY
 A4D 05088 4E590001 (02160)          ADD     (R1, 1, TP)      ACTIVATE CHANNEL #15
 A4E 0508A 4F1A0001 (02161)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #14 INTO MAP MEMORY
 A4F 0508C 50590021 (02162)          ADD     (R1, $21, TP)    SET CHANNEL # TO 00 AND ACTIVATE
 A50 0508E 511A0001 (02163)          ADD     (R0, 1, TM)      PUT DATA FROM CHANNEL #15 INTO MAP MEMORY
                    (02164) *
 A51 05090 52500030 (02165)          SUB     (R1, $30)        RESET S/H
                    (02166) *
           00005092 (02167)          A16$B2F=#L
 A52 05092 54280000 (02168)          JNE     (#2, R0, M$$)    LOOP AGAIN IF BUFFER NOT FILLED
 A53 05094 00000000
                    (02169) *
 A55 05098 56302000 (02170)          INT1                     OTHERWISE, INTERRUPT THE CSPU
 A56 0509A 58301000 (02171)          SKIPS   (SYNCSTOP)       AND STOP THE PROGRAM AT THE CSPU'S REQUEST
 A57 0509C 00000000
 A58 0509E 05300400 (02172)          JUMP    (A16$H1A)        ELSE, CONTINUE SAMPLING
                    (02173) *
 A59 050A0 5A300180 (02174) A16$STOP CF1                      TURN OFF ACQUISITION OF ADAM
 A5A 050A2 5B300780 (02175)           CF2
 A5B 050A4 5C306000 (02176)          STOP                     THEN STOP THE SCROLL
                    (02177) *
           0000005C (02178) A16$SIZ = #A
                    (02179) *
     050A6          (02180)          END
                    (02181) *
                    (02182) *  SET UP TRAILER INFORMATION
                    (02183) *
     050A6 0002     (02184)           DATA      2              SET TRANSFER DIRECTION TO WRITE
     050A7 0000     (02185)           DATA      0              # CHANNELS : ACQUISITION MODE
     050A8 0000     (02186)           DATA      0              RIO 3 : 2
     050A9 0000     (02187)           DATA      0              # SCROLL REGISTERS TO LOAD
     050AA 0000     (02188)           DATA      0              1ST SCROLL REGISTER TO LOAD
```

```
050AB FFFF     (02189)          DATA     -1                    INTEGER SCALAR ID OF OFFSET = NULL
050AC FFFF     (02190)          DATA     -1,-1,-1,-1           BINDING CHAIN ANCHORS = NULL
050AD FFFF
050AE FFFF
050AF FFFF
```

```
          * ADMRB      MODULE TO PROCESS ADAM ROTATE BUFFER FCB
          *
          *
00000003                 $M = 3                        SET TO MAP-300 ASSEMBLY
          *
                       OPADD TABS,(1 .LS. 14) + 19   DEFINE TAB PSEUDO OP
          *
          *
          *  FCB FORMAT   (16 BIT WORD FORMAT SHOWN)
          *
          *
          *        !                          !                          !
          *   WORD !        LEFT BYTE         !        RIGHT BYTE        !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    0   !   POINTER TO NEXT FCB AND FUNCTION LIST FLAG(LSB)   !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    1   !           229            !                          !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    2   !           YBUF           !           ISA            !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    3   !           UBUF           !                          !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    4   !            0             !            0             !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
          *        !                          !                          !
          *    5   !            0             !            0             !
          *        !                          !                          !
          *   -----!--------------------------!--------------------------!
```

```
                  (02230) *      SPECIAL BINDING MODULE FOR ROTATE ADAM BUFFER
                  (02231) *
                  (02232) *  THE INPUT BUFFER 'U' IS SEPARATED INTO TWO 'BUFFERS'
                  (02233) *  A AND B, WHERE
                  (02234) *
                  (02235) *        A = 'BUFFER' STARTING WITH THE TRIGGER ADDRESS + 1
                  (02236) *            AND EXTENDING TO THE LAST SAMPLE OF THE U BUFFER
                  (02237) *
                  (02238) *        B = 'BUFFER' STARTING WITH THE FIRST SAMPLE OF THE
                  (02239) *            U BUFFER AND EXTENDING TO THE SAMPLE INCLUDING THE
                  (02240) *            TRIGGER ADDRESS
                  (02241) *
                  (02242) *  THIS SPECIAL SUPPORT MODULE EXTRACTS THE TRIGGER ADDRESS
                  (02243) *  FROM A PAIR OF CONSECUTIVE INTEGER SCALARS AND BINDS
                  (02244) *  THE PERTINENT INFORMATION FOR 'BUFFERS' A AND B INTO THE
                  (02245) *  APS MODULE
                  (02246) *
                  (02247) *
050B0 8B100FFA    (02248) ADMRBS   CALL     R1, APSBNDR          PERFORM STANDARD BINDING FIRST
                  (02249) *
                  (02250) *     REG. 3 = HIGH ORDER BIT OF TRIGGER ADDRESS
                  (02251) *     REG. 4 = LOW ORDER 16 BITS OF TRIGGER ADDRESS
                  (02252) *
050B2 F0720001    (02253)          MOVMR    R7, 1(R1)            INTEGER SCALAR 'A' ID
050B4 9A7000FF    (02254)          ANDIR    R7, MSK$RBYT
                  (02255) *
050B6 C03E0502    (02256)          MOVMRL   R3, ISVTS(R7)        GET TRIGGER VALUE
                  (02257) *
050B8 2641        (02258)          INCR     R4, 1                ADJUST TO 'ABASE'
050B9 81405OBD    (02259)          JMP      ADMRBS0, NCRY        IF CARRY,
050BB D2000003    (02260)          SMB      0, R3                SET BIT 17 IN ADDRESS
                  (02261) *
050BD F0405159    (02262) ADMRBS0  MOVRM    R4, RBABAS+1         PUT TRIGGER ADDRESS IN APS
050BF F050515B    (02263)          MOVMR    R5, RBABAS           PICK UP 'ABASE' INSTRUCTION
050C1 56560001    (02264)          IORRR    R5, R3, 1            GET HIGH ORDER BIT
050C3 F050515B    (02265)          MOVRM    R5, RBABAS           STORE IN CONSTRUCTED BLOCK
                  (02266) *
050C5 D230515B    (02267)          SMB      3, RBABAS            SET SHORT BIT IN LOAD INSTRUCTIONS
050C7 D2305156    (02268)          SMB      3, RBABAS-2
050C9 D2305150    (02269)          SMB      3, RBABAS-8
                  (02270) *
050CB 3F31        (02271)          ROR      R3, 1                MOVE HIGH ORDER ADDRESS BIT TO POS 17
050CC 563BFFFF    (02272)          IORRR    R3, R4, $FFFF        CONCATENATE BIT 17 WITH ADDRESS
                  (02273) *
```

SPECIAL BINDING MODULE FOR ROTATE ADAM BUFFER

```
050CF F0220002 (02274)           MOVMR    R2, 2(R1)           GET UBUF
050D0 9A20FF00 (02275)           ANDIF    R2, MSKSLBYT        GET UBUF ID FROM LEFT BYTE
050D2 3C27     (02276)           LRS      R2, 7               CONVERT TO BCT INDEX
050D3 F0740582 (02277)           MOVMR    R7, BCTSBA(R2)      GET HIGH ORDER ADDRESS BIT
050D5 506E0001 (02278)           MOVKR    R6, R7, 1
050D7 3E61     (02279)           ROR      R6, 1               MOVE TO BIT POS 17
050D8 F0540583 (02280)           MOVMR    R5, BCTSBA+HS(R2)   LOW ORDER 16 BITS
050DA 566AFFFF (02281)           IORKR    R6, R5, $FFFF       CONCATENATE BIT 17 TO ADDRESS
               (02282) *
050DC F0740604 (02283)           MOVMR    R7, BCTSAD(R2)      GET SEPARATION
050DE 405C     (02284)           MOVRR    R5, R6             R5 = UBASE
050DF FC540605 (02285)           ADDMR    R5, BCTSAD+HS(R2)     COMPUTE BUFFER EXTENT
050E1 4256     (02286)           CMPRR    R5, R3                TRIG BEYOND BUFFER RANGE?
050E2 813050ED (02287)           JMP      ADMRBS1, GEZ          JUMP IF NOT
               (02288) *
050E4 403C     (02289)           MOVRR    R3, R6                TRIG = UBASE
050E5 F0305159 (02290)           MOVRM    R3, RBABAS+HS         STORE IN APS PROGRAM
050E7 0040     (02291)           CLR      R4
050E8 56470000 (02292)           IORKR    R4, R3, $10000        UPPER BIT SET?
050EA 1810     (02293)           SKP      EOZ                   SKIP IF NOT
050EB D200515B (02294)           SMB      0, RBABAS             ELSE, SET UPPER BIT
               (02295) *
050ED 4E66     (02296) ADMRBS1 SUBRR    R6, R3   UBASE - TRIG
050EE 1820     (02297)           SKP        GTZ
050EF 0860     (02298)           NEG        R6                OFFSET MUST BE NON - NEGATIVE
050F0 F02007B4 (02299)           MOVRM    R2, TEMSO           SAVE R2 = BCT INDEX
050F2 86001B50 (02300)           CALL     R0, IDIVS           BSIZE := ABS(UBASE-TRIG )/SEP
               (02301) *
050F4 F02007B4 (02302)           MOVMR    R2, TEMSO
050F6 F0540605 (02303)           MOVMR    R5, BCTSAD + HS(R2)  GET BUFFER SIZE - 1
050F8 4E5F     (02304)           SUBRR    R5, R7              ASIZE -1 := USIZE - BSIZE - 1
               (02305) *
050F9 F0505125 (02306)           MOVRM    R5, BINDBLOK+3      ASIZE - 1
050FB 2771     (02307)           DECR     R7, 1
050FC F070512B (02308)           MOVRM    R7, BINDBLOK+9      BSIZE - 1
               (02309) *
050FE 0F70     (02310)           RETURN
```

```
                    (02311) '       APU - ROTATE ADAM BUFFER
                    (02312) *
                    (02313) *  THIS ROUTINE ASSUMES THAT DATA IS IN 16 BIT FORMAT -
                    (02314) *  EITHER LONG & FIXED OR SHORT & FLOATING PT.
                    (02315) *
                    (02316) *    THERE IS ONE ADDRESSOR FOR BOTH DATA TYPES
                    (02317) *
                    (02318) *    THE SPECIAL BINDING MODULE FORCES THE DATA MODE
                    (02319) *    TO BE SHORT, SINCE THERE IS NO ARITHMETIC OPERATION
                    (02320) *    PERFORMED ON THE DATA.
                    (02321) *
                    (02322) *
    050FF 0800      (02323)          EVEN
                    (02324) *
    05100 0000      (02325)          DATA ADMRBSSA                START ADDRESS
    05101 000A      (02326)          DATA ADMRBSSZ                SIZE OF MODULE
                    (02327) *
                    (02328) ADMRBAPU BEGIN APU(ADMRB)
                    (02329) *
                    (02330) *
A00 05102 0BEC0000 (02331) ADMRBSSA MOV( IQA, OQ ) \ NOP
A01 05104 901C0000 (02332)          JUMPC( ADMRBSSA, FO )
A02 05106 20322032 (02333)          CLEAR( RA )
A03 05108 00000000 (02334)          NOP
A04 0510A 10000000 (02335)          JUMP( 0 )
                    (02336) *
A05 0510C 00000000 (02337)          NOP                          INSERT DUMMY NOP'S FOR 200/300
A06 0510E 00000000 (02338)          NOP                          OBJECT COMPATIBILITY
A07 05110 00000000 (02339)          NOP                          ( # NOP'S = # REAL INSTR'S ABOVE)
A08 05112 00000000 (02340)          NOP
A09 05114 00000000 (02341)          NOP
                    (02342) *
          0000000A (02343) ADMRBSSZ = #A - ADMRBSSA
                    (02344) *
    05116           (02345)          END
```

```
                        (02346) '      APS - ROTATE ADAM BUFFER
                        (02347) *
                        (02348) *  SPECIAL BINDING MODULE MODIFIES BASE ADDRESSES FOR ALL
                        (02349) *  THREE BUFFERS 'A', 'B', AND Y IN THE CONSTRUCTED
                        (02350) *  INSTRUCTION BLOCK
                        (02351) *       1) SETS UP 'ABASE'
                        (02352) *       2) FORCES THE SHORT BIT FOR THE LOAD INSTRUCTIONS
                        (02353) *
                        (02354) *
                        (02355) * INPUT STREAM - UK; FI
                        (02356) * OUTPUT STREAM - YK; FO
                        (02357) *
                        (02358)           EVEN
    05116 0000514E (02359)           ADDR RBAPSSI                 POINTER TO CONSTR. INST. BLOCK
    05118 0000511E (02360)           ADDR ADMRBAPS                POINTER TO SCALAR BLOCK ( NOT USED )
    0511A 0000     (02361)           DATA      0
    0511B 003C     (02362)           DATA RBAPSSZ                 MODULE SIZE
    0511C 00005146 (02363)           ADDR RBAPSSA                 CHAIN ANCHOR
                        (02364)           EVEN
                        (02365) *
                        (02366) *
                        (02367) ADMRBAPS BEGIN APS(ROTBUFAPS)
                        (02368) *
                        (02369) *
A00 0511E 00201040 (02370)           JSN(RBOUT, P2)               SET OUTPUT PC
A01 05120 02300030 (02371)           SET(RO)  ENABLE OUTPUT
                        (02372) *
                        (02373) *     INPUT PROGRAM
                        (02374) *
          00005122 (02375) BINDBLOK = #L
                        (02376) *
A02 05122 04481000 (02377)           LOAD(BR0, [1], S)            [ABASE] (TRIGGER)
A03 05124 06500000 (02378)           LOAD(BR1, M$$)               [ASIZE] - 1
A04 05126 08020000 (02379)           SUB(BR0, M$$)                ABASE - SEPARATION
                        (02380) *
A05 05128 0A681006 (02381)           LOAD(BR2, [1], S)            [BBASE]  ( = UBASE )
A06 0512A 0C700000 (02382)           LOAD(BR3, M$$)               [BSIZE] - 1
A07 0512C 0E220000 (02383)           SUB(BR2, M$$)                BBASE - SEPARATION
                        (02384) *
A08 0512E 10BA9006 (02385) ATRANS    ADD(BR0, [9], TE)
A09 05130 121908B1 (02386)           SUBL(BR1, 1), JUMPP(ATRANS)
                        (02387) *
A0A 05132 14390CBB (02388)           ADDL(BR3, 0), JUMPP(BTRANS)
A0B 05134 16000F60 (02389)           JUMP(RBENDIN)
```

```
                        (02390) *
A0C 05136 18AA9008 (02391) BTRANS   ADD(BR2, [9], TF)
A0D 05138 1A390CB1 (02392)          SUBL(BR3, 1), JUMPP(BTRANS)
                        (02393) *
A0E 0513A 1C200031 (02394) RBENDIN  CLEAR(RI)
A0F 0513C 1F000020 (02395)          NOP(0)
                        (02396) *
                        (02397) *
A10 0513E 20300032 (02398) RBOUT    SET(RA)
                        (02399) *
A11 05140 2248000A (02400)          LOAD(BW0, [0], S)          [YBASE]
A12 05142 24500000 (02401)          LOAD(BW1, MSS)             [YSIZE] - 1
A13 05144 26020000 (02402)          SUB(BW0, MSS)              BASE := BASE - SEPERATION
                        (02403) *
A14 05146 28BA8006 (02404) OUTTRANS ADD(BW0, [8], TF)
A15 05148 2A1114B1 (02405)          SUBL(BW1, 1), JUMPP(OUTTRANS)
                        (02406) *
A16 0514A 2C200030 (02407)          CLEAR(RO)
A17 0514C 2F000020 (02408)          NOP(0)
                        (02409) *
          00005146 (02410) RBAPSSA = #C
                        (02411) *
                        (02412) *
      0514E            (02413)          END
                        (02414) *
                        (02415) *          STORAGE FOR CONSTRUCTED INSTRUCTIONS
                        (02416) *
      0514E 00000000 (02417) RBAPSSI  DATA 6F'0,0'
       ...
                        (02418) *
          00005158 (02419) RBABAS = #L - 2                    POINT TO 'ABASE' SUBSTITUTION
                        (02420) *
          0000003C (02421) RBAPSSZ = #L - ADMRBAPS            MODULE SIZE
                        (02422) *
```

```
                (02423) * CROSS REFERENCE
                (02424) *
                (02425) *
                (02426) * END OF MODULE
                (02427) *
                (02428) *
       0000515A (02429)         TOFSCUR=#L                  CALCULATE END ADDRESS OF MODULE
                (02430) *
                (02431) *
   0515A        (02432)         END
```

```
A16SB1A:   00005 (02079) (02172)
A16SB1D:   04FFB (01983) (02078)
A16SB1F:   0503F (01993) (02118)
A16SB2D:   0504C (02005) (02127)
A16SB2F:   05092 (02015) (02167)
A16SFMA:   04FF0 (02072)
A16SSIZ:   0005C (02065) (02178)
A16SSTOP:  00059 (02123) (02174)
ACQSLIST:  04909 (00214) (00767) (01142) (01317) (01325)
ADAMS16S:  04F7C (00091) (01977)
ADAMSDCS:  04F30 (01791) (01885)
ADAMSSCS:  04FFA (01790) (01829)
ADAMSSS1:  04F46 (01679) (01735)
ADAMSSS2:  04F6C (01702) (01707)
ADAMSSS3:  04F92 (01719) (01733)
ADAMSSSP:  04F32 (00090) (01665)
ADCSB1AD:  00004 (01798) (01893) (01921)
ADCSB1ND:  0000B (01808) (01902)
ADCSB1SP:  00007 (01803) (01898)
ADCSB2AD:  00011 (01799) (01908)
ADCSB2ND:  00018 (01809) (01917)
ADCSB2SP:  00014 (01804) (01913)
ADCSFMA:   00001 (01794) (01889)
ADCSSIZE:  00021 (01883) (01925)
ADCSSMA1:  00006 (01811) (01897)
ADCSSMA2:  00013 (01812) (01912)
ADCSSTOP:  0001F (01906) (01923)
ADMS16C:   04FFF (02030) (02033) (02034) (02049) (02067)
ADMSBA:    04FD4 (01690) (01796)
ADMSBEND:  04FDC (01710) (01806)
ADMSCO:    04FBB (01998) (02021)
ADMSDECH:  04EF2 (01746) (01770) (01814) (02026) (02048)
ADMSFMA:   04ED0 (01674) (01793)
ADMSSEP:   04FD8 (01698) (01801)
ADMSSMA:   04EF0 (01721) (01811)
ADMSSEGN:  04FCC (01675) (01691) (01699) (01705) (01711) (01714) (01716) (01727) (01731) (01750)
                 (01754) (01756) (01771) (01790)
ADMRBS:    05080 (00101) (02248)
ADMRBS0:   0508D (02259) (02262)
ADMRBS1:   050FD (02287) (02296)
ADMRBSSA:  00000 (02325) (02331) (02332) (02343)
ADMRBSSZ:  0000A (02326) (02343)
ADMRBAPS:  0511F (00100) (02360) (02367) (02421)
ADMRBAPU:  05102 (00099) (02328)
```

```
  ADMRRECH:   000F5 (00019) (00097)
  AFDTSORG:   008FB (00020) (00097)
  APSRNDR:    00FEA (00021) (02248)
  ASCSB1AD:   00004 (01796) (01839) (01861)
  ASCSB1ND:   00009 (01806) (01845)
  ASCSB1SP:   00008 (01801) (01844)
  ASCSB2AD:   0000F (01797) (01851)
  ASCSB2ND:   00014 (01807) (01857)
  ASCSB2SP:   00013 (01802) (01856)
  ASCSEMA:    00001 (01793) (01835)
  ASCSSIZE:   0001D (01826) (01866)
  ASCSSTOP:   0001B (01849) (01863)
  ATRANS:     00008 (02385) (02386)
  BCTSAD:     00604 (00023) (00842) (00869) (01563) (01571) (01578) (01587) (01596) (01694) (01986)
                    (02008) (02283) (02285) (02303)
  BCTSAT:     006B6 (00024) (00600) (00601) (00602) (00603) (01008)
  BCTSBA:     00582 (00025) (00751) (01537) (01560) (01580) (01688) (01709) (01715) (01981) (01992)
                    (02003) (02014) (02277) (02280)
  BCTSSIZ:    00040 (00026) (01004)
  BIDSCHK:    04CBC (00790) (00795) (00821) (00826) (00998)
  BIDSRET:    04C9A (01000) (01013)
  BINDBLOK:   05122 (02306) (02308) (02375)
  BITSBD:     00008 (00027) (01008)
  BITSIORD:   00006 (00028) (00600) (00601)
  BITSIOWT:   00009 (00029) (00602) (00603)
  BTRANS:     0000C (02388) (02391) (02392)
  CSWSI161:   002B6 (00031) (00110) (01180)
  CSWSI162:   002B8 (00114) (01181)
  CSWSI163:   002BA (00115) (01182)
  CSWSI171:   002BC (00116)
  CSWSI172:   002BE (00117)
  CSWSI173:   002C0 (00118)
  CSWSI181:   002C2 (00119)
  CSWSI182:   002C4 (00120)
  CSWSI183:   002C6 (00121)
  CSWSI191:   002C8 (00122)
  CSWSI192:   002CA (00123)
  CSWSI193:   002CC (00124)
  CSWSI201:   002CE (00125)
  CSWSI202:   002D0 (00126)
  CSWSI203:   002D2 (00127)
  CSWSI211:   002D4 (00128)
  CSWSI212:   002D6 (00129)
  CSWSI213:   002D8 (00130)
```

CROSS REFERENCE

```
 CSWSI221:  0020A (00131)
 CSWSI222:  0020C (00132)
 CSWSI223:  0020E (00133)
 CSWSI231:  002E0 (00134)
 CSWSI232:  002E2 (00135)
 CSWSI233:  002E4 (00136)
 CSWSIOS:   00322 (00032) (00145) (01179)
 CSWSSTOR:  00405 (00033) (00113) (00114) (00115) (00116) (00117) (00118) (00119) (00120) (00121)
                  (00122) (00123) (00124) (00125) (00126) (00127) (00128) (00129) (00130) (00131)
                  (00132) (00133) (00134) (00135) (00136) (00148) (00149) (00150) (00151) (00152)
                  (00153) (00154) (00155) (00156) (00157) (00158) (00159) (00160) (00161) (00162)
                  (00163) (00164) (00165) (00166) (00167) (00168) (00169) (00170) (00171)
 D16SINT1:  049F2 (00113) (00148) (00248) (00256)
 D16SINT2:  04A00 (00114) (00149) (00262) (00270)
 D16SINT3:  04A0E (00115) (00150) (00276) (00280)
 D17SINT1:  04A16 (00116) (00151) (00285) (00293)
 D17SINT2:  04A24 (00117) (00152) (00299) (00307)
 D17SINT3:  04A32 (00118) (00153) (00313) (00317)
 D18SINT1:  04A3A (00119) (00154) (00322) (00330)
 D18SINT2:  04A48 (00120) (00155) (00336) (00344)
 D18SINT3:  04A56 (00121) (00156) (00350) (00354)
 D19SINT1:  04A5E (00122) (00157) (00359) (00367)
 D19SINT2:  04A6C (00123) (00158) (00373) (00381)
 D19SINT3:  04A7A (00124) (00159) (00387) (00391)
 D20SINT1:  04A82 (00125) (00160) (00396) (00404)
 D20SINT2:  04A90 (00126) (00161) (00410) (00418)
 D20SINT3:  04A9E (00127) (00162) (00424) (00428)
 D21SINT1:  04AA6 (00128) (00163) (00433) (00441)
 D21SINT2:  04AB4 (00129) (00164) (00447) (00455)
 D21SINT3:  04AC2 (00130) (00165) (00461) (00465)
 D22SINT1:  04ACA (00131) (00166) (00470) (00478)
 D22SINT2:  04AD8 (00132) (00167) (00484) (00492)
 D22SINT3:  04AE6 (00133) (00168) (00498) (00502)
 D23SINT1:  04AEE (00134) (00169) (00507) (00515)
 D23SINT2:  04AFC (00135) (00170) (00521) (00529)
 D23SINT3:  04B0A (00136) (00171) (00535) (00539)
 ERRSIOS:   00014 (00035) (01079)
 ERRSVAL:   04900 (00210) (00757) (01078)
 ERRORS:    01AFA (00036) (01080)
 EDTS:      007E8 (00038) (00077) (00083) (00088)
 HS:        00001 (00040) (00751) (00784) (00842) (00869) (00937) (01021) (01037) (01155) (01157)
                  (01159) (01379) (01384) (01453) (01455) (01521) (01522) (01580) (01587) (01596)
                  (01679) (01695) (01709) (01712) (01741) (01770) (01793) (01794) (01801) (01802)
                  (01803) (01804) (01806) (01807) (01808) (01809) (01811) (01812) (01977) (01992)
```

```
                    (01993) (01995) (02014) (02015) (02022) (02030) (02048) (02280) (02285) (02290)
                    (02303)
  IDEVS63:    0003F (00042) (00237) (00558) (00592) (01173)
  IDIVS:      01850 (00043) (02300)
  ILVLS1:     00001 (00044) (00237) (00558) (00592) (01173) (01235)
  ILVLS2:     00002 (00045) (01236)
  ILVLS3:     00003 (00046) (01237)
  IOSHMTBL:   04844 (00552) (00556) (00575) (00579) (00583) (00587) (00600)
  IOSHSR*1:   0484C (00545) (00565) (00612) (00781) (00782)
  IOSLBT:     0486C (00550) (00554) (00570) (00571) (00637) (00803) (00804) (00806) (00807) (00817)
                    (00818)
  IO2SHDR:    04DC8 (00934) (00950) (00955) (00961) (01505)
  IO2SHDR0:   04DDF (01537) (01614)
  IO2SHDR1:   04DEF (01560) (01615)
  IO2SHDR2:   04DFA (01571) (01616)
  IO2SHDR3:   04E00 (01578) (01617)
  IO2SHDR4:   04E08 (01587) (01618)
  IO2SHDR5:   04E0E (01595) (01619)
  IO2SHDR6:   04E18 (01605) (01620)
  IO2SHDRD:   04E22 (01532) (01614)
  IO2SHDRN:   04DD4 (01527) (01554)
  IOBDSLPT:   04E1E (01522) (01547) (01611)
  IOBDSOFF:   04E30 (00932) (01595) (01605) (01622)
  IOBDSPPT:   04E20 (01521) (01527) (01612)
  IOSSER1:    04CBC (00778) (01055)
  IOSSER10:   04CD7 (00929) (01073)
  IOSSER11:   04CDA (01075) (01553)
  IOSSER2:    04CBF (00792) (01057)
  IOSSER3:    04CC2 (00797) (01059)
  IOSSER4:    04CC5 (00823) (01061)
  IOSSER5:    04CC8 (00828) (01063)
  IOSSER6:    04CCB (00857) (01065)
  IOSSER7:    04CCE (00873) (01067)
  IOSSER8:    04CD1 (00900) (00902) (01069)
  IOSSER9:    04CD4 (00892) (00906) (01071)
  IOSSERR:    04CDD (01056) (01058) (01060) (01062) (01064) (01066) (01068) (01070) (01072) (01074)
                    (01076) (01078)
  IOSSFLG:    049D1 (00213) (00223) (00234) (00590) (01163)
  IOSSINL:    049E1 (00181) (00220)
  IOSSINT:    04B12 (00249) (00253) (00263) (00267) (00286) (00290) (00300) (00304) (00323) (00327)
                    (00337) (00341) (00360) (00364) (00374) (00378) (00397) (00401) (00411) (00415)
                    (00434) (00438) (00448) (00452) (00471) (00475) (00485) (00489) (00508) (00512)
                    (00522) (00526) (00545) (01169) (01171)
  IOSSINTC:   04B24 (00277) (00314) (00351) (00388) (00425) (00462) (00499) (00536) (00564)
```

```
ISVTS:      03502 (00047) (00931) (01378) (01443) (02256)
ISVTSSIZ:   00080 (00048) (00928)
LDSSCRL2:   04C7C (00917) (00937) (00971)
LDSSCRL3:   04CA0 (01022) (01030)
LDSSCRL4:   04CAC (01038) (01042)
LDSS:       04B9C (00079) (00748)
LDSS0:      04C4A (00891) (00915)
LDSS01:     04C98 (00877) (01020)
LDSS02:     04CA6 (00879) (01036)
LDSSA:      04C13 (00847) (00852) (00861)
LDSSADAM:   04BA2 (00753) (01777) (02056)
LDSSB:      04C27 (00864) (00875)
LDSSC:      04BBF (00775) (00780)
LDSSNULL:   04C59 (00927) (00932)
MSS:        00000 (00052) (01235) (01236) (01237) (01394) (01461) (01612) (01622) (01835) (01839)
                  (01840) (01844) (01845) (01851) (01852) (01856) (01857) (01889) (01893) (01894)
                  (01897) (01898) (01899) (01901) (01902) (01908) (01909) (01912) (01913) (01914)
                  (01916) (01917) (02079) (02080) (02119) (02128) (02129) (02168) (02378) (02379)
                  (02382) (02383) (02401) (02402)
MPRAAS:     04D66 (00081) (01292)
MPRAAS0:    04DBF (01318) (01324)
MPRAAS1:    04D99 (01326) (01332)
MSKSLBYT:   0FF00 (00050) (00947) (00958) (01295) (01298) (01666) (01724) (01996) (02275)
MSKSRBYT:   000FF (00051) (00748) (00766) (00785) (00813) (00839) (00862) (00867) (00940) (00953)
                  (01293) (01296) (01377) (01442) (01528) (01672) (01685) (01723) (01745) (01749)
                  (01763) (01978) (02025) (02029) (02043) (02254)
OUTTRANS:   00014 (02404) (02405)
RBARAS:     0515B (02262) (02263) (02265) (02267) (02268) (02269) (02290) (02294) (02419)
RBAPSSA:    05146 (02363) (02410)
RBAPSS1:    0514F (02359) (02417)
RBAPSSZ:    0003C (02362) (02421)
RBENDIN:    0000F (02389) (02394)
RBOUT:      00010 (02370) (02398)
RNSS:       04CF4 (00080) (01123) (01778) (02057)
RNSS01:     04D40 (01132) (01201)
RNSS02:     04D4A (01135) (01213)
RNSSFINT:   04D60 (01155) (01156) (01157) (01158) (01159) (01160) (01235)
RNSSINT:    04D06 (01125) (01155) (01302) (01306)
RNSSTOS2:   04D30 (01140) (01150) (01190) (01312) (01313) (01322) (01330) (01384) (01453) (01761)
                  (01765) (02040) (02045)
RNSSTOS3:   04D44 (01201) (01207)
RNSSTOS4:   04D50 (01215) (01221)
RSRS:       04D9A (00085) (01377)
RSRSBMOV:   04DB0 (01379) (01390) (01394)
```

```
PAGE   81:     SNAP-11 IOS PACKAGE  ---- APR 18, 1980 ----  PROGRAM # 840001.01A
               CROSS REFERENCE

  SETSSMB:    01C12 (00054) (00220)
  SNAPSINL:   00BEB (00055) (00179) (00225)
  START:      04900 (00067) (00185)
  TEMS0:      007B4 (00057) (00760) (00934) (02299) (02302)
  TEMS1:      007B5 (00058)
  TEMS2:      007B6 (00059) (00844) (00872)
  TOFSCUR:    0515A (00072) (02429)
  TOFSPTR:    00288 (00060) (00070)
  WS:         00002 (00062) (00552) (00575) (00579) (00583) (00587) (01004) (01158) (01160) (01179)
                    (01665) (01671) (01695) (01704) (01722) (01728) (01746) (01772) (01793) (01794)
                    (01796) (01797) (01798) (01799) (01801) (01802) (01803) (01804) (01806) (01807)
                    (01808) (01809) (01811) (01812) (01826) (01883) (02026) (02050) (02065)
  WAITS0:     049FD (00237)
  WAITSIOS:   049EA (00234) (00239) (00755) (01124) (01301) (01305)
  WSRS:       04DB2 (00086) (01439)
  WSRS1:      04DBC (00911) (01453)
  WSRSHMOV:   04DC6 (01455) (01457) (01461)

LINES WITH ERRORS:    0  (MAP VERSION 800101.10)   E=    0
```

T A B L E   O F   C O N T E N T S

```
;FAST FOURIER TRANSFORM TRANSFORM ALGORITHSM - FF2R, FF2RB     MAY 7, 1980
*
*    MODIFIED FOR GTE SYLVANIA      BY S. TERRACE
*
;MODIFICATIONS MADE TO CORRECT WBUFFER PROBLEMS....31 JANUARY 1979
;
; TWO REAL TRANSFORMS, Y=2.0*SA*FFT(X+JZ); X, Z REAL
; PERFORMS TWO REAL TRANSFORMS OF SIZE N ON
; THE U BUFFER.  ONE REAL FUNCTION, X, IS STORED IN
; THE REAL PART OF U AND THE OTHER, Z, IS STORED IN
; THE IMAGINARY PART OF U.  RESULTS ARE LEFT IN THE
; Y BUFFER WITH THE TRANSFORM OF X IN THE FIRST HALF
; AND THE TRANSFORM OF Z IN THE SECOND HALF.  SINCE
; THE RESULT SAMPLES 0 AND N/2 ARE KNOWN TO HAVE
; ONLY REAL PARTS, THEIR RESULTS ARE STORED IN BIN
; ZERO WITH RX(0)=RX(0) AND IX(0)=RX(N/2) AND
; RZ(0)=RZ(0) AND IZ(0)=RZ(N/2).
; THE ALGORITHM PRODUCES TWICE THE CORRECT VALUE FOR THE SPECTRAL
; OUTPUTS BEFORE THEY ARE WEIGHTED BY SCALAR A.
;
; THIS ROUTINE CANNOT BE DONE IN PLACE.  I.E.  THE
; Y CANNOT BE THE SAME AS W AND U CANNOT BE THE SAME
; AS W.  BUT Y CAN BE THE SAME AS U.
;
;
; THE BUFFER DESCRIPTIONS ARE:
;   Y BUFFER (10-39)  COMPACT, COMPLEX 32-BIT FLOATING POINT
;   U BUFFER (10-39)  COMPLEX 32-BIT FLOATING POINT
;   V BUFFER (10-39)  REAL 32-BIT FLOATING POINT
;   W BUFFER (10-39)  COMPACT, COMPLEX 32-BIT FLOATING POINT
;
; THE COSINE TABLE ENTRIES ARE:
;   CT(K)=COS(2*PI*K/CSIZE)
; WHERE CSIZE IS A MULTIPLE OF N
;
;
```

```
                    (00037) 'SYSTEM DEPENDENT VARIABLES (SNAP-300 REL. 3.5)
                    (00038) ;
                    (00039) ;  UPDATED TO RELEASE 3.5 BY KEN WILMER,  4-OCTOBER-1979
                    (00040) ;     CORRECTION MADE IN BINDING OF BU FOR CASE WHEN USEP*USIZE > OR = 2**14
                    (00041) ;
                    (00042) ;
                    (00043) ; ALL THESE VARIABLES MUST BE REDEFINED FOR USE
                    (00044) ; WITH A NEW RELEASE OF SNAP EXEC
                    (00045) ;
          000008EB (00046)          AFDTS = $8EB                      ARRAY FUNCTION DISPATCH
          000008EB (00047)          AFDTSORG = AFDTS                  DISPATCH TABLE ORIGIN
          000002BB (00048)          TOESPTR = $2BB                    POINTER TO TOP OF EXEC
          0000FF00 (00049)          MSKSLBYT = $FF00                  MASK LEFT BYTE
          000000FF (00050)          MSKSRBYT = $00FF                  MASK RIGHT BYTE
          00000001 (00051)          HS = 1                            1 HALFWORD = 1 HALFWORD
          00000002 (00052)          WS = 2*HS                         1 FULLWORD = 2 HALFWORDS
          00000582 (00053)          BCTSBA = $582                     BASE ADDRESS TABLE
          00000604 (00054)          BCTSAD = $604                     ARRAY DEFINITION TABLE
          00000686 (00055)          BCTSAT = $686                     BUFFER ATTRIBUTE TABLE
          00000000 (00056)          KSS = 0                           DUMMY ARGUMENT
          00000020 (00057)          FLGSSET = $20                     SET FLAG BIT
          00000004 (00058)          FLGSG0 = $4                       FLAG G0
          00000005 (00059)          FLGSG1 = $5                       FLAG G1
          00000382 (00060)          SVTS = $382                       SCALAR VALUE TABLE
          0000024D (00061)          APSBSL = $24D                     ADDR OF ADDR OF BINDING SUPPORT LIST
          000007BA (00062)          ZERO = $7BA                       ADDR OF ZERO
          0001FFCF (00063)          SYSSFLGS = $1FFCF
          00000F63 (00064)          APSBNDRO = $0F63
                    (00065) ;
          000002BB (00066) *L = TOESPTR
                    (00067) *
    002BB 001049C4 (00068)           ADDR   TOESCUR(,1)              UPDATE TOP OF EXEC POINTER
                    (00069) *
                    (00070) *
          00000003 (00071)           *M = 3
                    (00072) *
          00004690 (00073)           START = $4690                     DEFINE START LOCATION FOR MODULE
```

```
                  (00074) 'DISPATCH TABLE ENTRIES FOR FF2R, FF2RB
                  (00075) ;
                  (00076) ; FF2R HAS FULL BINDING
                  (00077) ; FF2RB ONLY BINDS YBASE AND UBASE AND WBASE
                  (00078) ;
         000000D6 (00079)          FF2R$=214
         000000D7 (00080)          FF2RB$=215
                  (00081) ;
         00000AFC (00082)          #L=AFDTS+3*WS*(FF2R$-128)
   00AFC 001F47BC (00083)          ADDR CSM2$(R7, 1)
   00AFE 001F48C4 (00084)          ADDR CSM$(R7, 1)
   00AF0 00104690 (00085)          ADDR FFTSSET(, 1, 0)
                  (00086) ;
         00000AF2 (00087)          #L=AFDTS+3*WS*(FF2RB$-128)
   00AF2 001F47BC (00088)          ADDR CSM2$(R7, 1)
   00AF4 001F48C4 (00089)          ADDR CSM$(R7, 1)
   00AF6 00104754 (00090)          ADDR SBM$FFT(, 1, 0)
                  (00091) ;
                  (00092) ;
         00004690 (00093) #L = START            SET PC TO START LOCATION OF MODULE
```

```
                   (00094) 'SPECIAL BINDING MODULE FOR FFT SETUP
                   (00095) ;
                   (00096) ; DOES ALL BINDING NOT NEEDED EVERY TIME
                   (00097) ; AN FFT IS DONE.
                   (00098) ;
                   (00099) ; ENTER WITH R1 POINTING TO FCB NUMBER
                   (00100) ;            R2 POINTING TO DISPATCH TABLE
                   (00101) ;
                   (00102) ;
04690 F0420002 (00103) FFTSSET MOVMR R4, 2*HS(R1)          GET U BUFFER ID
04692 9A40FF00 (00104)         ANDIR R4, MSKSLBYT          MASK OUT RIGHT HALF
04694 3C47     (00105)         LRS R4, 7                   SET FOR FULL WORD INDEX
04695 0800     (00106)         EVEN                        SKIP TO EVEN BOUNDARY
04696 F0580687 (00107)         MOVMR R5, BCTSAT+HS(R4)     R5= U BUFFER ATTRIBUTES
04698 F05047B8 (00108)         MOVRM R5, PWR2S             STORE FOR LATER REFERENCE
               (00109) ;
               (00110) ;BIND ALL USIZE-1'S AND USIZE-2
               (00111) ;
0469A C0580604 (00112)         MOVMRL R5, BCTSAD(R4)       R5<=USEP, R6<=USIZE-1
0469C F0604935 (00113)         MOVRM R6, CSMS+WS*CR4A03+HS   STORE ALL USIZE-1'S
0469E F06048C9 (00114)         MOVRM R6, CSMS+WS*CSML0S+HS   ...
046A0 F060497D (00115)         MOVRM R6, CSMS+WS*CR4ARS+HS   ...
046A2 F0604995 (00116)         MOVRM R6, CSMS+WS*CR4A9S+HS   STORE FFTSIZE-1
046A4 F060494D (00117)         MOVRM R6, CSMS+WS*CR4A13S+HS  ...
046A6 2761     (00118)         DECR R6, 1
046A7 0800     (00119)         EVEN
046A8 F0604957 (00120)         MOVRM R6, CSMS+WS*FOS03+HS    BIND USIZE-2
               (00121)
046AA 2662     (00122)         INCR R6, 2                  R6<=N(USIZE)
046AB 403A     (00123)         MOVRR R3, R5                R3<=USEP
046AC 485C     (00124)         MULRR R5, R6                R5<=2(USEP)(USIZE)
046AD 3C52     (00125)         LRS R5, 2                   R5<=0.5(USEP)(USIZE)
               (00126) ;
               (00127) ;BIND ALL BU'S
               (00128) ;
046AE 904047B1 (00129)         MOVIR   R4,BUSTBL-1         ; POINT TO TABLE OF BU BINDING LOC'S
046B0 304F     (00130) BUSLP   POPXI   R4,R7               ; GET NEXT BINDING ADDR
046B1 0270     (00131)         TEST    R7                  ; CHECK FOR END OF TABLE
046B2 801046BC (00132)         JMP     BUSDN,EQ            ; ZERO MARKS END OF TABLE
046B4 D00F0000 (00133)         CMB     0,0(R7)             ; RESET MSB OF DELTA FIELD
046B6 F05F0001 (00134)         MOVRM   R5,1(R7)            ; STORE LOW 16 BITS
046B8 1810     (00135)         SKPL    NE                  ; IF NOT 0, DELTA IS OK
046B9 D20F0000 (00136)         SMB     0,0(R7)             ; PRODUCT WAS 2**18 BEFORE CORRECTION
046BB 210C     (00137)         BOP     BUSLP               ; LOOP FOR ALL BU BINDING
```

```
                  (00138) ;
046BC 0250        (00139) BUSDN   TEST    R5                          ; CHECK PRODUCT
046BD 1B10        (00140)         SKPL    NE                          ; IF NOT 0, DELTA IS OK
046BE FC510000    (00141)         MOVAP   R5,$10000                   ; SET UP CORRECT DELTA FOR 2**18
                  (00142) ;
046C0 3C51        (00143)         LRS R5, 1                          R5<=BU/2 (QU)
046C1 0800        (00144)         EVEN                               TO TO EVEN BOUNDARY
                  (00145) ;
                  (00146) ;BIND ALL QU'S
                  (00147) ;
046C2 F0504B07    (00148)         MOVRM R5, CSMS+WS*CSML3S+HS        STORE QU'S
046C4 4E56        (00149)         SUBRR R5, R3                       R5<=QU-USEP
046C5 0800        (00150)         EVEN
046C6 F0504BCF    (00151)         MOVRM R5, CSMS+WS*CSML1S+HS        STORE QU-USEP
046C8 405C        (00152)         MOVRR R5, R6                       R5<=N
046C9 0800        (00153)         EVEN
046CA 98500003    (00154)         MULIR R5, 3                        R5<=4*(3N/2)
                  (00155) ;
                  (00156) ;BIND D0-D7'S
                  (00157) ;
046CC F0504BF5    (00158)         MOVRM R5, CSMS+WS*SCRM0+HS         STORE ALL D0'S
046CE F0504BF9    (00159)         MOVRM R5, CSMS+WS*SCRM2S+HS        ...
046D0 F0504BFD    (00160)         MOVRM R5, CSMS+WS*SCRM4S+HS        ...
046D2 F0504BF1    (00161)         MOVRM R5, CSMS+WS*SCRM6S+HS        ...
046D4 3C51        (00162)         LRS R5, 1                          R5<=D0/2 (D1)
046D5 0800        (00163)         EVEN
046D6 F0504BF7    (00164)         MOVRM R5, CSMS+WS*SCRM1S+HS        STORE ALL D1'S
046D8 F0504BEF    (00165)         MOVRM R5, CSMS+WS*SCRM5S+HS        ...
046DA 3C51        (00166)         LRS R5, 1                          R5<=D1/2 (D2)
046DB 0800        (00167)         EVEN
046DC F0504BEB    (00168)         MOVRM R5, CSMS+WS*SCRM3S+HS        STORE D2
046DE 3C51        (00169)         LRS R5, 1                          R5<=D2/2 (D3)
046DF 0800        (00170)         EVEN
046E0 F0504BF5    (00171)         MOVRM R5, CSMS+WS*SCRM7S+HS        STORE D3
046E2 3C51        (00172)         LRS R5, 1                          R5<=D3/2 (D4)
046E3 0800        (00173)         EVEN
046E4 F0504BEB    (00174)         MOVRM R5, CSMS+WS*SCRM8S+HS        STORE D4
046E6 3C51        (00175)         LRS R5, 1                          R5<=D4/2 (D5)
046E7 0800        (00176)         EVEN
046E8 F0504901    (00177)         MOVRM R5, CSMS+WS*SCRM9S+HS        STORE D5
046EA 3C51        (00178)         LRS R5, 1                          R5<=D5/2 (D6)
046EB 0800        (00179)         EVEN
046EC F0504907    (00180)         MOVRM R5, CSMS+WS*SCRM10S+HS        STORE D6
046EE 3C51        (00181)         LRS R5, 1                          R5<=D6/2 (D7)
```

```
046EF 0800     (00182)        EVEN
046F0 F050490B (00183)        MOVRM R5, CSMS+WS*SCRM7+HS     STORE D7
046F2 405C     (00184)        MOVRR R5, R6                   R5<=N (USIZE)
               (00185) ;
               (00186) ;BIND 2N'S AND 4N'S
               (00187) ;
046F3 3A51     (00188)        LLS R5, 1                      R5<=2*N
               (00189)        EVEN
046F4 F05049A3 (00190)        MOVRM R5, CSMS+WS*FOS3+HS      BIND 2N'S
046F6 F0504955 (00191)        MOVRM R5, CSMS+WS*FOSO2+HS     ...
046F8 3A51     (00192)        LLS R5, 1                      R5<=4N
046F9 0800     (00193)        EVEN
046FA F05049A9 (00194)        MOVRM R5, CSMS+WS*FOS4+HS      BIND 4N
046FC 2654     (00195)        INCR R5, 4                     R5<=4*N+4
046FD 0800     (00196)        EVEN
               (00197) ;
               (00198) ;BIND ALL 4N+4 AND 4N+12'S
               (00199) ;
046FE F0504937 (00200)        MOVRM R5, CSMS+WS*CR4A2S+HS    STORE 4N+4'S
04700 F050497B (00201)        MOVRM R5, CSMS+WS*CR4A3+HS     ...
04702 2658     (00202)        INCR R5, 8                     R5<=4*N+12
04703 0800     (00203)        EVEN
04704 F0504915 (00204)        MOVRM R5, CSMS+WS*CSMOL+HS     STORE 4N+12
               (00205) ;
               (00206) ;BIND SCALAR A
               (00207) ;
04706 F0420001 (00208)        MOVMR R4, HS(R1)               LOAD SCALAR A ID
04708 9A4000FF (00209)        ANDIR R4, MSKSRBYT             MASK IT OUT
0470A 9C4803B2 (00210)        ADDIR R4, SVTS(R4)             R4<=SVTS+WS*SAID
0470C F040499D (00211)        MOVRM R4, CSMS+WS*FOS1+HS      BIND SCALAR A
0470E F0420003 (00212)        MOVMR R4, 3*HS(R1)             LOAD C BUFFER ID
04710 9A40FF00 (00213)        ANDIR R4, MSKSLBYT
04712 3C47     (00214)        LRS R4, 7                      CREATE FULL WORD INDEX
04713 0800     (00215)        EVEN
               (00216) ;
               (00217) ;BIND CBASE
               (00218) ;
04714 C06805B2 (00219)        MOVMRL R6, BCTSBA(R4)          LOAD C BASE ADDRESS
04716 9050496A (00220)        MOVIR R5, CSMS+WS*CR4A4S       POINT TO LOAD CBASE INST.
04718 9A60000F (00221)        ANDIR R6, SF                   AND OUT LOW FOUR BITS
0471A 766AFFF0 (00222)        IORWR R6, R5, $FFF0            OR IN NEW FOUR BITS
0471C 846A0000 (00223)        MOVRML R6, 0(R5)               AND STORE BACK IN INST.
0471E C0580604 (00224)        MOVMRL R5, BCTSAD(R4)          R5<=CSEP, R6<=CSIZE-1
04720 2661     (00225)        INCR R6, 1                     R6<=CSIZE
```

```
04721 485C     (00226)          MOLRR R5, R6                 R5<=2*(CSEP)(CSIZE)
04722 3C53     (00227)          LRS R5, 3                    R5<=0.25*(CSEP)(CSIZE) (HPI)
04723 0800     (00228)          EVEN
               (00229) ;
               (00230) ;BIND ALL HPI'S
               (00231) ;
04724 F0504971 (00232)          MOVRM R5, CSMS+WS*CR4A5S+HS   STORE ALL HPI'S
04726 F0504975 (00233)          MOVRM R5, CSMS+WS*CR4A6S+HS   ...
04728 F0504979 (00234)          MOVRM R5, CSMS+WS*CR4A7S+HS   ...
               (00235) ;
               (00236) ;S1ANG=HPI/SSI. RIGHT NOW ONLY RADIX 2 AND RADIX 4
               (00237) ;HAVE BEEN IMPLEMENTED.  SO ALL THAT IS NECESSARY
               (00238) ;TO CALCULATE SSI IS TO KNOW IF USIZE IS AN EVEN OR
               (00239) ;ODD POWER OF TWO.  THIS IS DONE BY CHECKING THE
               (00240) ;POWER OF TWO ENTRY FOR THE U BUFFER IN THE BCTSAT
               (00241) ;TABLE.  IF BIT 0 IS ON, THEN IT'S AN ODD POWER.
               (00242) ;
0472A F030024D (00243)          MOVMR   R3, APSBSL             ; GET BINDING SUPPORT LIST POINTER
0472C D80647B8 (00244)          SMRC 0, PWR2S                SKIP IF EVEN POWER OF TWO
0472E 2006     (00245)          HOP SBMS1                    ... ODD POWER
0472F F80B4927 (00246)          MOVLM 4, CSMS+WS*CR4A01+HS    SET AND BIND SSI = 4
04731 F80A47B1 (00247)          MOVLM FLGSG1, GSFLGS+HS        RADIX 4 FIRST STEP
04733 3C52     (00248)          LRS R5, 2                    S1ANG=HPI/4
04734 2005     (00249)          HOP SBMS2
04735 F8044927 (00250) SBMS1    MOVLM 2, CSMS+WS*CR4A01+HS    SET AND BIND SSI = 2
04737 F84A47B1 (00251)          MOVLM FLGSG1+FLGSSET, GSFLGS+HS  RADIX 2 FIRST STEP
04739 3C51     (00252)          LRS R5, 1                    S1ANG=HPI/2
               (00253)          EVEN
               (00254) SBMS2
               (00255) ;
               (00256) ;BIND S1ANG TO S/ANG'S
               (00257) ;
0473A F05049B7 (00258)          MOVRM R5, CSMS+WS*ANGL1S+HS   STORE S1ANG
0473C 3C52     (00259)          LRS R5, 2
0473D 0800     (00260)          EVEN
0473E F05049B9 (00261)          MOVRM R5, CSMS+WS*ANGL2S+HS   STORE S2ANG
04740 3C52     (00262)          LRS R5, 2
04741 0800     (00263)          EVEN
04742 F05049BB (00264)          MOVRM R5, CSMS+WS*ANGL3S+HS   STORE S3ANG
04744 3C52     (00265)          LRS R5, 2
04745 0800     (00266)          EVEN
04746 F05049BD (00267)          MOVRM R5, CSMS+WS*ANGL4S+HS   STORE S4ANG
04748 3C52     (00268)          LRS R5, 2
04749 0800     (00269)          EVEN
```

```
0474A F05049BF (00270)        MOVRM R5, CSMS+WS*ANGL5S+HS   STORE S5ANG
0474C 3C52     (00271)        LRS R5, 2
0474D 0800     (00272)        EVEN
0474E F05049C1 (00273)        MOVRM R5, CSMS+WS*ANGL6S+HS   STORE S6ANG
04750 3C52     (00274)        LRS R5, 2
04751 0800     (00275)        EVEN
04752 F05049C3 (00276)        MOVRM R5, CSMS+WS*ANGL7S+HS   STORE S7ANG
```

```
                (00277) *          SPECIAL BINDING FOR WBASE, YBASE, AND UBASE
                (00278) ;
                (00279) ; THIS SECTION DOES THE FAST BINDING FOR
                (00280) ; CHANGES IN WBASE, YBASE, AND UBASE ONLY.
                (00281) ;
                (00282) SBMSEFT EVEN                        START ON EVEN BOUNDARY
04754 F0420004  (00283)         MOVMR R4, 4*HS(R1)          GET W BUFFER ID
04756 9A40FF00  (00284)         ANDIR R4, MSKSLBYT          MASK THE LEFT HALF
04758 3C47      (00285)         LRS R4, 7                   CREATE FULL WORD INDEX
04759 0800      (00286)         EVEN
                (00287) ;
                (00288) ;BIND ALL WBASE'S
                (00289) ;
0475A C068058?  (00290)         MOVMRL R6, BCTSBA(R4)       LOAD BASE ADDRESS IN R6, R7
0475C 90504910  (00291)         MOVIR R5, CSMS+WS*CSM01S    POINT TO LOAD INSTRUCTION
0475E 9A60000F  (00292)         ANDIR R6, SF                ONLY LOW FOUR BITS
04760 766AFFF0  (00293)         IORWK R6, R5, SFFF0         'OR'ED INTO LOAD INST.
04762 846A0000  (00294)         MOVRML R6, 0(R5)            STORE ALL WBASE'S
04764 9A60000F  (00295)         ANDIR R6, SF                MASK OUT OLD LOAD INST.
04766 9050499F  (00296)         MOVIR R5, CSMS+WS*F0S2      POINT TO NEXT LOAD
04768 766AFFF0  (00297)         IORWK R6, R5, SFFF0         OR INTO INSTRUCTION
0476A 846A0000  (00298)         MOVRML R6, 0(R5)            BIND WBASE
0476C 9A60000F  (00299)         ANDIR R6, SF                MASK OUT OLD LOAD INSTR
0476E 9050492C  (00300)         MOVIR R5, CSMS+WS*CR4A1S    POINT TO NEXT LOAD
04770 504C0001  (00301)         MOVKR R4, R6, S1              PLACE MSB INTO SEP REG
04772 9A61FFFF  (00302)         ANDIR R6, S1FFFF            CLR MSB IOF WBASE ADDR
04774 2774      (00303)         DECR R7, 4                  WBASE-4
04775 0640      (00304)         DAC R4                      CARRY INTO MSB
04776 56680001  (00305)         IORKR R6, R4, S1            PLACE PROPER MSB INTO RESULT
04778 766BFFF0  (00306)         IORWK R6, R5, SFFFF0         'OR' INTO LOAD INSTR
0477A 846A0000  (00307)         MOVRML R6, 0(R5)            LOAD WBASE-4 INTO APS
0477C 9A60000F  (00308)         ANDIR R6, SF                MASK OUT OLD LOAD INSTR
                (00309) ;
                (00310) ;BIND UBASE
                (00311) ;
0477E F0420002  (00312)         MOVMR R4, 2*HS(R1)          LOAD U BUFFER ID
04780 9A40FF00  (00313)         ANDIR R4, MSKSLBYT          MASK THE LEFT HALF
04782 3C47      (00314)         LRS R4, 7                   CREATE FULL WORD INDEX
04783 0800      (00315)         EVEN
04784 C068058?  (00316)         MOVMRL R6, BCTSBA(R4)       LOAD UBASE ADDRESS IN R6, R7
04786 905048C6  (00317)         MOVIR R5, CSMS+WS*CSML6S    POINT TO LOAD UBASE INST.
04788 9A60000F  (00318)         ANDIR R6, SF                ONLY LOW ORDER FOUR BITS
0478A 766AFFF0  (00319)         IORWK R6, R5, SFFF0         'OR' INTO INST.
0478C 846A0000  (00320)         MOVRML R6, 0(R5)            STORE UBASE
```

```
                   (00321) ;
                   (00322) ;BIND YBASE
                   (00323) ;
0478E E0420001 (00324)          MOVMR R4, BS(R1)             LOAD Y BUFFER ID
04790 9A40FF00 (00325)          ANDIR R4, MSKSLBYT           MASK LEFT HALF
04792 3C47     (00326)          LRS R4, 7                    CREATE FULL WORD INDEX
04793 0800     (00327)          EVEN
04794 C0680582 (00328)          MOVMRL R6, BCTSBA(R4)        LOAD YBASE ADDRESS IN R6, R7
04796 90504950 (00329)          MOVIR R5, CSMS+WS*FOS01      POINT TO LOAD INST.
04798 9A60000F (00330)          ANDIR R6, $F                 MASK LOW FOUR BITS
0479A 766AFFF0 (00331)          IORWK R6, R5, $FFF0          OR INTO INST.
0479C 846A0000 (00332)          MOVRML R6, 0(R5)             STORE YBASE
```

```
                 (00333) *         SET PENDING SLOTS TO PROPER VALUES
                 (00334) ;
0479E C6340BFB (00335)           PUSHMIL R3, AFDTSORG(R2)      STORE APU MODULE BUS ORIGIN
                 (00336)
047A0 9071FFFF (00337)           MOVIR R7, -WS
047A2 C6B7FFFF (00338)           PUSHMIL R3, M-WS(R3)         STORE APU MODULE START AND SIZE
047A4 C6340BFA (00339)           PUSHMIL R3, AFDTSORG+WS(R2)   STORE APS MODULE BUS ORIGIN
                 (00340) **        PUSHMIL R3, AFDTSORG + 2*WS(R2) ; STORE CSPU SUPPORT ADDR
047A6 2632     (00341)           INCR R3, 2
047A7 2771     (00342)           DECR R7, 1
                 (00343)           EVEN
047A8 C4B7FFFC (00344)           PUSHMI R3, M-2*WS(R3)         STORE APS MODULE SIZE
                 (00345) **        PUSHMI  R3, ZERO               ; NO SPECIAL SUPPORT
                 (00346) **        MOVRR R5, R1
                 (00347) **        INCR R5, HS                   POINT TO SCALAR A ID
                 (00348) **        MOVWR R4, R5, MSKSRBYT
                 (00349) **        PUSHXIL R3, R4                STORE SCALAR A IDENTIFIER
047AA 2634     (00350)           INCR    R3, 4
047AB C43047B1 (00351)           PUSHMI  R3, GSFLGS+HS         STORE FLAG G1
                 (00352) *
                 (00353) **        INCR R3, 2
                 (00354) **        PUSHMI R3, ZERO               SET FOR NO PBF
                 (00355) *
047AD 80000F63 (00356)           JMP    APSRNDRO               AND EXIT
                 (00357) ;
047AF 0800     (00358)           EVEN
047B0 0004     (00359) GSFLGS   DATA     $4, $25                 ; GO & G1 FLAG CONTROL STORAGE
047B1 0025
                 (00360) ;
047B2 48CC     (00361) HUSTBL   DATA     CSMS + WS*CSML          ; ADDRESS TABLE FOR HU BINDING
047B3 48D0     (00362)          DATA     CSMS + WS*CSME
047B4 48D4     (00363)          DATA     CSMS + WS*CSML2S
047B5 48D8     (00364)          DATA     CSMS + WS*CSML4S
047B6 48DF     (00365)          DATA     CSMS + WS*CSML5S
047B7 0000     (00366)          DATA     0
                 (00367) ;
047B8 0000     (00368) PWR2S    DATA     0                       ; STORAGE FOR POWER OF TWO
```

```
                (00369) 'FFT - AP PROGRAMS
       00000003 (00370)         #M=3                        THIS IS A MAP-300 PROGRAM
                (00371)         OPADD FWI, (3 .LS. 10)+(12 .LS. 5)+X'16'
                (00372) !   APU PROGRAMS
                (00373) ;
                (00374) ;BINDING SECTION FOR EXEC
                (00375) ;
  047B9 0800    (00376)         EVEN
  047BA 0000    (00377)         DATA CSM2SSA
  047BB 0080    (00378)         DATA CR4SSZ
                (00379)         EVEN
                (00380) ;
                (00381) ;
```

```
                    (00382) '         SCRAMBLE AND FIRST RADIX-2 STAGE, FORWARD
                    (00383) ; 03/08/78
                    (00384) ; SCRAMBLE AND FIRST STAGE OF FFT,GI SET
                    (00385) ;
                    (00386) ; FUNCTION
                    (00387) ; LETTING EIGHT SUCESSIVE INPUTS BEING DESIGNATED BY
                    (00388) ; RU0,RU1,IU1,IU0,IU2,IU3,RU3,RU2
                    (00389) ; THE EIGHT OUTPUTS ARE PROVIDED
                    (00390) ; RU0+RU1,RU0-RU1,IU0+IU1,IU0-IU1
                    (00391) ;  IU2+IU3,IU2-IU3,RU2+RU3,RU2-RU3
                    (00392) ; FOR RELATIONSHIP OF INPUTS TO U(K), AND OUTPUTS TO Y(K),
                    (00393) ; SEE SUPPORTING APS PROGRAM, CSM
                    (00394) ;
                    (00395) CSM2S    BEGIN APU(CSM2)
           00000000 (00496)          #A=00
                    (00397) ;
A00 047BC 90050000D (00398) CSM2SSA JUMPC(CSM4F, GI)
A01 047BE 10000003 (00399)          JUMP(CSM2F)
                    (00400) ;
A02 047C0 089C089C (00401) CSM2L    MOV(R,OO)                                       OO=IYO\IY1
                    (00402) ;
A03 047C2 08F008F0 (00403) CSM2F    MOV(IOA,A0)                                     A0=RU0
A04 047C4 08F108F1 (00404)          MOV(IOA,A1)                                     A1=RU1
A05 047C6 41004900 (00405)          ADD(A0,A1)\SUB(A0,A1)
                    (00406) ;
A06 047C8 08F308F3 (00407)          MOV(IOA,A3)                                     A3=IU1
A07 047CA 08F208F2 (00408)          MOV(IOA,A2)                                     A2=IU0
A08 047CC 435C485C (00409)          MOV(OO),ADD(A2,A3)\MOV(OO),SUB(A2,A3)          OO=RYO\RY1
                    (00410) ;
A09 047CE 90160002 (00411)          JUMPC(CSM2L,FWI)
                    (00412) ;
A0A 047D0 089C089C (00413)          MOV(R,OO)                                       OO=IYO\IY1
                    (00414) ;
A0B 047D2 00000000 (00415)          NOP
A0C 047D4 1000002B (00416)          JUMP(CR4FS)
                    (00417) ;
```

```
                      (00418) *          SCRAMBLE AND FIRST RADIX-4 STAGE, FORWARD
                      (00419) ; SCRAMBLE AND FORWARD RADIX 4 STAGE OF FFT,G1 CLEAR
                      (00420) ;
                      (00421) ; FUNCTION
                      (00422) ; THE EIGHT SUCESSIVE INPUTS ARE PROVIDED TO LOOP
                      (00423) ; R00,R01,I01,I00,I02,I03,R03,R02
                      (00424) ; THE INTERMEDIATE RADIX TWO RESULTS ARE CALCULATED
                      (00425) ; S0=U0+U1,S1=U0-U1,S2=U2+U3,S3=U2-U3
                      (00426) ; THE OUTPUTS THEN BEING GIVEN BY
                      (00427) ; Y0=S0+S2,Y1=S1-JS3,Y2=S0-S2,Y3=S1+JS3
                      (00428) ; THE ACTUAL OUTPUT SEQUENCE BEING
                      (00429) ; RY0,RY1,IY0,IY1,IY2,IY3,RY2,RY3
                      (00430) ;
                      (00431) ; APU INITIALIZATION
                      (00432) ;
                      (00433) ;
A0D 047D6 0BF00BF0 (00434) CSM4F    MOV(IOA,A0)                                   A0=R00\R00
A0E 047D8 0BF10BF1 (00435)          MOV(IOA,A1)                                   A1=R01\R01
A0F 047DA 0BF30BF3 (00436)          MOV(IOA,A3)                                   A3=I01\I01
A10 047DC 41004900 (00437)          ADD(A0,A1)\SUB(A0,A1)
A11 047DE 0BF20BF2 (00438)          MOV(IOA,A2)                                   A2=I00\I00
A12 047E0 10000018 (00439)          JUMP(CSM4FS)
                      (00440) ;
                      (00441) ;
                      (00442) ; CSM4F, APU INNER LOOP
                      (00443)
A13 047E2 46BC4EBC (00444) #1       MOV(OQ),ADD(A5,A6)\MOV(OQ),SUB(A5,A7)         OQ=RY0\RY1
A14 047E4 0BF00BF0 (00445)          MOV(IOA,A0)                                   A0=R00
                      (00446)
A15 047E6 4EBC47BC (00447)          MOV(OQ),SUB(A5,A6)\MOV(OQ),ADD(A5,A7)         OQ=IY0\IY1
A16 047E8 0BF10BF1 (00448)          MOV(IOA,A1)                                   A1=R01
                      (00449)
A17 047EA 4F9C4F9C (00450)          MOV(OQ),SUB(A4,A7)\MOV(OQ),SUB(A4,A6)         OQ=IY2\IY3
A18 047EC 0BF30BF3 (00451)          MOV(IOA,A3)                                   A3=I01
                      (00452)
A19 047EE 411C491C (00453)          MOV(OQ),ADD(A0,A1)\MOV(OQ),SUB(A0,A1)         OQ=RY2\RY3
A1A 047F0 0BF20BF2 (00454)          MOV(IOA,A2)                                   A2=I00
                      (00455)
A1B 047F2 43544B54 (00456) CSM4FS   MOV(A4),ADD(A2,A3)\MOV(A4),SUB(A2,A3)         A4=RS0\RS1
A1C 047F4 0BF00BF0 (00457)          MOV(IOA,A0)                                   A0=I02
A1D 047F6 0BF10BF1 (00458)          MOV(IOA,A1)                                   A1=I03
                      (00459)
A1E 047F8 41154915 (00460)          MOV(A5),ADD(A0,A1)\MOV(A5),SUB(A0,A1)         A5=IS0\IS1
A1F 047FA 0BF30BF3 (00461)          MOV(IOA,A3)                                   A3=R03
```

```
A20 047EC 08F208F2 (00462)        MOV(IOA,A2)                                   A2=RO2
                   (00463)
A21 047EE 43564B56 (00464)        MOV(A6),ADD(A2,A3)\MOV(A6),SUB(A2,A3)         A6=IS2\IS3
                   (00465)
A22 04800 47974697 (00466)        MOV(A7),ADD(A4,A7)\MOV(A7),ADD(A4,A6)         A7=RS2\RS3
                   (00467)
A23 04802 90160013 (00468)        JUMPC(#1,FWI)
                   (00469)
A24 04804 46BC4EBC (00470)        MOV(OO),ADD(A5,A6)\MOV(OO),SUB(A5,A7)         OO=RYO\RY1
A25 04806 4EBC47BC (00471)        MOV(OO),SUB(A5,A6)\MOV(OO),ADD(A5,A7)         OO=IYO\IY1
A26 04808 4F9C4F9C (00472)        MOV(OO),SUB(A4,A7)\MOV(OO),SUB(A4,A6)         OO=IY2\IY3
A27 0480A 089C089C (00473)        MOV(R,OO)                                     OO=RY2\RY3
```

```
 '          SUCCESSIVE RADIX-4 STAGES, FORWARD
 ;
 ;USES APS PROGRAM CR4A
 ;
 ;MATHEMATICS
 ;
 ; W=A+BF+CF2+DF3=P+Q
 ; X=A-JBF-CF2+JDF3=R-S
 ; Y=A-BF+CF2-DF3=P-Q
 ; Z=A+JBF-CF2-JDF3=R+S

 ; P=A+CF2,R=A-CF2
 ; Q=F(B+DF2),S=JF(B-DF2)

 ; PR=AR+CRCOS2X+CISIN2X
 ; PI=AI+CICOS2X-CRSIN2X
 ; RR=AR-CRCOS2X-CISIN2X
 ; RI=AI-CICOS2X+CRSIN2X
 ; QR=BRCOSX+BISINX+DRCOS3X+DISIN3X
 ; QI=BICOSX-BRSINX+DICOS3X-DRSIN3X
 ; SR=BRSINX-BICOSX+DICOS3X-DRSIN3X
 ; SI=BRCOSX+BISINX-DRCOS3X-DISIN3X
 ;
 ; SINX STORED IN M1\M5
 ; SIN2X STORED IN M6\M2
 ; SIN3X STORED IN M7\M3
 ; COSX STORED IN M5\M1
 ; COS2X STORED IN M2\M6
 ; COS3X STORED IN M3\M7

          EJECT
```

```
                       (00505) ;CR4F-PIPELINE STARTUP
                       (00506)
                       (00507)
A28 0480C 20282026 (00508) CR4FS    CLEAR(AF3)
A29 0480E 202A202A (00509)          CLEAR(AF2)
A2A 04810 20292029 (00510) CR4FSA   CLEAR(AF1)
A2B 04812 20282028 (00511)          CLEAR(AF0)
                       (00512)
A2C 04814 16801680 (00513)          F(1)
A2D 04816 08090BF0 (00514)          MOV(ZERO,M1)\MOV(IQA,A0)          A0=DI\DR
A2E 04818 08F0080D (00515)          MOV(IQA,A0)\MOV(ZERO,M5)          SINX=0=M1\M5
                       (00516)
                       (00517)
A2F 0481A 08D00889 (00518)          MOV(R,M5)\MOV(R,M1)               COSX=1=M5\M1
A30 0481C 088A088E (00519)          MOV(R,M2)\MOV(R,M6)               COS2X=1=M2\M6
A31 0481E 088B088F (00520)          MOV(R,M3)\MOV(R,M7)               COS3X=1=M3\M7
                       (00521)
A32 04820 08F1080A (00522)          MOV(IQA,A1)\MOV(ZERO,M2)          A1=BI\BR
A33 04822 080F08F1 (00523)          MOV(ZERO,M6)\MOV(IQA,A1)          SIN2X=0=M6\M2
A34 04824 080F080B (00524)          MOV(ZERO,M7)\MOV(ZERO,M3)         SIN3X=0=M7\M3
                       (00525)
A35 04826 08F808F8 (00526)          MOV(IQA,M0)                       M0=CR
A36 04828 84408440 (00527)          MUL(M0,M6)                        P=CRSIN2X\CRCOS2X
A37 0482A 0A200220 (00528)          NEG(A1)\R(A1)                     R=-BI\BR
                       (00529)
A38 0482C 10000054 (00530)          JUMP(CR4FE)
                       (00531)
                       (00532) ;CR4F:COSINE ENTRY
                       (00533) ;
A39 0482E 08FD08F9 (00534) CR4FC    MOV(IQA,M5)\MOV(IQA,M1)           COSX TO M5\M1
A3A 04830 08F908FD (00535)          MOV(IQA,M1)\MOV(IQA,M5)           SINX TO M1\M5
A3B 04832 08A008A0 (00536)          MOV(P,NULL)
A3C 04834 08FE08FA (00537)          MOV(IQA,M6)\MOV(IQA,M2)           SIN2X TO M6\M2
A3D 04836 08FA08FE (00538)          MOV(IQA,M2)\MOV(IQA,M6)           COS2X TO M2\M6
A3E 04838 08FB08FF (00539)          MOV(IQA,M3)\MOV(IQA,M7)           COS3X TO M3\M7
A3F 0483A 08FF08FB (00540)          MOV(IQA,M7)\MOV(IQA,M3)           SIN3X TO M7\M3
                       (00541) ;
                       (00542) ;CR4F-BUTTERFLY
                       (00543)
A40 0483C 08F808F8 (00544) CR4FB    MOV(IQA,M0)                       DR TO M0
A41 0483E 8A748A74 (00545)          MOV(A4),MULN(M0,M7)\MOV(A4),MUL(M0,M7)   A4=CICOS2X\CISIN2X
A42 04840 4A954455 (00546)          MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)    A5=DI\DR
A43 04842 08FC08FC (00547)          MOV(IQA,M4)                       DI TO M4
A44 04844 4A724A72 (00548)          MOV(A2),SUB(A3,A2)                A2=CICOS2X-CRSIN2X
```

```
                     (00549) ;                                                        \CISIN2X+CRCOS2X
                     (00550)
                     (00551)
A45 04846 8590A590  (00552)          MOV(A0),MUL(M3,M4)\MOV(A0),MULN(M3,M4)     A0=DRSIN3X\DRCOS3X
A46 04848 427B427B  (00553)          MOV(FX0),ADD(A3,A2)                        FX=RI\RR
A47 0484A 0B560B56  (00554)          MOV(FXI,A6)                                A6=RR\RI
A48 0484C 0BFB0BFB  (00555)          MOV(IOA,M0)                                RI TO M0
A49 0484E 47D447D4  (00556)          MOV(A4),ADD(A6,A7)                         A4=PI\PR
                     (00557)
                     (00558)
A4A 04850 8431A431  (00559)          MOV(A1),MUL(M0,M5)\MOV(A1),MULN(M0,M5)     A1=DICOS3X\DISIN3X
A4B 04852 483C103C  (00560)          MOV(OO),SUB(A1,A0)\MOV(OO),ADD(A1,A0)      OO=ZR\ZI
A4C 04854 0BFC0BFC  (00561)          MOV(IOA,M4)                                RR TO M4
A4D 04856 40904090  (00562)          MOV(A0),SUB(A4,A5)                         A0=DICOS3X-DRSIN3X
                     (00563) ;                                                        \DISIN3X+DRCOS3X
                     (00564)
                     (00565)
                     (00566)
A4E 04858 A491B491  (00567)          MOV(A1),MULN(M1,M4)\MOV(A1),MUL(M1,M4)     A1=BICOSX\BISINX
A4F 0485A 4FDC4FDC  (00568)          MOV(OO),SUB(A6,A7)                         OO=YI\YR
A50 0485C 0BFB0BFB  (00569)          MOV(IOA,M0)                                CR TO M0
A51 0485E 459C459C  (00570)          MOV(OO),ADD(A4,A5)                         OO=XR\XI
                     (00571)
                     (00572)
A52 04860 A4528452  (00573)          MOV(A2),MULN(M0,M6)\MOV(A2),MUL(M0,M6)     A2=BRSINX\BRCOSX
A53 04862 495C423C  (00574)          MOV(OO),SUB(A2,A1)\MOV(OO),ADD(A1,A2)      OO=WR\WI
A54 04864 0BFC0BFC  (00575) CR4FF    MOV(IOA,M4)                                CI TO M4
A55 04866 41114831  (00576)          MOV(A1),ADD(A0,A1)\MOV(A1),SUB(A1,A0)      A1=BRSINX-BICOSX
                     (00577) ;                                                        \BRCOSX+BISINX
                     (00578)
                     (00579)
A56 04868 8512A512  (00580)          MOV(A2),MUL(M2,M4)\MOV(A2),MULN(M2,M4)     A2=CRSIN2X\CRCOS2X
A57 0486A 0BF34037  (00581)          MOV(IOA,A3)\MOV(A7),ADD(A1,A0)             A3=AI \ A7=SI
A58 0486C 49170BF3  (00582)          MOV(A7),SUB(A0,A1)\MOV(IOA,A3)             A7=SR \ A3=AR
A59 0486E 90280040  (00583)          JUMPC(CR4FB,AF0),CLEAR                     AF1;STAGE DONE
A5A 04870 90290039  (00584)          JUMPC(CR4FC,AF1),CLEAR                     AF0;COSINES USED
                     (00585)
                     (00586)
                     (00587) ; CR4F-PIPELINE CLEANUP
A5B 04872 0BB40BB4  (00588)          MOV(P,A4)                                  A4=CICOS2X\CISIN2X
                     (00589)
                     (00590)
A5C 04874 4A954455  (00591)          MOV(A5),SUB(A4,A2)\MOV(A5),ADD(A2,A4)      A5=QI\QR
A5D 04876 4A724A72  (00592)          MOV(A2),SUB(A3,A2)                         A2=CICOS2X-CRSIN2X
```

```
                       (00593) ;                                        \CISIN2X+CRCOS2X
                       (00594)
A5E 0487B 4278127B (00595)    MOV(FX0),ADD(A3,A2)              FX=RI\RR
A5F 0487A 08560856 (00596)    MOV(FX1,A6)                      A6=RR\RI
                       (00597)
A60 0487C 47044704 (00598)    MOV(A4),ADD(A6,A7)               A4=PI\PR
A61 0487E 4D9C0000 (00599)    MOV(OO),SUB(A4,A5)\NOP           OO=ZR
A62 04880 00004D9C (00600)    NOP\MOV(OO),SUB(A4,A5)           OO=ZI
A63 04882 4FDC0000 (00601)    MOV(OO),SUB(A6,A7)\NOP           OO=YI
A64 04884 00004FDC (00602)    NOP\MOV(OO),SUB(A6,A7)           OO=YR
A65 04886 204A204A (00603)    SET(AF2)                         APS/OTPT STAGE DN
A66 04888 459C0000 (00604)    MOV(OO),ADD(A4,A5)\NOP           OO=XR
A67 0488A 0000459C (00605)    NOP\MOV(OO),ADD(A4,A5)           OO=XI
A68 0488C 089C0000 (00606)    MOV(R,OO)\NOP                    OO=WR
A69 0488E 0000089C (00607)    NOP\MOV(R,OO)                    OO=WI
                       (00608)
A6A 04890 9008002A (00609)    JUMPC(CR4FSA,AF3)
                       (00610)
                       (00611) ;
```

```
                           *          EVEN-ODD SEPARATE
                           ;
                           ;
                           ; X(K)=A*[W(K)+W'(N-K)]
                           ; Z(K)=-J*A*[W(K)-W'(N-K)]
                           ; X(0)=2*A*[RW(0)+JRW(N/2)]
                           ; Z(0)=2*A*[IW(0)+JIW(N/2)]
                           ;
                           ; RX(K)=A(RW(K)+RW(N-K))
                           ; IX(K)=A(IW(K)-IW(N-K))
                           ; RZ(K)=A(IW(K)+IW(N-K))
                           ; IZ(K)=A(RW(N-K)-RW(K))
                           ;
                           ;
A6B 04892 0BF80BF8                 MOV(IOA, M0)                          M0=A
A6C 04894 0BF00BF0                 MOV(IOA, A0)                          A0=RW(N/2)
A6D 04896 0BF10BF1                 MOV(IOA, A1)                          A1=IW(N/2)
A6E 04898 0BF20BF2                 MOV(IOA, A2)                          A2=RW(0)
A6F 0489A 0BF30BF3                 MOV(IOA, A3)                          A3=IW(0)
A70 0489C 42404360                 ADD(A2, A2)\ADD(A3, A3)
A71 0489E 400C412C                 MOV(M4),ADD(A0,A0)\MOV(M4),ADD(A1,A1)  M4=2RW(0)\2IW(0)
A72 048A0 84008400                 MUL(M0, M4)
A73 048A2 0BB00BB0         #1      MOV(B, NULL)
A74 048A4 0BF40BF4                 MOV(IOA, A4)                          A4=IW(N-K)
A75 048A6 0BF50BF5                 MOV(IOA, A5)                          A5=RW(K)
A76 048A8 0BF60BF6                 MOV(IOA, A6)                          A6=RW(N-K)
A77 048AA 0BF70BF7                 MOV(IOA, A7)                          A7=IW(K)
A78 048AC 46A0478D                 MOV(M5),ADD(A5,A6)\MOV(M5),ADD(A4,A7)  M5=IW(K)-IW(N-K)
                                                                          \RW(N-K)-RW(K)
                           ;
A79 048AE 843C843C                 MOV(OO), MUL(M0, M5)                  OO=RX(K)\RZ(K)
A7A 048B0 4CEC4DCC                 MOV(M4),SUB(A7,A4)\MOV(M4),SUB(A6,A5)  M4=RW(K)+RW(N-K)
                                                                          \IW(K)+IW(N-K)
                           ;
A7B 048B2 841C841C                 MOV(OO), MUL(M0, M4)                  OO=IX(K)\IZ(K)
A7C 048B4 901C0073                 JUMPC(#1, EO)
A7D 048B6 20322032                 CLEAR(RA)
A7E 048B8 00000000                 NOP
A7F 048BA 10000000                 JUMP(CSM2$SA)
                           ;
          00000080                 CR4$SZ = #A-CSM2$SA
    048BC                          END
                           ;
```

```
                EVEN-ODD SEPARATE

                (00656)
               -(00657)
                (00658)
                (00659) ;
```

```
                  (00660) *   APS PROGRAMS
                  (00661) ;
                  (00662) ;
                  (00663) ; START OF HEADER BLOCK
                  (00664) ;
                  (00665)         EVEN                        START ON WORD BOUNDARY
04BBC 000049C4    (00666)         ADDR CSMS1                  PTR TO CONSTR INSTR BLOCK(NONE)
04BBE 00004BC4    (00667)         ADDR CSMS+2*CSMSS           PTR TO SCALAR BLOCK (NONE)
04BC0 0000        (00668)         DATA 0                      NO SCALARS
04BC1 0100        (00669)         DATA CR4ASZ                 MODULE SIZE
04BC2 00000000    (00670)         ADDR CR4ASA                 PTR TO CHAIN ANCHOR
                  (00671)         EVEN                        END OF BOUNDARY
                  (00672) ;
                  (00673) ;
```

```
                         '         CSM - COMPLEX FFT SCRAMBLE (P0 - INPUT)
                         ; APS PROGRAM PROVIDING SCRAMBLE FOR COMPLEX FFT
                         ; MAY BE USED WITH THE MAP-300 APU PROGRAMS
                         ; CSM2(Y,U), CSM4F(Y,U), CSM4I(Y,U)
                         ;
                         ; RESTRICTIONS
                         ; IN PLACE OPERATION NOT PERMITTED
                         ; BUFFER SIZES MUST BE 1024 OR LESS
                         ; Y BUFFER MUST BE COMPACT 32 BIT FLOATING
                         ;
                         ;
                         ; BINDING PARAMETERS
                         ;
                         ;         N=# OF POINTS IN FFT (USIZE)
                         ;         HU=USEP*N/2
                         ;         QU=HU/2
                         ; APS-INPUT ADDRESS SEQUENCE
                         ; RU(K), RU(K+N/2), IU(K+N/2), IU(K), IU(K+N/4)
                         ; IU(K+3N/4), RU(K+3N/4), RU(K+N/4)], FOR 0<=K<N/4
                         ; THUS PROVIDING REVERSAL OF TWO BITS
                         ;
                         ; APS-OUTPUT ADDRESS SEQUENCE
                         ; RW(J), RW(J+1), IW(J), IW(J+1), RW(J+2),
                         ; RW(J+3), IW(J+2), IW(J+3)
                         ; WHERE J= BIT REVERSAL OF K
                         ; THE DIFFERENCE, D(K) = J(K+1)-J(K) IS
                         ; PROVIDED BY THE P3 SUBROUTINE
                         ;
                         ;
                         CSMS    BEGIN APS(CSM)
                         ;
A00 048C4 00202540       CSMSS   JSN(CSM0,P2)
                         ;
A01 048C6 02C00000       CSML6S  LOAD(BR0,MSS,L,TF)           INPT RU(0) [UBASE BOUND]
A02 048C8 04500000       CSML0S  LOAD(BR1,MSS)                [USIZE-1 BOUND]
A03 048CA 06300672               JUMP(CSMF,RA),SET            TURN ON APU
                         ;
A04 048CC 08820000       CSML    SUB(BR0,MSS,TF)              INPT RU(K+N/4) [HU BOUND]
                         ;
A05 048CE 0A820000       CSML1S  SUB(BR0,MSS,TF)              INPT RU(K) [QU-USEP BOUND]
A06 048D0 0CBA0000       CSMF    ADD(BR0,MSS,TF)              INPT RU(K+N/2) [HU BOUND]
A07 048D2 0EBA0002               ADD(BR0,2,TF)                INPT IU(K+N/2)
A08 048D4 10820000       CSML2S  SUB(BR0,MSS,TF)              INPT IU(K) [HU BOUND]
                         ;
```

```
A09 04BD6 12BA0000 (00718) CSML3S  ADD(BRO,KSS,TF)            INPT IU(K+N/4) [OU BOUND]
A0A 04BD8 14BA0000 (00719) CSML4S  ADD(BRO,MSS,TF)            INPT IU(K+3N/4) [HU BOUND]
A0B 04BDA 16B20002 (00720)         SUB(BRO,2,TF)              INPT RU(K+3N/4)
                   (00721) ;
A0C 04BDC 1B1904B4 (00722)         SUBL(BR1,4),JUMPP(CSML)
                   (00723) ;
A0D 04BDE 1AB20000 (00724) CSML5S  SUB(BRO,MSS,TF)            INPT RU(LAST) [HU BOUND]
A0E 04BE0 1C304E77 (00725)         JUMP(CR4AE,WI),SET
                   (00726) ;
```

```
                         (00727) *          CSM-SCRAMBLE SUBROUTINE (P3 - OUTPUT)
                         (00728) ;
                         (00729) ; DATA POINT ADDRESS AT ENTRY 4*[J(K)-N]+WBASE
                         (00730) ; DATA OUTPUT ADDRESS GENERATED [4J(K+1)]+WBASE
                         (00731) ;
                         (00732) ; ITERATION EQUATION, J=J(K)=K,BIT REVERSED
                         (00733) ; 4J(K+1)=4*[J(K)-N]+DM
                         (00734) ; WHERE M=NUMBER OF TRAILING 1'S IN (K)
                         (00735) ;
                         (00736) ; BINDING CONSTANTS
                         (00737) ;
                         (00738) ;          D0=4*(3*N/2)
                         (00739) ;          D1=D0/2
                         (00740) ;          D2=D1/2
                         (00741) ;          D3=D2/2
                         (00742) ;          D4=D3/2
                         (00743) ;          D5=D4/2
                         (00744) ;          D6=D5/2
                         (00745) ;          D7=D6/2
                         (00746) ;;
                         (00747) ;
A0F 04BF2 1F107840 (00748) SCRM     JSN(ANGLE,P1)                LOADS P1 FOR CR4A
                         (00749) ;
A10 04BF4 20BF0000 (00750) SCRM0    ADD(BW0,MSS,TF,C)            [D0 BOUND]
A11 04BE6 22BF0000 (00751) SCRM1S   ADD(BW0,MSS,TF,C)            [D1 BOUND]
A12 04BF8 248F0000 (00752) SCRM2S   ADD(BW0,MSS,TF,C)            [D0 BOUND]
A13 04BEA 26BF0000 (00753) SCRM3S   ADD(BW0,MSS,TF,C)            [D2 BOUND]
A14 04BFC 28BF0000 (00754) SCRM4S   ADD(BW0,MSS,TF,C)            [D0 BOUND]
A15 04BFF 2AAE0000 (00755) SCRM5S   ADD(BW0,MSS,TF,C)            [D1 BOUND]
A16 04BF0 2CBF0000 (00756) SCRM6S   ADD(BW0,MSS,TF,C)            [D0 BOUND]
                         (00757) ;
A17 04BF2 2E201AEB (00758)          JUMPS(SCRM4,AF0),CLEAR
A18 04BF4 30BF0000 (00759) SCRM7S   ADD(BW0,MSS,TF,C)            [D3 BOUND]
A19 04BF6 32301068 (00760)          JUMP(SCRM0,AF0),SET
                         (00761) ;
A1A 04BFB 34201DF9 (00762) SCRM4    JUMPS(SCRM5,AF1),CLEAR
A1B 04BFA 36BF0000 (00763) SCRM8S   ADD(BW0,MSS,TF,C)            [D4 BOUND]
A1C 04BFC 38301069 (00764)          JUMP(SCRM0,AF1),SET
                         (00765) ;
                         (00766) ;
A1D 04BFF 3A2020EA (00767) SCRM5    JUMPS(SCRM6,AF2),CLEAR
A1E 04900 3CBF0000 (00768) SCRM9S   ADD(BW0,MSS,TF,C)            [D5 BOUND]
A1F 04902 3E30106A (00769)          JUMP(SCRM0,AF2),SET
                         (00770) ;
```

```
A20 04904 402023EB (00771) SCRM6   JUMPS(SCRM7,AF3),CLEAR
A21 04906 428E0000 (00772) SCRM10$ ADD(RW0,MSS,TF,C)              [D6 BOUND]
A22 04908 4430106B (00773)         JUMP(SCRM0,AF3),SET
                   (00774) ;
A23 0490A 468E0000 (00775) SCRM7   ADD(RW0,MSS,TF,C)              [D7 BOUND]
A24 0490C 48001060 (00776)         JUMP(SCRM0)
                   (00777) ;
```

```
                        (00778) '         CSM-SCRAMBLE SUBROUTINE (OUTPUT - P2)
                        (00779) ;
                        (00780) ;
 A25 0490F 4A300F40 (00781) CSMO     JSN(SCRM,P3)
 A26 04910 4CC00000 (00782) CSMOIS   LOAD(BW0,MSS,L,TF)             OTPT RW(0) [WBASE BOUND]
 A27 04912 4F112953 (00783)          MOVR(BW1,BR1),JUMP(CSMOE)      BW1=WSIZE-1
                        (00784) ;
 A28 04914 50060000 (00785) CSMOL    SUB(BW0,MSS,C)                 [4N+12 BOUND]
                        (00786) ;
                        (00787) ; PRIOR TO JUMP TO P3,BW0=4J(K)-4N
                        (00788) ; SCRAMBLE SUBROUTINE OUTPUTS RW(J)
                        (00789) ; LEAVING BW0=4J(K+1)
                        (00790)
 A29 04916 528A0004 (00791) CSMOE    ADD(BW0,4,TF)                  OTPT RW(J+1)
 A2A 0491B 54820002 (00792)          SUB(BW0,2,TF)                  OTPT IW(J)
 A2B 0491A 568A0004 (00793)          ADD(BW0,4,TF)                  OTPT IW(J+1)
                        (00794) ;
 A2C 0491C 588A0004 (00795)          ADD(BW0,4,TF)                  OTPT IW(J+2)
 A2D 0491E 5A8A0004 (00796)          ADD(BW0,4,TF)                  OTPT IW(J+3)
 A2E 04920 5C820006 (00797)          SUB(BW0,6,TF)                  OTPT RW(J+2)
 A2F 04922 5E8A0004 (00798)          ADD(BW0,4,TF)                  OTPT RW(J+3)
                        (00799) ;
 A30 04924 601128B4 (00800)          SUBL(BW1,4),JUMPP(CSMOL)
                        (00801) ;
```

```
'          SUCCESSIVE RADIX-4 STAGES (OUTPUT - P2)
;          PROGRAM ASSUMES SCRAMBLE COMPLETED


; 12/14/77
; SCRAMBLE LEAVES APS AS FOLLOWS
; INPUT - LAST COMMAND,JUMP(CR4AF,WI),SET
; INPUT-P1 LOCATED AT ANGLE
; OUTPUT - ACTIVE, IN P2
;
; SS=STAGE SEPARATION, DATA SEPARATION A B C D
; STARTING VALUES(/4)ARE
; SSI=1, SCRAMBLE ONLY PRECEEDING        N=4^M
; SSI=2, SCRAMBLE PLUS RADIX2 PRECEEDING N=2*4^M
; SSI=3, SCRAMBLE PLUS RADIX3 PRECEEDING N=3*4^M
; SSI=4, SCRAMBLE PLUS RADIX4 PRECEEDING N=4*4^M
; SSI=6, SCRAMBLE PLUS RADIX6 PRECEEDING N=6*4^M
;
;
; COSINE TABLE IS A REAL BUFFER WITH CONTENTS:
;      CT(K)=COS(2*PI*K/CSIZE)
; RESTRICTION.
; CSIZE MUST BE MULTIPLE OF N, FFT SIZE
;
; COSINE TABLE BINDING PARAMETERS
;
;          HPI=CSEP*(CSIZE/4)           90 DEG SEPARATION
;
; REGISTER USAGE
;          BR0/BW0                      DATA ADDRESSES
;          BR1/BW1                      # OF USES OF A COSINE, COUNTER
;          BW2                          # OF COSINES USED, COUNTER
;          BW3                          STAGE SEPARATION
;          BR2                          SEP BETWEEN COSINES WITHIN A SET
;          BR3                          SEP BETWEEN SETS OF COSINES
         EJECT
```

```
                         (00838)
A31 04926 62700000 (00839) CR4A01 LOAD(BW3,MSS)                    (SSI BOUND)
                         (00840) ;
                         (00841) ;  CR4A-APS OUTPUT PROGRAM-STAGE INITIALIZATION
                         (00842) ;
                         (00843) ;
A32 04928 6431002F (00844) CR4A02  ADDB(BW3,BW3)
A33 0492A 6631002F (00845)         ADDB(BW3,BW3)           SET SEPARATION FOR STAGE
A34 0492C 68400000 (00846) CR4A1S  LOAD(BW0,MSS)           SET BW0=WB-4 (WBASE BOUND)
A35 0492E 6A210016 (00847)         MOVB(BW2,BW3)
A36 04930 6C090010 (00848)         MOVB(BR0,BW0)           SET UP INPUT BASE
A37 04932 6E200037 (00849)         CLEAR(WI)               START UP INPUT
                         (00850)
A38 04934 70500000 (00851) CR4A03  LOAD(BW1,MSS)           SET BUTTERFLY COUNT (USIZE-1 BOUND)
A39 04936 720A0000 (00852) CR4A2S  ADD(BW0,MSS)            SET STARTING ADDRESS (4N+4 BOUND)
                         (00853)
                         (00854)
A3A 04938 74810026 (00855) CR4A04  SUBB(BW0,BW3,TF)        ZR
A3B 0493A 768A0002 (00856)         ADD(BW0,2,TF)           ZI
A3C 0493C 78810026 (00857)         SUBB(BW0,BW3,TF)        YI
A3D 0493E 7A820002 (00858)         SUB(BW0,2,TF)           YR
A3E 04940 7C810026 (00859)         SUBB(BW0,BW3,TF)        XR
A3F 04942 7E8A0002 (00860)         ADD(BW0,2,TF)           XI
A40 04944 80810026 (00861)         SUBB(BW0,BW3,TF)        WI
A41 04946 82820002 (00862)         SUB(BW0,2,TF)           WR
                         (00863) ;TEST FOR NEW COSINES, ELSE OUTPUT NEXT 4 POINTS
                         (00864) ;
A42 04948 84113AA6 (00865)         SUBB(BW1,BW3),JUMPP(CR4A04)
                         (00866) ;TEST STAGE DONE, ELSE GET NEW COSINES
                         (00867) ;
A43 0494A 86203BAA (00868)         JUMPC(CR4A03,AF2),CLEAR
                         (00869) ;
                         (00870) ;STOPPING SHORT CHECK IS HERE
                         (00871) ;
A44 0494C 88500000 (00872) CR4A13S LOAD(BW1, MSS)          (FFTSIZE-1 BOUND)
A45 0494E 8A1132A6 (00873)         SUBB(BW1, BW3), JUMPP(CR4A02)
```

```
                        (00874) '          GENERATE OUTPUT ADDRESSES FOR EVEN-ODD SEPARATE
                        (00875) ;
                        (00876) ;
                        (00877) ; W BUFFER CONTAINS FFT OF X+JZ
                        (00878) ; TWO REAL FFT'S ARE EXTRACTED AS FOLLOWS
                        (00879) ;   X(K)=A*[W(K)+W'(N-K)]
                        (00880) ;   Z(K)=A*[W(K)-W'(N-K)]
                        (00881) ; WHERE, W'=CONJUGATE OF W
                        (00882) ;
                        (00883) ; RX(0)=RW(0)
                        (00884) ; IX(0)=RW(N/2)
                        (00885) ; RZ(0)=IW(0)
                        (00886) ; IZ(0)=IW(N/2)
                        (00887) ;
                        (00888) ; X(K) IS STORED IN Y(K)
                        (00889) ; Z(K) IS STORED IN Y(K+N/2)
                        (00890) ;
                        (00891) ; RX(K)=RW(K)+RW(N-K), IX(K)=IW(K)-IW(N-K)
                        (00892) ; RZ(K)=RW(K)-RW(N-K), IZ(K)=IW(K)+IW(N-K)
                        (00893) ;
                        (00894) ; THE OUTPUT ADDRESS SEQUENCE IS:
                        (00895) ;   RX(0), RZ(0), IX(0), IZ(0),
                        (00896) ;   RX(K), RZ(K), IX(K), IZ(K)
                        (00897) ;
                        (00898) ;
A46 04950 RCC00000 (00899) EOS01    LOAD(RW0, MSS, TF)           <RX(0) [YBASE BOUND]
A47 04952 8F110010 (00900)          MOVB(BW1, BW0)
A48 04954 909A0000 (00901) EOS02    ADD(BW1, MSS, TF)            <RZ(0) [2N BOUND]
A49 04956 92600000 (00902) EOS03    LOAD(BW2, MSS)               [USIZE-2 BOUND]
A4A 04958 94BA0002 (00903) EOS04    ADD(BW0, 2, TF)              <X(K)
A4B 0495A 969A0002 (00904)          ADD(BW1, 2, TF)              <Z(K)
A4C 0495C 98214AB1 (00905)          SUBL(BW2, 1), JUMPP(EOS04)
A4D 0495E 9A203470 (00906)          JUMP(CR4A1S, R0), CLEAR      HALT APS OUTPUT
                   (00907) ;
                   (00908)
```

```
                           *        SUCCESSIVE RADIX-4 STAGES (INPUT - P0)
A4F 04960 9C300037 CR4A1   SET(WI)
                   ;STAGE INITIALIZATION
                   ;
A4F 04962 9F600000 CR4AF   LOAD(BR2,0)                    SET COSINE VALUE=0
A50 04964 A03D0015         MOVB(BF3,BR2,C)
                   ;
                   ; ANGLE SUBROUTINE,P1,INSERTS COSINE SEPARATION
                   ; FOR STAGE INTO BF3
A51 04966 A2215B71         SUBL(BW2,1),JUMP(CR4A3)        BW2=#OF COSINES-1
A52 04968 A4890032 CR4A2   SUBL(BR0,2,TF)                 AR
                   ;COSINE ENTRY
                   ;
A53 0496A A6500000 CR4A4S  LOAD(BR1,MSS,L)                [CHASE BOUND]
A54 0496C AB29002F         ADDB(BR2,BR3)                  BR2=NEXT ANGLE
A55 0496F AA99002D         ADDB(BR1,BR2,TF)               COSX
A56 04970 AC9A0000 CR4A5S  ADD(BR1,MSS,TF)                SINX [HPI BO ND]
A57 04972 AF99002D         ADDB(BR1,BR2,TF)               SIN2X
A58 04974 B0920000 CR4A6S  SUB(BR1,MSS,TF)                COS2X [HPI BOUND]
A59 04976 B299002D         ADDB(BR1,BR2,TF)               COS3X
A5A 04978 B49A0000 CR4A7S  ADD(BR1,MSS,TF)                SIN3X [HPI BOUND]
A5B 0497A B6DA0000 CR4A3   ADD(BR0,MSS)                   SET BR0 TO 4N+COS# [4N+4 BOUND]
A5C 0497C B8500000 CR4A8S  LOAD(BR1,MSS)                  SET COSINE USE COUNTER [USIZE-1 BOUND]
                   ; CR4A-APS BUTTERFLY INPUT
                   ;
A5D 0497F BA890026 CR4A4   SUBB(BR0,BW3,TF)               DR
A5E 04980 BC89003A         ADDL(BR0,2,TF)                 DI
A5F 04982 BF890026         SUBB(BR0,BW3,TF)               BI
A60 04984 C0890032         SUBL(BR0,2,TF)                 BR
A61 04986 C2890026         SUBB(BR0,BW3,TF)               CR
A62 04988 C489003A         ADDL(BR0,2,TF)                 CI
A63 0498A C6890026         SUBB(BR0,BW3,TF)               AI
                   ; TEST IF NEW COSINE NEEDED, ELSE INPUT NEXT 4 POINTS
                   ;
A64 0498C C81976A6         SUBB(BR1,BW3),JUMPP(CR4A5)
A65 0498F CA300028         SET(AFO)                       TELL APU COSINES COMING
```

```
                          (00953) ; TEST IF STAGE DONE, ELSE GET NEW COSINES
                          (00954)
 A66 04990 CC215284 (00955)          SUBL(BW2,4),JUMPP(CR4A2)
 A67 04992 CF300029 (00956)          SET(AF1)                        TELL APU STAGE DONE
                          (00957) ;********* CHANGE VALUE HERE TO STOP SHORT ***********
                          (00958) ;TO STOP SHORT 1 STAGE, BIND USIZE/4-1 HERE
 A68 04994 D0500000 (00959) CR4A9$  LOAD(BR1,MSS)                 (FFTSIZE-1 BOUND)
 A69 04996 D2890032 (00960)          SUBL(BR0,2,TF)                  LAST DATA POINT
                          (00961)
                          (00962) ; TEST IF FFT DONE, ELSE START NEXT STAGE
                          (00963)
 A6A 04998 D4194FA6 (00964)          SUBB(BR1,BW3),JUMPP(CR4A1)    GO BACK
                          (00965) ;
```

```
                          (00966) *          GENERATE INPUT ADDRESSES FOR EVEN-ODD SEPARATE
                          (00967) ;
                          (00968) ;
                          (00969) ; INPUT SEQUENCE:
                          (00970) ;  SCALAR A,
                          (00971) ;  RW(N/2), IW(N/2), RW(0), IW(0),
                          (00972) ;  [IW(N-K), RW(K), RW(N-K), IW(K)]
                          (00973) ;
                          (00974) ;
 A68 0499A D6300028 (00975)            SET(AE3)                    SIGNAL APU TO DO EOS
 A6C 0499C DBC20000 (00976) EOS1    LOAD(BRO, MSS(1), TE)           <SA [SCALAR A BOUND]
 A6D 0499E DA400000 (00977) EOS2    LOAD(BRO, MSS)              RW(0) [WBASE BOUND]
 A6E 049A0 DC190011 (00978)         MOVB(BR1, BRO)
 A6F 049A2 DEBA0000 (00979) EOS3    ADD(BRO, MSS, TE)           <RW(N/2) [2N BOUND]
 A70 049A4 F08A0002 (00980)         ADD(BRO, 2, TE)             <IW(N/2)
 A71 049A6 F2890013 (00981)         MOVB(BRO, BR1, TE)          <RW(0)
 A72 049A8 F41A0000 (00982) EOS4    ADD(BR1, MSS)               RW(N) [4N BOUND]
 A73 049AA F68A0002 (00983) EOS5    ADD(BRO, 2, TE)             <W(K)
 A74 049AC F8920002 (00984)         SUB(BR1, 2, TE)             <W(N-K)
 A75 049AE FA007360 (00985)         JUMP(EOS5)
                          (00986)
 A76 049B0 FC890032 (00987) CR4A5   SUBL(BRO,2,TE)              AR
 A77 049B2 FE005D60 (00988)         JUMP(CR4A4)
                          (00989)
```

```
                            (00990) *          ANGULAR SEPARATION SUBROUTINE (INPUT - P1)
                            (00991) ; S1ANG=PPI/SSI
                            (00992) ; S2ANG=S1ANG/4
                            (00993) ; S3ANG=S2ANG/4
                            (00994) ; S4ANG=S3ANG/4
                            (00995) ; S5ANG=S4ANG/4
                            (00996) ; S6ANG=S5ANG/4
                            (00997) ; S7ANG=S6ANG/4
                            (00998)
                            (00999)
 A78 049B4 F0300030 (01000) ANGLE    SET(R0)
 A79 049B6 F23F0000 (01001) ANGL1$   ADD(BR3,MSS,C)               [S1ANG BOUND]
 A7A 049B8 F43F0000 (01002) ANGL2$   ADD(BR3,MSS,C)               [S2ANG BOUND]
 A7B 049BA F63F0000 (01003) ANGL3$   ADD(BR3,MSS,C)               [S3ANG BOUND]
 A7C 049BC F83F0000 (01004) ANGL4$   ADD(BR3,MSS,C)               [S4ANG BOUND]
 A7D 049BE FA3F0000 (01005) ANGL5$   ADD(BR3,MSS,C)               [S5ANG BOUND]
 A7E 049C0 FC3F0000 (01006) ANGL6$   ADD(BR3,MSS,C)               [S6ANG BOUND]
 A7F 049C2 FE3F0000 (01007) ANGL7$   ADD(BR3,MSS,C)               [S7ANG BOUND]
                            (01008)
                            (01009) ;
           00000000 (01010)  CR4ASA=#C                            ASSIGN VALUE TO CHAIN
     049C4          (01011)          END                          #A-1  END OF MODULE
                            (01012) ;
           000049C4 (01013) CSMSI    #L=#L+2*0
                            (01014) ;
           00000100 (01015)  CR4ASZ=#L-CSMS
                            (01016) ;
                            (01017) *  DEFINE TOP OF MODULE
                            (01018) *
           000049C4 (01019) TOESCUR = #L
                            (01020) *
```

```
                (01021) 'SYMBOL TABLE
   049C4        (01022)       END
```

SYMBOL TABLE

```
AFDIS:      008F8 (00046) (00047) (00082) (00087)
AFDTSORG:   008F8 (00047) (00335) (00339)
ANGL1S:     00079 (00258) (01001)
ANGL2S:     0007A (00261) (0100[illegible])
ANGL3S:     0007B (00264) (010[illegible])
ANGL4S:     0007C (00267) (01004)
ANGL5S:     0007D (00270) (01005)
ANGL6S:     0007E (00273) (01006)
ANGL7S:     0007F (00276) (01007)
ANGLE:      00078 (00748) (01000)
APSENDRO:   00F63 (00064) (00356)
APSBSL:     0024D (00061) (00243)
BCTSAD:     00604 (00054) (00112) (00224)
BCTSAT:     00686 (00055) (00107)
BCTSBA:     00582 (00053) (00219) (00290) (00316) (00328)
CR4SSZ:     00080 (00378) (00650)
CR4ASA:     00000 (00670) (01010)
CR4ASZ:     00100 (00669) (01015)
CR4A1:      0004F (00911) (00964)
CR4A1S:     00034 (00300) (00846) (00906)
CR4A13S:    00044 (00117) (00872)
CR4A2:      00052 (00923) (00955)
CR4A2S:     00039 (00200) (00852)
CR4A3:      0005B (00201) (00921) (00935)
CR4A4:      0005D (00941) (00988)
CR4A4S:     00053 (00220) (00926)
CR4A5:      00076 (00951) (00987)
CR4A5S:     00056 (00232) (00930)
CR4A6S:     00058 (00233) (00932)
CR4A7S:     0005A (00234) (00934)
CR4A8S:     0005C (00115) (00936)
CR4A9S:     00068 (00116) (00959)
CR4AF:      0004F (00725) (00914)
CR4A01:     00031 (00246) (00250) (00839)
CR4A02:     00032 (00844) (00873)
CR4A03:     0003B (00113) (00851) (00868)
CR4A04:     0003A (00855) (00865)
CR4FB:      00040 (00544) (00583)
CR4FC:      00039 (00534) (00584)
CR4FF:      00054 (00530) (00575)
CR4FS:      00028 (00416) (00508)
CR4FSA:     0002A (00510) (00609)
CSMS:       048C4 (00084) (00089) (00113) (00114) (00115) (00116) (00117) (00120) (00148) (00151)
                  (00158) (00159) (00160) (00161) (00164) (00165) (00168) (00171) (00174) (00177)
```

SYMBOL TABLE

```
                 (00180) (00183) (00190) (00191) (00194) (00200) (00201) (00204) (00211) (00220)
                 (00232) (00233) (00234) (00246) (00250) (00258) (00261) (00264) (00267) (00270)
                 (00273) (00276) (00291) (00296) (00300) (00317) (00329) (00361) (00362) (00363)
                 (00364) (00365) (00667) (00703) (01015)
CSMSI:     049C4 (00666) (01013)
CSMSS:     00000 (00667) (00705)
CSM2S:     047BC (00083) (00088) (00395)
CSM2SSA:   00000 (00377) (00398) (00648) (00650)
CSM2F:     00003 (00399) (00403)
CSM2L:     00002 (00401) (00411)
CSM4F:     0000P (00398) (00434)
CSM4FS:    00018 (00439) (00456)
CSMF:      00006 (00362) (00709) (00714)
CSML:      00004 (00361) (00711) (00722)
CSML0S:    00002 (00114) (00708)
CSML1S:    00005 (00151) (00713)
CSML2S:    00008 (00363) (00716)
CSML3S:    00009 (00148) (00718)
CSML4S:    0000A (00364) (00719)
CSML5S:    0000P (00365) (00724)
CSML6S:    00001 (00317) (00707)
CSM0:      00025 (00705) (00781)
CSM01S:    00026 (00291) (00782)
CSM0F:     00029 (00783) (00791)
CSM0L:     00028 (00204) (00785) (00800)
FOS1:      0006C (00211) (00976)
FOS2:      0006D (00296) (00977)
FOS3:      0006F (00190) (00979)
FOS4:      00072 (00194) (00982)
FOS5:      00073 (00983) (00985)
FOS01:     00046 (00329) (00899)
FOS02:     00048 (00191) (00901)
FOS03:     00049 (00120) (00902)
FOS04:     0004A (00903) (00905)
FF2RS:     00006 (00079) (00082)
FF2RBS:    000D7 (00080) (00087)
FFTSSFT:   04690 (00085) (00103)
FLGSG0:    00004 (00058)
FLGSG1:    00005 (00059) (00247) (00251)
FLGSSFT:   00020 (00057) (00251)
GSFLGS:    047B0 (00247) (00251) (00351) (00359)
HS:        00001 (00051) (00052) (00103) (00107) (00113) (00114) (00115) (00116) (00117) (00120)
                 (00148) (00151) (00158) (00159) (00160) (00161) (00164) (00165) (00168) (00171)
                 (00174) (00177) (00180) (00183) (00190) (00191) (00194) (00200) (00201) (00204)
```

SYMBOL TABLE

```
                   (00208) (00211) (00212) (00232) (00233) (00234) (00246) (00247) (00250) (00251)
                   (00258) (00261) (00264) (00267) (00270) (00273) (00276) (00283) (00312) (00324)
                   (00351)
  HUSPN:     046BC (00132) (00139)
  HUSLP:     046B0 (00130) (00137)
  HUSTHL:    047B2 (00129) (00361)
  MSS:       00000 (00056) (00707) (00708) (00711) (00713) (00714) (00716) (00718) (00719) (00724)
                   (00750) (00751) (00752) (00753) (00754) (00755) (00756) (00759) (00763) (00768)
                   (00772) (00775) (00782) (00785) (00839) (00846) (00851) (00852) (00872) (00899)
                   (00901) (00902) (00926) (00930) (00932) (00934) (00935) (00936) (00959) (00976)
                   (00977) (00979) (00982) (01001) (01002) (01003) (01004) (01005) (01006) (01007)
  MSKSLBYT:  0FF00 (00049) (00104) (00213) (00284) (00313) (00325)
  MSKSRBYT:  000FF (00050) (00209)
  PWR2S:     047BB (00108) (00244) (00368)
  SHMS1:     04735 (00245) (00250)
  SHMS2:     0473A (00249) (00254)
  SHMSFFT:   04754 (00090) (00282)
  SCRM:      0000F (00748) (00781)
  SCRM0:     00010 (00158) (00750) (00760) (00764) (00769) (00773) (00776)
  SCRM1S:    00011 (00164) (00751)
  SCRM10S:   00021 (00180) (00772)
  SCRM2S:    00012 (00159) (00752)
  SCRM3S:    00013 (00168) (00753)
  SCRM4:     0001A (00758) (00762)
  SCRM4S:    00014 (00160) (00754)
  SCRM5:     0001D (00762) (00767)
  SCRM5S:    00015 (00165) (00755)
  SCRM6:     00020 (00767) (00771)
  SCRM6S:    00016 (00161) (00756)
  SCRM7:     00023 (00183) (00771) (00775)
  SCRM7S:    00018 (00171) (00759)
  SCRM8S:    0001B (00174) (00763)
  SCRM9S:    0001F (00177) (00768)
  START:     04690 (00073) (00093)
  SVTS:      00382 (00060) (00210)
  SYSSFLGS:  1FFCF (00063)
  TOFSCUR:   049C4 (00068) (01019)
  TOFSPTR:   002BB (00048) (00066)
  WS:        00002 (00052) (00082) (00087) (00113) (00114) (00115) (00116) (00117) (00120) (00148)
                   (00151) (00158) (00159) (00160) (00161) (00164) (00165) (00168) (00171) (00174)
                   (00177) (00180) (00183) (00190) (00191) (00194) (00200) (00201) (00204) (00211)
                   (00220) (00232) (00233) (00234) (00246) (00250) (00258) (00261) (00264) (00267)
                   (00270) (00273) (00276) (00291) (00296) (00300) (00317) (00329) (00337) (00338)
                   (00339) (00344) (00361) (00362) (00363) (00364) (00365)
```

SYMBOL TABLE

```
ZERO:      007BA (00062)

LINES WITH ERRORS:    0  (MAP VERSION 800101,10)   F=    0
```

T A B L E   O F   C O N T E N T S

```
                       *      APSDONE    AP PROCESS INTERRUPT HANDLER    MAY 28, 1980
                       *
                       *
                       *
                       *  THE AP DONE INTERRUPT ROUTINE HAS BEEN MODIFIED
                       *  FOR GTE SYLVANIA TO SPEED UP PROCESSING.
                       *  IN ADDITION, SPECIAL CODE HAS BEEN ADDED TO CONVERSE W/ TH
                       *  ATC DISPATCH PROGRAM "CSPUEK.TXT".
                       *
                       *    THE LIMITATIONS ARE DESCRIBED IN THE DOCUMENTATION.
                       *
                       *
          000002F6     *L = $02F6                       SET PC TO CSW FOR AP DONE INTERRUPT
                       *
002F6 0F82                       DATA APSDONE           SET CSW TO NEW START ADDRESS
                       *
```

```
         (00017) *  DEFINE NECESSARY SYMBOLS FROM SNAP-II EXECUTIVE
         (00018) *                                   RELEASE 3.05
         (00019) *
00001390 (00020)          ABORTS = $1390
00000245 (00021)          APSASSS = $245
0000024A (00022)          APSCSL = $24A
00000006 (00023)          APSCSPU = $6
00000008 (00024)          APSECB = $8
0000000D (00025)          APSGT = $D
00000244 (00026)          APSPAF = $244
00000010 (00027)          APSPBFF = 16
00000004 (00028)          APSSBMO = $4
0001FFC0 (00029)          APSSLA = $1FFC0
00000008 (00030)          APSSSIZE = $8
00000000 (00031)          APUSBMO = $0
0001FFCA (00032)          APUSLA = $1FFCA
0001FFC8 (00033)          APUSPC = $1FFC8
00000003 (00034)          APUSSIZE = $3
00000002 (00035)          APUSSTRT = $2
         (00036) *
         (00037) *
00000011 (00038)          FLGSRT = $11
00000020 (00039)          FLGSSET = $20
         (00040) *
00000001 (00041)          HS = 1
00000502 (00042)         ISVTS=$502
0000057F (00043)         ISAS125=ISVTS+D'125'
00000580 (00044)         ISAS126=ISVTS+D'126'
00000020 (00045)          IDEVS32 = $20
0000003F (00046)          IDEVS62 = $3F
0000003F (00047)          IDEVS63 = $3F
00000001 (00048)          ILVLS1 = $1
00000002 (00049)          ILVLS2 = $2
         (00050) *
00005000 (00051)          OFFSET = $5000
         (00052) *
00000E02 (00053)          SCDSENTRY = $E02
00001C12 (00054)          SETSSNB = $1C12
0001FFCF (00055)          SYSSFLGS = $1FFCF
         (00056) *
00000002 (00057)          WS = $2
         (00058) *
         (00059) *
         (00060)          EJECT
```

DEFINE NECESSARY SYMBOLS FROM SNAP-II EXECUTIVE

```
          *
00000FB2  *L = $0FB2          SET PC TO OLD APSDONE + HS   (EVEN)
          *
          *  THIS MODULE IS ENTERED WHENEVER RA MAKES THE
          *  TRANSITION FROM ON TO OFF OR THE EXECUTIVE HAS
          *  GENERATED THE INTERRUPT BECAUSE THE AP IS IDLE AND
          *  A TASK IS PENDING
          *
          *  THE MODULE RELEASES THE CURRENT ACTIVE TASK
          *  AND ACTIVATES ANY PENDING TASK.
          *
          *  THE CONTENTS OF THE AP-PROCESSOR CONTROL BLOCK
          *  ARE USED TO CONTROL THE FUNCTIONS OF THIS MODULE
          *
          *  THE MODULE ASSUMES THAT REGISTERS R1-R7 ARE
          *  PRESERVED BY THE INTERRUPT.
          *
          *
00FB2 C060024A  APSDONE   MOVMRL   R6,APSCSL            GET POINTERS TO SUPPORT LISTS
          **              ANDIR    R6, $FFFF            MASK ADDRESS FOR SIGN EXTENSION
          **              ANDIR    R7, $FFFF            IF ABOVE 16K HALF-WRDS
00FB4 0260                TEST     R6
00FB5 B0100FC0            JMP      APSDNE10,EQZ         JUMP IF NULL
          *
00FB7 0800                EVEN
00FB8 E01C000B            MOVMR    R1, APSFCB(R6)       GET CURRENT AP-FCB POINTER
          **              DECM     APSASSS              IF SPECIAL SUPPORT SEMAPHORE NON-ZERO
          **              JMP      @APSCSPU(R6), GEZ    GOTO DESIGNATED MODULE
          *
          *
          * RETURN HERE IF JUMP WAS EXECUTED WHEN NO SUPPORT WAS REQUESTED
          *
          *
00FBA 12FC003F  APSDONER  AINT     IDEVS63, ILVL$2      FLAG AP DONE TO WAITING PROGRAM
          **              MOVKR    R4,R1,,NOT. MSK$BTO  GET POINTER TO FCB JUST PROCESSED
          **              MOVMR    R5, HS(R4)           AND FCB NUMBER WITH CONTROL BITS
          **              MOVRML   R4, APSLFCB          SAVE LAST FCB PROCESSED
          **              MOVMR    R3, APSPBFF(R6)      IF PERM. BOUND FUNCTION
          **              IORMR    R3, FLGSNOBT         OR BUFFER TESTING INHIBITED
          **              JMP      APSDNE05, NEZ        BYPASS BUFFER IN USE CLEANUP
          **
          **              MOVIR    R3, FCBSMNLB-1       ELSE RESET BUFFER IN-USE FLAGS
          **              MOVIR    R5,.NOT. MSKSCUBU
          **              ADDIR    R4, WS(R3)           POINT TO LAST POSSIBLE LOGICAL BUFFER ID
```

DEFINE NECESSARY SYMBOLS FROM SNAP-II EXECUTIVE

```
                 (00105) **
                 (00106) ***22      MOVAR    R2,R4,MSKSLBYT
                 (00107) **         LRS      R2,7
                 (00108) **         DECR     R4,1
                 (00109) **         ANDRM    R5,BCTSAT(R2)
                 (00110) **         OJP      R3,#22
                 (00111) *
00FBC 2802       (00112) APSDNE05   SRBC     0, R1               IF FCB NOT IN FUNCTION LIST
00FBD 2002       (00113)            BOP      APSDNE10            BYPASS RELEASE OF THE FCB
                 (00114) *
00FBE 12000020   (00115)            AINT     IDEVS32, ILVLS1     RELEASE THE FCB
                 (00116) *
                 (00117) *  PROCESS PENDING TASK
                 (00118) *
00FC0 406F       (00119) APSDNE10   MOVRR    R6, R7              IF PENDING FCB NULL,
00FC1 0800       (00120)            EVEN
00FC2 80100FF0   (00121)            JMP      APSDNE20,EQZ        GO SET AP TO IDLE STATE
                 (00122) *
00FC4 0D70       (00123)            CLR      R7                  SET PENDING LIST TO NULL
00FC5 0800       (00124)            EVEN
00FC6 FCBC0000   (00125)            MOVAR    R3,RAPUSBMO(R6)     NOW LOAD APU
                 (00126) **         CMPMR    R3, APSUBMO
                 (00127) **         JMP      APSDNE30,EQ         SET NEW CURRENT APU MODULE ORIGIN
                 (00128) **
                 (00129) **         MOVRM    R3,APSUBMO          UPDATE CURRENT APU MEMORY ORIGIN
                 (00130) **         MOVMR    R2,APUSSTRT(R6)     SET MODULE START IN AP
                 (00131) **         MOVRM    R2,APUSPC
00FC8 CC01FFC8   (00132)            MOVZM    APUSPC              ASSUME APU STARTS AT 0
00FCA F02C0003   (00133)            MOVMR    R2,APUSSIZE(R6)     GET MODULE SIZE
00FCC 2721       (00134)            DECR     R2, HS
                 (00135) **         SMBSL    BITSPT, SYSSFLGS    DETERMINE PROCESSOR TYPE
                 (00136) **         JMP      APSDNE40
                 (00137) **
                 (00138) **         DECR     R3,HS               MAP 100, 200
                 (00139) **         EVEN
                 (00140) **         LPROC    R3,R2,APUSLA
                 (00141) **         JMP      APSDNE30
                 (00142) *
00FCD 2732       (00143) APSDNE40   DECR     R3, WS              MAP 300
                 (00144)            EVEN
00FCE D635FFCA   (00145)            LPROCL   R3,R2,APUSLA
                 (00146) *
                 (00147) *R2        CCS      2
                 (00148) *R2        SCS      3
```

DEFINE NECESSARY SYMBOLS FROM SNAP-II EXECUTIVE

```
00FD0 FCHC0004 APSDNE30  MOVAR    R3, @APS$HMO(R6)
00FD2 2732                DECR     R3,WS
00FD3 0600                EVEN
               *B2       SUBIR    R3, OFFSET                SUBTRACT OFFSET ON BUS 2
               *
00FD4 F02C0008            MOVAR    R2,APS$SIZE(R6)
00FD6 3C21                LRS      R2, 1                    COUNT = # WORDS TO BE MOVED
00FD7 2721                DECR     R2, HS
00FD8 D635FFC0            LPROCL   R3,R2,APS$LA
               *
               *B2       CCS   3
               *B2       SCS   2
               *
               *  UPGRADE PENDING AP-FUNCTION TO CURRENT
               *
               **        MOVMRL   R4, APSG0(R6)         PICK UP G0, G1 FLAG STATES
00FDA F04C000D            MOVMR    R4, APSG1(R6)
00FDC F041F1CF            MOVRM    R4, SYS$FLGS          SET OR CLEAR G1
               **        MOVRM    R5, SYS$FLGS          SET OR CLEAR G1
               **        MOVZM    APSASSS               SET SPECIAL SUPP. SEMAPHORE
               **        MOVMR    R4, APSSSS(R6)        SET SPECIAL SUPPORT SEMAPHORE
               **        MOVRM    R4, APSASSS
               **        MOVMRL   R3, APSSSCLR(R6)      SET SPECIAL SCALAR
               **        MOVRML   R3, APSCSSC
00FDE F863FFCF            MOVLM    FLG$SET+FLG$RI, SYS$FLGS  START APS
               **
               **  CHECK FOR PERMENENTLY BOUND FUNCTION OR BUFFER CHECK INHIBIT
               **
               **        MOVMR    R3, APSPBFF(R6)       GET PERMANENTLY BOUND FUNCTION FLAG
               **        IORMR    R3, FLG$NOBT          COMBINE PBF FLAGS WITH BUFFER INHIBIT
               **        JMP      APSDNE15, NEZ         BYPASS BUFFER USAGE UPGRADE IF EITHER SET
               **
               **  BUFFER USAGE UPGRADE - PENDING TO CURRENT
               **
               **
               **        MOVMR    R1, APSFCB(R6)
               **        MOVIR    R3, FCB$MNLB-1
               **        MOVIR    R5,.NOT.MSK$PUBU
               **        MOVKR    R4, R1, .NOT. MSK$BTO
               **        ADDIR    R4,WS(R3)
               **
               **
               **   LOOP TO TRANSFER BUFFER USAGE BITS
               **
```

```
                    (00193) **#21     MOVWR    R2,R4,MSKSLBYT
                    (00194) **        ERS      R2,7
                    (00195) **        EVEN
                    (00196) **        MOVIR    R1,MSKSPUBU
                    (00197) **        ANDMR    R1,BCTSAT(R2)
                    (00198) **        LRS      R1,1
                    (00199) **        EVEN
                    (00200) **        ANDRM    R5,BCTSAT(R2)
                    (00201) **        IORRM    R1,BCTSAT(R2)
                    (00202) **        DECR     R4,1
                    (00203) **        EVEN
                    (00204) **        GAP      R3,#21
                    (00205) *
00FF0 B460024A (00206) APSDNE15  MOVRML   R6, APSCSL          UPDATE SUPPORT LIST POINTERS
00FF2 90500001 (00207)           MOVIR    R5, 1               FLAG AP BUSY
00FF3 E0500214 (00208)           MOVRM    R5, APSPAF
00FF6 00900020 (00209)           WAITC    IDEVS32, ILVLS1     WAIT TILL FCB RELEASE DONE
               (00210) *****************************************************************
               (00211) *  ADD INTERFACE CODE FOR "CSPUFK"
00FF8 E8020580 (00212)          MOVLM HS,ISAS126                        SET "APDONE" SCALAR
00FFA E500057E (00213)          INCM ISAS125                            FCB CALLING CARD. =0 @ T
               (00214) :                                                OP OF MPWHL
               (00215) *****************************************************************
00FFC 0E70     (00216)          RET                                     RETURN,BACK TO CSPUFK?
00FFD 0800     (00217)          EVEN
00FFE B000OFB2 (00218)          JMP APSDONE                             RESTART ON NEXT INTERRUP
               (00219) :                                                T
               (00220) *
               (00221) *
               (00222) *  PATH FOR NULL PENDING FCB
               (00223) *
               (00224) *
00FF0 CC00024A (00225) APSDNE20  MOVZM    APSCSL              SET CURRENT FCB NULL
00FF2 CC000244 (00226)           MOVZM    APSPAF              SET PROCESSOR AVAILABLE
00FF4 00900020 (00227)           WAITC    IDEVS32, ILVLS1     WAIT TILL FCB RELEASE DONE
00FF6 1200003F (00228)           AINT     IDEVS62, ILVLS1     TELL WAIT ROUTINE AP IS IDLE
               (00229) *****************************************************************
               (00230) *  ADD INTERFACE CODE FOR "CSPUFK"
00FF8 E8020580 (00231)          MOVLM HS,ISAS126                        SET "APDONE" SCALAR
00FFA E500057E (00232)          INCM ISAS125                            FCB CALLING CARD. =0 @ T
               (00233) :                                                OP OF MPWHL
               (00234) *****************************************************************
00FFC 0E70     (00235)          RET                                     RETURN,BACK TO CSPUFK?
00FFD 0800     (00236)          EVEN
```

```
AP PROCESS INTERRUPT HANDLER    MAY 28, 1980
   SYMBOLS FROM SNAP-II EXECUTIVE

         JMP APSDONE                                   RESTART ON NEXT INTERRUP
      ;                                                T
         END
```

DEFINE NECESSARY SYMBOLS FROM SNAP-II EXECUTIVE

```
APORTS:     01390 (00020)
APSASSS:    00215 (00021)
APSCSL:     0024A (00022) (00079) (00206) (00225)
APSCSPD:    00006 (00023)
APSDNE05:   00FBC (00112)
APSDNE10:   00FC0 (00083) (00113) (00119)
APSDNE15:   00FE0 (00206)
APSDNE20:   00FF0 (00121) (00225)
APSDNE30:   00FD0 (00149)
APSDNE40:   00FCD (00143)
APSDONE:    00F82 (00015) (00079) (00218) (00247)
APSDONER:   00F8A (00094)
APSFCR:     00006 (00024) (00086)
APSG1:      0000D (00025) (00165)
APSPAF:     00244 (00026) (00208) (00226)
APSPHFF:    00010 (00027)
APSSHMD:    00004 (00028) (00149)
APSSLA:     1FFC0 (00029) (00157)
APSSSIZE:   00008 (00030) (00154)
APUSEMD:    00000 (00031) (00125)
APUSLA:     1FFCA (00032) (00145)
APUSPC:     1FFCB (00033) (00132)
APUSSIZE:   00003 (00034) (00133)
APUSSTRT:   00002 (00035)
FLGSET:     00011 (00038) (00173)
FLGSSET:    00020 (00039) (00173)
HS:         00001 (00041) (00134) (00156) (00212) (00231)
IDEVS32:    00020 (00045) (00115) (00209) (00227)
IDEVS62:    0003E (00046) (00228)
IDEVS64:    0003F (00047) (00094)
ILVLS1:     00001 (00048) (00115) (00209) (00227) (00228)
ILVLS2:     00002 (00049) (00094)
ISAS125:    0057F (00043) (00213) (00232)
ISAS126:    00580 (00044) (00212) (00231)
ISVTS:      00502 (00042) (00043) (00044)
OFFSET:     05000 (00051)
SCDSENTR:   00F02 (00053)
SETSSMR:    01C12 (00054)
SYSSFLGS:   1FFCF (00055) (00166) (00173)
WS:         00002 (00057) (00143) (00150)

LINES WITH ERRORS:     0  (MAP VERSION R00101.10)    E=     0
```

PAGE 1: PROG, CSPDSP.TXT

```
          ; PROG, CSPDSP.TXT
          ; CSPU SUPPORT PROGRAMS TO PERFORM I-O SYNCHRONIZATION,
          ; SERIALIZATION,DESERIALIZATION, ERROR CODING AND ERROR DECODING
          ;
          ;                                      ORIGINATED:17-DEC-79
          ;                                      UPDATED:   11-AUG-80
          ; DEFINE GLOBAL SYMBOLS
                 OPADD LOAD,(1 .LS. 14)+(29 .LS. 8)+X'A6'
                 OPADD STORE,(1 .LS. 14)+(29 .LS. 8)+X'F1'
000008E8         SEDISORG=$8E8
000021FC         CSPU$VOS=$21FC
00000018         IDEVS24=24
00000001         ILVLS1=1
00000405         CS&STDP=$405
00000004         CLRSG0=$4
00000006         CLRSG2=$6
00000007         CLRSG4=$7
00000024         SETSG0=$24
00000026         SETSG2=$26
00000027         SETSG3=$27
0001FFCF         SYSFLGS=$1FFCF
00000794         DMYS=$794
00000502         ISVTS=$0502
00000503         ISAS01=ISVTS+D'01'
00000566         ISAS100=ISVTS+D'100'
00000567         ISAS101=ISVTS+D'101'
00000568         ISAS102=ISVTS+D'102'
00000569         ISAS103=ISVTS+D'103'
0000056A         ISAS104=ISVTS+D'104'
0000056B         ISAS105=ISVTS+D'105'
0000056C         ISAS106=ISVTS+D'106'
0000056D         ISAS107=ISVTS+D'107'
0000056E         ISAS108=ISVTS+D'108'
0000056F         ISAS109=ISVTS+D'109'
00000570         ISAS110=ISVTS+D'110'
00000571         ISAS111=ISVTS+D'111'
00000572         ISAS112=ISVTS+D'112'
00000573         ISAS113=ISVTS+D'113'
00000574         RGO=ISVTS+D'114'
00000575         XGO=ISVTS+D'115'
00000576         RGO2=ISVTS+D'116'                    SYNC WITH APU FOR DESER.
00000577         SYNCS=ISVTS+D'117'
00000578         ESYN=ISVTS+D'118'
00000579         ERRSYN=ISVTS+D'119'
```

```
00000575 (00045)        DLSYF=ISVIS+D'120'
00000576 (00046)        ISTEP=ISVIS+D'121'
0000057C (00047)        FLSYF=ISVIS+D'122'
00000581 (00048)        ISAS127=ISVIS+D'127'
00000003 (00049)        K1=3
00000001 (00050)        NS=1
00000000 (00051)        MS=0
00000000 (00052)        MSS=0
00008A00 (00053)        START=$8A00
00000482 (00054)        SVIS=$0482
00000498 (00055)        SAS11=SVIS+2*D'11'
00000002 (00056)        WS=2
00000004 (00057)        ZS=4
         (00058) ;
         (00059) ;
         (00060) ;  DEFINITIONS OF BUFFER PARAMETERS
         (00061) ;
000000F6 (00062)        BL10=246                          'SPEECH BUFFER LENGTH
00000100 (00063)        L10=256
00000040 (00064)        L101=64
00000171 (00065)        LSFN=369                          'DATA BUFFER LENGTH
0000000D (00066)        LPARM=13                          'BASIS INFO BUFFER LENGT
         (00067) ;                                        H
0000003E (00068)        BLT01=62
0000007C (00069)        BLT012=124
0000003E (00070)        LSFN1=62
0000007C (00071)        LSFN12=124
0000003D (00072)        LSFN2=61
0000007A (00073)        LSFN22=122
         (00074) ;
         (00075) ;  DEFINITIONS OF I-O BUFFER FLAGS
         (00076) ;
00000566 (00077)        ALAS=ISAS100                      'READY TO ANALYZE:0=YES
         (00078) ;                                        1=NO
00000567 (00079)        AFLGS=ISAS101                     'OUTPUT XMIT DATA BUFFER
         (00080) ;                                        :
00000568 (00081)        SYNS=ISAS102                      'READY TO SYNTHESIZE:
00000569 (00082)        SFLGS=ISAS103                     'INPUT RCVR DATA BUFFER:
0000056A (00083)        OUTS=ISAS104                      'READY TO OUTPUT SPEECH:
0000056B (00084)        OFLGS=ISAS105                     'OUTPUT SPCH BUFFER:
0000056C (00085)        INPS=ISAS106                      'READY TO INPUT SPCH:
0000056D (00086)        IFLGS=ISAS107                     'INPUT SPCH BUFFER:
00000581 (00087)        ICVOFF=ISAS127                    '*** FOR TIMER PROGRAM O
         (00088) ;                                        NLY ***
```

```
                  ;
                  ;
                  ;  EXTRA I-O FLAGS (PRIMARILY TO INDICATE ERRORS)
                  ;
        0000056E                KEATAS=ISAS108
        0000056F                XGATAS=ISAS109
        00000570                INFOLL=ISAS110
        00000571                OUTFUL=ISAS111
        00000572                KOUNTS=ISAS112
        00000574                INITS=ISAS113
        00000580                APPNELS=S502+126              ISVTS+126=IS126
                  ;
                  ;
                  *
                  *
                  ;  INCLUDE INTERRUPT TRANSFER VECTOR
                  *
        000000C2                #L=SC2
  000C2 871069E1A               IPTKPT R1,CS#SD24
                  ;  ENABLE
        00000D76                #L=S0D76
  00D76 12900018                EINT IOEVS24,ILVLS1
                  ;PATCH SCHEDULE TABLE D24 INSTEAD OF 2ND GO, AND END TABLE
        0000037A                #L=S37A
  0037A 0018                    DATA 24
                  ;CHANGE PRIORITIES OF IO SCROLLS
        0000036A                #L=S36A
  0036A 0018                    DATA S18,S16,S17
  0036A 0016
  0036C 0017
                  ;PATCH "SCD" TO INTERRUPT TO CSPD
        00000DFE  SCDS19=S0DFE
        00000F50                #L=S0F50
  00F50 80001D50                JMP S1D50
        00001D50                #L=S1D50
  01D50 12000018                AIST IOEVS24,ILVLS1
  01D52 00000018                WAITC IOEVS24,ILVLS1
  01D54 80000DFE                JMP SCDS19
                                EJECT
```

PAGE 4: PROG. CSPOSP.TXT

```
                  (00126) *
                  (00127) *>>>>>>>>>>>>>>>>>>>>>>>>>>>>> C S P H <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
                  (00128) *
                  (00129)              ROS 2
        00000D2A  (00130)              #L=3370
0D2A 0000         (00131) MIN1S        DATA D'0'
        00000E20  (00132)              #L=3616
0E20 0000         (00133) MIN2S        DATA D'0'
        0000100C  (00134)              #L=4108
0100C 0000        (00135) NOUT1S       DATA D'0'
        00001102  (00136)              #L=4354
01102 0000        (00137) NOUT2S       DATA D'0'
        000011FE  (00138)              #L=4606
011FE 0000        (00139) RECV1S       DATA D'0'
        00001369  (00140)              #L=4969
01369 0000        (00141) RECV2S       DATA D'0'
        0000140C  (00142)              #L=5140
0140C 0000        (00143) SEN1S        DATA D'0'
        0000164E  (00144)              #L=5710
0164E 0000        (00145) SEN2S        DATA D'0'
                  (00146) ;
                  (00147) ;
                  (00148) ;
                  (00149) ;
                  (00150) ;
                  (00151)              ROS 1
        00008A00  (00152)              #L=35328
                  (00153) QPBS
08A00 0000        (00154) QIPARS       DATA 8D'0'
 ...
08A08 0000        (00155) QIVAR        DATA D'0'
08A09 0000        (00156) QTOCS        DATA D'0'
08A0A 0000        (00157) QTMS         DATA D'0'
08A0B 0000        (00158) QIVS         DATA D'0'
08A0C 0000        (00159) QICS         DATA D'0'
        00008A0E  (00160)              #L=35342
08A0E 0000        (00161) QTPCTS       DATA D'0'
        00008B0E  (00162)              #L=35598
08B0E 0000        (00163) TGTIS        DATA D'0'
        00008C0E  (00164)              #L=35854
08C0E 0000        (00165) SENS         DATA D'0'
        00008D80  (00166)              #L=36224
08D80 0000        (00167) TGPOS        DATA D'0'
        00008E80  (00168)              #L=36480
```

```
08E80 0000        (00169) RE1S     DATA D'0'
      00008EE1    (00170)          *L=36849
08EE1 0000        (00171) RE2S     DATA D'0'
      00009162    (00172)          *L=37218
09162 0000        (00173) RE3S     DATA D'0'
      00009204    (00174)          *L=37587
09204 0000        (00175) ZROS     DATA D'0'
                  (00176)
      00009414    (00177)          *L=37956
09414 0000        (00178) OUTRS    DATA D'0'
      00009514    (00179)          *L=38212
09514 0000        (00180) RECVS    DATA D'0'
      00009666    (00181)          *L=38582
09666 0000        (00182) RQPRS    DATA D'0'
      000096C4    (00183)          *L=38596
096C4 0000        (00184) RQTDCIS  DATA D'0'
      000097C4    (00185)          *L=38852
097C4 0000        (00186) SYNB1S   DATA D'0'
      000097C4    (00187)          RSSSS=SYNB1S
      00009935    (00188)          *L=39221
09935 0000        (00189) SYNB2S   DATA D'0'
      00009935    (00190)          RSSPE=SYNB2S
      00009AA6    (00191)          *L=39590
09AA6 0000        (00192) RERITS   DATA D'0'
      00009C40    (00193)          *L=40000
                  (00194)          EVEN
09C40 0000A088    (00195) INDX1S   ADDR BIT0S,BIT5S,BIT4S,BIT4S,BIT1S,BIT1S
09C42 0000A074
09C44 0000A078
09C46 0000A074
09C48 0000A084
09C4A 0000A084
09C4C 0000A066    (00196)          ADDR BIT2S,BIT2S,BIT3S,BIT3S,BIT4S,BIT4S
09C4E 0000A080
09C50 0000A07C
09C52 0000A07C
09C54 0000A070
09C56 0000A078
                  (00197)          EVEN
09C58 0000A242    (00198) INDX2S   ADDR DBIT0S,DBIT1S,DBIT2S,DBIT3S,DBIT4S
09C5A 0000A1DE
09C5C 0000A176
09C5E 0000A110
09C60 0000A0A2
```

```
PAGE    62        PROG. CSFOAF.DEI

                     (00199)           EVEN
09C62 0000A96A (00200) INDX4S   ADDR FBIT0S,FBIT5S,FBIT4S,FBIT4S,FBIT1S,FBIT1S
09C64 0000A92C
09C66 0000A942
09C68 0000A942
09C6A 0000A944
09C6C 0000A944
09C6E 0000A93E (00201)           ADDR FBIT2S,FBIT2S,FBIT3S,FBIT3S,FBIT4S,FBIT4S
09C70 0000A93E
09C72 0000A938
09C74 0000A938
09C76 0000A932
09C78 0000A932
                     (00202)           EVEN
09C7A 0000AA44 (00203) INDX5S   ADDR GBIT0S,GBIT1S,GBIT2S,GBIT3S,GBIT4S
09C7C 0000AADE
09C7E 0000AA78
09C80 0000AA12
09C82 0000A 60
                     (00204)           EVEN
09C84 0000           (00205) FRSTF    DATA 0                                      SET TO INDICATE FIRST RC
                     (00206) ;                                                    VR FRAME AFTER SYNC
09C85 0060           (00207) POST2S   DATA D'109'
09C86 00AC           (00208) POST3S   DATA D'172'
09C87 8CEA           (00209) POST3S1  DATA SEQS+172
09C88 95E0           (00210) POST3S2  DATA RECVS+172
09C89 0170           (00211) POST4S   DATA LSER-1
09C8A 0000           (00212) ZEROS    DATA D'0'
09C8B 00FF           (00213) FRMLNGS  DATA  LTB-1
09C8C 0016           (00214) PARMSM2  DATA LPARM+LPARM-4
09C8D 0000           (00215) #ASKS    DATA D'0'
09C8E 0012           (00216) FICTERS  DATA D'18'
09C8F 002C           (00217) FORFOES  DATA D'44'
09C90 0001           (00218) ONES     DATA D'1'
09C91 0000           (00219) ERRS     DATA D'0'
09C92 0000           (00220) SAVES    DATA D'0'
09C93 0000           (00221) ZZ1S     DATA D'0'
09C94 0020           (00222) THRSLDS  DATA D'15'
09C95 0005           (00223) BARS     DATA 5,5,3,1,1,1,2,2,5,5,6,1,2
09C96 0005
09C97 0003
09C98 0001
09C99 0001
09C9A 0001
```

```
09C9B 0002
09C9C 0002
09C9D 0005
09C9E 0005
09C9F 0006
09CA0 0001
09CA1 0002
09CA2 7059     (00224) TDENCS  DATA D'28761',D'7'          ENCODER TABLE FOR A (63,
09CA3 0007
               (00225) ;                                   1ST BCH CODE
09CA4 10EA     (00226)         DATA D'4330',D'1'           IN REVERSE ORDER
09CA5 0001
09CA6 21D4     (00227)         DATA D'8660',D'2'
09CA7 0002
09CA8 43A8     (00228)         DATA D'17320',D'4'
09CA9 0004
09CAA 7708     (00229)         DATA D'30472',D'7'
09CAB 0007
09CAC 1E48     (00230)         DATA D'7752',D'1'
09CAD 0001
09CAE 3C90     (00231)         DATA D'15504',D'2'
09CAF 0002
09CB0 7920     (00232)         DATA D'31008',D'4'
09CB1 0001
09CB2 0218     (00233)         DATA D'536',D'7'
09CB3 0007
09CB4 0431     (00234)         DATA D'1073',D'6'
09CB5 0006
09CB6 0863     (00235)         DATA D'2147',D'4'
09CB7 0001
09CB8 10C7     (00236)         DATA D'4295',D'0'
09CB9 0000
09CBA 218E     (00237)         DATA D'8590',D'0'
09CBB 0000
09CBC 431C     (00238)         DATA D'17180',D'0'
09CBD 0000
09CBE 7661     (00239)         DATA D'30305',D'7'
09CBF 0007
09CC0 1C9A     (00240)         DATA D'7322',D'1'
09CC1 0001
09CC2 3934     (00241)         DATA D'14644',D'2'
09CC3 0002
09CC4 7268     (00242)         DATA D'29288',D'4'
09CC5 0004
```

```
09CC6 1488      (00243)         DATA D'5256',D'7'
09CC7 0007
09CC8 2911      (00244)         DATA D'10513',D'6'
09CC9 0006
09CCA 5223      (00245)         DATA D'21027',D'4'
09CCB 0004
09CCC 541E      (00246)         DATA D'21534',D'7'
09CCD 0007
09CCE 5864      (00247)         DATA D'22628',D'1'
09CCF 0001
09CD0 4091      (00248)         DATA D'16529',D'5'
09CD1 0005
09CD2 717A      (00249)         DATA D'29050',D'5'
09CD3 0005
09CD4 12AC      (00250)         DATA D'4780',D'5'
09CD5 0005
09CD6 2559      (00251)         DATA D'9561',D'2'
09CD7 0002
09CD8 4AB2      (00252)         DATA D'19122',D'4'
09CD9 0004
09CDA 653C      (00253)         DATA D'25916',D'7'
09CDB 0007
09CDC 3A20      (00254)         DATA D'14880',D'1'
09CDD 0001
09CDE 7440      (00255)         DATA D'29760',D'2'
09CDF 0002
09CE0 18D9      (00256)         DATA D'6361',D'3'
09CE1 0003
09CE2 31B2      (00257)         DATA D'12722',D'6'
09CE3 0006
09CE4 6365      (00258)         DATA D'25445',D'4'
09CE5 0004
09CE6 3692      (00259)         DATA D'13970',D'7'
09CE7 0007
09CE8 6D25      (00260)         DATA D'27941',D'6'
09CE9 0006
09CEA 2A12      (00261)         DATA D'10770',D'3'
09CEB 0003
09CEC 5424      (00262)         DATA D'21540',D'6'
09CED 0006
09CEE 5810      (00263)         DATA D'22544',D'3'
09CEF 0003
09CF0 4079      (00264)         DATA D'16505',D'1'
09CF1 0001
```

```
09CF2 70AB      (00265)          DATA D'28843',D'5'
09CF3 0005
09CF4 110E      (00266)          DATA D'4366',D'5'
09CF5 0005
09CF6 2210      (00267)          DATA D'8744',D'2'
09CF7 0002
09CF8 443A      (00268)          DATA D'17466',D'4'
09CF9 0004
09CFA 782C      (00269)          DATA D'30764',D'7'                                  RESIDUE DUE TO FIRST INF
09CFB 0007
                (00270) ;                                                            ORMATION BIT
09CFC 0001      (00271) IRPSWS   DATA D'1',D'1',D'1'                                 POWER SUM TABLE IN REVER
09CFD 0001
09CFE 0001
                (00272) ;                                                            SE ORDER
09CFF 0002      (00273)          DATA D'2',D'8',D'32'
09D00 0008
09D01 0020
09D02 0004      (00274)          DATA D'4',D'3',D'48'
09D03 0003
09D04 0030
09D05 0008      (00275)          DATA D'8',D'24',D'40'
09D06 0018
09D07 0028
09D08 0010      (00276)          DATA D'16',D'5',D'60'
09D09 0005
09D0A 003C
09D0B 0020      (00277)          DATA D'32',D'40',D'34'
09D0C 0028
09D0D 0022
09D0E 0003      (00278)          DATA D'3',D'15',D'51'
09D0F 000F
09D10 0033
09D11 0006      (00279)          DATA D'6',D'59',D'11'
09D12 003B
09D13 000B
09D14 000C      (00280)          DATA D'12',D'17',D'47'
09D15 0011
09D16 002F
09D17 0018      (00281)          DATA D'24',D'14',D'25'
09D18 000E
09D19 0019
09D1A 0030      (00282)          DATA D'48',D'51',D'52'
09D1B 0033
```

```
09D1C 0023      (00284)        DATA D'35',D'18',D'46'
09D1D 0012
09D1E 002E
09D20 0005      (00283)        DATA D'5',D'22',D'57'
09D21 0016
09D22 0039
09D23 000D      (00285)        DATA D'13',D'54',D'4'
09D24 0036
09D25 0004
09D26 0014      (00286)        DATA D'20',D'58',D'6'
09D27 003A
09D28 0006
09D29 0028      (00287)        DATA D'40',D'25',D'5'
09D2A 0019
09D2B 0005
09D2C 0013      (00288)        DATA D'19',D'13',D'38'
09D2D 000D
09D2E 0026
09D2F 0026      (00289)        DATA D'38',D'43',D'53'
09D30 002B
09D31 0035
09D32 000F      (00290)        DATA D'15',D'24',D'14'
09D33 0018
09D34 000E
09D35 001E      (00291)        DATA D'30',D'62',D'9'
09D36 003E
09D37 0009
09D38 003C      (00292)        DATA D'60',D'57',D'44'
09D39 0039
09D3A 002C
09D3B 003B      (00293)        DATA D'59',D'1',D'58'
09D3C 0001
09D3D 003A
09D3E 0035      (00294)        DATA D'53',D'8',D'39'
09D3F 0008
09D40 0027
09D41 0029      (00295)        DATA D'41',D'3',D'21'
09D42 0003
09D43 0015
09D44 0011      (00296)        DATA D'17',D'24',D'62'
09D45 0018
09D46 003E
09D47 0022      (00297)        DATA D'34',D'5',D'33'
```

```
09D48 0005
09D49 0021
09D4A 0007      (00298)      DATA D'7',D'40',D'16'
09D4B 0028
09D4C 0010
09D4D 000E      (00299)      DATA D'14',D'15',D'24'
09D4E 000F
09D4F 0018
09D50 001C      (00300)      DATA D'28',D'53',D'20'
09D51 0035
09D52 0014
09D53 0038      (00301)      DATA D'56',D'17',D'30'
09D54 0011
09D55 001E
09D56 0033      (00302)      DATA D'51',D'14',D'17'
09D57 000E
09D58 0011
09D59 0025      (00303)      DATA D'37',D'51',D'56'
09D5A 0033
09D5B 0038
09D5C 0009      (00304)      DATA D'9',D'18',D'36'
09D5D 0012
09D5E 0024
09D5F 0012      (00305)      DATA D'18',D'22',D'54'
09D60 0016
09D61 0036
09D62 0024      (00306)      DATA D'36',D'54',D'45'
09D63 0036
09D64 002D
09D65 000B      (00307)      DATA D'11',D'58',D'26'
09D66 003A
09D67 001A
09D68 0016      (00308)      DATA D'22',D'25',D'23'
09D69 0019
09D6A 0017
09D6B 002C      (00309)      DATA D'44',D'13',D'61'
09D6C 000D
09D6D 003D
09D6E 001B      (00310)      DATA D'27',D'43',D'2'
09D6F 002B
09D70 0002
09D71 0036      (00311)      DATA D'54',D'23',D'3'
09D72 0017
09D73 0003
```

```
PAGE   121        PROG. CS005F.TXT

09D74 002F     (00312)       DATA D'47',D'62',D'45'
09D75 003E
09D76 002D
09D77 001D     (00313)       DATA D'29',D'57',D'19'
09D78 0039
09D79 0013
09D7A 003A     (00314)       DATA D'58',D'1',D'59'
09D7B 0001
09D7C 003B
09D7D 0037     (00315)       DATA D'55',D'8',D'7'
09D7E 0008
09D7F 0007
09D80 002D     (00316)       DATA D'45',D'3',D'37'
09D81 0003
09D82 0025
09D83 0019     (00317)       DATA D'25',D'24',D'22'
09D84 0018
09D85 0016
09D86 0032     (00318)       DATA D'50',D'5',D'29'
09D87 0005
09D88 001D
09D89 0027     (00319)       DATA D'39',D'40',D'50'
09D8A 0028
09D8B 0032
09D8C 000D     (00320)       DATA D'13',D'15',D'43'
09D8D 000F
09D8E 002B
09D8F 001A     (00321)       DATA D'26',D'59',D'31'
09D90 003B
09D91 001F
09D92 0034     (00322)       DATA D'52',D'17',D'49'
09D93 0011
09D94 0031
09D95 002B     (00323)       DATA D'43',D'14',D'8'
09D96 000E
09D97 0008
09D98 0015     (00324)       DATA D'21',D'51',D'12'
09D99 0033
09D9A 000C
09D9B 002A     (00325)       DATA D'42',D'18',D'10'
09D9C 0012
09D9D 000A
09D9E 0017     (00326)       DATA D'23',D'22',D'15'
09D9F 0016
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

7 OF 9
AD
A091663

```
09DA0 00[illegible]
09DA1 002E      (00327)          DATA D'46',D'54',D'41'
09DA2 0036
09DA3 0029
09DA4 001F      (00328)          DATA D'31',D'58',D'28'
09DA5 003A
09DA6 001C
09DA7 003E      (00329)          DATA D'62',D'25',D'18'
09DA8 0019
09DA9 0012
09DAA 003F      (00330)          DATA D'63',D'13',D'27'
09DAB 000D
09DAC 001B
09DAD 003D      (00331)          DATA D'61',D'43',D'55'
09DAE 002B
09DAF 0037
09DB0 0039      (00332)          DATA D'57',D'23',D'13'
09DB1 0017
09DB2 000D
09DB3 0031      (00333)          DATA D'49',D'62',D'42'
09DB4 003E
09DB5 002A
09DB6 0021      (00334)          DATA D'33',D'57',D'63'                    POWER SUMS S1,S3,S5 DUE
09DB7 0039
09DB8 003F
                (00335) ;                                                  TO FIRST INFORMATION BIT
09DB9 0001      (00336) TAPRCS   DATA D'1'                                 POWER TO RESIDUE CONVERS
                (00337) ;                                                  ION TABLE
09DBA 0002      (00338)          DATA D'2'                                 FOR A (63,45) BCH CODE
09DBB 0004      (00339)          DATA D'4'
09DBC 0008      (00340)          DATA D'8'
09DBD 0010      (00341)          DATA D'16'
09DBE 0020      (00342)          DATA D'32'
09DBF 0003      (00343)          DATA D'3'
09DC0 0006      (00344)          DATA D'6'
09DC1 000C      (00345)          DATA D'12'
09DC2 0018      (00346)          DATA D'24'
09DC3 0030      (00347)          DATA D'48'
09DC4 0023      (00348)          DATA D'35'
09DC5 0005      (00349)          DATA D'5'
09DC6 000A      (00350)          DATA D'10'
09DC7 0014      (00351)          DATA D'20'
09DC8 0028      (00352)          DATA D'40'
09DC9 0013      (00353)          DATA D'19'
```

```
09DCA 0026      (00354)      DATA D'38'
09DCB 000F      (00355)      DATA D'15'
09DCC 001E      (00356)      DATA D'30'
09DCD 003C      (00357)      DATA D'60'
09DCE 003B      (00358)      DATA D'59'
09DCF 0035      (00359)      DATA D'53'
09DD0 0029      (00360)      DATA D'41'
09DD1 0011      (00361)      DATA D'17'
09DD2 0022      (00362)      DATA D'34'
09DD3 0007      (00363)      DATA D'7'
09DD4 000E      (00364)      DATA D'14'
09DD5 001C      (00365)      DATA D'28'
09DD6 0038      (00366)      DATA D'56'
09DD7 0033      (00367)      DATA D'51'
09DD8 0025      (00368)      DATA D'37'
09DD9 0009      (00369)      DATA D'9'
09DDA 0012      (00370)      DATA D'18'
09DDB 0024      (00371)      DATA D'36'
09DDC 000B      (00372)      DATA D'11'
09DDD 0016      (00373)      DATA D'22'
09DDE 002C      (00374)      DATA D'44'
09DDF 001B      (00375)      DATA D'27'
09DE0 0036      (00376)      DATA D'54'
09DE1 002F      (00377)      DATA D'47'
09DE2 001D      (00378)      DATA D'29'
09DE3 003A      (00379)      DATA D'58'
09DE4 0037      (00380)      DATA D'55'
09DE5 002D      (00381)      DATA D'45'
09DE6 0019      (00382)      DATA D'25'
09DE7 0032      (00383)      DATA D'50'
09DE8 0027      (00384)      DATA D'39'
09DE9 000D      (00385)      DATA D'13'
09DEA 001A      (00386)      DATA D'26'
09DEB 0034      (00387)      DATA D'52'
09DEC 002B      (00388)      DATA D'43'
09DED 0015      (00389)      DATA D'21'
09DEE 002A      (00390)      DATA D'42'
09DEF 0017      (00391)      DATA D'23'
09DF0 002E      (00392)      DATA D'46'
09DF1 001F      (00393)      DATA D'31'
09DF2 003E      (00394)      DATA D'62'
09DF3 003F      (00395)      DATA D'63'
09DF4 003D      (00396)      DATA D'61'
09DF5 0039      (00397)      DATA D'57'
```

```
09DF6 0031    (00398)          DATA D'49'
09DF7 0021    (00399)          DATA D'33'
09DF8 0000    (00400) TRRPCS   DATA D'0'       DUMMY DATA MEMORY
09DF9 0000    (00401)          DATA D'0'       RESIDUE TO POWER CONVERS
              (00402) ;                        ION TABLE
09DFA 0001    (00403)          DATA D'1'       FOR A (63,45) BCH CODE
09DFB 0006    (00404)          DATA D'6'
09DFC 0002    (00405)          DATA D'2'
09DFD 000C    (00406)          DATA D'12'
09DFE 0007    (00407)          DATA D'7'
09DFF 001A    (00408)          DATA D'26'
09E00 0003    (00409)          DATA D'3'
09E01 0020    (00410)          DATA D'32'
09E02 000D    (00411)          DATA D'13'
09E03 0023    (00412)          DATA D'35'
09E04 0008    (00413)          DATA D'8'
09E05 0030    (00414)          DATA D'48'
09E06 001B    (00415)          DATA D'27'
09E07 0012    (00416)          DATA D'18'
09E08 0004    (00417)          DATA D'4'
09E09 0018    (00418)          DATA D'24'
09E0A 0021    (00419)          DATA D'33'
09E0B 0010    (00420)          DATA D'16'
09E0C 000E    (00421)          DATA D'14'
09E0D 0034    (00422)          DATA D'52'
09E0E 0024    (00423)          DATA D'36'
09E0F 0036    (00424)          DATA D'54'
09E10 0009    (00425)          DATA D'9'
09E11 002D    (00426)          DATA D'45'
09E12 0031    (00427)          DATA D'49'
09E13 0026    (00428)          DATA D'38'
09E14 001C    (00429)          DATA D'28'
09E15 0029    (00430)          DATA D'41'
09E16 0013    (00431)          DATA D'19'
09E17 0038    (00432)          DATA D'56'
09E18 0005    (00433)          DATA D'5'
09E19 003E    (00434)          DATA D'62'
09E1A 0019    (00435)          DATA D'25'
09E1B 000B    (00436)          DATA D'11'
09E1C 0022    (00437)          DATA D'34'
09E1D 001F    (00438)          DATA D'31'
09E1E 0011    (00439)          DATA D'17'
09E1F 002F    (00440)          DATA D'47'
09E20 000F    (00441)          DATA D'15'
```

PAGE 16: [illegible] CSPDSP.TXT

```
09F21 0017     (00442)          DATA D'23'
09F22 0035     (00443)          DATA D'53'
09F23 0033     (00444)          DATA D'51'
09F24 0025     (00445)          DATA D'37'
09F25 002C     (00446)          DATA D'44'
09F26 0037     (00447)          DATA D'55'
09F27 0028     (00448)          DATA D'40'
09F28 000A     (00449)          DATA D'10'
09F29 003D     (00450)          DATA D'61'
09F2A 002E     (00451)          DATA D'46'
09F2B 001E     (00452)          DATA D'30'
09F2C 0032     (00453)          DATA D'50'
09F2D 0016     (00454)          DATA D'22'
09F2E 0027     (00455)          DATA D'39'
09F2F 002B     (00456)          DATA D'43'
09F30 001D     (00457)          DATA D'29'
09F31 003C     (00458)          DATA D'60'
09F32 002A     (00459)          DATA D'42'
09F33 0015     (00460)          DATA D'21'
09F34 0014     (00461)          DATA D'20'
09F35 003B     (00462)          DATA D'59'
09F36 0039     (00463)          DATA D'57'
09F37 003A     (00464)          DATA D'58'
09F38 0000     (00465) TBAL1S   DATA D'0'                MULT. TABLE OF ALPA FOR
               (00466) ;                                 A (63,45)BCH CODE
09F39 0002     (00467)          DATA D'2'
09F3A 0004     (00468)          DATA D'4'
09F3B 0006     (00469)          DATA D'6'
09F3C 0008     (00470)          DATA D'8'
09F3D 000A     (00471)          DATA D'10'
09F3E 000C     (00472)          DATA D'12'
09F3F 000E     (00473)          DATA D'14'
09F40 0010     (00474)          DATA D'16'
09F41 0012     (00475)          DATA D'18'
09F42 0014     (00476)          DATA D'20'
09F43 0016     (00477)          DATA D'22'
09F44 0018     (00478)          DATA D'24'
09F45 001A     (00479)          DATA D'26'
09F46 001C     (00480)          DATA D'28'
09F47 001E     (00481)          DATA D'30'
09F48 0020     (00482)          DATA D'32'
09F49 0022     (00483)          DATA D'34'
09F4A 0024     (00484)          DATA D'36'
09F4B 0026     (00485)          DATA D'38'
```

```
09F4C 0028      (00486)      DATA D'40'
09F4D 002A      (00487)      DATA D'42'
09F4E 002C      (00488)      DATA D'44'
09F4F 002E      (00489)      DATA D'46'
09F50 0030      (00490)      DATA D'48'
09F51 0032      (00491)      DATA D'50'
09F52 0034      (00492)      DATA D'52'
09F53 0036      (00493)      DATA D'54'
09F54 0038      (00494)      DATA D'56'
09F55 003A      (00495)      DATA D'58'
09F56 003C      (00496)      DATA D'60'
09F57 003E      (00497)      DATA D'62'
09F58 0003      (00498)      DATA D'3'
09F59 0001      (00499)      DATA D'1'
09F5A 0007      (00500)      DATA D'7'
09F5B 0005      (00501)      DATA D'5'
09F5C 000B      (00502)      DATA D'11'
09F5D 0009      (00503)      DATA D'9'
09F5E 000F      (00504)      DATA D'15'
09F5F 000D      (00505)      DATA D'13'
09F60 0013      (00506)      DATA D'19'
09F61 0011      (00507)      DATA D'17'
09F62 0017      (00508)      DATA D'23'
09F63 0015      (00509)      DATA D'21'
09F64 001B      (00510)      DATA D'27'
09F65 0019      (00511)      DATA D'25'
09F66 001F      (00512)      DATA D'31'
09F67 001D      (00513)      DATA D'29'
09F68 0023      (00514)      DATA D'35'
09F69 0021      (00515)      DATA D'33'
09F6A 0027      (00516)      DATA D'39'
09F6B 0025      (00517)      DATA D'37'
09F6C 002B      (00518)      DATA D'43'
09F6D 0029      (00519)      DATA D'41'
09F6E 002F      (00520)      DATA D'47'
09F6F 002D      (00521)      DATA D'45'
09F70 0033      (00522)      DATA D'51'
09F71 0031      (00523)      DATA D'49'
09F72 0037      (00524)      DATA D'55'
09F73 0035      (00525)      DATA D'53'
09F74 003B      (00526)      DATA D'59'
09F75 0039      (00527)      DATA D'57'
09F76 003F      (00528)      DATA D'63'
09F77 003D      (00529)      DATA D'61'
```

MULT. TABLE OF ALPA**2

```
09E78 0000     (00530) TBAL2S  DATA D'0'
09E79 0004     (00531)         DATA D'4'
09E7A 0008     (00532)         DATA D'8'
09E7B 000C     (00533)         DATA D'12'
09E7C 0010     (00534)         DATA D'16'
09E7D 0014     (00535)         DATA D'20'
09E7E 0018     (00536)         DATA D'24'
09E7F 001C     (00537)         DATA D'28'
09E80 0020     (00538)         DATA D'32'
09E81 0024     (00539)         DATA D'36'
09E82 0028     (00540)         DATA D'40'
09E83 002C     (00541)         DATA D'44'
09E84 0030     (00542)         DATA D'48'
09E85 0034     (00543)         DATA D'52'
09E86 0038     (00544)         DATA D'56'
09E87 003C     (00545)         DATA D'60'
09E88 0003     (00546)         DATA D'3'
09E89 0007     (00547)         DATA D'7'
09E8A 000B     (00548)         DATA D'11'
09E8B 000F     (00549)         DATA D'15'
09E8C 0013     (00550)         DATA D'19'
09E8D 0017     (00551)         DATA D'23'
09E8E 001B     (00552)         DATA D'27'
09E8F 001F     (00553)         DATA D'31'
09E90 0023     (00554)         DATA D'35'
09E91 0027     (00555)         DATA D'39'
09E92 002B     (00556)         DATA D'43'
09E93 002F     (00557)         DATA D'47'
09E94 0033     (00558)         DATA D'51'
09E95 0037     (00559)         DATA D'55'
09E96 003B     (00560)         DATA D'59'
09E97 003F     (00561)         DATA D'63'
09E98 0006     (00562)         DATA D'6'
09E99 0002     (00563)         DATA D'2'
09E9A 000E     (00564)         DATA D'14'
09E9B 000A     (00565)         DATA D'10'
09E9C 0016     (00566)         DATA D'22'
09E9D 0012     (00567)         DATA D'18'
09E9E 001E     (00568)         DATA D'30'
09E9F 001A     (00569)         DATA D'26'
09EA0 0026     (00570)         DATA D'38'
09EA1 0022     (00571)         DATA D'34'
09EA2 002E     (00572)         DATA D'46'
09EA3 002A     (00573)         DATA D'42'
```

```
09FA4 0036      (00574)          DATA D'54'
09FA5 0032      (00575)          DATA D'50'
09FA6 003E      (00576)          DATA D'62'
09FA7 003A      (00577)          DATA D'58'
09FA8 0005      (00578)          DATA D'5'
09FA9 0001      (00579)          DATA D'1'
09FAA 000D      (00580)          DATA D'13'
09FAB 0009      (00581)          DATA D'9'
09FAC 0015      (00582)          DATA D'21'
09FAD 0011      (00583)          DATA D'17'
09FAE 001D      (00584)          DATA D'29'
09FAF 0019      (00585)          DATA D'25'
09FB0 0025      (00586)          DATA D'37'
09FB1 0021      (00587)          DATA D'33'
09FB2 002D      (00588)          DATA D'45'
09FB3 0029      (00589)          DATA D'41'
09FB4 0035      (00590)          DATA D'53'
09FB5 0031      (00591)          DATA D'49'
09FB6 003D      (00592)          DATA D'61'
09FB7 0039      (00593)          DATA D'57'
09FB8 0000      (00594) TBAL3S   DATA D'0'                              MULT. TABLE OF ALPA**3
09FB9 0008      (00595)          DATA D'8'
09FBA 0010      (00596)          DATA D'16'
09FBB 0018      (00597)          DATA D'24'
09FBC 0020      (00598)          DATA D'32'
09FBD 0028      (00599)          DATA D'40'
09FBE 0030      (00600)          DATA D'48'
09FBF 0038      (00601)          DATA D'56'
09FC0 0003      (00602)          DATA D'3'
09FC1 000B      (00603)          DATA D'11'
09FC2 0013      (00604)          DATA D'19'
09FC3 001B      (00605)          DATA D'27'
09FC4 0023      (00606)          DATA D'35'
09FC5 002B      (00607)          DATA D'43'
09FC6 0033      (00608)          DATA D'51'
09FC7 003B      (00609)          DATA D'59'
09FC8 0006      (00610)          DATA D'6'
09FC9 000E      (00611)          DATA D'14'
09FCA 0016      (00612)          DATA D'22'
09FCB 001E      (00613)          DATA D'30'
09FCC 0026      (00614)          DATA D'38'
09FCD 002E      (00615)          DATA D'46'
09FCE 0036      (00616)          DATA D'54'
09FCF 003E      (00617)          DATA D'62'
```

```
09FD0 0005     (00618)          DATA D'5'
09FD1 000D     (00619)          DATA D'13'
09FD2 0015     (00620)          DATA D'21'
09FD3 001D     (00621)          DATA D'29'
09FD4 0025     (00622)          DATA D'37'
09FD5 002D     (00623)          DATA D'45'
09FD6 0035     (00624)          DATA D'53'
09FD7 003D     (00625)          DATA D'61'
09FD8 000C     (00626)          DATA D'12'
09FD9 0004     (00627)          DATA D'4'
09FDA 001C     (00628)          DATA D'28'
09FDB 0014     (00629)          DATA D'20'
09FDC 002C     (00630)          DATA D'44'
09FDD 0024     (00631)          DATA D'36'
09FDE 003C     (00632)          DATA D'60'
09FDF 0034     (00633)          DATA D'52'
09FE0 000F     (00634)          DATA D'15'
09FE1 0007     (00635)          DATA D'7'
09FE2 001F     (00636)          DATA D'31'
09FE3 0017     (00637)          DATA D'23'
09FE4 002F     (00638)          DATA D'47'
09FE5 0027     (00639)          DATA D'39'
09FE6 003F     (00640)          DATA D'63'
09FE7 0037     (00641)          DATA D'55'
09FE8 000A     (00642)          DATA D'10'
09FE9 0002     (00643)          DATA D'2'
09FEA 001A     (00644)          DATA D'26'
09FEB 0012     (00645)          DATA D'18'
09FEC 002A     (00646)          DATA D'42'
09FED 0022     (00647)          DATA D'34'
09FEE 003A     (00648)          DATA D'58'
09FEF 0032     (00649)          DATA D'50'
09FF0 0009     (00650)          DATA D'9'
09FF1 0001     (00651)          DATA D'1'
09FF2 0019     (00652)          DATA D'25'
09FF3 0011     (00653)          DATA D'17'
09FF4 0029     (00654)          DATA D'41'
09FF5 0021     (00655)          DATA D'33'
09FF6 0039     (00656)          DATA D'57'
09FF7 0031     (00657)          DATA D'49'
09FF8 0000     (00658) NESTS    DATA 4D'0'                 NCH
 ...
09FFC 0000     (00659) NESS     DATA 1D'0'                 NESS=2(FRE, # FOR BLOCK
 ...
```

```
                  (00660) ;                                                     1),NESS-1(ERR, # FOR BL,
09F00 0000        (00661) SISPS   DATA D'0'                                     NESS(ERR, # FOR BLOCK 3)
09F01 0000        (00662) REGS    DATA 8D'0'                                    DATA MEM, FOR REGISTER R
  ...
                  (00663) ;                                                     1-R7
                  (00664) *
                  (00665) *  END OF DATA MEMORY
                  (00666)         EJECT
```

```
                     (00667) *
                     (00668) *
                     (00669) *
                     (00670) ********************************************
                     (00671) ;  CSPU PROGRAM STARTS HERE
                     (00672) ********************************************
09F08 0800           (00673)         EVEN
09F0A 9F1C           (00674) GTIOSTS DATA IODSPTS
09F0B AC5E           (00675) GTIGENS DATA GTIDLSS+2
09F0C 00000000       (00676) R1SAV   DATA 1E'0,0'
09F0E 00000000       (00677) R2SAV   DATA 1E'0,0'
09F10 00000000       (00678) R3SAV   DATA 1E'0,0'
09F12 00000000       (00679) R4SAV   DATA 1E'0,0'
09F14 00000000       (00680) R5SAV   DATA 1E'0,0'
09F16 00000000       (00681) R6SAV   DATA 1E'0,0'
09F18 00000000       (00682) R7SAV   DATA 1E'0,0'
                     (00683) *
                     (00684) ;
                     (00685) ;
                     (00686) ;  DEFINITION FOR CSW AND PATCH TO AVOID 16K INTERRUPT PROBLEM
                     (00687) ;
                     (00688)         EVEN
09F1A 9F1C1014       (00689) CSWSD24 CSW CSSSTDB,IODSPTS                CSW OF TRANSMITTER
09F1C 2000           (00690) IODSPTS NOP                                ***FOR DEBUG ONLY***
09F1D CC000570       (00691)         MOVZM INFULL                       INITIALIZE
09F1E CC000571       (00692)         MOVZM OUTFUL
09F21 0800           (00693) IODSPTS EVEN
09F22 DA000575       (00694) TSTS0   SMBCL 0,XGO                        TRANSMIT REQUEST IS TIED
09F24 80009FA2       (00695)         JMP XMITS                          G3=1 SO EXECUTE TRANSMIT
                     (00696) ;                                          TER("XMIT")
09F26 DE000580       (00697)         SMBSL 0,APDNFLS
09F28 2005           (00698)         NOP #L+6
09F29 0800           (00699)         EVEN
09F2A CC000580       (00700)         MOVZM APDNFLS
09F2C 0E70           (00701)         RET
09F2D 0800           (00702)         EVEN
09F2E 80009F3A       (00703)         JMP TSTS1                          WHEN WE COME BACK TEST 1
09F30 EROFFFCE       (00704) TRANSRA MOVEM CLRSG3,SYSFLGS               ***FOR TRAN TIMING ONLY
                     (00705) ;                                          ***
09F32 DE000580       (00706)         SMBSL 0,APDNFLS                    HAS APDONE HAPPENED?
09F33 2005           (00707)         NOP #L+6
09F35 0800           (00708)         EVEN
09F36 CC000580       (00709)         MOVZM APDNFLS
09F38 0E70           (00710)         RET
```

```
09F39 0800      (00711)          EVEN
09F3A DE00056C  (00712) TSTS1    SMRSL 0,IDPS                           DO WE HAVE NEW A/D SPEEC
                (00713) ;                                               H TO PROCESS?
09F3C 80009EAA  (00714)          JMP MOVES                              YES
09F3E DE000580  (00715)          SMRSL 0,APDNFLS
09F40 2005      (00716)          ROP #L+6
09F41 0800      (00717)          EVEN
09F42 CC000580  (00718)          MOVZM APDNFLS
09F44 0E70      (00719)          RET
09F45 0800      (00720)          EVEN
09F46 80009E52  (00721)          JMP TSTS2
09F48 E8020570  (00722) MOVESRA  MOVLM 1,INFULL                         SAY INPUT (TO AP) FULL
09F4A DE000580  (00723) MOVSRA1  SMRSL 0,APDNFLS                        HAS APDONE HAPPENED?
09F4C 2005      (00724)          ROP #L+6
09F4D 0800      (00725)          EVEN
09F4E CC000580  (00726)          MOVZM APDNFLS
09F50 0E70      (00727)          RET
09F51 0800      (00728)          EVEN
09F52 FA000574  (00729) TSTS2    SMRCL 0,RGO                            RECEIVE REQUEST IS TIED
                (00730) ;                                               TO RGO
09F54 8000A496  (00731)          JMP XRECVS                             GO=1 SO EXECUTE RECEIVER
                (00732) ;                                               ("XRECV")
09F56 DE000580  (00733)          SMRSL 0,APDNFLS
09F58 2005      (00734)          ROP #L+6
09F59 0800      (00735)          EVEN
09F5A CC000580  (00736)          MOVZM APDNFLS
09F5C 0E70      (00737)          RET
09F5D 0800      (00738)          EVEN
09F5E 80009E6E  (00739)          JMP TSTS3                              WHEN WE COME BACK TEST G
                (00740) ;                                               3!
                (00741) RECVSRA  EVEN
09F60 E809EECE  (00742)          MOVLM CLRSGO,SYSFLGS                   ***FOR RECV TIMING ONLY
                (00743) ;                                               ***
09F62 9031FFFF  (00744)          MOVIR R1,-1                            -1 IMPLIES THAT RCVR IS
                (00745) ;                                               OFF
09F64 E0100581  (00746)          MOVRM R1,RCVOFF                        WHICH LETS "VMOV1" STALL
09F66 DE000580  (00747)          SMRSL 0,APDNFLS                        HAS APDONE HAPPENED?
09F68 2005      (00748)          ROP #L+6
09F69 0800      (00749)          EVEN
09F6A CC000580  (00750)          MOVZM APDNFLS
09F6C 0E70      (00751)          RET
09F6D 0800      (00752)          EVEN
09F6E DE00056A  (00753) TSTS3    SMRSL 0,OUTS                           D/A READY FOR MORE OUTPU
                (00754) ;                                               T?
```

```
09F70 8000A6E8 (00755)          JMP LDADES               YES
09F72 FE000580 (00756)          SMBSL 0,APDNFLS
09F74 2005     (00757)          HOP #L+6
09F75 0800     (00758)          EVEN
09F76 CC000580 (00759)          MOVZM APDNFLS
09F78 0F70     (00760)          RET
09F79 0800     (00761)          EVEN
09F7A 80009F22 (00762)          JMP TSTS0
09F7C CC000571 (00763) LOPUSRA  MOVZM OUTFUL             SAY AP OUT BUFF EMPTY
09F7E DE000580 (00764) LOPSRA1  SMBSL 0,APDNFLS          HAS APDONE HAPPENED?
09F80 2005     (00765)          HOP #L+6
09F81 0800     (00766)          EVEN
09F82 CC000580 (00767)          MOVZM APDNFLS
09F84 0F70     (00768)          RET
09F85 0800     (00769)          EVEN
09F86 80009F22 (00770)          JMP TSTS0
               (00771) ;
               (00772)          EJECT
```

PAGE 25: PROG. CSPDSP.TXT

```
                  (00773) ;
                  (00774) ;
                  (00775) ;  EXECUTE INSTRUCTIONS
                  (00776) ;
                  (00777)          EVEN
09E88 FC209C8E    (00778) CNTR1S   ADD4R R2,EIGTENS              INCREMENT COUNTER R2 BY
                                                                 18
                  (00779) ;
                  (00780)          EVEN
                  (00781) ;
                  (00782) ;  ERROR INDICATORS
                  (00783) ;
                  (00784) *
09E8A D2009C91    (00785) ERR1S    SMB 0,ERRS                    'ERROR CONDITION FOR TES
                  (00786) ;                                      T 1
09E8C 80009E92    (00787)          JMP ERRSRET                   'ERROR:INPUT SPEECH BUFF
                  (00788) ;                                      ER IS INPUT TWICE
09E8E D2109C91    (00789) ERR2S    SMB 1,ERRS                    'ERROR CONDITION FOR TES
                  (00790) ;                                      T 2
09E90 80009E92    (00791)          JMP ERRSRET                   'ERROR:XMITTER DATA BUFF
                  (00792) ;                                      ER IS OUTPUT TWICE
09E92 E8000573    (00793) ERRSRET  MOVEM 0,INITS                 'RESET INIT FOR INITIALI
                  (00794) ;                                      ZATION AGAIN
09E94 80009E30    (00795)          JMP TRANSEA                   'GO TO XMITTER RETURN
                  (00796) ;
                  (00797) ;;
                  (00798) ;
09E96 D2209C91    (00799) ERR3S    SMB 2,ERRS                    'ERROR CONDITION FOR TES
                  (00800) ;                                      T 3
09E98 80009E9E    (00801)          JMP ERRSREF                   'ERROR:RECV DATA BUFFER
                  (00802) ;                                      IS UNLOADED TWICE
09E9A D2309C91    (00803) ERR4S    SMB 3,ERRS                    'ERROR CODITION FOR TEST
09E9C 80009E9E    (00804)          JMP ERRSREF                   'ERROR:RECV SPEECH BUFFE
                  (00805) ;                                      R IS UNLOADED TWICE
09E9E E8020573    (00806) ERRSREF  MOVEM 1,INITS                 'CLEAR INIT
09EA0 80009E60    (00807)          JMP RECVSEA                   'RETURN
                  (00808)          EJECT
```

```
                    (00809) *
                    (00810) *
                    (00811) *
                    (00812) *
                    (00813) ***********************************************
                    (00814) *
                    (00815) * DOUBLE:XMIT.TXT
                    (00816) *
                    (00817) * TO UNLOAD SPEECH SAMPLES FROM DOUBLE BUFFER
                    (00818) *
                    (00819) ***********************************************
                    (00820) *
                    (00821)         EVEN
09FA2 CC000575 (00822) XMITS   MOVZM,XGO                       CLEAR XMIT GOO FLG
09FA4 FR4FFFCF (00823)         MOVLM SETSG1,SYSFLGS
09FA6 FR020566 (00824)         MOVLM 1,ANAS                     ;RESET THE ANALYZER READ
                    (00825) ;                                   Y FLAG
09FA8 80009FFR (00826)         JMP SRFUFS
                    (00827)         EJECT
```

```
                       (00828) ***** MOVE SPEECH DATA FROM NIN1,NIN2 TO INPB
09FAA EA000570 (00829) MOVES    SMPCL 0,INFULL
09FAC 80009F4A (00830)          JMP MOVSRA1                       WAIT TILLAP HAS TAKEN LA
               (00831) ;                                          ST ONE
09FAE F8A1FFCF (00832)          MOVLM SETSC3,SYSFLGS
09FB0 F8020560 (00833)          MOVLM 1,INPS                      CLEAR A/D INTERRUPT FLAG
09FB2 02A0     (00834)          CCS 2                             FROM SRB=1...
09FB3 0660     (00835)          SCS 3                             TO SRB=2
09FB4 0200     (00836)          CCS 5                             FROM MRB=2...
09FB5 06C0     (00837)          SCS 4                             TO MRB=1
09FB6 9040003E (00838)          MOVIR R4,BLTH1-1                  NUMBER OF FULL WORDS/BLO
               (00839) ;                                          CK
09FB8 F200058E (00840)          CMPMZ IFLGS                       'IS IFLG EQ 0?
09FBA 80109FD0 (00841)          JMP FLRS,EQ                       'GOTO FLR IF IT IS EQ
09FBC 90200F1E (00842)          MOVIR R2,NIN2S-2                  'PICK UP DATA FROM NIN2
09FBE D728AE80 (00843) UNLS     RMOVEL R2,R4,INPBS                'MOVE BUFFER AS 2 BLOCKS
09FC0 EE000580 (00844)          SMPSL 0,APDNFLS                   HAS APDONE HAPPENED?
09FC2 2005     (00845)          LOP #L+6
09FC3 0800     (00846)          EVEN
09FC4 CC000580 (00847)          MOVZM APDNFLS
09FC6 0F70     (00848)          RET
09FC7 0800     (00849)          EVEN
09FC8 90200F9A (00850)          MOVIR R2,NIN2S-2+BLTH12
09FCA 9040003C (00851)          MOVIR R4,BLTH1-2
09FCC D7238DFC (00852)          RMOVEL R2,R4,INPBS+BLTH12
09FCE 80009FF2 (00853)          JMP FLR1S
09FD0 90200D28 (00854) FLRS     MOVIR R2,NIN1S-2
09FD2 D728BD80 (00855)          RMOVEL R2,R4,INPBS
09FD4 DE000580 (00856)          SMPSL 0,APDNFLS                   HAS APDONE HAPPENED?
09FD6 2005     (00857)          LOP #L+6
09FD7 0800     (00858)          EVEN
09FD8 CC000580 (00859)          MOVZM APDNFLS
09FDA 0F70     (00860)          RET
09FDB 0800     (00861)          EVEN
09FDC 90200DA1 (00862)          MOVIR R2,NIN1S-2+BLTH12
09FDE 9040003C (00863)          MOVIR R4,BLTH1-2
09FE0 D728B0FC (00864)          RMOVEL R2,R4,INPBS+BLTH12
09FE2 02A0     (00865) FLR1S    CCS 3                             FROM SRB=2...
09FE3 06A0     (00866)          SCS 2                             TO SRB=1
09FE4 F80FFFCF (00867)          MOVLM CLRSC3,SYSFLGS
09FE6 80009F4A (00868)          JMP MOVESRA                       BACK TO DISPATCH
               (00869)          EJECT
```

```
                   (00870) ***** SERIALIZE SIDEBAND AND MAINBAND PARAMETERS
                   (00871) SERDES  EVEN
09FF8 0200         (00872)         CCS 3                       FROM SRB=2...
09FF9 0EA0         (00873)         SCS 2                       TO SRB=1
09FFA 0200         (00874)         CCS 5                       FROM MRB=2...
09FFB 06C0         (00875)         SCS 4                       TO MRB=1
09FFC DE000580     (00876)         SMBSE 0,APDOFLS             HAS APDONE HAPPENED?
09FFE 2021         (00877)         HOP *L+34                   NO, HOP TO EVEN LOC AFTE
                   (00878) ;                                   R RET
09FFF 0800         (00879)         EVEN
09FF0 CC000560     (00880)         MOVZ* APDOFLS
09FF2 E1109F0C     (00881)         STORE R1,R1SAV
09FF4 E1209F0E     (00882)         STORE R2,R2SAV
09FF6 E1309F10     (00883)         STORE R3,R3SAV
09FF8 E1409F12     (00884)         STORE R4,R4SAV
09FFA E1509F14     (00885)         STORE R5,R5SAV
09FFC E1609F16     (00886)         STORE R6,R6SAV
09FFE E1709F18     (00887)         STORE R7,R7SAV
0A000 0E70         (00888)         RET
0A001 0800         (00889)         EVEN
0A002 A6109F0C     (00890)         LOAD R1,R1SAV
0A004 A6209F0E     (00891)         LOAD R2,R2SAV
0A006 A6309F10     (00892)         LOAD R3,R3SAV
0A008 A6409F12     (00893)         LOAD R4,R4SAV
0A00A A6509F14     (00894)         LOAD R5,R5SAV
0A00C A6609F16     (00895)         LOAD R6,R6SAV
0A00E A6709F18     (00896)         LOAD R7,R7SAV
0A010 90209202     (00897)         MOVIR R2,ZROS-1
0A012 9040003D     (00898)         MOVIR R4,LSEN1-1
0A014 D728RC0E     (00899)         RMOVEL R2,R4,SENS           TRANSFER 3 BLOCKS OF 0'S
0A016 90209202     (00900)         MOVIR R2,ZROS-1
0A018 9040003D     (00901)         MOVIR R4,LSEN1-1
0A01A D728RCRA     (00902)         RMOVEL R2,R4,SENS+2*LSEN1
0A01C 90209202     (00903)         MOVIR R2,ZROS-1
0A01E 9040003C     (00904)         MOVIR R4,LSEN1-2
0A020 D728RD06     (00905)         RMOVEL R2,R4,SENS+4*LSEN1
                   (00906) *
                   (00907)         EVEN
0A022 DE000580     (00908)         SMBSE 0,APDOFLS             HAS APDONE HAPPENED?
0A024 2021         (00909)         HOP *L+34                   NO, HOP TO EVEN LOC AFTE
                   (00910) ;                                   R RET
0A025 0800         (00911)         EVEN
0A026 CC000580     (00912)         MOVZ* APDOFLS
0A028 E1109F0C     (00913)         STORE R1,R1SAV
```

```
0A02A F1209F0E (00913)          STORE R2,R2SAV
0A02C F1309F10 (00915)          STORE R3,R3SAV
0A02E F1409F12 (00916)          STORE R4,R4SAV
0A030 F1509F14 (00917)          STORE R5,R5SAV
0A032 F1609F16 (00918)          STORE R6,R6SAV
0A034 F1709F18 (00919)          STORE R7,R7SAV
0A036 0E70     (00920)          RET
0A037 0800     (00921)          EVEN
0A038 A6109F0C (00922)          LOAD R1,R1SAV
0A03A A6209F0E (00923)          LOAD R2,R2SAV
0A03C A6309F10 (00924)          LOAD R3,R3SAV
0A03E A6409F12 (00925)          LOAD R4,R4SAV
0A040 A6509F14 (00926)          LOAD R5,R5SAV
0A042 A6609F16 (00927)          LOAD R6,R6SAV
0A044 A6709F18 (00928)          LOAD R7,R7SAV
               (00929) ;
0A046 00200001 (00930)          MOVIR R2,1                  SKIP SYNC BIT
0A048 0E40     (00931)          CLR R4
0A049 0800     (00932)          EVEN
0A04A F0509C8C (00933)          MOVMR R5,PARMSM2
0A04C F06B8A00 (00934) LOOPS    MOVMR R6,OPBS(R4)           STORE THE PARAMETER IN R
               (00935) ;                                    6
0A04E DE000580 (00936)          SMBSL 0,APDNFLS             HAS APDONE HAPPENED?
0A050 2021     (00937)          HOP $L+34                   NO, HOP TO EVEN LOC AFTE
               (00938) ;                                    R RET
0A051 0800     (00939)          EVEN
0A052 CC000580 (00940)          MOVZM APDNFLS
0A054 F1109F0C (00941)          STORE R1,R1SAV
0A056 F1209F0E (00942)          STORE R2,R2SAV
0A058 F1309F10 (00943)          STORE R3,R3SAV
0A05A F1409F12 (00944)          STORE R4,R4SAV
0A05C F1509F14 (00945)          STORE R5,R5SAV
0A05E F1609F16 (00946)          STORE R6,R6SAV
0A060 F1709F18 (00947)          STORE R7,R7SAV
0A062 0E70     (00948)          RET
0A063 0800     (00949)          EVEN
0A064 A6109F0C (00950)          LOAD R1,R1SAV
0A066 A6209F0E (00951)          LOAD R2,R2SAV
0A068 A6309F10 (00952)          LOAD R3,R3SAV
0A06A A6409F12 (00953)          LOAD R4,R4SAV
0A06C A6509F14 (00954)          LOAD R5,R5SAV
0A06E A6609F16 (00955)          LOAD R6,R6SAV
0A070 A6709F18 (00956)          LOAD R7,R7SAV
0A072 808A9CA0 (00957)          JOP WITDXIS(R5)
```

```
                  (00958) *
                  (00959) *
                  (00960)         EVEN
0A074 2A5C        (00961) BIT5S   SRBCL 5,R6          IS PARM BUFFER BIT 5 CLE
                  (00962) ;                           AR?
0A075 D204BC0F    (00963)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00964) ;                           OF SEN ARRAY TO BE A 1
0A077 2621        (00965)         INCR R2,1           INCREMENT THE COUNTER R2
                  (00966) *
                  (00967)         EVEN
0A078 2A4C        (00968) BIT4S   SRBCL 4,R6          IS PARM BUFFER BIT 4 CLE
                  (00969) ;                           AR?
0A079 D204BC0F    (00970)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00971) ;                           OF SEN ARRAY TO BE A 1
0A07B 2621        (00972)         INCR R2,1           INCREMENT THE COUNTER R2
                  (00973) *
                  (00974)         EVEN
0A07C 2A3C        (00975) BIT3S   SRBCL 3,R6          IS PARM BUFFER BIT 3 CLE
                  (00976) ;                           AR?
0A07D D204BC0F    (00977)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00978) ;                           OF SEN ARRAY TO BE A 1
0A07F 2621        (00979)         INCR R2,1           INCREMENT THE COUNTER R2
                  (00980) *
                  (00981)         EVEN
0A080 2A2C        (00982) BIT2S   SRBCL 2,R6          IS PARM BUFFER BIT 2 CLE
                  (00983) ;                           AR?
0A081 D204BC0F    (00984)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00985) ;                           OF SEN ARRAY TO BE A 1
0A083 2621        (00986)         INCR R2,1           INCREMENT THE COUNTER R2
                  (00987) *
                  (00988)         EVEN
0A084 2A1C        (00989) BIT1S   SRBCL 1,R6          IS PARM BUFFER BIT 1 CLE
                  (00990) ;                           AR?
0A085 D204BC0F    (00991)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00992) ;                           OF SEN ARRAY TO BE A 1
0A087 2621        (00993)         INCR R2,1           INCREMENT THE COUNTER R2
                  (00994) *
                  (00995)         EVEN
0A088 2A0C        (00996) BIT0S   SRBCL 0,R6          IS PARM BUFFER BIT 0 CLE
                  (00997) ;                           AR?
0A089 D204BC0F    (00998)         SBB 0,SENS(R2)      OTHERWISE, SET LOCATION
                  (00999) ;                           OF SEN ARRAY TO BE A 1
0A08B 2621        (01000)         INCR R2,1           INCREMENT THE COUNTER R2
                  (01001)         EVEN
```

```
                    (01002) *
0A08C 2631          (01003)        INCR R3,1                              INCREMENT THE PARAMETER
                    (01004) ;                                             COUNTER
0A08D 0800          (01005)        EVEN
0A08E 8E50A04C      (01006)        DEJP R5,LOOPS                          DECREMENT R5,JUMP IF R5
                    (01007) ;                                             IS +VE
                    (01008) *
                    (01009) *
                    (01010) ;  SERIALIZE THE VERY LAST PARAMETER(I.E. OTG)
                    (01011) ;  AFTER INSERTING 18 ZEROES
                    (01012) *
                    (01013)        EVEN
0A090 1E0005B0      (01014)        SMOSE 0,APDOELS                        HAS APDONE HAPPENED?
0A092 2621          (01015)        HOP $L+31                              NO, HOP TO EVEN LOC AFTE
                    (01016) ;                                             R RET
0A093 0800          (01017)        EVEN
0A094 CC0005B0      (01018)        MOVZW APDOELS
0A096 F1109F0C      (01019)        STORE R1,R1SAV
0A098 F1209F0E      (01020)        STORE R2,R2SAV
0A09A F1309F10      (01021)        STORE R3,R3SAV
0A09C F1409F12      (01022)        STORE R4,R4SAV
0A09E F1509F14      (01023)        STORE R5,R5SAV
0A0A0 F1609F16      (01024)        STORE R6,R6SAV
0A0A2 F1709F18      (01025)        STORE R7,R7SAV
0A0A4 0F70          (01026)        RET
0A0A5 0800          (01027)        EVEN
0A0A6 A6109F0C      (01028)        LOAD R1,R1SAV
0A0A8 A6209F0E      (01029)        LOAD R2,R2SAV
0A0AA A6309F10      (01030)        LOAD R3,R3SAV
0A0AC A6409F12      (01031)        LOAD R4,R4SAV
0A0AE A6509F14      (01032)        LOAD R5,R5SAV
0A0B0 A6609F16      (01033)        LOAD R6,R6SAV
0A0B2 A6709F18      (01034)        LOAD R7,R7SAV
0A0B4 FC209CBE      (01035)        ADDMR R2,EIGTENS                       SKIP 18 LOCATIONS IN SEN
0A0B6 DA1BBA00      (01036)        SMBCL 1,OPBS(R4)                       IS BIT 1 OF OTG SET?SKIP
0A0B8 D204BC0E      (01037)        SMP 0,SENS(R2)                         INCREMENT SEN ARRAY IF I
                    (01038) ;                                             T IS NOT
0A0BA 2621          (01039)        INCR R2,1                              INCREMENT COUNTER R2
0A0BB 0800          (01040)        EVEN
0A0BC DA0BBA00      (01041)        SKBCL 0,OPBS(R4)                       CHECK IF BIT 0 OF OTG SE
                    (01042) ;                                             T? SKIP IF NOT
0A0BE D204BC0E      (01043)        SMP 0,SENS(R2)                         INCREMENT SEN ARRAY IF I
                    (01044) ;                                             T IS NOT
0A0C0 2621          (01045)        INCR R2,1                              INCREMENT COUNTER R2
```

```
0A0C1 0800      (01046)          EVEN
                (01047) *
                (01048) *
                (01049) *
                (01050) ;  SERIALIZE THE QUANTIZED DCT COEFS
                (01051) ;  AND ASSUMING HORIZONTAL CODING
                (01052) *
0A0C2 F0500C0F  (01053)          MOVPR R5,FORFOUR                     MOVE FORTY FOUR IN R5
0A0C4 2752      (01054)          DECR R5,2
0A0C5 0800      (01055)          EVEN
0A0C6 F070R00F  (01056)          MOVRP R7,IBITS                       SAVE THE FIRST IBIT VALU
                (01057) ;                                              E
0A0C8 2771      (01058)          DECR R7,1                            DECREMENT R7 BY 1
0A0C9 0800      (01059)          EVEN
0A0CA 3A71      (01060)          LLS R7,1                             SHIFT LEFT BY 1
0A0CB 0800      (01061)          EVEN
                (01062) ***********************************************************
                (01063) *  SPECIAL CODE TO CORRECT FOR IBIT=0 CASE 5/16/80
0A0CC 0270      (01064)          TEST R7                              WHEN IBIT=0,R7<0 AND A I
                (01065) ;                                              S NG!
0A0CD 0800      (01066)          EVEN                                 SO TEST & JUMP AROUND SE
                (01067) ;                                              RIALIZATION SINCE
0A0CE 8030A2E0  (01068)          JMP EPDSRS,LT                        IBIT=0 IMPLIES NO BITS T
                (01069) ;                                              O SERIALIZE!!
                (01070) *************************************************************
0A0D0 808E9C58  (01071)          JMP HIGDX2S(R7)                      GOTO THE CORR.SERIALIZAT
                (01072) ;                                              ION ROUTINE
                (01073) *
                (01074) *
                (01075)          EVEN
                (01076) *
                (01077) *
0A0D2 F070R00F  (01078) DBIT4S   MOVRP R7,IBITS                       SAVE THE 1ST IBIT VALUE
0A0D3 0D40      (01079)          CLR R4
0A0D5 0800      (01080)          EVEN
0A0D6 F2788B0F  (01081) DBIT4S1  CMPRP R7,IBITS(R4)                   COMPARE THE BIT ASSIGNME
                (01082) ;                                              NT WITH THE MARK
0A0D8 8020A110  (01083)          JMP DBIT3S,GT                        GOTO NEXT BIT DECODING R
                (01084) ;                                              OUTINE IF BIT ASSIGN. CH
                (01085) ;                                              ANGES
0A0DA DE0005E0  (01086)          SKRSL 0,APDNFLS                      HAS APDONE HAPPENED?
0A0DC 2621      (01087)          HOP $L434                            NO, HOP TO EVEN LOC AFTE
                (01088) ;                                              R RET
0A0DE 0800      (01089)          EVEN
```

```
0A0DE CC0005B6 (01090)          MOVZ9 APDDELS
0A0E0 E1109E0C (01091)          STORE R1,R1SAV
0A0E2 E1209E0E (01092)          STORE R2,R2SAV
0A0E4 E1309E10 (01093)          STORE R3,R3SAV
0A0E6 E1409E12 (01094)          STORE R4,R4SAV
0A0E8 E1509E14 (01095)          STORE R5,R5SAV
0A0EA E1609E16 (01096)          STORE R6,R6SAV
0A0EC E1709E18 (01097)          STORE R7,R7SAV
0A0EE 0E70     (01098)          RET
0A0EF 0800     (01099)          EVEN
0A0F0 A6109E0C (01100)          LOAD R1,R1SAV
0A0F2 A6209E0E (01101)          LOAD R2,R2SAV
0A0F4 A6309E10 (01102)          LOAD R3,R3SAV
0A0F6 A6409E12 (01103)          LOAD R4,R4SAV
0A0F8 A6509E14 (01104)          LOAD R5,R5SAV
0A0FA A6609E16 (01105)          LOAD R6,R6SAV
0A0FC A6709E18 (01106)          LOAD R7,R7SAV
0A0FE 0A48EA0E (01107)          SMBCL 4,OTDCTS(R4)          SKIP IF BIT 4 OF OCT(R4)
0A100 D204BC0E (01108)          SMB 0,SENS(R2)
0A102 2641     (01109)          INCR R4,1
0A103 0800     (01110)          EVEN
0A104 2621     (01111)          INCR R2,1                   INCREMENT THE BUFFER POS
               (01112) ;                                    ITION COUNTER
0A105 0800     (01113)          EVEN
0A106 8C50A0E6 (01114)          DJP R5,DBIT4S1              CHECK IF R5 +VE
0A108 E0509C8E (01115)          MOVBR R5,FORFORS            RELOAD THE COUNTER WITH
               (01116) ;                                    44
0A10A E2209C86 (01117)          CMPMB R2,POST3S             IS R2 GREATER THAN POST
               (01118) ;                                    3
0A10C 84209E8E (01119)          XCT CNTP1S,LE               ADD 18 TO R2 IF LESS THA
               (01120) ;                                    N OR EQUAL
0A10E 8000A0E6 (01121)          JMP DBIT4S1                 GO BACK TO DBITS1
               (01122) *
               (01123) *
               (01124) ;  SERIALIZATION FOR BIT 3
               (01125) *
               (01126)          EVEN
0A110 E2209C89 (01127) DBIT3S   CMPMB R2,POST4S
0A112 8010A2E6 (01128)          JMP ENDSPS,EQ
0A114 DE0005B0 (01129)          SMBSL 0,APDNELS             HAS APDONE HAPPENED?
0A116 2021     (01130)          HOP #L+44                   NO, HOP TO EVEN LOC AFTE
               (01131) ;                                    R RET
0A117 0800     (01132)          EVEN
0A118 CC0005B6 (01133)          MOVZ9 APDDELS
```

```
0A11A E1109E0C (01134)          STORE R1,R1SAV
0A11C E1209E0E (01135)          STORE R2,R2SAV
0A11E E1309E10 (01136)          STORE R3,R3SAV
0A120 E1409E12 (01137)          STORE R4,R4SAV
0A122 E1509E14 (01138)          STORE R5,R5SAV
0A124 E1609E16 (01139)          STORE R6,R6SAV
0A126 E1709E18 (01140)          STORE R7,R7SAV
0A128 0E70     (01141)          RET
0A129 0800     (01142)          EVEN
0A12A A6109E0C (01143)          LOAD R1,R1SAV
0A12C A6209E0E (01144)          LOAD R2,R2SAV
0A12E A6309E10 (01145)          LOAD R3,R3SAV
0A130 A6409E12 (01146)          LOAD R4,R4SAV
0A132 A6509E14 (01147)          LOAD R5,R5SAV
0A134 A6609E16 (01148)          LOAD R6,R6SAV
0A136 A6709E18 (01149)          LOAD R7,R7SAV
0A138 E078860E (01150)          MOVBR R7,IBITS(R4)                          SAVE THE NEXT IBIT VALUE
0A13A 0040     (01151)          CLR R4
0A13B 0800     (01152)          EVEN
0A13C E278860E (01153) DBIT3S1  CMPBR R7,IBITS(R4)                          COMPARE THE BIT ASSIGNME
               (01154) ;                                                    NT WITH THE MARK
0A13E 8020A176 (01155)          JMP DBIT2S,GT                               FI NOT EQUAL,GOTO DECODE
0A140 DE0005E0 (01156)          SERSE 0,APDSELS                             HAS APDONE HAPPENED?
0A142 2021     (01157)          HOP #L+34                                   NO, HOP TO EVEN LOC AFTE
               (01158) ;                                                    R RET
0A143 0800     (01159)          EVEN
0A144 CC0005E0 (01160)          MOVZM APDSELS
0A146 E1109E0C (01161)          STORE R1,R1SAV
0A148 E1209E0E (01162)          STORE R2,R2SAV
0A14A E1309E10 (01163)          STORE R3,R3SAV
0A14C E1409E12 (01164)          STORE R4,R4SAV
0A14E E1509E14 (01165)          STORE R5,R5SAV
0A150 E1609E16 (01166)          STORE R6,R6SAV
0A152 E1709E18 (01167)          STORE R7,R7SAV
0A154 0E70     (01168)          RET
0A155 0800     (01169)          EVEN
0A156 A6109E0C (01170)          LOAD R1,R1SAV
0A158 A6209E0E (01171)          LOAD R2,R2SAV
0A15A A6309E10 (01172)          LOAD R3,R3SAV
0A15C A6409E12 (01173)          LOAD R4,R4SAV
0A15E A6509E14 (01174)          LOAD R5,R5SAV
0A160 A6609E16 (01175)          LOAD R6,R6SAV
0A162 A6709E18 (01176)          LOAD R7,R7SAV
               (01177) *
```

```
PAGE   352        PROG. CSPUCP.TXT

0A163 EA358700  (01178)         SBHCL 3,OTDCIS(R4)      SKIP IF BIT 3 OF DCT(R4)
0A166 E2238C0E  (01179)         SNK 0,SENS(R2)
0A168 2641      (01180)         INCR R4,1
0A169 C800      (01181)         EVEN
0A16A 2621      (01182)         INCR R2,1               INCREMENT THE BUFFER POS
                (01183) ;                               ITION COUNTER
0A16B 0800      (01184)         EVEN
0A16C RC50F13C  (01185)         BJE R5,DBIT3S1          GO BACK TO DBIT3S1
0A16E F0509C8E  (01186)         MOVBE R5,FORFORS        RELOAD THE COUNTER WITH
                (01187) ;                               44
0A170 F2209C86  (01188)         CMPRR R2,POST3S         IS R2 GREATER THAN POST
                (01189) ;                               3
0A172 R4203E8A  (01190)         ACI CNTP1S,1E           ADD 1E TO LOOP COUNTER I
                (01191) ;                               F LESS THAN OR EQUAL TO
0A174 E000A13C  (01192)         JMP DBIT3S1             GO BACK TO DBIT3S1
                (01193) *
                (01194) *
                (01195) ;  SERIALIZATION FOR BIT 2
                (01196) *
                (01197)         EVEN
                (01198) DBIT2S  EVEN
0A176 F2209CF9  (01199)         CMPRR R2,POST4S
0A178 R010A2FR  (01200)         JPP ENDSRS,E0
0A17A DE000580  (01201)         SMBSL 0,APDNFLS         HAS APDONE HAPPENED?
0A17C 2021      (01202)         HOP #1,+44              NO, HOP TO EVEN LOC AFTE
                (01203) ;                               R RET
0A17E 0800      (01204)         EVEN
0A17F CC000580  (01205)         MOVZM APDNFLS
0A180 F1109F0C  (01206)         STORE R1,R1SAV
0A182 F1209F0E  (01207)         STORE R2,R2SAV
0A184 F1309F10  (01208)         STORE R3,R3SAV
0A186 F1409F12  (01209)         STORE R4,R4SAV
0A188 F1509F14  (01210)         STORE R5,R5SAV
0A18A F1609F16  (01211)         STORE R6,R6SAV
0A18C F1709F18  (01212)         STORE R7,R7SAV
0A18E 0E70      (01213)         RET
0A18F 0800      (01214)         EVEN
0A190 A6109F0C  (01215)         LOAD R1,R1SAV
0A192 A6209F0E  (01216)         LOAD R2,R2SAV
0A194 A6309F10  (01217)         LOAD R3,R3SAV
0A196 A6409F12  (01218)         LOAD R4,R4SAV
0A198 A6509F14  (01219)         LOAD R5,R5SAV
0A19A A6609F16  (01220)         LOAD R6,R6SAV
0A19C A6709F18  (01221)         LOAD R7,R7SAV
```

```
0A19E E07880OF (01222)          MOVBR R7,INITS(R4)        SAVE THE INIT VALUE
0A1A0 0040     (01223)          CLR R4
0A1A1 0800     (01224)          EVEN
0A1A2 E2785 OF (01225) DBIT2S1  CMPBR R7,IBITS(R4)        COMPARE THE BIT ASSIGNME
               (01226) ;                                  NT WITH THE MARK
0A1A4 8020A1DC (01227)          JMP DBITIS,GF             IF NOT EQUAL,GOTO DECODE
0A1A6 DE000580 (01228)          SMBSL 0,APDOFLS           HAS APDONE HAPPENED?
0A1A8 2021     (01229)          HOP *L+16                 NO, HOP TO EVEN LOC AFTE
               (01230) ;                                  R RET
0A1A9 0800     (01231)          EVEN
0A1AA CC000580 (01232)          MOVZM APDOFLS
0A1AC F1109F0C (01233)          STORE R1,R1SAV
0A1AE F1209F0E (01234)          STORE R2,R2SAV
0A1B0 F1309F10 (01235)          STORE R3,R3SAV
0A1B2 F1409F12 (01236)          STORE R4,R4SAV
0A1B4 F1509F14 (01237)          STORE R5,R5SAV
0A1B6 F1609F16 (01238)          STORE R6,R6SAV
0A1B8 F1709F18 (01239)          STORE R7,R7SAV
0A1BA 0F70     (01240)          RET
0A1BB 0800     (01241)          EVEN
0A1BC A6109F0C (01242)          LOAD R1,R1SAV
0A1BE A6209F0E (01243)          LOAD R2,R2SAV
0A1C0 A6309F10 (01244)          LOAD R3,R3SAV
0A1C2 A6409F12 (01245)          LOAD R4,R4SAV
0A1C4 A6509F14 (01246)          LOAD R5,R5SAV
0A1C6 A6609F16 (01247)          LOAD R6,R6SAV
0A1C8 A6709F18 (01248)          LOAD R7,R7SAV
               (01249) *
0A1CA DA288A0F (01250)          SMBCL 2,OTDCIS(R4)        SKIP IF BIT 2 OF DCT(R4)
0A1CC D2048C0F (01251)          SMB 0,SENS(R2)
0A1CE 2641     (01252)          INCR R4,1
0A1CF 0800     (01253)          EVEN
0A1D0 2621     (01254)          INCR R2,1                 INCREMENT THE BUFFER POS
               (01255) ;                                  ITION COUNTER
0A1D1 0800     (01256)          EVEN
0A1D2 8C50A1A2 (01257)          LJP R5,DBIT2S1            CHECK IF R5 +VE
0A1D4 F0509C8F (01258)          MOVBR R5,FOREORS          RELOAD THE COUNTER WITH
               (01259) ;                                  44
0A1D6 F2209C86 (01260)          CMPBR R2,POST3S           IS R2 GREATER THAN POST
               (01261) ;                                  3
0A1D8 8E209F88 (01262)          XCT CLIPIS,LF             ADD 18 TO LOOP COUNTER I
               (01263) ;                                  F LESS THAN OR EQUAL
0A1DA 8000A1A2 (01264)          JMP DBIT2S1               GO BACK TO DBIT2S1
               (01265) *
```

```
                  (01266) *
                  (01267) *
                  (01268) ;  SERIALIZATION FOR BIT 1
                  (01269)         EVEN
                  (01270) DBIT1S  EVEN
0A1DC DE0005E0    (01271)         SKRSE 0,APDNELS                HAS APDONE HAPPENED?
0A1DE 2021        (01272)         HOP *L+31                      NO, HOP TO EVEN LOC AFTE
                  (01273) ;                                      R RET
0A1DF 0800        (01274)         EVEN
0A1E0 CC0005E0    (01275)         MOVZB APDNELS
0A1E2 E1109E0C    (01276)         STORE R1,R1SAV
0A1E4 E1209E0E    (01277)         STORE R2,R2SAV
0A1E6 E1309E10    (01278)         STORE R3,R3SAV
0A1E8 E1409E12    (01279)         STORE R4,R4SAV
0A1EA E1509E14    (01280)         STORE R5,R5SAV
0A1EC E1609E16    (01281)         STORE R6,R6SAV
0A1EE E1709E18    (01282)         STORE R7,R7SAV
0A1F0 0E70        (01283)         RET
0A1F1 0800        (01284)         EVEN
0A1F2 A6109E0C    (01285)         LOAD R1,R1SAV
0A1F4 A6209E0E    (01286)         LOAD R2,R2SAV
0A1F6 A6309E10    (01287)         LOAD R3,R3SAV
0A1F8 A6409E12    (01288)         LOAD R4,R4SAV
0A1FA A6509E14    (01289)         LOAD R5,R5SAV
0A1FC A6609E16    (01290)         LOAD R6,R6SAV
0A1FE A6709E18    (01291)         LOAD R7,R7SAV
0A200 E2209C89    (01292)         CMPRB R2,POST4S
0A202 8010A2E8    (01293)         JMP EPDSRS,EQ
0A204 E07D8E0E    (01294)         MOVMB R7,IBITS(R4)             SAVE THE IBIT VALUE
0A206 0D40        (01295)         CLR R4
0A207 0800        (01296)         EVEN
                  (01297) *
                  (01298) *
0A208 E27D8E0E    (01299) DBIT1S1 CMPMB R7,IBITS(R4)             COMPARE THE BIT ASSIGNME
                  (01300) ;                                      NT WITH THE MARK
0A20A 8020A242    (01301)         JMP DBIT0S,GT                  GOTO SERIALIZATION ROUTI
                  (01302) ;                                      NE IF NOT EQUAL
0A20C DE0005E0    (01303)         SKRSE 0,APDNELS                HAS APDONE HAPPENED?
0A20E 2021        (01304)         HOP *L+31                      NO, HOP TO EVEN LOC AFTE
                  (01305) ;                                      R RET
0A20F 0800        (01306)         EVEN
0A210 CC0005E0    (01307)         MOVZB APDNELS
0A212 E1109E0C    (01308)         STORE R1,R1SAV
0A214 E1209E0E    (01309)         STORE R2,R2SAV
```

```
0A216 E1309E10 (01310)          STORE R3,R3SAV
0A218 E1409E12 (01311)          STORE R4,R4SAV
0A21A E1509E14 (01312)          STORE R5,R5SAV
0A21C E1609E16 (01313)          STORE R6,R6SAV
0A21E E1709E18 (01314)          STORE R7,R7SAV
0A220 0E70     (01315)          RET
0A221 0800     (01316)          EVEN
0A222 AE109E0C (01317)          LOAD R1,R1SAV
0A224 AE209E0E (01318)          LOAD R2,R2SAV
0A226 AE309E10 (01319)          LOAD R3,R3SAV
0A228 AE409E12 (01320)          LOAD R4,R4SAV
0A22A AE509E14 (01321)          LOAD R5,R5SAV
0A22C AE609E16 (01322)          LOAD R6,R6SAV
0A22E AE709E18 (01323)          LOAD R7,R7SAV
               (01324) *
0A230 DA1B6A0E (01325)          SKBCL 1,0,DCTS(R4)       SKIP IF BIT 1 OF DCT(R4)
0A232 E201BC0E (01326)          SOP 0,SEGS(R2)
0A234 2641     (01327)          INCR R4,1
0A235 0800     (01328)          EVEN
0A236 2621     (01329)          INCR R2,1                INCREMENT THE BUFFER POS
               (01330) :                                 ITION COUNTER
0A237 0800     (01331)          EVEN
0A238 BC50A208 (01332)          DJP R5,DBITIS1           CHECK IF R5 +VE
0A23A E0509CBE (01333)          MOVOF R5,FORFOUR$        RELOAD THE COUNTER WITH
               (01334) :                                 44
0A23C E2209CBE (01335)          CMPMI R2,POST3S          IS R2 GREATER THAN POST
               (01336) :                                 3
0A23E 8320 9E86 (01337)         YCT CNT18S,LE            ADD 18 TO R2 IF LESS THA
               (01338) :                                 N OR EQUAL TO
0A240 8000A208 (01339)          JSP DBITIS1              GO BACK TO DBITIS1
               (01340) *
               (01341) *
               (01342) :  SERIALIZATION OF BIT 0
               (01343)          EVEN
0A242 E2209C80 (01344) DBIT0S   CMPNE R2,POST4S          COMPARE R2 WITH THE FRAM
               (01345) :                                 E LENGTH
0A244 8010A2E8 (01346)          JPE ENDSRS,EQ            LEAVE IF THE ENTIRE FRAM
               (01347) :                                 E IS FINISHED
0A246 1E0005A0 (01348)          SKBSE 0,APDFLS           HAS APDONE HAPPENED?
0A248 2621     (01349)          HOP $L+34                NO, HOP TO EVEN LOC AFTE
               (01350) :                                 R RET
0A249 0800     (01351)          EVEN
0A24A CC0005A0 (01352)          MOVZP APDFLS
0A24C E1109E0C (01354)          STORE R1,R1SAV
```

```
0A24E E1209E0E (01354)          STORE R2,R2SAV
0A250 E1309E10 (01355)          STORE R3,R3SAV
0A252 E1409E12 (01356)          STORE R4,R4SAV
0A254 E1509E14 (01357)          STORE R5,R5SAV
0A256 E1609E16 (01358)          STORE R6,R6SAV
0A258 E1709E18 (01359)          STORE R7,R7SAV
0A25A 0E70     (01360)          RET
0A25B 0800     (01361)          EVEN
0A25C A6109E0C (01362)          LOAD R1,R1SAV
0A25E A6209E0E (01363)          LOAD R2,R2SAV
0A260 A6309E10 (01364)          LOAD R3,R3SAV
0A262 A6409E12 (01365)          LOAD R4,R4SAV
0A264 A6509E14 (01366)          LOAD R5,R5SAV
0A266 A6609E16 (01367)          LOAD R6,R6SAV
0A268 A6709E18 (01368)          LOAD R7,R7SAV
0A26A E2209CA6 (01369)          CMPMR R2,POST3$       COMPARE R2 WITH POST3
0A26C 8620A262 (01370)          JMP ERRTOS,GT         IF GREATER THAN POST3, N
               (01371) ;                              O NEED TO PUT IN PARITY
               (01372) ;                              BITS
               (01373) *
               (01374) *
0A26E E0609C89 (01375)          MOVMR R6,POST4$       MOVE THE FRAME LENGTH TO
0A270 4E61     (01376)          SUBTR R6,R2           R6=LSEN-1-R2
0A271 0800     (01377)          EVEN
0A272 9040B20D (01378)          MOVIR R4,OTDCTS-1     MOVE ADDRESS OTDCT-1 TO
               (01379) ;                              R4
0A274 9C20BC0E (01380)          ADDIR R2,SENS         SET R2 TO BE THE ADDRESS
               (01381) ;                              :SENS$(R2)
0A276 E0309C90 (01382)          MOVMR R3,ONES         SAVE THE MASK IN R3
0A278 3042     (01383) PRITOS1  POPXI R4,R1           POP OTDCT(R4)-->R1...R4=
               (01384) ;                              R4+1
0A279 0800     (01385)          EVEN
0A27A 4A16     (01386)          ANDRR R1,R3           SAVE ONLY THE LAST BIT
0A27B 0800     (01387)          EVEN
0A27C 3422     (01388)          PUSHXI R2,R1          SAVE THE VALUE IN SENS(R
               (01389) ;                              2).. R2=R2+1
0A27D 0800     (01390)          EVEN
0A27E DE000580 (01391)          SPRSL 0,APDNFLS       HAS APDONE HAPPENED?
0A280 2021     (01392)          HOP #L+3A             NO, HOP TO EVEN LOC AFTE
               (01393) ;                              R RET
0A281 0800     (01394)          EVEN
0A282 CC0005E0 (01395)          MOVZM APDOFLS
0A284 E1109E0C (01396)          STORE R1,R1SAV
0A286 E1209E0E (01397)          STORE R2,R2SAV
```

```
0A28A E1309E10 (01398)          STORE R3,R3SAV
0A28A E1409E12 (01399)          STORE R4,R4SAV
0A28C E1509E14 (01400)          STORE R5,R5SAV
0A28E E1609E16 (01401)          STORE R6,R6SAV
0A290 E1709E18 (01402)          STORE R7,R7SAV
0A292 CE70     (01403)          RET
0A293 0800     (01404)          EVEN
0A294 AE109E0E (01405)          LOAD R1,R1SAV
0A296 AE209E0E (01406)          LOAD R2,R2SAV
0A298 AE309E10 (01407)          LOAD R3,R3SAV
0A29A AE409E12 (01408)          LOAD R4,R4SAV
0A29C AE509E14 (01409)          LOAD R5,R5SAV
0A29E AE609E16 (01410)          LOAD R6,R6SAV
0A2A0 AE709E18 (01411)          LOAD R7,R7SAV
               (01412) *
0A2A2 8C50A2AE (01413)          DJP R5,ENDS0               CHECK IF R5 +VE
0A2A4 F0509C8E (01414)          FOVMR R5,FORFORS           RELOAD THE COUNTER WITH
               (01415) ;                                   44
0A2A6 F2209C67 (01416)          CMPMR R2,POST3$1           IS R2 GREATER THAN POST
               (01417) ;                                   3
0A2A8 8020A2AE (01418)          JMP ENDS0,GT               JUMP TO ENDS0 IF GREATER
0A2AA FC209C8E (01419)          ADDMR R2,EIGTENS           ADD 18 TO R2
0A2AC FE609C8E (01420)          SUBMR R6,EIGTENS           SUBTRACT 18 FROM R6
0A2AE 8C60A27E (01421) ENDS0    DJP R6,DRITOS1             LOOP BACK UNITIL THE ENT
               (01422) ;                                   IRE FRAME IS DONE
0A2B0 8000A2E4 (01423)          JMP ENDSRS
               (01424) *
               (01425) *
               (01426) ;  BIT SERIALIZATION WITHOUT ERROR CONTROL
               (01427) *
0A2B2 F0609C85 (01428) ERITOS   FOVMR R6,POS14S            MOVE THE FRAME LENGTH TO
0A2B4 3E64     (01429)          SUBER R6,R2                R6=LSEN-1-R2
0A2B5 0800     (01430)          EVEN
0A2B6 9040820D (01431)          MOVIR R4,QTDCTS-1          MOVE ADDRESS QTDCT-1 TO
               (01432) ;                                   R4
0A2B8 9C20BC0E (01433)          ADEIR R2,SENS              SET R2 TO BE THE ADDRESS
               (01434) ;                                   :SEN1S(R2)
0A2BA F0309C9E (01435)          FLVSR R3,ONES              SAVE THE MASK IN R3
0A2BC 3042     (01436) ERITOS1  POEXI R3,R1                POP QTDCT(R4)-->R1...R4=
               (01437) ;                                   R4+1
0A2BD 0400     (01438)          EVEN
0A2BE DE0005A0 (01439)          SPOSI 0,APDONES            HAS APDONS HAPPENED?
0A2C0 2621     (01440)          HOP #L434                  NO, HOP TO EVEN LOC AFTE
               (01441) ;                                   R RET
```

```
0A2C1 0800       (01442)          EVEN
0A2C2 CC000550   (01443)          MOVZM APDNELS
0A2C3 E1109E0C   (01444)          STORE R1,R1SAV
0A2C6 E1209E0E   (01445)          STORE R2,R2SAV
0A2C8 E1309E10   (01446)          STORE R3,R3SAV
0A2CA E1409E12   (01447)          STORE R4,R4SAV
0A2CC E1509E14   (01448)          STORE R5,R5SAV
0A2CE E1609E16   (01449)          STORE R6,R6SAV
0A2D0 E1709E18   (01450)          STORE R7,R7SAV
0A2D2 0E70       (01451)          RET
0A2D3 0800       (01452)          EVEN
0A2D4 A6109E0C   (01453)          LOAD R1,R1SAV
0A2D6 A6209E0E   (01454)          LOAD R2,R2SAV
0A2D8 A6309E10   (01455)          LOAD R3,R3SAV
0A2DA A6409E12   (01456)          LOAD R4,R4SAV
0A2DC A6509E14   (01457)          LOAD R5,R5SAV
0A2DE A6609E16   (01458)          LOAD R6,R6SAV
0A2E0 A6709E18   (01459)          LOAD R7,R7SAV
0A2E2 4A16       (01460)          ANDRR R1,R3             SAVE ONLY THE LAST BIT
0A2E3 0800       (01461)          EVEN
0A2E4 3422       (01462)          PUSHXI R2,R1            SAVE THE VALUE IN SENS(R
                 (01463) ;                                2).. R2=R2+1
0A2E5 0800       (01464)          EVEN
0A2E6 8C60A2BC   (01465)          DJP R6,FBITOS1          LOOP BACK UNITIL THE ENT
                 (01466) ;                                IRE FRAME IS DONE
                 (01467) ENDSRS   EVEN
0A2E8 02B0       (01468)          CCS 3                   FROM SRB=2...
0A2E9 06A0       (01469)          SCS 2                   TO SRB=1
0A2EA 02D0       (01470)          CCS 5                   FROM MRB=2...
0A2EB 06C0       (01471)          SCS 4                   TO MRB=1
                 (01472) *  JMP LODATS      NOT ENOUGH TIME TO PROTECT
0A2EC 8000A2FF   (01473)          JMP ACHEOS              PROTECT SIDEBAND
                 (01474)          EJECT
```

```
                  (01475) ***** BCH ENCODING OF SIDEBAND & PART OF MAINBAND DATA
                  (01476) BCHE0S  EVEN
0A2EE 90200001    (01477)         MOVIR R2,1                      SET INPUT BUFFER POINTER
                  (01478) BCHE1S  EVEN
0A2F0 CE000580    (01479)         SSESL 0,APDNFLS                 HAS APDONE HAPPENED?
0A2F2 2021        (01480)         HOP $L434                       NO, HOP TO EVEN LOC AFTE
                  (01481) ;                                       R RET
0A2F3 0800        (01482)         EVEN
0A2F4 CC000580    (01483)         MOVZE APDNFLS
0A2F6 F1109F0C    (01484)         STORE R1,R1SAV
0A2F8 F1209F0E    (01485)         STORE R2,R2SAV
0A2FA F1309F10    (01486)         STORE R3,R3SAV
0A2FC F1409F12    (01487)         STORE R4,R4SAV
0A2FE F1509F14    (01488)         STORE R5,R5SAV
0A300 F1609F16    (01489)         STORE R6,R6SAV
0A302 F1709F18    (01490)         STORE R7,R7SAV
0A304 0E70        (01491)         RET
0A305 0800        (01492)         EVEN
0A306 A6109F0C    (01493)         LOAD R1,R1SAV
0A308 A6209F0E    (01494)         LOAD R2,R2SAV
0A30A A6309F10    (01495)         LOAD R3,R3SAV
0A30C A6409F12    (01496)         LOAD R4,R4SAV
0A30E A6509F14    (01497)         LOAD R5,R5SAV
0A310 A6609F16    (01498)         LOAD R6,R6SAV
0A312 A6709F18    (01499)         LOAD R7,R7SAV
0A314 0D30        (01500)         CLR R3                          CLEAR 15 BITS REMAINDER
0A315 0D40        (01501)         CLR R4                          CLEAR BITS 16,17,18
0A316 9050005B    (01502)         MOVIR R5,88                     R5=88
0A318 F2048C0E    (01503) BCHE2S  CMPMZ SENS(R2)                  READ SENS(R2)
0A31A 8010A320    (01504)         JMP BCHE3S,EQ                   GO TO BCHE3S IF SENS(R2)
                  (01505) ;                                       =0
0A31C C06A9CA2    (01506)         MOVPRL R6,TBENCS(R5)            R6=TBENC(R5),R7=TBENC(R5
                  (01507) ;                                       +1)
0A31E 443C        (01508)         XORRR R3,R6                     R3=R3.XOR.R6
0A31F 444F        (01509)         XORRR R4,R7                     R4=R4.XOR.R7
0A320 2621        (01510) BCHE3S  INCR R2,1                       INCREAMENT INPUT BUFFER
                  (01511) ;                                       POINTER
0A321 0800        (01512)         EVEN
0A322 8E50A318    (01513)         DBJR R5,BCHE2S                  DECREAMENT TABLE POINTER
                  (01514) *
                  (01515) ;  START SERIALIZE R3 AND R4
                  (01516) ;
                  (01517) ;
0A324 90500001    (01518)         MOVIR R5,1
```

```
0A326 2AE6       (01519)          SRRCL 14,R3
0A327 F054BC0E   (01520)          MOVRM R5,SENS(R2)
0A329 2AD6       (01521)          SRRCL 13,R3
0A32A F054BC0F   (01522)          MOVRM R5,SENS+1(R2)
0A32C 2AC6       (01523)          SRRCL 12,R3
0A32D F054BC10   (01524)          MOVRM R5,SENS+2(R2)
0A32F 2AB6       (01525)          SRRCL 11,R3
0A330 F054BC11   (01526)          MOVRM R5,SENS+3(R2)
0A332 2AA6       (01527)          SRRCL 10,R3
0A333 F054BC12   (01528)          MOVRM R5,SENS+4(R2)
0A335 2A96       (01529)          SRRCL 9,R3
0A336 F054BC13   (01530)          MOVRM R5,SENS+5(R2)
0A338 2A86       (01531)          SRRCL 8,R3
0A339 F054BC14   (01532)          MOVRM R5,SENS+6(R2)
0A33B 2A76       (01533)          SRRCL 7,R3
0A33C F054BC15   (01534)          MOVRM R5,SENS+7(R2)
0A33E 2A66       (01535)          SRRCL 6,R3
0A33F F054BC16   (01536)          MOVRM R5,SENS+8(R2)
0A341 2A56       (01537)          SRRCL 5,R3
0A342 F054BC17   (01538)          MOVRM R5,SENS+9(R2)
0A344 2A46       (01539)          SRRCL 4,R3
0A345 F054BC18   (01540)          MOVRM R5,SENS+10(R2)
0A347 2A36       (01541)          SRRCL 3,R3
0A348 F054BC19   (01542)          MOVRM R5,SENS+11(R2)
0A34A 2A26       (01543)          SRRCL 2,R3
0A34B F054BC1A   (01544)          MOVRM R5,SENS+12(R2)
0A34D 2A16       (01545)          SRRCL 1,R3
0A34E F054BC1B   (01546)          MOVRM R5,SENS+13(R2)
0A350 2A06       (01547)          SRRCL 0,R3
0A351 F054BC1C   (01548)          MOVRM R5,SENS+14(R2)
0A353 2A28       (01549)          SRRCL 2,R4
0A354 F054BC1D   (01550)          MOVRM R5,SENS+15(R2)
0A356 2A18       (01551)          SRRCL 1,R4
0A357 F054BC1E   (01552)          MOVRM R5,SENS+16(R2)
0A359 2A08       (01553)          SRRCL 0,R4
0A35A F054BC1F   (01554)          MOVRM R5,SENS+17(R2)
                 (01555) *  ADDIR R2,18
                 (01556) *  CMPIR R2,190
                 (01557) *  JMP ENDECS,EQ
                 (01558) *  CMPIR R2,560
                 (01559) *  JMP RCHEIS,LT
                 (01560) *
                 (01561) *
                 (01562) ;  END OF ENCODER ROUTINE
```

```
                 (01563) *
                 (01564) *
                 (01565) ENDECS   EVEN
0A35C 8000A35E   (01566)          JMP LODATS
                 (01567) *
                 (01568) *
                 (01569)          EJECT
```

```
                    (01570) ***** LOAD DIGITAL FRAME DATA INTO SEN1 OR SEN2
                    (01571) LODATS  EVEN
0A35E 0280          (01572)         CCS 3                           FROM SRB=2...
0A35F 06A0          (01573)         SCS 2                           TO SRB=1
0A360 02C0          (01574)         CCS 4                           FROM MRB=1...
0A361 06E0          (01575)         SCS 5                           TO MRB=2
0A362 DE000580      (01576)         SMBSL 0,APDNFLS                 HAS APDONE HAPPENED?
0A364 2021          (01577)         HOP $L434                       NO, HOP TO EVEN LOC AFTE
                    (01578) :                                       R RET
0A365 0800          (01579)         EVEN
0A366 CC000580      (01580)         MOVZM APDNFLS
0A368 F1109F0C      (01581)         STORE R1,R1SAV
0A36A F1209F0E      (01582)         STORE R2,R2SAV
0A36C F1309F10      (01583)         STORE R3,R3SAV
0A36E F1409F12      (01584)         STORE R4,R4SAV
0A370 F1509F14      (01585)         STORE R5,R5SAV
0A372 F1609F16      (01586)         STORE R6,R6SAV
0A374 F1709F18      (01587)         STORE R7,R7SAV
0A376 0E70          (01588)         RET
0A377 0800          (01589)         EVEN
0A378 A6109F0C      (01590)         LOAD R1,R1SAV
0A37A A6209F0E      (01591)         LOAD R2,R2SAV
0A37C A6309F10      (01592)         LOAD R3,R3SAV
0A37E A6409F12      (01593)         LOAD R4,R4SAV
0A380 A6509F14      (01594)         LOAD R5,R5SAV
0A382 A6609F16      (01595)         LOAD R6,R6SAV
0A384 A6709F18      (01596)         LOAD R7,R7SAV
0A386 F0200567      (01597)         MOVRR R2,AFLGS                  'STORE AFLG TEMP. IN R2
0A388 F220056F      (01598)         CMPRM R2,XDATAS                 'COMPARE R2 WITH XDATA
                    (01599) *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                    (01600) *  THIS ERROR TRAP MAY NEEDED,DELETED FOR DEBUG 5/17/80
                    (01601) *  JMP ERR2S,EQ     'ERROR OCCURS SINCE THE SAME BUF IS USED TWICE
                    (01602) *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0A38A F020056F      (01603)         MOVRM R2,XDATAS                 'RESET XDATA
0A38C 90208C0C      (01604)         MOVIR R2,SENS-2
0A38E 9040003D      (01605)         MOVIR R4,LSEN1-1
0A390 F2000567      (01606)         CMPMZ AFLGS                     WHICH BUFFER SHOULD THE
                    (01607) :                                       SERIALIZED DATA GO?
0A392 8010A414      (01608)         JMP FL9S,EQ                     IF AFLG=0, LOAD SEN 1
                    (01609) *
0A394 F8028C0F      (01610)         MOVLM 1,SENS                    MAKE SYNC BIT 1 FOR BUFF
                    (01611) :                                       ER 2
0A396 D7281C4F      (01612)         BMOVEL R2,R4,SEN2S
0A398 DE000580      (01613)         SMBSL 0,APDNFLS                 HAS APDONE HAPPENED?
```

```
0A39A 2021      (01614)        HOP #L+34                    NO, HOP TO EVEN LOC AFTE
                (01615) :                                   R RET
0A39B 0800      (01616)        EVEN
0A39C CC0005B0  (01617)        MOVZM APDNFLS
0A39E F1109F0C  (01618)        STORE R1,R1SAV
0A3A0 F1209F0E  (01619)        STORE R2,R2SAV
0A3A2 F1309F10  (01620)        STORE R3,R3SAV
0A3A4 F1409F12  (01621)        STORE R4,R4SAV
0A3A6 F1509F14  (01622)        STORE R5,R5SAV
0A3A8 F1609F16  (01623)        STORE R6,R6SAV
0A3AA F1709F18  (01624)        STORE R7,R7SAV
0A3AC 0F70      (01625)        RET
0A3AD 0800      (01626)        EVEN
0A3AE A6109F0C  (01627)        LOAD R1,R1SAV
0A3B0 A6209F0E  (01628)        LOAD R2,R2SAV
0A3B2 A6309F10  (01629)        LOAD R3,R3SAV
0A3B4 A6409F12  (01630)        LOAD R4,R4SAV
0A3B6 A6509F14  (01631)        LOAD R5,R5SAV
0A3B8 A6609F16  (01632)        LOAD R6,R6SAV
0A3BA A6709F18  (01633)        LOAD R7,R7SAV
0A3BC 90208C88  (01634)        MOVIR R2,SENS-2+LSEN12
0A3BE 9040003D  (01635)        MOVIR R4,LSEN1-1
0A3C0 L72816CA  (01636)        BMOVEL R2,R4,SEN2S+LSEN12
                (01637) *  MOVLM CLRSG2,SYSFLGS
0A3C2 DE0005B0  (01638)        SMBSE 0,APDNFLS              HAS APDONE HAPPENED?
0A3C4 2021      (01639)        HOP #L+34                    NO, HOP TO EVEN LOC AFTE
                (01640) :                                   R RET
0A3C5 0800      (01641)        EVEN
0A3C6 CC0005B0  (01642)        MOVZM APDNFLS
0A3C8 F1109F0C  (01643)        STORE R1,R1SAV
0A3CA F1209F0E  (01644)        STORE R2,R2SAV
0A3CC F1309F10  (01645)        STORE R3,R3SAV
0A3CE F1409F12  (01646)        STORE R4,R4SAV
0A3D0 F1509F14  (01647)        STORE R5,R5SAV
0A3D2 F1609F16  (01648)        STORE R6,R6SAV
0A3D4 F1709F18  (01649)        STORE R7,R7SAV
0A3D6 0F70      (01650)        RET
0A3D7 0800      (01651)        EVEN
0A3D8 A6109F0C  (01652)        LOAD R1,R1SAV
0A3DA A6209F0E  (01653)        LOAD R2,R2SAV
0A3DC A6309F10  (01654)        LOAD R3,R3SAV
0A3DE A6409F12  (01655)        LOAD R4,R4SAV
0A3E0 A6509F14  (01656)        LOAD R5,R5SAV
0A3E2 A6609F16  (01657)        LOAD R6,R6SAV
```

```
0A3E3 A6709E18 (01658)        LOAD R7,R7SAV
0A3E6 90208E04 (01659)        MOVIR R2,SENS-2+2*LSEN12
0A3E8 9040003C (01660)        MOVIR R4,LSEN1-2
0A3EA D72B1746 (01661)        RMOVEL R2,R4,SEN2S+2*LSEN12
0A3EC DE000580 (01662)        SMESL 0,APDNFLS                       HAS APDONE HAPPENED?
0A3EE 2021     (01663)        HOP #L+34                             NO, HOP TO EVEN LOC AFTE
               (01664) ;                                            R RET
0A3EF 0800     (01665)        EVEN
0A3F0 CC000580 (01666)        MOVZM APDNFLS
0A3F2 E1109E0C (01667)        STORE R1,R1SAV
0A3F4 E1209E0E (01668)        STORE R2,R2SAV
0A3F6 E1309E10 (01669)        STORE R3,R3SAV
0A3F8 E1409E12 (01670)        STORE R4,R4SAV
0A3FA E1509E14 (01671)        STORE R5,R5SAV
0A3FC E1609E16 (01672)        STORE R6,R6SAV
0A3FE E1709E18 (01673)        STORE R7,R7SAV
0A400 0E70     (01674)        RET
0A401 0800     (01675)        EVEN
0A402 A6109E0C (01676)        LOAD R1,R1SAV
0A403 A6209E0E (01677)        LOAD R2,R2SAV
0A406 A6309E10 (01678)        LOAD R3,R3SAV
0A408 A6409E12 (01679)        LOAD R4,R4SAV
0A40A A6509E14 (01680)        LOAD R5,R5SAV
0A40C A6609E16 (01681)        LOAD R6,R6SAV
0A40E A6709E18 (01682)        LOAD R7,R7SAV
0A410 E0200567 (01683)        MOVMR R2,AFLGS                        'STORE AFLG TEMP. IN R2
0A412 8000A490 (01684)        JMP XMTISF                            END OF "TRANSMITTER"
0A414 CC008C0E (01685) EL9$   MOVZM SENS                            MAKE SYNC BIT 0 FOR BUFF
               (01686) ;                                            ER 1
0A416 D72814DC (01687)        RMOVEL R2,R4,SEN1S
0A418 DE000580 (01688)        SMBSL 0,APDNFLS                       HAS APDONE HAPPENED?
0A41A 2021     (01689)        HOP #L+34                             NO, HOP TO EVEN LOC AFTE
               (01690) ;                                            R RET
0A41B 0800     (01691)        EVEN
0A41C CC000580 (01692)        MOVZM APDNFLS
0A41E E1109E0C (01693)        STORE R1,R1SAV
0A420 E1209E0E (01694)        STORE R2,R2SAV
0A422 E1309E10 (01695)        STORE R3,R3SAV
0A424 E1409E12 (01696)        STORE R4,R4SAV
0A426 E1509E14 (01697)        STORE R5,R5SAV
0A428 E1609E16 (01698)        STORE R6,R6SAV
0A42A E1709E18 (01699)        STORE R7,R7SAV
0A42C 0E70     (01700)        RET
0A42D 0800     (01701)        EVEN
```

```
0A42E A6109F0C (01702)          LOAD R1,R1SAV
0A430 A6209F0E (01703)          LOAD R2,R2SAV
0A432 A6309F10 (01704)          LOAD R3,R3SAV
0A434 A6409F12 (01705)          LOAD R4,R4SAV
0A436 A6509F14 (01706)          LOAD R5,R5SAV
0A438 A6609F16 (01707)          LOAD R6,R6SAV
0A43A A6709F18 (01708)          LOAD R7,R7SAV
0A43C 90208C88 (01709)          MOVIR R2,SENS-2+LSEN12
0A43E 9040003E (01710)          MOVIR R4,LSEN1-1
0A440 D72815S8 (01711)          BMOVEL R2,R4,SEN1S+LSEN12
0A442 DE000580 (01712)          SMBSL 0,APDNFLS                         HAS APDONE HAPPENED?
0A444 2021     (01713)          HOP *L+34                               NO, HOP TO EVEN LOC AFTE
               (01714) ;                                                R RET
0A445 0800     (01715)          EVEN
0A446 CC000580 (01716)          MOVZM APDNFLS
0A448 F1109F0C (01717)          STORE R1,R1SAV
0A44A F1209F0E (01718)          STORE R2,R2SAV
0A44C F1309F10 (01719)          STORE R3,R3SAV
0A44E F1409F12 (01720)          STORE R4,R4SAV
0A450 F1509F14 (01721)          STORE R5,R5SAV
0A452 F1609F16 (01722)          STORE R6,R6SAV
0A454 F1709F18 (01723)          STORE R7,R7SAV
0A456 0E70     (01724)          RET
0A457 0800     (01725)          EVEN
0A458 A6109F0C (01726)          LOAD R1,R1SAV
0A45A A6209F0E (01727)          LOAD R2,R2SAV
0A45C A6309F10 (01728)          LOAD R3,R3SAV
0A45E A6409F12 (01729)          LOAD R4,R4SAV
0A460 A6509F14 (01730)          LOAD R5,R5SAV
0A462 A6609F16 (01731)          LOAD R6,R6SAV
0A464 A6709F18 (01732)          LOAD R7,R7SAV
0A466 90208D04 (01733)          MOVIR R2,SENS-2+2*LSEN12
0A468 9040003C (01734)          MOVIR R4,LSEN1-2
0A46A D72815D4 (01735)          BMOVEL R2,R4,SEN1S+2*LSEN12
0A46C DE000580 (01736)          SMBSL 0,APDNFLS                         HAS APDONE HAPPENED?
0A46E 2021     (01737)          HOP *L+34                               NO, HOP TO EVEN LOC AFTE
               (01738) ;                                                R RET
0A46F 0800     (01739)          EVEN
0A470 CC000580 (01740)          MOVZM APDNFLS
0A472 F1109F0C (01741)          STORE R1,R1SAV
0A474 F1209F0E (01742)          STORE R2,R2SAV
0A476 F1309F10 (01743)          STORE R3,R3SAV
0A478 F1409F12 (01744)          STORE R4,R4SAV
0A47A F1509F14 (01745)          STORE R5,R5SAV
```

```
0A47C F1609F16 (01746)          STORE R6,R6SAV
0A47E F1709F18 (01747)          STORE R7,R7SAV
0A480 0E70     (01748)          RET
0A481 0800     (01749)          EVEN
0A482 A6109F0C (01750)          LOAD R1,R1SAV
0A484 A6209F0E (01751)          LOAD R2,R2SAV
0A486 A6309F10 (01752)          LOAD R3,R3SAV
0A488 A6409F12 (01753)          LOAD R4,R4SAV
0A48A A6509F14 (01754)          LOAD R5,R5SAV
0A48C A6609F16 (01755)          LOAD R6,R6SAV
0A48E A6709F18 (01756)          LOAD R7,R7SAV
0A490 2000     (01757) XMITSF   NOP             *** FOR DEBUG ONLY ***
0A491 0200     (01758)          CCS 5           FROM MRB=2...
0A492 0EC0     (01759)          SCS 4           TO MRB=1
0A493 80009F30 (01760)          JMP TRANSRA     RETURN TO XMIT UPON THE
               (01761) ;                        NEXT G3 INTERRUPT
               (01762)          EJECT
```

```
                    (01763) *
                    (01764) *
                    (01765) ;
                    (01766) ;
                    (01767) ;++++++++++++++++++++++++++++++++++++++++
                    (01768) ;
                    (01769) ; MODULE XRECV.TXT: SET UP RECEIVER
                    (01770) ;
                    (01771) ;++++++++++++++++++++++++++++++++++++++++
                    (01772) ;
0A495 0800          (01773)          EVEN
0A496 CC000574      (01774) XRECVS   MOVZB,PGO                  CLEAR RCVR GO FLG
0A49B EE020581      (01775)          MOVEM HS,RCVOFF            TELL "TIMER" FL THAT CSP
                    (01776) ;                                   U RCVR CODE IS ON
                    (01777) *  MOVEB SETSG2,SYSFLGS
0A49A EE02056B      (01778)          MOVEK 1,SYNS               RESET THE SYNTHESIZER RE
                    (01779) ;                                   ADY FLAG
                    (01780)          EJECT
```

```
0A49C E819FFCF (01781) MOVESR  MOVEP SETSCO,SYSFLGS
0A49E 02A0     (01782)         CCS 2                                            FROM SRB=1...
0A49F 0E80     (01783)         SCS 3                                            TO SRB=2
0A4A0 0200     (01784)         CCS 5                                            FROM MRB=2...
0A4A1 0EC0     (01785)         SCS 4                                            TO MRB=1
0A4A2 E0200569 (01786)         MOVBR R2,SFLGS                                   'STORE TEMP. SFLG IN R2
0A4A4 F220056F (01787)         CMPMR R2,RDATAS                                  COMPARE IT WITH RDATA
               (01788) *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
               (01789) *  THIS ERROR TRAP MAY NEEDED,DELETED FOR DEBUG 5/17/80
               (01790) *  JMP EEL4S,EQ     GOTO ERROR ROUTINE IF RDATA=SFLG
               (01791) *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0A4A6 E020056F (01792)         MOVRM R2,RDATAS                                  STORE R2 IN SFLG
               (01793) ;
               (01794) ;
0A4A8 9040003C (01795)         MOVIR R4,LSEN2-1
0A4AA F200056F (01796)         CMPMZ RDATAS                                     IS SFLG =0?
0A4AC 8010A530 (01797)         JMP FL11S,EQ                                     UNLOAD BUF1 IF SFLG=0
               (01798) ;
0A4AE 90201368 (01799)         MOVIR R2,RECV2S-1                                SET UP TO UNLOAD RECV2
0A4B0 C820BFF1 (01800)         EOPMI R2,BF2S                                    'MOVE 1ST HALF-WORD, ODD
0A4B2 2721     (01801)         DECR R2,1                                        'SET UP FOR BMOVE
0A4B3 0800     (01802)         EVEN
0A4B4 D728BFF2 (01803)         BMOVEL R2,R4,BF2S+1                              'START EVEN
0A4B6 DE000580 (01804)         SMBSL 0,APDNFLS                                  HAS APDONE HAPPENED?
0A4B8 2021     (01805)         BOF #L+34                                        NO, HOP TO EVEN LOC AFTE
               (01806) ;                                                        R RET
0A4B9 0800     (01807)         EVEN
0A4BA CC000580 (01808)         MOVZM APDNFLS
0A4BC E1109F0C (01809)         STORE R1,R1SAV
0A4BE E1209F0E (01810)         STORE R2,R2SAV
0A4C0 E1309F10 (01811)         STORE R3,R3SAV
0A4C2 E1409F12 (01812)         STORE R4,R4SAV
0A4C4 E1509F14 (01813)         STORE R5,R5SAV
0A4C6 E1609F16 (01814)         STORE R6,R6SAV
0A4C8 E1709F18 (01815)         STORE R7,R7SAV
0A4CA 0E70     (01816)         RET
0A4CB 0800     (01817)         EVEN
0A4CC A6109F0C (01818)         LOAD R1,R1SAV
0A4CE A6209F0E (01819)         LOAD R2,R2SAV
0A4D0 A6309F10 (01820)         LOAD R3,R3SAV
0A4D2 A6409F12 (01821)         LOAD R4,R4SAV
0A4D4 A6509F14 (01822)         LOAD R5,R5SAV
0A4D6 A6609F16 (01823)         LOAD R6,R6SAV
0A4D8 A6709F18 (01824)         LOAD R7,R7SAV
```

```
0A4DA 902613E2 (01825)        MOVIR R2,RECV2S-1+LSEN22
0A4DC 9040003C (01826)        MOVIR R4,LSEN2-1
0A4DE B72B906C (01827)        RMOVEL R2,R4,RE2S+LSEN22+1
0A4E0 0E000580 (01828)        SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A4E2 2021     (01829)        HOP *L+34                          NO, HOP TO EVEN LOC AFTE
               (01830) ;                                         R RET
0A4E3 0800     (01831)        EVEN
0A4E4 CC000580 (01832)        MOVZM APDNELS
0A4E6 F1109F0C (01833)        STORE R1,R1SAV
0A4E8 F1209F0E (01834)        STORE R2,R2SAV
0A4EA F1309F10 (01835)        STORE R3,R3SAV
0A4EC F1409F12 (01836)        STORE R4,R4SAV
0A4EE F1509F14 (01837)        STORE R5,R5SAV
0A4F0 F1609F16 (01838)        STORE R6,R6SAV
0A4F2 F1709F18 (01839)        STORE R7,R7SAV
0A4F4 0F70     (01840)        RET
0A4F5 0800     (01841)        EVEN
0A4F6 A6109F0C (01842)        LOAD R1,R1SAV
0A4F8 A6209F0E (01843)        LOAD R2,R2SAV
0A4FA A6309F10 (01844)        LOAD R3,R3SAV
0A4FC A6409F12 (01845)        LOAD R4,R4SAV
0A4FE A6509F14 (01846)        LOAD R5,R5SAV
0A500 A6609F16 (01847)        LOAD R6,R6SAV
0A502 A6709F18 (01848)        LOAD R7,R7SAV
0A504 9020145C (01849)        MOVIR R2,RECV2S-1+2*LSEN22
0A506 9040003C (01850)        MOVIR R4,LSEN2-1
0A508 B72B90E6 (01851)        RMOVEL R2,R4,RE2S+2*LSEN22+1
0A50A 0E000580 (01852)        SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A50C 2021     (01853)        HOP *L+34                          NO, HOP TO EVEN LOC AFTE
               (01854) ;                                         R RET
0A50D 0800     (01855)        EVEN
0A50E CC000580 (01856)        MOVZM APDNELS
0A510 F1109F0C (01857)        STORE R1,R1SAV
0A512 F1209F0E (01858)        STORE R2,R2SAV
0A514 F1309F10 (01859)        STORE R3,R3SAV
0A516 F1409F12 (01860)        STORE R4,R4SAV
0A518 F1509F14 (01861)        STORE R5,R5SAV
0A51A F1609F16 (01862)        STORE R6,R6SAV
0A51C F1709F18 (01863)        STORE R7,R7SAV
0A51E 0F70     (01864)        RET
0A51F 0800     (01865)        EVEN
0A520 A6109F0C (01866)        LOAD R1,R1SAV
0A522 A6209F0E (01867)        LOAD R2,R2SAV
0A524 A6309F10 (01868)        LOAD R3,R3SAV
```

```
0A526 A6409E12 (01869)          LOAD R4,R4SAV
0A528 A6509E14 (01870)          LOAD R5,R5SAV
0A52A A6609E16 (01871)          LOAD R6,R6SAV
0A52C A6709E18 (01872)          LOAD R7,R7SAV
               (01873) ;
0A52E 8000A592 (01874)          JMP FL12S
               (01875) ;
               (01876) ;
0A530 902011E6 (01877) FL11S    MOVIR R2,RECV1S-2                  SET UP TO UNLOAD RECV1
0A532 D7288E80 (01878)          BMOVEL R2,R4,BF1S
0A534 DE000580 (01879)          SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A536 2005     (01880)          BOP #L+6
0A537 0800     (01881)          EVEN
0A538 CC000580 (01882)          MOVZM APDNELS
0A53A 0E70     (01883)          RET
0A53B 0800     (01884)          EVEN
0A53C 90201270 (01885)          MOVIR R2,RECV1S-2+LSEN22
0A53E 9040003C (01886)          MOVIR R4,LSEN2-1
0A540 D7288EEA (01887)          BMOVEL R2,R4,BF1S+LSEN22
0A542 DE000580 (01888)          SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A544 2005     (01889)          BOP #L+6
0A545 0800     (01890)          EVEN
0A546 CC000580 (01891)          MOVZM APDNELS
0A548 0E70     (01892)          RET
0A549 0800     (01893)          EVEN
0A54A 902012EA (01894)          MOVIR R2,RECV1S-2+2*LSEN22
0A54C 9040003D (01895)          MOVIR R4,LSEN2
0A54E D7288E74 (01896)          BMOVEL R2,R4,BF1S+2*LSEN22
0A550 DE000580 (01897)          SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A552 2005     (01898)          BOP #L+6
0A553 0800     (01899)          EVEN
0A554 CC000580 (01900)          MOVZM APDNELS
0A556 0E70     (01901)          RET
0A557 0800     (01902)          EVEN
0A558 90201365 (01903)          MOVIR R2,RECV1S-2+3*LSEN22+1
0A55A C820BEF0 (01904)          POPMI R2,BF1S+LSEN-1               'MOVE LAST HALF WORD
0A55C 2721     (01905)          DECR R2,1
0A55D 0800     (01906)          EVEN
0A55E C82092D2 (01907)          POPMI R2,BF3S+LSEN-1               'TO BOTH BUFFERS
0A560 DE000580 (01908)          SMBSL 0,APDNELS                    HAS APDONE HAPPENED?
0A562 2005     (01909)          BOP #L+6
0A563 0800     (01910)          EVEN
0A564 CC000580 (01911)          MOVZM APDNELS
0A566 0E70     (01912)          RET
```

```
0A567 0800       (01913)          EVEN
0A568 90201FE6   (01914)          MOVIR R2,RECV1S-2                    'MOVE SAME BLOCKS TO BUF
                 (01915) ;                                              3
0A56A 9040003C   (01916)          MOVIR R4,LSEN2-1
0A56C E72891E2   (01917)          RMOVEL R2,R4,BF3S
0A56E DE000580   (01918)          SMBSL 0,APDNELS                      HAS APDONE HAPPENED?
0A570 2005       (01919)          HOP *L+6
0A571 0800       (01920)          EVEN
0A572 CC000580   (01921)          MOVZM APDNELS
0A574 0E70       (01922)          RET
0A575 0800       (01923)          EVEN
0A576 90201270   (01924)          MOVIR R2,RECV1S-2+LSEN22
0A578 9040003C   (01925)          MOVIR R4,LSEN2-1
0A57A D72891DC   (01926)          RMOVEL R2,R4,BF3S+LSEN22
0A57C DE000580   (01927)          SMBSL 0,APDNELS                      HAS APDONE HAPPENED?
0A57E 2005       (01928)          HOP *L+6
0A57F 0800       (01929)          EVEN
0A580 CC000580   (01930)          MOVZM APDNELS
0A582 0E70       (01931)          RET
0A583 0800       (01932)          EVEN
0A584 902012FA   (01933)          MOVIR R2,RECV1S-2+2*LSEN22
0A586 9040003D   (01934)          MOVIR R4,LSEN2
0A588 E7289256   (01935)          RMOVEL R2,R4,BF3S+2*LSEN22
0A58A DE000580   (01936)          SMBSL 0,APDNELS                      HAS APDONE HAPPENED?
0A58C 2005       (01937)          HOP *L+6
0A58D 0800       (01938)          EVEN
0A58E CC000580   (01939)          MOVZM APDNELS
0A590 0E70       (01940)          RET
0A591 0800       (01941)          EVEN
0A592 0280       (01942) FL12S    CCS 3                                FROM SRB=2...
0A593 06A0       (01943)          SCS 2                                TO SRB=1
                 (01944) *SYNC CHECK AND UPDATE,SEARCH IF NECESSARY.
                 (01945) ;   R1 - PTR TO THE RMODEM BUFFER BEING SEARCHED
                 (01946)
                 (01947) ;   R2 - OFFSET INTO THE RMODEM, RSSPE, RSSSS BUFFERS
                 (01948) ;   R3 - SCRATCH AC
                 (01949) ;   R4 - MAXCNT, THE MAX. OF RSSSS(I)
                 (01950) ;   R5 - NMAX, THE # OF INSTANCES OF MAXCNT IN RSSSS(I)
                 (01951) ;   R6 - IMAX, THE FRAME OFFSET WHERE MAXCNT WAS FOUND
                 (01952) ; VARIABLES:
                 (01953) ;   SYNCS - BEGINNING OF FRAME OFFSET TO THE SYNC WORD IN THE
                 (01954) ;  RMODEM BUFFER
0A594 9010RER0   (01955) SCHKS    MOVIR R1,RE1S                        PREPARE TO USE BF1 AS NE
                 (01956) ;                                              W BUFFER
```

```
0A596 F200056F (01957)          CMPMZ RDATAS                          CHECK ASSUMPTION
0A598 8010A59C (01958)          JMP SCHKIS,FO
0A59A 9010BFF1 (01959)          MOVIR R1,BF2S                         NON ZERO MEANS USE BF2
0A59C F2000578 (01960) SCHKIS   CMPMZ RSYN                            ;WHAT DO WE KNOW ABOUT S
               (01961) ;                                              YNC?
               (01962)          EVEN
0A59E 8610A5A4 (01963)          JMP RMISLS,EQZ                        ;0 MEANS NIL, SO INIT
0A5A0 8630A5AC (01964)          JMP RMISSP,LTZ                        ;-1 MEANS WE LACK IT, SO
0A5A2 8000A656 (01965)          JMP UNFEELS                           SYNCH OK, SO COPY BITS
      00000001 (01966)          IPWSRD=1
      00000004 (01967)          LSTHP=4
      0000000A (01968)          ACOTH=10
      00000171 (01969)          MBLTH=369
               (01970)          EVEN
0A5A4 9031FFFF (01971) RMISLS:  MOVIR R3,-1
0A5A6 F0300578 (01972)          MOVRM R3,RSYN                         ;SAY WE'VE LOST IT
0A5A8 8610A5AF (01973) RMISSIN  CALL R1,SYNIN
0A5AA 80009F66 (01974)          JMP RECVSRA
0A5AC 8000A5C4 (01975) RMISSP   JMP SYNSR
               (01976) ; SYNIN: INITIALIZE BUFFERS, ETC. FOR SYNC SEARCHING.  CALL WITH
               (01977) ;   R1 POINTING TO RMODEM BUFFER.
               (01978) ; ALSO ZEROS THE ENTIRE "LAST" RBITS BUFFER.
               (01979)           EVEN
0A5AF 90200170 (01980) SYNIN    MOVIR R2,MBLTH-1                      ;BUFFER OFFSET = LAST WO
               (01981) ;                                              RD IN BUFFER
0A5B0 4C14     (01982)          ADDRR R1,R2
0A5B1 70320001 (01983) SISLP    MOVWR R3,R1,1                         ;PICK UP DATA BIT (BIT 0
               (01984) ;                                              )
0A5B3 F0349935 (01985)          MOVRM R3,RSSPF(R2)                    ; AND STASH IT IN RSSPF(
               (01986) ;                                              I)
0A5B5 2711     (01987)          DECR R1,1
0A5B6 8C20A5B1 (01988)          DJP R2,SISLP
0A5B8 F0200578 (01989)          MOVRM R2,RSYN                         ;SET RSYN=-1 TO MEAN SEA
               (01990) ;                                              RCH FOR SYNC
0A5BA 964097C2 (01991) SISBMO   MOVIR R4,RSSSS-WS                     ;CLEAR THE RSSSS BUFFER
0A5BC C10097C4 (01992)          MOVZML RSSSS
0A5BE 962000B8 (01993)          MOVIR R2,MBLTH/2
0A5C0 D74497C6 (01994)          BMOVEL R4,R2,RSSSS+WS
0A5C2 0F70     (01995)          RETURN
               (01996) ;
               (01997) ;
               (01998) ; SYNSR: SEARCH INCOMING RMODEM BUFFER FOR SYNC, AND IF SYNC
               (01999) ;   IS FOUND, SET RSYN TO +1 AND SET SYNCS TO THE FRAME OFFSET
               (02000) ;   CORRESPONDING TO THE LOGICAL START OF THE FRAME.
```

```
                (02001) ; CALL WITH R1 POINTING TO THE START OF THE RMODEM BUFFER TO BE
                (02002) ;    SEARCHED.
                (02003) ; REGISTER USAGE:
                (02004) ; BUFFERS:
                (02005) ;    RSSPF - SYNC-SEARCH PREVIOUS FRAME - STORES THE DATA BIT OF THE
                (02006) ;   PREVIOUS RMODEM BUFFER
                (02007) ;    RSSSS - SYNC-SEARCH SUM-OF-SYNC COUNTS - FOR EACH RMODEM WORD,
                (02008) ;   STORES THE NUMBER OF CONSECUTIVE FRAMES OF DATA BIT
                (02009) ;   ALTERNATION.
0A5C3 0800      (02010)           EVEN
0A5C4 90200170  (02011) SYNSR     MOVIR R2,MBLTH-1               ;INIT BUFFER OFFSET TO L
                (02012) ;                                        AST IN FRAME
0A5C6 4C14      (02013)           ADDRR R1,R2                    ;MAKE R1 POINT THERE TOO
0A5C7 0D40      (02014)           CLR R4                         ;MAXCNT <- 0
0A5C8 DE0005B0  (02015) SSSLP     SMBSL 0,APDNFLS                HAS APDONE HAPPENED?
0A5CA 2021      (02016)           HOP *L+34                      NO, HOP TO EVEN LOC AFTE
                (02017) ;                                        R RET
0A5CB 0800      (02018)           EVEN
0A5CC CC0005B0  (02019)           MOVZR APDNFLS
0A5CE F1109F0C  (02020)           STORE R1,R1SAV
0A5D0 F1209F0E  (02021)           STORE R2,R2SAV
0A5D2 F1309F10  (02022)           STORE R3,R3SAV
0A5D4 F1409F12  (02023)           STORE R4,R4SAV
0A5D6 F1509F14  (02024)           STORE R5,R5SAV
0A5D8 F1609F16  (02025)           STORE R6,R6SAV
0A5DA F1709F18  (02026)           STORE R7,R7SAV
0A5DC 0F70      (02027)           RET
0A5DD 0800      (02028)           EVEN
0A5DE A6109F0C  (02029)           LOAD R1,R1SAV
0A5E0 A6209F0E  (02030)           LOAD R2,R2SAV
0A5E2 A6309F10  (02031)           LOAD R3,R3SAV
0A5E4 A6409F12  (02032)           LOAD R4,R4SAV
0A5E6 A6509F14  (02033)           LOAD R5,R5SAV
0A5E8 A6609F16  (02034)           LOAD R6,R6SAV
0A5EA A6709F18  (02035)           LOAD R7,R7SAV
0A5EC 70320001  (02036)           MOVAR R3,R1,1LWSRD             ;R3 <- NEW DATA BIT (OTH
                (02037) ;                                        ER BITS MASKED)
0A5EE F2349945  (02038)           CMPRB R3,RSSPF(R2)             ;DOES THIS BIT ALTERNATE
0A5F0 B010A662  (02039)           JMP SSSNAL,EQ                  ;JUMP IF NO ALTERNATION
0A5F2 F50497C4  (02040) SSSALT    INCR RSSSS(R2)                 ;YES: COUNT THE BIT ALTE
                (02041) ;                                        RNATION
0A5F4 F24497C4  (02042)           CMPRF R4,RSSSS(R2)             ;IS MAXCNT >= RSSSS(I)?
0A5F6 B130A5FF  (02043)           JMP NOMAX,GE                   ;YES
0A5F8 F04497C4  (02044) NEWMAX    MOVMR R4,RSSSS(R2)             ;NO: RSSSS(I) BECOMES TH
```

```
              (02045) ;                                                  E NEW MAXCNT
0A5FA 96500001 (02046)          MOVIR R5,1                               ;NMAX <- 1 (UNIQUE SO FA
              (02047) ;                                                  R)
0A5FC 4064    (02048)          MOVRR R6,R2                               ;AND WE SAVE THE OFFSET
              (02049) ;                                                  IN IMAX
0A5FD 2006    (02050)          BOP SSSLP1
              (02051) ;
              (02052)          EVEN
0A5FE B110A604 (02053) NOMAX   JMP SSSLP1,NE                            ;MAXCNT = RSSSS(I)?
0A600 2651    (02054)          INCR R5,1                                 ;YES, SO THE MAX ISN'T U
              (02055) ;                                                  NIQUE
0A601 2002    (02056)          BOP SSSLPT
              (02057) ;
              (02058)          EVEN
0A602 CC0497C4 (02059) SSSMAL  MOVZM RSSSS(R2)                          ;CLEAR RSSSS(I)
0A604 F0349935 (02060) SSSLPT  MOVRM R3,RSSPF(R2)                       ;SAVE NEW DATA BIT IN RS
              (02061) ;                                                  SPF(I)
0A606 2711    (02062)          DECR R1,1                                 ;DECR RMODEM BUFFER PTR
0A607 0800    (02063)          EVEN
0A608 8C20A5C8 (02064)         DJP R2,SSSLP                              ;DECR OFFSET AND TEST IF
              (02065) ; WE HAVE NOW SCANNED THE WHOLE BUFFER.  IS MAXCNT >= THE SYNC
              (02066) ;   ACQUISITION THRESHOLD, AND IT IT A UNIQUE MAX?  IF SO, WE HAVE
              (02067) ;   FOUND SYNC.
0A60A 9240006A (02068)         CMPIR R4,ACQTH                            ;MAXCNT >= ACQTH?
0A60C B030A62A (02069)         JMP SSSNO,LT                              ;NO
0A60E 92500001 (02070)         CMPIR R5,1                                ;YES: IS NMAX=1?
0A610 B110A62A (02071)         JMP SSSNO,NE                              ;NO
0A612 F0500578 (02072)         MOVRM R5,RSYN                             ;YES: WE HAVE SYNC. SET
              (02073) ;                                                  RSYN TO +1
0A614 F0600577 (02074)         MOVRM R6,SYNCS                            ;SAVE OFFSET IN SYNCS
0A616 90300004 (02075)         MOVIR R3,LSTHR
0A618 F030057C (02076)         MOVRM R3,FRSYN                            ;INIT FRSYN TO COUNT ERR
              (02077) ;                                                  ORS
0A61A 9031FFFC (02078)         MOVIR R3,-4
0A61C F0300579 (02079)         MOVRM R3,NEWSYN                           SET # OF FRAMES TO WAIT
              (02080) ;                                                  BEFORE PUTTIN OUT SPEECH
0A61E F0509C84 (02081)         MOVRM R5,FRSTF                            SET FIRST FRAME AFTER SY
              (02082) ;                                                  NC FLAG
0A620 CC00057B (02083)         MOVZM LSTER                               ; FOR SUPDAT NEXT FRAME
0A622 9C1C0001 (02084)         ADDIR R1,1(R6)                            ;R1 POINTS TO WORD HOLDI
              (02085) ;                                                  NG SYNC
0A624 70320001 (02086)         MOVWR R3,R1,1DWSRD                        ;PICK UP SYNC BIT AND SA
              (02087) ;                                                  VE IT IN
0A626 F030057A (02088)         MOVRM R3,OLSYN                            ;OLSYN FOR USE IN SUPDAT
```

```
0A628 80009E60 (02089)          JMP RECVSEA
               (02090) ;
               (02091)          EVEN
0A62A F0200578 (02092) SSSSO    MOVEM R2,RSYN                    ;NO SYNC YET: SET TO -1
               (02093) ;                                         (=SEARCH)
0A62C 80009E60 (02094)          JMP RECVSEA                      ; FOR GOOD MEASURE
               (02095) ;
               (02096) ;
               (02097) ; SUPDAT:  CHECK RMODEM BUFFER TO SEE IF THE DATA BIT AT THE
               (02098) ;   EXPECTED SYNC POSITION (START OF BUFFER+SYNCS) HAS THE EXPECTED
               (02099) ;   VALUE.  IF THERE ARE LSTHR ERRORS WITHOUT 2 CONSECUTIVE CORRECT
               (02100) ;   COMPARISONS, CLEAR RSYN TO SIGNIFY LOSS OF FRAME SYNCHRONIZATION.
               (02101) ; ENTER WITH R1 POINTING TO THE LATEST RMODEM BUFFER.  ALSO, THE
               (02102) ;   WORD SYNCS HOLDS THE OFFSET IN THE FRAME THAT POINTS TO THE
               (02103) ;   SYNC WORD TO BE VERIFIED.
               (02104) ;  CALLED FROM RCVR MODEM INTERRUPT SERVICE ROUTINE.
               (02105) ; VARIABLES:
               (02106) ;   SYNCS - BEGINNING OF FRAME OFFSET
               (02107) ;   OLSYN - SYNC BIT FROM PREVIOUS FRAME
               (02108) ;   LSTER - NZ IF SYNC ERROR ON PREVIOUS FRAME
               (02109) ;   ERSYN - COUNTS SYNC ERRORS (FROM LSTHR DOWN TO 0)
               (02110)          EVEN
0A62E EC100577 (02111) SUPDAT   ADDER R1,SYNCS                   ;R1 NOW POINTS TO NEW SY
               (02112) ;                                         NC WORD
0A630 F0A0A654 (02113)          MOVER R2,BBUS2                   0(2) INDEXED BY BUFN+SYN
               (02114) ;                                         CS
0A632 E420057A (02115)          XORMR R2,OLSYN                   ;XOR WITH LAST SYNC BIT
0A634 2C04     (02116)          SRBS 0,R2                        ;SKIP IF DATA BIT SET (G
               (02117) ;                                         OOD SYNC)
0A635 200C     (02118)          BOP SUSERR
0A636 F2000578 (02119)          CMPMZ LSTER                      ;WAS THERE AN ERROR LAST
0A638 811OA63E (02120)          JMP SUSELE,NEZ                   ;YES
0A63A 90300004 (02121)          MOVIR R3,LSTHR                   ;NO:
0A63C F030057C (02122)          MOVRM R3,ERSYN                   ; CORRECT FRAMES, SO RES
               (02123) ;                                         ET ERROR COUNT
0A63E CC000578 (02124) SUSELE   MOVZM LSTER                      ;REMEMBER NONERROR FOR N
               (02125) ;                                         EXT FRAME
0A640 2009     (02126)          BOP SUSVAL
               (02127) ;
0A641 0800     (02128)          EVEN
0A642 90200001 (02129) SUSERR   MOVIR R2,1                       ;COME HERE ON SYNC BIT E
               (02130) ;                                         RROR
0A644 F0200578 (02131)          MOVRM R2,LSTER                   ;REMEMBER ERROR FOR NEXT
0A646 E400057C (02132)          DECM ERSYN                       ;HAVE WE COUNTED LSTHR E
```

```
                      (02133) ;                                   RRORS?
0A638 E1200A650       (02134)          JMP SUSLS,LEZ              ;YES, THEREFORE WE'VE LO
                      (02135) ;                                   ST SYNC
0A63A 90100001        (02136) SUSVAL   MOVIR R1,ID*SED            ;COMPLEMENT OLSYN IN PRE
                      (02137) ;                                   PARATION
0A63C F510057A        (02138)          XORRM R1,OLSYN             ; FOR USE NEXT FRAME
0A64E 0F70            (02139)          RETURN
                      (02140) ;
0A64F 0800            (02141)          EVEN
0A650 CC00057B        (02142) SUSLS    MOVZM RSYN
0A652 0F70            (02143)          RETURN
0A653 0800            (02144)          EVEN
0A654 00220000        (02145) BUS2     ADDR 0(1,2)                0(2) INDEXED BY R1
                      (02146) ;
                      (02147) UNPEELS  EVEN
0A656 02B0            (02148)          CCS 3                      FROM SRB=2...
0A657 06A0            (02149)          SCS 2                      TO SRB=1
0A658 02D0            (02150)          CCS 5                      FROM MRB=2...
0A659 06C0            (02151)          SCS 4                      TO MRB=1
0A65A F200056F        (02152)          CMEMZ RDATAS
0A65C 8010A662        (02153)          JMP FL16S,EQ
0A65E 9020BE7F        (02154)          MOVIR R2,BF1S-1            NEW DATA IN BUFF 2, SO R
                      (02155) ;                                   EAD BUFF1+SYNC
0A660 8000A664        (02156)          JMP FL17S
0A662 9020BFF0        (02157) FL16S    MOVIR R2,BF2S-1            BUFF1(BUFF3) NEW, SO REA
                      (02158) ;                                   D BUFF2+SYNC
0A664 FC200577        (02159) FL17S    ADDMR R2,SYNCS
0A666 9040003D        (02160)          MOVIR R4,LSEN1-1
0A668 D5285544        (02161)          RMOVE R2,R1,RECVS
0A66A DF000580        (02162)          SMBSL 0,APDNELS            HAS APDONE HAPPENED?
0A66C 2021            (02163)          HOP *L+34                  NO, HOP TO EVEN LOC AFTE
                      (02164) ;                                   R RET
0A66D 0800            (02165)          EVEN
0A66E CC000580        (02166)          MOVZM APDNELS
0A670 F1109F0C        (02167)          STORE R1,R1SAV
0A672 F1209F0E        (02168)          STORE R2,R2SAV
0A674 F1309F10        (02169)          STORE R3,R3SAV
0A676 F1409F12        (02170)          STORE R4,R4SAV
0A678 F1509F14        (02171)          STORE R5,R5SAV
0A67A F1609F16        (02172)          STORE R6,R6SAV
0A67C F1709F18        (02173)          STORE R7,R7SAV
0A67E 0F70            (02174)          RET
0A67F 0800            (02175)          EVEN
0A680 A6109F0C        (02176)          LOAD R1,R1SAV
```

```
0A682 A6209E0E (02177)          LOAD R2,R2SAV
0A684 A6309E10 (02178)          LOAD R3,R3SAV
0A686 A6409E12 (02179)          LOAD R4,R4SAV
0A688 A6509E14 (02180)          LOAD R5,R5SAV
0A68A A6609E16 (02181)          LOAD R6,R6SAV
0A68C A6709E18 (02182)          LOAD R7,R7SAV
0A68E 9040003D (02183)          MOVIR R4,LSEN1-1
0A690 D5289582 (02184)          RMOVE R2,R4,RECVS+LSEN1
0A692 DE000580 (02185)          SMRSL 0,APDNELS          HAS APDONE HAPPENED?
0A694 2021     (02186)          HOP #L+34                NO, HOP TO EVEN LOC AFTE
               (02187) ;                                 R RET
0A695 0800     (02188)          EVEN
0A696 CC000580 (02189)          MOVZM APDNELS
0A698 E1109E0C (02190)          STORE R1,R1SAV
0A69A E1209E0E (02191)          STORE R2,R2SAV
0A69C E1309E10 (02192)          STORE R3,R3SAV
0A69E E1409E12 (02193)          STORE R4,R4SAV
0A6A0 E1509E14 (02194)          STORE R5,R5SAV
0A6A2 E1609E16 (02195)          STORE R6,R6SAV
0A6A4 E1709E18 (02196)          STORE R7,R7SAV
0A6A6 0E70     (02197)          RET
0A6A7 0800     (02198)          EVEN
0A6A8 A6109E0C (02199)          LOAD R1,R1SAV
0A6AA A6209E0E (02200)          LOAD R2,R2SAV
0A6AC A6309E10 (02201)          LOAD R3,R3SAV
0A6AE A6409E12 (02202)          LOAD R4,R4SAV
0A6B0 A6509E14 (02203)          LOAD R5,R5SAV
0A6B2 A6609E16 (02204)          LOAD R6,R6SAV
0A6B4 A6709E18 (02205)          LOAD R7,R7SAV
0A6B6 9040003D (02206)          MOVIR R4,LSEN1-1
0A6B8 D52895C0 (02207)          RMOVE R2,R4,RECVS+2*LSEN1
0A6BA DE000580 (02208)          SMRSL 0,APDNELS          HAS APDONE HAPPENED?
0A6BC 2021     (02209)          HOP #L+34                NO, HOP TO EVEN LOC AFTE
               (02210) ;                                 R RET
0A6BD 0800     (02211)          EVEN
0A6BE CC000580 (02212)          MOVZM APDNELS
0A6C0 E1109E0C (02213)          STORE R1,R1SAV
0A6C2 E1209E0E (02214)          STORE R2,R2SAV
0A6C4 E1309E10 (02215)          STORE R3,R3SAV
0A6C6 E1409E12 (02216)          STORE R4,R4SAV
0A6C8 E1509E14 (02217)          STORE R5,R5SAV
0A6CA E1609E16 (02218)          STORE R6,R6SAV
0A6CC E1709E18 (02219)          STORE R7,R7SAV
0A6CE 0E70     (02220)          RET
```

```
0A6CE 0800      (02221)       EVEN
0A6D0 A6109F0C  (02222)       LOAD R1,R1SAV
0A6D2 A6209F0E  (02223)       LOAD R2,R2SAV
0A6D4 A6309F10  (02224)       LOAD R3,R3SAV
0A6D6 A6409F12  (02225)       LOAD R4,R4SAV
0A6D8 A6509F14  (02226)       LOAD R5,R5SAV
0A6DA A6609F16  (02227)       LOAD R6,R6SAV
0A6DC A6709F18  (02228)       LOAD R7,R7SAV
0A6DE 9040003D  (02229)       MOVIR R4,LSEN1-1
0A6E0 D52895FF  (02230)       RMOVE R2,R4,RECVS+4*LSEN1
0A6E2 DE000580  (02231)       SMBSL 0,APDNELS                HAS APDONE HAPPENED?
0A6E4 2021      (02232)       HOP #L+34                      NO, HOP TO EVEN LOC AFTE
                (02233) ;                                    R RET
0A6E5 0800      (02234)       EVEN
0A6E6 CC000580  (02235)       MOVZM APDNELS
0A6E8 E1109F0C  (02236)       STORE R1,R1SAV
0A6EA E1209F0E  (02237)       STORE R2,R2SAV
0A6EC E1309F10  (02238)       STORE R3,R3SAV
0A6EE E1409F12  (02239)       STORE R4,R4SAV
0A6F0 E1509F14  (02240)       STORE R5,R5SAV
0A6F2 E1609F16  (02241)       STORE R6,R6SAV
0A6F4 E1709F18  (02242)       STORE R7,R7SAV
0A6F6 0E70      (02243)       RET
0A6F7 0800      (02244)       EVEN
0A6F8 A6109F0C  (02245)       LOAD R1,R1SAV
0A6FA A6209F0E  (02246)       LOAD R2,R2SAV
0A6FC A6309F10  (02247)       LOAD R3,R3SAV
0A6FE A6409F12  (02248)       LOAD R4,R4SAV
0A700 A6509F14  (02249)       LOAD R5,R5SAV
0A702 A6609F16  (02250)       LOAD R6,R6SAV
0A704 A6709F18  (02251)       LOAD R7,R7SAV
0A706 9040003D  (02252)       MOVIR R4,LSEN1-1
0A708 D528963C  (02253)       RMOVE R2,R4,RECVS+4*LSEN1
0A70A DE000580  (02254)       SMBSL 0,APDNELS                HAS APDONE HAPPENED?
0A70C 2021      (02255)       HOP #L+34                      NO, HOP TO EVEN LOC AFTE
                (02256) ;                                    R RET
0A70D 0800      (02257)       EVEN
0A70E CC000580  (02258)       MOVZM APDNELS
0A710 E1109F0C  (02259)       STORE R1,R1SAV
0A712 E1209F0E  (02260)       STORE R2,R2SAV
0A714 E1309F10  (02261)       STORE R3,R3SAV
0A716 E1409F12  (02262)       STORE R4,R4SAV
0A718 E1509F14  (02263)       STORE R5,R5SAV
0A71A E1609F16  (02264)       STORE R6,R6SAV
```

```
0A71C E1709F18 (02265)         STORE R7,R7SAV
0A71E 0E70     (02266)         RET
0A71F 0800     (02267)         EVEN
0A720 A6109F0C (02268)         LOAD R1,R1SAV
0A722 A6209F0E (02269)         LOAD R2,R2SAV
0A724 A6309F10 (02270)         LOAD R3,R3SAV
0A726 A6409F12 (02271)         LOAD R4,R4SAV
0A728 A6509F14 (02272)         LOAD R5,R5SAV
0A72A A6609F16 (02273)         LOAD R6,R6SAV
0A72C A6709F18 (02274)         LOAD R7,R7SAV
0A72E 9040003A (02275)         MOVIR R4,LSEN1-4
0A730 D5289E7A (02276)         RMOVE R2,R4,RECVS+5*LSEN1
0A732 DE0005R0 (02277)         SMBSL 0,APDNELS                 HAS APDONE HAPPENED?
0A734 2C21     (02278)         HOP #L+14                       NO, HOP TO EVEN LOC AFTE
               (02279) ;                                       R RET
0A735 0800     (02280)         EVEN
0A736 CC0005R0 (02281)         MOVZB APDNELS
0A738 E1109F0C (02282)         STORE R1,R1SAV
0A73A E1209F0E (02283)         STORE R2,R2SAV
0A73C E1309F10 (02284)         STORE R3,R3SAV
0A73E E1409F12 (02285)         STORE R4,R4SAV
0A740 E1509F14 (02286)         STORE R5,R5SAV
0A742 E1609F16 (02287)         STORE R6,R6SAV
0A744 E1709F18 (02288)         STORE R7,R7SAV
0A746 0E70     (02289)         RET
0A747 0800     (02290)         EVEN
0A748 A6109F0C (02291)         LOAD R1,R1SAV
0A74A A6209F0E (02292)         LOAD R2,R2SAV
0A74C A6309F10 (02293)         LOAD R3,R3SAV
0A74E A6409F12 (02294)         LOAD R4,R4SAV
0A750 A6509F14 (02295)         LOAD R5,R5SAV
0A752 A6609F16 (02296)         LOAD R6,R6SAV
0A754 A6709F18 (02297)         LOAD R7,R7SAV
               (02298) *  JMP DSRBUFS     NOT ENOUGH TIME TO CORRECT
0A756 8000A758 (02299)         JMP BCHDOS                      CORRECT SIDEBAND
               (02300)         EJECT
```

```
                   (02301) ***** BCH DECODING OF SIDEBAND & PART OF MAINBAND DATA
                   (02302) BCHD0S  EVEN
0A758 90100001 (02303)         MOVIR R1,1                    SET INPUT BUFFER POINTER
0A75A CF009FEA (02304)         MOVZML NESS-2                 CLEAR ERROR COUNTERS(R1,
                   (02305) ;                                 R2)
0A75C DF000580 (02306) BCHD1S  SMBSL 0,APDNELS               HAS APDONE HAPPENED?
0A75E 2021       (02307)         HOP #1+34                   NO, HOP TO EVEN LOC AFTE
                   (02308) ;                                 R RET
0A75F 0800       (02309)         EVEN
0A760 CC000580 (02310)         MOVZM APDNELS
0A762 F1109F0C (02311)         STORE R1,R1SAV
0A764 F1209F0E (02312)         STORE R2,R2SAV
0A766 F1309F10 (02313)         STORE R3,R3SAV
0A768 F1409F12 (02314)         STORE R4,R4SAV
0A76A F1509F14 (02315)         STORE R5,R5SAV
0A76C F1609F16 (02316)         STORE R6,R6SAV
0A76E F1709F18 (02317)         STORE R7,R7SAV
0A770 0F70       (02318)         RET
0A771 0800       (02319)         EVEN
0A772 A6109F0C (02320)         LOAD R1,R1SAV
0A774 A6209F0E (02321)         LOAD R2,R2SAV
0A776 A6309F10 (02322)         LOAD R3,R3SAV
0A778 A6409F12 (02323)         LOAD R4,R4SAV
0A77A A6509F14 (02324)         LOAD R5,R5SAV
0A77C A6609F16 (02325)         LOAD R6,R6SAV
0A77E A6709F18 (02326)         LOAD R7,R7SAV
0A780 905000BA (02327)         MOVIR R5,1R6                  R5=# OF POWER SUM TABLE
0A782 0020       (02328)         CLR R2                      CLEAR POWER SUM S1
0A783 0030       (02329)         CLR R3                      CLEAR POWER SUM S3
0A784 0040       (02330)         CLR R4                      CLEAR POWER SUM S5
0A785 0800       (02331)         EVEN
0A786 DF029544 (02332) BCHD2S  SMBSL 0,RECVS(R1)             READ INPUT DATA
0A788 2009       (02333)         HOP BCHD3S
0A789 0800       (02334)         EVEN
0A78A F06A9CFC (02335)         MOVMR R6,TBPSMS(R5)
0A78C 442C       (02336)         XORRR R2,R6
0A78D 0800       (02337)         EVEN
0A78E C06A9CFD (02338)         MOVMRL R6,TBPSMS+1(R5)
0A790 443C       (02339)         XORRR R3,R6
0A791 444F       (02340)         XORRR R4,R7
0A792 DF000580 (02341) BCHD3S  SMBSL 0,APDNELS               HAS APDONE HAPPENED?
0A794 2021       (02342)         HOP #1+34                   NO, HOP TO EVEN LOC AFTE
                   (02343) ;                                 R RET
0A795 0800       (02344)         EVEN
```

```
0A796 CC000580          MOVZM APDNFLS
0A798 E1109E0C          STORE R1,R1SAV
0A79A E1209E0E          STORE R2,R2SAV
0A79C E1309E10          STORE R3,R3SAV
0A79E E1409E12          STORE R4,R4SAV
0A7A0 E1509E14          STORE R5,R5SAV
0A7A2 E1609E16          STORE R6,R6SAV
0A7A4 E1709E18          STORE R7,R7SAV
0A7A6 0E70              RET
0A7A7 0800              EVEN
0A7A8 A6109E0C          LOAD R1,R1SAV
0A7AA A6209E0E          LOAD R2,R2SAV
0A7AC A6309E10          LOAD R3,R3SAV
0A7AE A6409E12          LOAD R4,R4SAV
0A7B0 A6509E14          LOAD R5,R5SAV
0A7B2 A6609E16          LOAD R6,R6SAV
0A7B4 A6709E18          LOAD R7,R7SAV
0A7B6 2611              INCR R1,1                             INCREAMENT INPUT BUFFER
                      ;                                       POINTER
0A7B7 2753              DECR R5,3
0A7B8 R130A786          JMP BCBD2S,GE
                      ; POWER SUMS(S1,S3,S5) ARE CALCULATEED AND STORED IN R2,R3,R4
                      ; CHECK THE RANGE OF ERRORS
0A7BA DE000580          SMBSL 0,APDNFLS                       HAS APDONE HAPPENED?
0A7BC 2021              HOP #L+34                             NO, HOP TO EVEN LOC AFTE
                      ;                                       R RET
0A7BD 0800              EVEN
0A7BE CC000580          MOVZM APDNFLS
0A7C0 E1109E0C          STORE R1,R1SAV
0A7C2 E1209E0E          STORE R2,R2SAV
0A7C4 E1309E10          STORE R3,R3SAV
0A7C6 E1409E12          STORE R4,R4SAV
0A7C8 E1509E14          STORE R5,R5SAV
0A7CA E1609E16          STORE R6,R6SAV
0A7CC E1709E18          STORE R7,R7SAV
0A7CE 0E70              RET
0A7CF 0800              EVEN
0A7D0 A6109E0C          LOAD R1,R1SAV
0A7D2 A6209E0E          LOAD R2,R2SAV
0A7D4 A6309E10          LOAD R3,R3SAV
0A7D6 A6409E12          LOAD R4,R4SAV
0A7D8 A6509E14          LOAD R5,R5SAV
0A7DA A6609E16          LOAD R6,R6SAV
0A7DC A6709E18          LOAD R7,R7SAV
```

```
0A7DE 1054      (02389)          MOVRR R5,R2
0A7DF 1056      (02390)          ADDRR R5,R3
0A7E0 1058      (02391)          ADDRR R5,R4
0A7E1 0800      (02392)          EVEN
0A7E2 8010AR94  (02393)          JMP BCHDAS,EQ                        NO ERROR IF R5=0
                (02394) ;  CALCULATE DET3=R7=S1**3+S3=R2**3+R3
0A7E4 CC009FFA  (02395)          MOVZM ERSTS                          CLEAR ERROR COUNTER
0A7E6 1074      (02396)          MOVRR R7,R2
0A7E7 0800      (02397)          EVEN
0A7E8 8010AR00  (02398)          JMP BCHDSS,EQ
0A7EA F0749EF8  (02399)          MOVMR R7,TRRPCS(R2)
0A7EC 406E      (02400)          MOVRR R6,R7
0A7ED 3CCC      (02401)          ADDRR R6,R6
0A7EE 9260003F  (02402)          CMPIR R6,63                          CHECK R6<63
0A7F0 1A30      (02403)          SKPL LT
0A7F1 9E60003F  (02404)          SUBIR R6,63
0A7F3 0800      (02405)          EVEN
0A7F4 F0609F00  (02406)          MOVRM R6,S1SPS                       STORE POWER INDEX OF S1*
                (02407) ;                                             *2
0A7F6 3C6E      (02408)          ADDRR R6,R7
0A7F7 0600      (02409)          EVEN
0A7F8 9260003F  (02410)          CMPIR R6,63
0A7FA 1A30      (02411)          SKPL LT
0A7FB 9E60003F  (02412)          SUBIR R6,63
                (02413) ;  READ PRC TABLE FOR S1**3
0A7FD 0600      (02414)          EVEN
0A7FE F07C9DB9  (02415)          MOVMR R7,TRRPCS(R6)
0A800 4476      (02416) BCHDSS   XORRR R7,R3                          R7=S1**3+S3
0A801 0800      (02417)          EVEN
0A802 8010ABAA  (02418)          JMP BCHDBS,EQ                        GO TO BCHDBS IF R7=0
                (02419) ;  MORE THAN TWO ERRORSOCCUR
                (02420) ;  CALCULATE SIGMA2 AND SIGMA3
                (02421) ;  CALCULATE S1**2*S3+S5 FOR SIGMA2 CALC.
0A804 0220      (02422)          TEST R2
0A805 1910      (02423)          SKP NE
0A806 0640      (02424)          CLR R3
0A807 0230      (02425)          TEST R3
0A808 8010AB16  (02426)          JMP BCHDTS,EQ
0A80A F0569BF8  (02427)          MOVMR R5,TRRPCS(R3)
0A80C FC509F00  (02428)          ADDMR R5,S1SPS
0A80E 9250003F  (02429)          CMPIR R5,63
0A810 1A30      (02430)          SKPL LT
0A811 9E50003F  (02431)          SUBIR R5,63
                (02432) ;  READ PRC TABLE
```

PAGE 662 PROG. CSPUSP.TXT

```
0AB13 0800       (02433)            EVEN
0AB14 E13A9EB9   (02434)            MOVSE R3,TBPRCS(R5)
0AB16 CE000580   (02435) BCBLIS   SPBSE 0,APDIELS          HAS APDONE HAPPENED?
0AB18 2021       (02436)            HOP #L+14                NO, HOP TO EVEN LOC AFTE
                 (02437) ;                                   R RET
0AB19 0800       (02438)            EVEN
0AB1A CC000580   (02439)            MOVZM APDNELS
0AB1C E1109E0C   (02440)            STORE R1,R1SAV
0AB1E E1209E0E   (02441)            STORE R2,R2SAV
0AB20 E1309E10   (02442)            STORE R3,R3SAV
0AB22 E1309E12   (02443)            STORE R4,R4SAV
0AB23 E1509E14   (02444)            STORE R5,R5SAV
0AB26 E1609E16   (02445)            STORE R6,R6SAV
0AB26 E1709E18   (02446)            STORE R7,R7SAV
0AB28 0E30       (02447)            RET
0AB26 0800       (02448)            EVEN
0AB2C A6109E0C   (02449)            LOAD R1,R1SAV
0AB2E A6209E0E   (02450)            LOAD R2,R2SAV
0AB30 A6309E10   (02451)            LOAD R3,R3SAV
0AB32 A6409E12   (02452)            LOAD R4,R4SAV
0AB33 A6509E14   (02453)            LOAD R5,R5SAV
0AB36 A6609E16   (02454)            LOAD R6,R6SAV
0AB38 A6709E18   (02455)            LOAD R7,R7SAV
0AB3A 4438       (02456)            XORRR R3,R4              R3=S1**2*S3+S5
0AB3B 4046       (02457)            MOVRR R4,R3              R4=R3
0AB3C 8010AB58   (02458)            JMP BCHDUS,E0
0AB3E E0669DE8   (02459)            MOVMR R6,TBRPCS(R3)
                 (02460) ;  READ RPC TABLE FOR R7
0AB40 E03E9DE8   (02461)            MOVMR R3,TBRPCS(R7)
0AB42 4E66       (02462)            SUBRR R6,R3              R6=R6/R3
0AB43 1830       (02463)            SKPL GE
0AB43 9C60003E   (02464)            ADDIR R6,63
0AB46 E03C9D69   (02465)            MOVMR R3,TBPRCS(R6)      SIGMA2=R3
                 (02466) ;  CALCULATE SIGMA3
0AB48 4044       (02467)            MOVRR R4,R2
0AB49 0800       (02468)            EVEN
0AB4A 8010AB58   (02469)            JMP BCHDUS,E0
0AB4C E0549DE8   (02470)            MOVMR R5,TBRPCS(R2)
0AB4E 4C5C       (02471)            ADDRR R5,R6
0AB4F 0800       (02472)            EVEN
0AB50 9250003E   (02473)            CMPIR R5,63
0AB52 1A30       (02474)            SKPL LT
0AB53 9E50003E   (02475)            SUBIR R5,63
0AB55 0800       (02476)            EVEN
```

```
0AR56 F03A9CB9 (02477)          MOVMR R4,TBPRCS(R5)
0AR58 441F     (02478) BCHDDS   XORRR R4,R7              R4=R7+S1*SIGMA2
0AR59 0RC0     (02479)          EVEN
0AR5A 801CAREO (02480)          JMP BCHDCS,EO            TWO ERRORS IF R4=0
               (02481) ;  3 OR MORE ERRORS OCCUR
0AR5C E8069FEC (02482)          MOVEM 3,NESS
0AR5E 9051FFC2 (02483)          MOVIR R5,-62
               (02484) ;  CHECK THE FIRST POSITION
0AR60 4064     (02485)          MOVRR R6,R2
0AR61 4466     (02486)          XORRR R6,R3
0AR62 4468     (02487)          XORRR R6,R4
0AR63 2761     (02488)          DECR R6,1
0AR64 8110AR6A (02489)          JMP BCHD5S,NE
               (02490) ;  STORE ERROR POSITION
0AR66 E5009FER (02491)          INCM NESTS
0AR68 F0109FE0 (02492)          MOVRM R1,NESS+1
               (02493) ;  MULTIPLY ALPA TO R2
0AR6A F0249F3R (02494) BCHD5S   MOVMR R2,TBAL1S(R2)
               (02495) ;  MULTIPLY ALPA**2 TO R3
0AR6C F0369F78 (02496)          MOVMR R3,TBAL2S(R3)
0AR6E F0489FRR (02497)          MOVMR R4,TBAL3S(R4)
0AR70 4064     (02498)          MOVRR R6,R2
0AR71 4466     (02499)          XORRR R6,R3
0AR72 4468     (02500)          XORRR R6,R4
0AR73 2761     (02501)          DECR R6,1
0AR74 8110AR7E (02502)          JMP BCHD4S,NE
               (02503) ;  STORE ERROR POSITION
0AR76 E5009FER (02504)          INCM NESTS
0AR78 407A     (02505)          MOVRR R7,R5
0AR79 4C72     (02506)          ADDRR R7,R1              R7 IS THE ERROR POSITION
0AR7A F0609FER (02507)          MOVMR R6,NESTS
0AR7C F07C9FEC (02508)          MOVRM R7,NESS(R6)
0AR7E R950AR6A (02509) BCHD4S   IJN R5,BCHD5S
               (02510) ;  CHECK ERROR CORRECTION STATUS
0AR80 F0509FEC (02511)          MOVMR R5,NESS
0AR82 F2509FER (02512)          CMPMR R5,NESTS
0AR84 8110ARA6 (02513)          JMP BCHDDS,NE
               (02514) ;  CORRECT 3 ERRORS
0AR86 90600001 (02515)          MOVIR R6,1
0AR88 F0709FED (02516)          MOVMR R7,NESS+1
0AR8A F56F9543 (02517)          XORRM R6,RECVS-1(R7)
0AR8C F0709FEE (02518)          MOVMR R7,NESS+2
0AR8E F56F9543 (02519)          XORRM R6,RECVS-1(R7)
0AR90 F0709FEF (02520)          MOVMR R7,NESS+3
```

```
0AB92 F56E9543 (02521)          XORRM R6,RECVS-1(R7)
0AB94 F0609FFC (02522) BCHDAS   MOVRM R6,MESS
0AB96 92100040 (02523)          CMPIR R1,64
0AB98 1610     (02524)          SKPL GE
0AB99 F0609FFA (02525)          MOVRM R6,MESS-2
0AB9B 0800     (02526)          EVEN
0AB9C 9210007F (02527)          CMPIR R1,127
0AB9E 1610     (02528)          SKPL NE
0AB9F F0609FFB (02529)          MOVRM R6,MESS-1
0ABA1 0800     (02530)          EVEN
               (02531) *  CMPIR R1,189
               (02532) *  JMP BCHD1S,LT
0ABA2 8000ABD4 (02533)          JMP DSRBDES                     CORRECT ONLY SIDEBAND
               (02534) *
               (02535) *
               (02536) :  END OF BCH DECODER EXCUTION
               (02537) *
0ABA4 8000ABD2 (02538)          JMP ENDDCS
               (02539) *
               (02540) *
               (02541) *
               (02542) *
0ABA6 90309FFC (02543) BCHDDS   MOVIR 4,MESS
0ABA8 8000AB94 (02544)          JMP BCHDAS
               (02545) ;  CORRECT 1 OR 2 ERRORS
0ABAA FF029FFC (02546) BCHDBS   MOVLM 1,MESS
0ABAC 0F30     (02547)          CLR R3
0ABAD 0800     (02548)          EVEN
0ABAE 8000ABB2 (02549)          JMP BCHDCS+2
0ABB0 FF019FFC (02550) BCHDCS   MOVLM 2,MESS                    2 ERRORS
0ABB2 9051FFC2 (02551)          MOVIR R5,-62
0ABB4 4061     (02552)          MOVRR R6,R2
0ABB5 4466     (02553)          XORRR R6,R3
0ABB6 2761     (02554)          DECR R6,1
0ABB7 0800     (02555)          EVEN
0ABB8 8110ABBE (02556)          JMP BCH25S,NE
               (02557) ;  CORRECT ERROR
0ABBA 90600001 (02558)          MOVIR R6,1
0ABBC F5629543 (02559)          XORRM R6,RECVS-1(R1)
0ABBE F0249E3B (02560) BCH25S   MOVMR R2,TBAL1S(R2)
0ABC0 F0369E7B (02561)          MOVMR R3,TBAL2S(R3)
0ABC2 4064     (02562)          MOVRR R6,R2
0ABC3 4466     (02563)          XORRR R6,R3                     R6=R2+R3
0ABC4 2761     (02564)          DECR R6,1
```

```
0ABC5 0800      (02565)          EVEN
0ABC6 8110ABCF  (02566)          JMP BCH24S,9F
                (02567) ;  CORRECT ERRORS
0ABC8 407A      (02568)          MOVRR R7,R5
0ABC9 4C72      (02569)          ADDRR R7,R1
0ABCA 96600001  (02570)          MOVIR R6,1
0ABCC E56F9543  (02571)          XORRM R6,RECVS-1(R7)
0ABCE 8950ABBF  (02572) BCH24S   IJN R5,BCH25S
0ABD0 8000AB94  (02573)          JMP BCHDAS
                (02574) *
                (02575) *
                (02576) ;  END OF BCH DECODING LIST
                (02577) ENDDCS   EVEN
0ABD2 8000ABD4  (02578)          JMP DSPBUFS
                (02579)          EJECT
```

```
                    (02580) ***** DESERIALIZE SIDEBAND & MAINBAND PARAMETTERS
                    (02581) DSRBUFS EVEN
0A8D4 0200          (02582)         CCS 3                          FROM SRB=2...
0A8D5 06A0          (02583)         SCS 2                          TO SRB=1
0A8D6 0200          (02584)         CCS 5                          FROM MRB=2...
0A8D7 06C0          (02585)         SCS 4                          TO MRB=1
0A8D8 DE000580      (02586)         SMBSL 0,APDNELS                HAS APDONE HAPPENED?
0A8DA 2021          (02587)         HOP #L+34                      NO, HOP TO EVEN LOC AFTE
                    (02588) ;                                      R RET
0A8DB 0800          (02589)         EVEN
0A8DC CC000580      (02590)         MOVZM APDNELS
0A8DE F1109F0C      (02591)         STORE R1,R1SAV
0A8E0 F1209F0E      (02592)         STORE R2,R2SAV
0A8E2 F1309F10      (02593)         STORE R3,R3SAV
0A8E4 F1409F12      (02594)         STORE R4,R4SAV
0A8E6 F1509F14      (02595)         STORE R5,R5SAV
0A8E8 F1609F16      (02596)         STORE R6,R6SAV
0A8EA F1709F18      (02597)         STORE R7,R7SAV
0A8EC 0F70          (02598)         RET
0A8ED 0800          (02599)         EVEN
0A8EE A6109F0C      (02600)         LOAD R1,R1SAV
0A8F0 A6209F0E      (02601)         LOAD R2,R2SAV
0A8F2 A6309F10      (02602)         LOAD R3,R3SAV
0A8F4 A6409F12      (02603)         LOAD R4,R4SAV
0A8F6 A6509F14      (02604)         LOAD R5,R5SAV
0A8F8 A6609F16      (02605)         LOAD R6,R6SAV
0A8FA A6709F18      (02606)         LOAD R7,R7SAV
0A8FC 0D20          (02607)         CLR R2
0A8FD 0800          (02608)         EVEN
0A8FE 2621          (02609)         INCR R2,1                      SKIP THE SYNC BIT
0A8FF 0800          (02610)         EVEN
0A900 0D40          (02611)         CLR R4
0A901 0800          (02612)         EVEN
0A902 F0509C8C      (02613)         MOVMR R5,PARMSM2               LOAD R5 WITH THE NUMBER
                    (02614) ;                                      OF PARAMETERS-2
                    (02615) *
0A904 CC0696B6      (02616) LOOPS1  MOVZM  RUPPS(R4)               ZERO OUT THE REC'D PARM
                    (02617) ;                                      BUFFER
0A906 DE000580      (02618)         SMBSL 0,APDNELS                HAS APDONE HAPPENED?
0A908 2021          (02619)         HOP #L+34                      NO, HOP TO EVEN LOC AFTE
                    (02620) ;                                      R RET
0A909 0800          (02621)         EVEN
0A90A CC000580      (02622)         MOVZM APDNELS
0A90C F1109F0C      (02623)         STORE R1,R1SAV
```

```
0A90E E1209E0E (02624)          STORE R2,R2SAV
0A910 E1309E10 (02625)          STORE R3,R3SAV
0A912 E1309E12 (02626)          STORE R4,R4SAV
0A914 E1509E14 (02627)          STORE R5,R5SAV
0A916 E1609E16 (02628)          STORE R6,R6SAV
0A918 E1709E18 (02629)          STORE R7,R7SAV
0A91A 0E70     (02630)          RET
0A91B 0800     (02631)          EVEN
0A91C A6109E0C (02632)          LOAD R1,R1SAV
0A91E A6209E0E (02633)          LOAD R2,R2SAV
0A920 A6309E10 (02634)          LOAD R3,R3SAV
0A922 A6409E12 (02635)          LOAD R4,R4SAV
0A924 A6509E14 (02636)          LOAD R5,R5SAV
0A926 A6609E16 (02637)          LOAD R6,R6SAV
0A928 A6709E18 (02638)          LOAD R7,R7SAV
0A92A 808A9C62 (02639)          JMP @INDX4S(R5)
               (02640)          EVEN
0A92C DA049544 (02641) FBIT5S   SMBCL 0,RECVS(R2)                          SKIP IF THE REC'D BIT IS
0A92E D2589666 (02642)          SMB 5,ROPBS(R4)                            SET BIT 5 OF PARM TO BE
               (02643) ;                                                   A 1
0A930 2621     (02644)          INCR R2,1                                  INCREMENT R2 BY 1
0A931 0800     (02645)          EVEN
               (02646) *
0A932 DA049544 (02647) FBIT4S   SMBCL 0,RECVS(R2)                          SKIP IF THE REC'D BIT IS
0A934 D2489666 (02648)          SMB 4,ROPBS(R4)                            SET BIT 4 OF ROPB TO BE
               (02649) ;                                                   1
0A936 2621     (02650)          INCR R2,1                                  INCREMENT R2 BY 1
0A937 0800     (02651)          EVEN
               (02652) *
0A938 DA049544 (02653) FBIT3S   SMBCL 0,RECVS(R2)                          SKIP IF THE REC'D BIT IS
0A93A D2389666 (02654)          SMB 3,ROPBS(R4)                            SET BIT 3 OF ROPB TO BE
               (02655) ;                                                   1
0A93C 2621     (02656)          INCR R2,1                                  INCREMENT R2 BY 1
0A93D 0800     (02657)          EVEN
               (02658) *
0A93E DA049544 (02659) FBIT2S   SMBCL 0,RECVS(R2)                          SKIP IF THE REC'D BIT IS
0A940 D2289666 (02660)          SMB 2,ROPBS(R4)                            SET BIT 2 OF ROPB TO BE
               (02661) ;                                                   1
0A942 2621     (02662)          INCR R2,1                                  INCREMENT R2 BY 1
0A943 0800     (02663)          EVEN
               (02664) *
0A944 DA049544 (02665) FBIT1S   SMBCL 0,RECVS(R2)                          SKIP IF THE REC'D BIT IS
0A946 D2189666 (02666)          SMB 1,ROPBS(R4)                            SET BIT 2 OF ROPB TO BE
               (02667) ;                                                   1
```

```
0A948 2621     (02668)         INCR R2,1                        INCREMENT R2 BY 1
0A949 0800     (02669)         EVEN
               (02670) *
0A94A DA049544 (02671) FBITOS  SMBCL 0,RECVS(R2)                SKIP IF THE REC'D BIT IS
0A94C D2089686 (02672)         SMB 0,ROPBS(R4)                  SET BIT 0 OF ROPB TO BE
               (02673) ;                                        1
0A94E 2621     (02674)         INCR R2,1                        INCREMENT R2 BY 1
0A94F 0800     (02675)         EVEN
               (02676) *
0A950 2641     (02677)         INCR R4,1                        INCREMENT THE PARM. COUN
               (02678) ;                                        TER BY 1
0A951 0800     (02679)         EVEN
0A952 RESOA904 (02680)         DBJP R5,LOOPS1                   AND GOTO THE NEXT ONE
               (02681) *
               (02682) ;
0A954 EC209CBE (02683)         ADDMR R2,FIGIENS                 SKIP THE NEXT 18 POSITIO
               (02684) ;                                        NS SINCE THESE ARE PARIT
               (02685) ;                                        Y BITS FOR ERROR CONTROL
0A956 CC089666 (02686)         MOVZB ROPBS(R4)                  CLEAR THE PARM BUFFER
0A958 DA049544 (02687)         SMBCL 0,RECVS(R2)                SKIP IF THE REC'D BIT IS
0A95A D2189686 (02688)         SMB 1,ROPBS(R4)                  SET BIT 1 OF ROPB TO BE
               (02689) ;                                        1
0A95C 2621     (02690)         INCR R2,1                        INCREMENT THE LOOP COUNT
               (02691) ;                                        ER BY ONE
0A95D 0800     (02692)         EVEN
0A95E DA049544 (02693)         SMBCL 0,RECVS(R2)                SKIP IF THE REC'D BIT IS
0A960 D2089686 (02694)         SMB 0,ROPBS(R4)                  SET BIT 0 OF ROPB TO BE
               (02695) ;                                        1
0A962 2621     (02696)         INCR R2,1                        INCREMENT R2 BY 1
0A963 0800     (02697)         EVEN
               (02698) *
               (02699) *
               (02700) *
               (02701) *
               (02702) ;  FROM THESE DECODED PARAMETERS
               (02703) ;  BIT ASSIGNMENT RULE FOR DESERIALIZATION
               (02704) ;  OF DCT COEFS IS COMPUTED
               (02705) *
               (02706) *
               (02707) *
               (02708) *
               (02709) *
               (02710) *
               (02711) *
```

```
                    (02712) ;  DESERIALIZE THE QUANTIZED DCT COEFS
                    (02713) ;  ASSUMING HORIZONTAL CODING
                    (02714) *
0A964 E609FECE      (02715)        MOVLM CLRSGO,SYSFLGS
0A966 DE000560      (02716) DSQDS  SMRSL 0,APDNFLS                HAS APDONE HAPPENED?
0A968 2021          (02717)        HOP *I+34                      NO, HOP TO EVEN LOC AFTE
                    (02718) ;                                     R RET
0A969 0800          (02719)        EVEN
0A96A CE000560      (02720)        MOVZM APDNFLS
0A96C F1109F0C      (02721)        STORE R1,R1SAV
0A96E F1209F0E      (02722)        STORE R2,R2SAV
0A970 F1309F10      (02723)        STORE R3,R3SAV
0A972 F1409F12      (02724)        STORE R4,R4SAV
0A974 F1509F14      (02725)        STORE R5,R5SAV
0A976 F1609F16      (02726)        STORE R6,R6SAV
0A978 F1709F18      (02727)        STORE R7,R7SAV
0A97A 0E70          (02728)        RET
0A97B 0800          (02729)        EVEN
0A97C A6109F0C      (02730)        LOAD R1,R1SAV
0A97E A6209F0E      (02731)        LOAD R2,R2SAV
0A980 A6309F10      (02732)        LOAD R3,R3SAV
0A982 A6409F12      (02733)        LOAD R4,R4SAV
0A984 A6509F14      (02734)        LOAD R5,R5SAV
0A986 A6609F16      (02735)        LOAD R6,R6SAV
0A988 A6709F18      (02736)        LOAD R7,R7SAV
0A98A FE000576      (02737)        SMRSL 0,RG02                   HAS AP WRITTEN RIBIT YET
                    (02738) ;                                     ?
0A98C 8000A966      (02739)        JMP DSQDS                      NO, WAIT FOR OK
0A98E CE000576      (02740)        MOVZM RG02                     RESET GO FLAG
0A990 E849FECE      (02741)        MOVLM SETSGO,SYSFLGS
                    (02742) *????????????????????????????????????????????????????????????????????????
                    (02743) *  THE "BMOVEL" OP'S CAUSE A MEMORY BUS ERROR!
                    (02744) *  REASON IS UNKNOWN, 5/17/80
                    (02745) *  MOVIR R3,ZROS-1
                    (02746) *  MOVIR R4,LTH1-1
                    (02747) *  BMOVEL R3,R4,ROTDCTS
                    (02748) *  BMOVEL R3,R4,ROTDCTS+2*LTH1      MOVE 2 BLOCKS OF 0'S TO ROTDCT
                    (02749) *????????????????????????????????????????????????????????????????????????
0A992 F1409F12      (02750)        STORE R4,R4SAV                 NEED IT FOR RPT
0A994 114000FF      (02751)        LDRPT R4,LTH-1                 MOVE LONG BUT DECR BY 2!
0A996 CE0896C4      (02752)        MOVZML ROTDCTS(R4)             ROTDCT(I)=0 FOR 0.LE.I.L
                    (02753) ;                                     E.255
0A998 F2009C84      (02754)        CMPMZ FRSTF
0A99A 8010A9A0      (02755)        JMP DSQD1S,EQ                  DON'T DESER IF FIRST FRA
```

```
                   ;                                                    ME
0A99C CC009CB1              MOVZM ERSTF
0A99E 80009E60              JMP RECVSRA
0A9A0 A6409E12     DSOD1S   LOAD R4,R4SAV                               RESTORE R4
0A9A2 F0509CBF              MOVMR R5,FORFORS                            MOVE FORTY FOUR IN R5
0A9A4 2752                  DECR R5,2
0A9A5 0800                  EVEN
0A9A6 F0709AA6              MOVMR R7,RIBITS                             SAVE THE FIRST IBIT VALU
                   ;                                                    E
0A9A8 2771                  DECR R7,1                                   DECREMENT R7 BY 1
0A9A9 3A71                  LLS R7,1                                    SHIFT LEFT BY 1
                            EVEN
                   *************************************************************
                   *  SPECIAL CODE TO CORRECT FOR RIBIT=0 CASE 6/12/80
0A9AA 0270                  TEST R7                                     WHEN RIBIT=0,R7<0 AND @
                   ;                                                    IS NG!
0A9AB 0800                  EVEN                                        SO TEST & JUMP AROUND SE
                   ;                                                    RIALIZATION SINCE
0A9AC 8030APE6              JMP EXITS,LT                                IBIT=0 IMPLIES NO BITS T
                   ;                                                    O SERIRIALIZE!!
                   ***************************************************************
0A9AE 808E9C7A              JMP @IGDX5S(R7)                             GOTO THE CORR.DESERIALIZ
                   ;                                                    ATION ROUTINE
                   *
                   ;  DESERIALIZATION ROUTINE FOR BIT 4
                   *
                            EVEN
0A9B0 DE0005R0     GBIT4S   SMBSL 0,APDNFLS                             HAS APDONE HAPPENED?
0A9B2 2021                  HOP @L+34                                   NO, HOP TO EVEN LOC AFTE
                   ;                                                    R RET
0A9B3 0800                  EVEN
0A9B4 CC000590              MOVZM APDNFLS
0A9B6 F1109E0C              STORE R1,R1SAV
0A9B8 F1209E0E              STORE R2,R2SAV
0A9BA F1309E10              STORE R3,R3SAV
0A9BC F1409E12              STORE R4,R4SAV
0A9BE F1509E14              STORE R5,R5SAV
0A9C0 F1609E16              STORE R6,R6SAV
0A9C2 F1709E18              STORE R7,R7SAV
0A9C4 0E70                  RET
0A9C5 0800                  EVEN
0A9C6 A6109E0C              LOAD R1,R1SAV
0A9C8 A6209E0E              LOAD R2,R2SAV
0A9CA A6309E10              LOAD R3,R3SAV
```

```
0A9CC A6409F12 (02800)         LOAD R4,R4SAV
0A9CE A6509F14 (02801)         LOAD R5,R5SAV
0A9D0 A6609F16 (02802)         LOAD R6,R6SAV
0A9D2 A6709F18 (02803)         LOAD R7,R7SAV
0A9D4 F6709AA6 (02804)         MOVMR R7,R1BITS                SAVE THE 1ST IBIT VALUE
0A9D6 0D40     (02805)         CLR R4
0A9D7 0800     (02806)         EVEN
0A9D8 F2789AA6 (02807) GBIT4$1 CMPMR R7,R1BITS(R4)            COMPARE THE BIT ASSIGNME
               (02808) ;                                      NT WITH THE MARK
0A9DA 8020AA12 (02809)         JMP GBIT3S,GT                  GOTO NEXT BIT DECODING R
               (02810) ;                                      OUTINE IF BIT ASSIGN. CH
               (02811) ;                                      ANGES
0A9DC DE000580 (02812)         SMBSL 0,APDNFLS                HAS APDONE HAPPENED?
0A9DE 2021     (02813)         HOP *L+34                      NO, HOP TO EVEN LOC AFTE
               (02814) ;                                      R RET
0A9DF 0800     (02815)         EVEN
0A9E0 CC000580 (02816)         MOVZM APDNFLS
0A9E2 F1109F0C (02817)         STORE R1,R1SAV
0A9E4 F1209F0E (02818)         STORE R2,R2SAV
0A9E6 F1309F10 (02819)         STORE R3,R3SAV
0A9E8 F1409F12 (02820)         STORE R4,R4SAV
0A9EA F1509F14 (02821)         STORE R5,R5SAV
0A9EC F1609F16 (02822)         STORE R6,R6SAV
0A9EE F1709F18 (02823)         STORE R7,R7SAV
0A9F0 0F70     (02824)         RET
0A9F1 0800     (02825)         EVEN
0A9F2 A6109F0C (02826)         LOAD R1,R1SAV
0A9F4 A6209F0E (02827)         LOAD R2,R2SAV
0A9F6 A6309F10 (02828)         LOAD R3,R3SAV
0A9F8 A6409F12 (02829)         LOAD R4,R4SAV
0A9FA A6509F14 (02830)         LOAD R5,R5SAV
0A9FC A6609F16 (02831)         LOAD R6,R6SAV
0A9FE A6709F18 (02832)         LOAD R7,R7SAV
0AA00 DA049544 (02833)         SMBCL 0,RECVS(R2)              SKIP IF BIT 0 OF RECV(R2
               (02834) ;                                      ) IS CLEARED
0AA02 D24896C4 (02835)         SMB 4,ROTDCTS(R4)
0AA04 2641     (02836)         INCR R4,1
0AA05 0800     (02837)         EVEN
0AA06 2621     (02838)         INCR R2,1                      INCREMENT THE BUFFER POS
               (02839) ;                                      ITION COUNTER
0AA07 0800     (02840)         EVEN
0AA08 8C50A9D8 (02841)         PJP R5,GBIT4$1                 CHECK IF R5 +VE
0AA0A F0509CFF (02842)         MOVMR R5,FORFORS               RELOAD THE COUNTER WITH
               (02843) ;                                      44
```

```
0AA0C F2209C86 (02843)          CMPMR R2,POST3S         IS R2 GREATER THAN POST
               (02845) ;                                3
0AA0E 8320 9E88 (02846)         XCT CNT3IS,LE           ADD 18 TO R2 IF LESS THA
               (02847) ;                                N OR EQUAL
0AA10 8000A9E8 (02848)          JMP GBIT4S1             GO BACK TO DBITS1
               (02849) *
               (02850) *
               (02851) ;  DESERIALIZATION FOR BIT 3
               (02852) *
               (02853)          EVEN
0AA12 F2209C89 (02854) GBIT3S   CMPMR R2,POST4S
0AA14 8010A0E6 (02855)          JMP EXITS,EQ
0AA16 1E000580 (02856)          SBBSL 0,APDNELS         HAS APDONE HAPPENED?
0AA18 2021     (02857)          HOP #L+34               NO, HOP TO EVEN LOC AFTE
               (02858) ;                                R RET
0AA19 0800     (02859)          EVEN
0AA1A CC000580 (02860)          MOVZM APDNELS
0AA1C E1109E0C (02861)          STORE R1,R1SAV
0AA1E E1209E0E (02862)          STORE R2,R2SAV
0AA20 E1309E10 (02863)          STORE R3,R3SAV
0AA22 E1409E12 (02864)          STORE R4,R4SAV
0AA24 E1509E14 (02865)          STORE R5,R5SAV
0AA26 E1609E16 (02866)          STORE R6,R6SAV
0AA28 E1709E18 (02867)          STORE R7,R7SAV
0AA2A 0E70     (02868)          RET
0AA2B 0800     (02869)          EVEN
0AA2C A6109E0C (02870)          LOAD R1,R1SAV
0AA2E A6209E0E (02871)          LOAD R2,R2SAV
0AA30 A6309E10 (02872)          LOAD R3,R3SAV
0AA32 A6409E12 (02873)          LOAD R4,R4SAV
0AA34 A6509E14 (02874)          LOAD R5,R5SAV
0AA36 A6609E16 (02875)          LOAD R6,R6SAV
0AA38 A6709E18 (02876)          LOAD R7,R7SAV
0AA3A F0789AA6 (02877)          MOVMR R7,RIBITS(R4)     SAVE THE NEXT IBIT VALUE
0AA3C 0D40     (02878)          CLR R4
0AA3D 0800     (02879)          EVEN
0AA3E F2789AA6 (02880) GBIT3S1  CMPMR R7,RIBITS(R4)     COMPARE THE BIT ASSIGNME
               (02881) ;                                NT WITH THE MARK
0AA40 8020AA78 (02882)          JMP GBIT2S,GT           FT NOT EQUAL,GOTO DECODE
0AA42 1E000580 (02883)          SBBSL 0,APDNELS         HAS APDONE HAPPENED?
0AA44 2021     (02884)          HOP #L+34               NO, HOP TO EVEN LOC AFTE
               (02885) ;                                R RET
0AA45 0800     (02886)          EVEN
0AA46 CC000580 (02887)          MOVZM APDNELS
```

```
0AA48 E1109E0C (02888)          STORE R1,R1SAV
0AA4A E1209E0E (02889)          STORE R2,R2SAV
0AA4C E1309E10 (02890)          STORE R3,R3SAV
0AA4E E1409E12 (02891)          STORE R4,R4SAV
0AA50 E1509E14 (02892)          STORE R5,R5SAV
0AA52 E1609E16 (02893)          STORE R6,R6SAV
0AA54 E1709E18 (02894)          STORE R7,R7SAV
0AA56 0E70     (02895)          RET
0AA57 0800     (02896)          EVEN
0AA58 A6109E0C (02897)          LOAD R1,R1SAV
0AA5A A6209E0E (02898)          LOAD R2,R2SAV
0AA5C A6309E10 (02899)          LOAD R3,R3SAV
0AA5E A6409E12 (02900)          LOAD R4,R4SAV
0AA60 A6509E14 (02901)          LOAD R5,R5SAV
0AA62 A6609E16 (02902)          LOAD R6,R6SAV
0AA64 A6709E18 (02903)          LOAD R7,R7SAV
0AA66 DA019544 (02904)          SMBCL 0,RECVS(R2)               SKIP IF BIT 0 OF RECV(R2
                (02905) ;                                        ) IS CLEARED
0AA68 0230A6C4 (02906)          SMR 3,ROTPCTS(R4)
0AA6A 2611     (02907)          INCR R4,1
0AA6B 0800     (02908)          EVEN
0AA6C 2621     (02909)          INCR R2,1                       INCREMENT THE BUFFER POS
                (02910) ;                                        ITION COUNTER
0AA6D 0800     (02911)          EVEN
0AA6E BC50AA3E (02912)          DJP R5,GBIT3S1                  GO BACK TO GBIT3S1
0AA70 E0509C6E (02913)          MOVMR R5,FORFORS                RELOAD THE COUNTER WITH
                (02914) ;                                        44
0AA72 E2209C86 (02915)          CMPMR R2,POST3S                 IS R2 GREATER THAN POST
                (02916) ;                                        3
0AA74 B3209E88 (02917)          XCT CNTP1S,LE                   ADD 18 TO LOOP COUNTER I
                (02918) ;                                        F LESS THAN OR EQUAL TO
0AA76 B000AA3E (02919)          JMP GBIT3S1                     GO BACK TO GBIT3S1
                (02920) *
                (02921) *
                (02922) ;  DESERIALIZATION FOR BIT 2
                (02923) *
                (02924)         EVEN
0AA78 E2209C89 (02925) GBIT2S  CMPMR R2,POST4S
0AA7A B010ABE6 (02926)          JMP EXITS,EQ
0AA7C DE0005B0 (02927)          SMESL 0,APDNELS                 HAS APDONE HAPPENED?
0AA7E 2021     (02928)          HOP #L+34                       NO, HOP TO EVEN LOC AFTE
                (02929) ;                                        R RET
0AA7F 0800     (02930)          EVEN
0AA80 CC0005B0 (02931)          MOVZM APDNELS
```

```
0AA82 E1109F0C (02932)          STORE R1,R1SAV
0AA84 E1209F0E (02933)          STORE R2,R2SAV
0AA86 E1309F10 (02934)          STORE R3,R3SAV
0AA88 E1409F12 (02935)          STORE R4,R4SAV
0AA8A E1509F14 (02936)          STORE R5,R5SAV
0AA8C E1609F16 (02937)          STORE R6,R6SAV
0AA8E E1709F18 (02938)          STORE R7,R7SAV
0AA90 0F70     (02939)          RET
0AA91 0800     (02940)          EVEN
0AA92 A6109F0C (02941)          LOAD R1,R1SAV
0AA94 A6209F0E (02942)          LOAD R2,R2SAV
0AA96 A6309F10 (02943)          LOAD R3,R3SAV
0AA98 A6409F12 (02944)          LOAD R4,R4SAV
0AA9A A6509F14 (02945)          LOAD R5,R5SAV
0AA9C A6609F16 (02946)          LOAD R6,R6SAV
0AA9E A6709F18 (02947)          LOAD R7,R7SAV
0AAA0 F078AAA6 (02948)          MOVBR R7,RIBITS(R4)             SAVE THE IBIT VALUE
0AAA2 0040     (02949)          CLR R4
0AAA3 0800     (02950)          EVEN
0AAA4 F278AAA6 (02951) GBIT2S1  CMPBR R7,RIBITS(R4)             COMPARE THE BIT ASSIGNME
               (02952) ;                                        NT WITH THE MARK
0AAA6 8020AADE (02953)          JMP GBIT1S,GT                   IF NOT EQUAL,GOTO DECODE
0AAA8 DE0005B0 (02954)          SMBSL 0,APDNFLS                 HAS APDONE HAPPENED?
0AAAA 2021     (02955)          HOP #L+34                       NO, HOP TO EVEN LOC AFTE
               (02956) ;                                        R RET
0AAAB 0800     (02957)          EVEN
0AAAC CC0005B0 (02958)          MOVZM APDNFLS
0AAAE E1109F0C (02959)          STORE R1,R1SAV
0AAB0 E1209F0E (02960)          STORE R2,R2SAV
0AAB2 E1309F10 (02961)          STORE R3,R3SAV
0AAB4 E1409F12 (02962)          STORE R4,R4SAV
0AAB6 E1509F14 (02963)          STORE R5,R5SAV
0AAB8 E1609F16 (02964)          STORE R6,R6SAV
0AABA E1709F18 (02965)          STORE R7,R7SAV
0AABC 0F70     (02966)          RET
0AABD 0800     (02967)          EVEN
0AABE A6109F0C (02968)          LOAD R1,R1SAV
0AAC0 A6209F0E (02969)          LOAD R2,R2SAV
0AAC2 A6309F10 (02970)          LOAD R3,R3SAV
0AAC4 A6409F12 (02971)          LOAD R4,R4SAV
0AAC6 A6509F14 (02972)          LOAD R5,R5SAV
0AAC8 A6609F16 (02973)          LOAD R6,R6SAV
0AACA A6709F18 (02974)          LOAD R7,R7SAV
0AACC DA049544 (02975)          SMBCL 0,RECVS(R2)               SKIP IF BIT 0 OF RECV(R2
```

```
                     (02976) ;                                                  ) IS CLEARED
0AACE E2289EC4 (02977)          SMR 2,EOTDCES(R4)
0AAD0 2641     (02978)          INCR R4,1
0AAD1 0800     (02979)          EVEN
0AAD2 2621     (02980)          INCR R2,1                                       INCREMENT THE BUFFER POS
               (02981) ;                                                        ITION COUNTER
0AAD3 0800     (02982)          EVEN
0AAD4 8C50AAA4 (02983)          DJP R5,GRIT2$1                                  CHECK IF R5 +VE
0AAD6 E0509CBE (02984)          MOVMR R5,FORFOPS                                RELOAD THE COUNTER WITH
               (02985) ;                                                        44
0AAD8 E2209C86 (02986)          CMPMR R2,POST3S                                 IS R2 GREATER THAN POST
               (02987) ;                                                        3
0AADA 83209E86 (02988)          XCT CNTP1S,LE                                   ADD 18 TO LOOP COUNTER I
               (02989) ;                                                        F LESS THAN OR EQUAL
0AADC 8000AAA1 (02990)          JMP GRIT2$1                                     GO BACK TO GRIT2$1
               (02991) *
               (02992) *
               (02993) *
               (02994) ;  DESERIALIZATION FOR BIT 1
               (02995)          EVEN
0AADE E2209C89 (02996) GRIT1S   CMPMR R2,POST4S
0AAE0 8010ABE6 (02997)          JMP EXITS,EQ
0AAE2 DE000580 (02998)          SNBSL 0,APDNELS                                 HAS APDONE HAPPENED?
0AAE4 2021     (02999)          HOP #L+34                                       NO, HOP TO EVEN LOC AFTE
               (03000) ;                                                        R RET
0AAE5 0800     (03001)          EVEN
0AAE6 CC000580 (03002)          MOVZM APDNELS
0AAE8 E1109E0C (03003)          STORE R1,R1SAV
0AAEA E1209E0E (03004)          STORE R2,R2SAV
0AAEC E1309E10 (03005)          STORE R3,R3SAV
0AAEE E1409E12 (03006)          STORE R4,R4SAV
0AAF0 E1509E14 (03007)          STORE R5,R5SAV
0AAF2 E1609E16 (03008)          STORE R6,R6SAV
0AAF4 E1709E18 (03009)          STORE R7,R7SAV
0AAF6 0E70     (03010)          RET
0AAF7 0800     (03011)          EVEN
0AAF8 A6109E0C (03012)          LOAD R1,R1SAV
0AAFA A6209E0E (03013)          LOAD R2,R2SAV
0AAFC A6309E10 (03014)          LOAD R3,R3SAV
0AAFE A6409E12 (03015)          LOAD R4,R4SAV
0AB00 A6509E14 (03016)          LOAD R5,R5SAV
0AB02 A6609E16 (03017)          LOAD R6,R6SAV
0AB04 A6709E18 (03018)          LOAD R7,R7SAV
0AB06 E0789AA6 (03019)          MOVRR R7,RIBITS(R4)                             SAVE THE IBIT VALUE
```

```
0AB08 0C30     (03020)         CLR R4
0AB09 0800     (03021)         EVEN
               (03022) *
               (03023) *
0AB0A E27B9AA6 (03024) GBIT1S1 CMPMR R7,BIBITS(R4)          COMPARE THE BIT ASSIGNME
               (03025) ;                                    NT WITH THE MARK
0AB0C 9620AB14 (03026)         JMP GBIT0S,GT                GOTO SERIALIZATION ROUTI
               (03027) ;                                    NE IF NOT EQUAL
0AB0E BE000580 (03028)         SMBSL 0,APDNELS              HAS APDONE HAPPENED?
0AB10 2021     (03029)         HOP *L+34                    NO, HOP TO EVEN LOC AFTE
               (03030) ;                                    R RET
0AB11 0800     (03031)         EVEN
0AB12 CC000580 (03032)         MOVZM APDNELS
0AB13 E1109E0C (03033)         STORE R1,R1SAV
0AB16 E1209E0E (03034)         STORE R2,R2SAV
0AB18 E1309E10 (03035)         STORE R3,R3SAV
0AB1A E1409E12 (03036)         STORE R4,R4SAV
0AB1C E1509E14 (03037)         STORE R5,R5SAV
0AB1E E1609E16 (03038)         STORE R6,R6SAV
0AB20 E1709E18 (03039)         STORE R7,R7SAV
0AB22 0E70     (03040)         RET
0AB23 0800     (03041)         EVEN
0AB24 A6109E0C (03042)         LOAD R1,R1SAV
0AB26 A6209E0E (03043)         LOAD R2,R2SAV
0AB28 A6309E10 (03044)         LOAD R3,R3SAV
0AB2A A6409E12 (03045)         LOAD R4,R4SAV
0AB2C A6509E14 (03046)         LOAD R5,R5SAV
0AB2E A6609E16 (03047)         LOAD R6,R6SAV
0AB30 A6709E18 (03048)         LOAD R7,R7SAV
0AB32 DA049544 (03049)         SMBCL 0,RECVS(R2)            SKIP IF BIT 0 OF RECV(R2
               (03050) ;                                    ) IS CLEARED
0AB33 D21896C4 (03051)         SMB 1,POTECTS(R4)
0AB36 2641     (03052)         INCR R4,1
0AB37 0800     (03053)         EVEN
0AB38 2621     (03054)         INCR R2,1                    INCREMENT THE BUFFER POS
               (03055) ;                                    ITION COUNTER
0AB39 0800     (03056)         EVEN
0AB3A 8C50AB0A (03057)         DJP R5,GBIT1S1               CHECK IF R5 +VE
0AB3C F0509CBE (03058)         MOVMR R5,FOREORS             RELOAD THE COUNTER WITH
               (03059) ;                                    44
0AB3E E2209CA6 (03060)         CMPMR R2,POST1$              IS R2 GREATER THAN POST
               (03061) ;                                    1
0AB40 83209EBB (03062)         XCT CETBIS,LE                ADD 18 TO R2 IF LESS THA
               (03063) ;                                    N OR EQUAL TO
```

```
0AB42 8000AB02 (03064)          JMP GBIT1S1                GO BACK TO GBIT1S1
               (03065) *
               (03066) *
               (03067) ;  DESERIALIZATION OF BIT 0
               (03068)          EVEN
0AB43 E2209C89 (03069) GBIT0S   CMPSR R2,POST4S            COMPARE R2 WITH THE FRAM
               (03070) ;                                   E LENGTH
0AB46 8010AB66 (03071)          JMP EXITS,EQ               LEAVE IF THE ENTIRE FRAM
               (03072) ;                                   E IS FINISHED
0AB48 DE0005B0 (03073)          SMRSL 0,APDNELS            HAS APDONE HAPPENED?
0AB4A 2021     (03074)          HOP #L+33                  NO, HOP TO EVEN LOC AFTE
               (03075) ;                                   R RET
0AB4B 0800     (03076)          EVEN
0AB4C CC0005B0 (03077)          MOVZM APDNELS
0AB4E E1109E0C (03078)          STORE R1,R1SAV
0AB50 E1209E0E (03079)          STORE R2,R2SAV
0AB52 E1309E10 (03080)          STORE R3,R3SAV
0AB54 E1409E12 (03081)          STORE R4,R4SAV
0AB56 E1509E14 (03082)          STORE R5,R5SAV
0AB58 E1609E16 (03083)          STORE R6,R6SAV
0AB5A E1709E18 (03084)          STORE R7,R7SAV
0AB5C 0E70     (03085)          RET
0AB5D 0800     (03086)          EVEN
0AB5E A6109E0C (03087)          LOAD R1,R1SAV
0AB60 A6209E0E (03088)          LOAD R2,R2SAV
0AB62 A6309E10 (03089)          LOAD R3,R3SAV
0AB64 A6409E12 (03090)          LOAD R4,R4SAV
0AB66 A6509E14 (03091)          LOAD R5,R5SAV
0AB68 A6609E16 (03092)          LOAD R6,R6SAV
0AB6A A6709E18 (03093)          LOAD R7,R7SAV
0AB6C E2209C56 (03094)          CMPMR R2,POST3S            COMPARE R2 WITH POST3
0AB6E 8020AB82 (03095)          JMP BBIT0S,GT              IF GREATER THAN POST3, N
               (03096) ;                                   O NEED TO PUT IN PARITY
               (03097) ;                                   BITS
               (03098) *
               (03099) *
0AB70 E0609C89 (03100)          MOVMR R6,POST4S            MOVE THE FRAME LENGTH TO
0AB72 4E64     (03101)          SUBRR R6,R2                R6=LSEN-1-R2
0AB73 0800     (03102)          EVEN
0AB74 0140     (03103)          CLR R4                     CLEAR R4
0AB75 0800     (03104)          EVEN
0AB76 9C209543 (03105)          ADDIR R2,RECVS-1           SET R2 TO BE THE ADDRESS
               (03106) ;                                   :RECVS(R2)
0AB78 3022     (03107) GBIT0S1  POPXI R2,R1                POP RECV(R2)-->R1...R2=R
```

```
                        ;                                2+1
0AB79 2A02                     SRBCL 0,R1              DON'T CHANGE ROTDCT IF 0
0AB7A E2089EC4                 SBB 0,ROTDCTS(R4)       SET LOW BIT OF THE ROTDC
                        ;                                T PARM
0AB7C 2641                     INCR R4,1               INCREMENT R4 BY 1
0AB7D 0800                     EVEN
0AB7E BE000580                 SBBSL 0,APDNELS         HAS APDONE HAPPENED?
0AB80 2621                     HOP #L+34               NO, HOP TO EVEN LOC AFTE
                        ;                                R RET
0AB81 0800                     EVEN
0AB82 CC000580                 MOVZM APDNELS
0AB83 E1109E0C                 STORE R1,R1SAV
0AB86 E1209E0E                 STORE R2,R2SAV
0AB88 E1309E10                 STORE R3,R3SAV
0AB8A E1409E12                 STORE R4,R4SAV
0AB8C E1509E14                 STORE R5,R5SAV
0AB8E E1609E16                 STORE R6,R6SAV
0AB90 E1709E18                 STORE R7,R7SAV
0AB92 0F70                     RET
0AB93 0800                     EVEN
0AB94 A6109E0C                 LOAD R1,R1SAV
0AB96 A6209E0E                 LOAD R2,R2SAV
0AB98 A6309E10                 LOAD R3,R3SAV
0AB9A A6409E12                 LOAD R4,R4SAV
0AB9C A6509E14                 LOAD R5,R5SAV
0AB9E A6609E16                 LOAD R6,R6SAV
0ABA0 A6709E18                 LOAD R7,R7SAV
0ABA2 8C50ABAE                 DJP R5,GENDS0           CHECK IF R5 +VE
0ABA4 E0509CBE                 MOVMR R5,FORFORS        RELOAD THE COUNTER WITH
                        ;                                44
0ABA6 F2209CB6                 CMPMR R2,POST3S2        IS R2 GREATER THAN POST
                        ;                                3
0ABA8 8020ABAE                 JMP GENDS0,GT           LEAVE IF GREATER THAN
0ABAA FC209CBE                 ADDMR R2,EIGTENS        ADD 18 TO R2
0ABAC FE609CBE                 SUBMR R6,EIGTENS        SUBTRACT 18 FROM R6
0ABAE 8C60AB78          GENDS0 DJP R6,GBITOS1          LOOP BACK UNITIL THE ENT
                        ;                                IRE FRAME IS DONE
0ABB0 8000ABEE                 JMP EXITS
                        *
                        *
                        ;  BIT DESERIALIZATION WITHOUT ERROR CONTROL
                        *
0ABB2 E0609CA9          BBITOS MOVMR R6,POST4S         MOVE THE FRAME LENGTH TO
0ABB3 3E64                     SUBRR R6,R2             R6=LSEN-1-R2
```

PAGE 83: PROG. CSPOSP.TXT

```
0ABB5 0800      (03152)          EVEN
0ABB6 0D40      (03153)          CLR R4                  CLEAR R4
0ABB7 0800      (03154)          EVEN
0ABB8 9C209513  (03155)          ADDIR R2,RECVS-1        SET R2 TO BE THE ADDRESS
                (03156) ;                                :RECVS(R2)
0ABBA 3022      (03157) HBITOS1  POPXI R2,R1             POP RECV(R2)-->R1...R2=R
                (03158) ;                                2+1
0ABBB 2A02      (03159)          SEBCL 0,R1
0ABBC D2089EC4  (03160)          SPB 0,ROTDCTS(R4)       SET LOW BIT IN ROTDCT
0ABBE EE0005B0  (03161)          SMBSL 0,APDNFLS         HAS APDONE HAPPENED?
0ABC0 2021      (03162)          HOP $L+34               NO, HOP TO EVEN LOC AFTE
                (03163) ;                                R RET
0ABC1 0800      (03164)          EVEN
0ABC2 CC0005B0  (03165)          MOVZB APDNFLS
0ABC4 F1109F0C  (03166)          STORE R1,R1SAV
0ABC6 F1209F0E  (03167)          STORE R2,R2SAV
0ABC8 F1309F10  (03168)          STORE R3,R3SAV
0ABCA F1409F12  (03169)          STORE R4,R4SAV
0ABCC F1509F14  (03170)          STORE R5,R5SAV
0ABCE F1609F16  (03171)          STORE R6,R6SAV
0ABD0 F1709F18  (03172)          STORE R7,R7SAV
0ABD2 0E70      (03173)          RET
0ABD3 0800      (03174)          EVEN
0ABD4 A6109F0C  (03175)          LOAD R1,R1SAV
0ABD6 A6209F0E  (03176)          LOAD R2,R2SAV
0ABD8 A6309F10  (03177)          LOAD R3,R3SAV
0ABDA A6409F12  (03178)          LOAD R4,R4SAV
0ABDC A6509F14  (03179)          LOAD R5,R5SAV
0ABDE A6609F16  (03180)          LOAD R6,R6SAV
0ABE0 A6709F18  (03181)          LOAD R7,R7SAV
0ABE2 2641      (03182)          INCR R4,1               INCREMENT R4 BY 1
0ABE3 0800      (03183)          EVEN
0ABE4 8C60ABBA  (03184)          DJP R6,HBITOS1          LOOP BACK UNTIL THE ENT
                (03185) ;                                IRE FRAME IS DONE
                (03186) EXITS    EVEN
0ABE6 B0009E60  (03187)          JMP RECVSRA
                (03188)          EJECT
```

```
                      (03189) LGBUFS  EVEN
0ABE6 DE000571  (03190)         SMBSL 0,OUTFUL
0ABEA 80009E7E  (03191)         JMP LOBSRA1                    WAIT TIL AP OUT BUFF FUL
                      (03192) ;                                L
0ABEC E8020S6A  (03193)         MOVLM 1,OUTS                   CLEAR D/A INTERRUPT FLAG
0ABEE E849EECE  (03194)         MOVLM SETSGO,SYSFLGS
0ABF0 0260      (03195)         CCS 3                          FROM SRB=2...
0ABF1 06A0      (03196)         SCS 2                          TO SRB=1
0ABF2 02C0      (03197)         CCS 4                          FROM MRB=1...
0ABF3 0600      (03198)         SCS 5                          TO MRB=2
0ABF4 DE000580  (03199)         SMBSL 0,APDNFLS                HAS APDONE HAPPENED?
0ABF6 2005      (03200)         HOP #L+6
0ABF7 0800      (03201)         EVEN
0ABF8 CC000580  (03202)         MOVZM APDNFLS
0ABFA 0E70      (03203)         RET
0ABFB 0800      (03204)         EVEN
0ABFC E2000578  (03205)         CMPMZ RSYN
0ABFE 8120AC06  (03206)         JMP LOZS,LE                    OUTPUT SILENCE IF NO SYN
                      (03207) ;                                C
0AC00 E2000579  (03208)         CMPMZ NEWSYN
0AC02 8140AC32  (03209)         JMP LOBS,GE                    OUTPUT SPEECH IS WE'VE W
                      (03210) ;                                AITED LONG ENUFF
0AC04 E5000579  (03211)         INCM NEWSYN                    COUNT THIS AS ONE MORE F
                      (03212) ;                                RAME WAITED
0AC06 9040003D  (03213) LOZS    MOVIR R4,BLTH1-1
0AC08 90209202  (03214)         MOVIR R2,ZEOS-1
0AC0A E200056B  (03215)         CMPMZ OFLGS
0AC0C 8010AC20  (03216)         JMP LOZ1S,EQ
0AC0E D7281102  (03217)         BMOVEL R2,R4,NOUT2S            LOAD BUFFER #2 IF OFLG S
                      (03218) ;                                ET
0AC10 DE000580  (03219)         SMBSL 0,APDNFLS
0AC12 2005      (03220)         HOP #L+6
0AC13 0800      (03221)         EVEN
0AC14 CC000580  (03222)         MOVZM APDNFLS
0AC16 0E70      (03223)         RET
0AC17 0800      (03224)         EVEN
0AC18 9040003D  (03225)         MOVIR R4,BLTH1-1
0AC1A 90209202  (03226)         MOVIR R2,ZEOS-1
0AC1C D728117E  (03227)         BMOVEL R2,R4,NOUT2S+BLTH12
0AC1E 8000AC5C  (03228)         JMP GTIDLSS
0AC20 D728100C  (03229) LOZ1S   BMOVEL R2,R4,NOUT1S            LOAD BUFFER #1
0AC22 DE000580  (03230)         SMBSL 0,APDNFLS
0AC24 2005      (03231)         HOP #L+6
0AC25 0800      (03232)         EVEN
```

```
0AC26 CC000580 (03233)          MOVZB APDNFLS
0AC28 0E70     (03234)          RET
0AC29 0800     (03235)          EVEN
0AC2A 9040003C (03236)          MOVIR R3,BLTH1-1
0AC2C 90209202 (03237)          MOVIR R2,ZROS-1
0AC2E D728108B (03238)          BMOVEL R2,R4,NOUT1S+BLTH12
0AC30 8000AC5C (03239)          JMP GTIOLSS
0AC32 9030003C (03240) LOBS     MOVIR R4,BLTH1-1
0AC34 96209442 (03241)          MOVIR R2,OUTBS-2             'PREPARE FOR BLOCK MOVE
0AC36 E2000568 (03242)          CMPBZ OFLGS                  EXAMINE OFLG
0AC38 8010AC4C (03243)          JMP FL20S,EQ                 IF OFLG=0, GOTO FL20
0AC3A D7281192 (03244)          BMOVEL R2,R4,NOUT2S
0AC3C DE000580 (03245)          SMBSL 0,APDNFLS              HAS APDONE HAPPENED?
0AC3E 2005     (03246)          BOP #L+6
0AC3F 0800     (03247)          EVEN
0AC40 CC000580 (03248)          MOVZB APDNFLS
0AC42 0E70     (03249)          RET
0AC43 0800     (03250)          EVEN
0AC44 962094BE (03251)          MOVIR R2,OUTBS-2+BLTH12
0AC46 9040003C (03252)          MOVIR R4,BLTH1-2
0AC48 D728117E (03253)          BMOVEL R2,R4,NOUT2S+BLTH12
0AC4A 8000AC5C (03254)          JMP GTIOLSS
               (03255) ;
               (03256) ;
0AC4C D728100C (03257) FL20S    BMOVEL R2,R4,NOUT1S
0AC4E DE000580 (03258)          SMBSL 0,APDNFLS              HAS APDONE HAPPENED?
0AC50 2005     (03259)          BOP #L+6
0AC51 0800     (03260)          EVEN
0AC52 CC000580 (03261)          MOVZB APDNFLS
0AC54 0E70     (03262)          RET
0AC55 0800     (03263)          EVEN
0AC56 902094EE (03264)          MOVIR R2,OUTBS-2+BLTH12
0AC58 9040003C (03265)          MOVIR R4,BLTH1-2
0AC5A D728108B (03266)          BMOVEL R2,R4,NOUT1S+BLTH12
               (03267) ;
               (03268) ;
               (03269) GTIOLSS EVEN
0AC5C 0200     (03270)          CCS 5                        FROM MRB=2...
0AC5D 06C0     (03271)          SCS 4                        TO MRB=1
0AC5E EB09FFCF (03272)          MOVLM CLRSCO,SYSFLGS
0AC60 80009E7C (03273)          JMP LOBUSRA                  GO BACK TO DISPATCHER
               (03274) *  DEFINE GLOBAL SYMBOLS
     000049E2 (03275)          D16SS1=$49E2
     00004A00 (03276)          D16SS2=$4A00
```

```
        00004ACA (03277)          D22SS1=$4ACA
        00001AFF (03278)          F23SS1=$4AFF
        00004B12 (03279)          IOSSINT=$4B12
                 (03280) *
                 (03281) *A/D,D/A,MODEM INTERRUPT SERVICE RTNS
0AC62 CC000566 (03282) SENS1F  MOVZM ISAS100                     ANA=0
0AC64 CC000567 (03283)         MOVZM ISAS101                     AFLG=0
0AC66 80004B12 (03284)         JMP IOSSINT
0AC68 CC000566 (03285) SENS2F  MOVZM ISAS100                     ANA=0
0AC6A F0500503 (03286)         MOVMR R5,ISAS01
0AC6C F0500567 (03287)         MOVRM R5,ISAS101                  AFLG=1
0AC6E 80004B12 (03288)         JMP IOSSINT
0AC70 CC000568 (03289) RCVRS1F MOVZM ISAS102                     SYN=0
0AC72 CC000569 (03290)         MOVZM ISAS103                     SFLG=0
0AC74 90101IFF (03291)         MOVIR R1,RECV1S
0AC76 F200057B (03292)         CMRMZ KSYN
0AC78 2000     (03293)         NOP
0AC79 1620     (03294)         SKPL LE
0AC7A 8600A62F (03295)         CALL R0,SUPDAT                    DO SYNC UPDATE
0AC7C 80004B12 (03296)         JMP IOSSINT
0AC7E CC000568 (03297) RCVRS2F MOVZM ISAS102                     SYN=0
0AC80 F0500503 (03298)         MOVMR R5,ISAS01
0AC82 F0500569 (03299)         MOVRM R5,ISAS103                  SFLG=1
0AC84 90101369 (03300)         MOVIR R1,RECV2S
0AC86 F200057B (03301)         CMRMZ KSYN
0AC88 2000     (03302)         NOP
0AC89 1620     (03303)         SKPL LE
0AC8A 8600A62F (03304)         CALL R0,SUPDAT                    DO SYNC UPDATE
0AC8C 80004B12 (03305)         JMP IOSSINT
0AC8E CC00056A (03306) OUTS1F  MOVZM ISAS104                     OUT=0
0AC90 CC00056B (03307)         MOVZM ISAS105                     OFLG=0
0AC92 80004B12 (03308)         JMP IOSSINT
0AC94 CC00056A (03309) OUTS2F  MOVZM ISAS104                     OUT=0
0AC96 F0500503 (03310)         MOVMR R5,ISAS01
0AC98 F050056B (03311)         MOVRM R5,ISAS105                  OFLG=1
0AC9A 80004B12 (03312)         JMP IOSSINT
0AC9C CC00056C (03313) INS1F   MOVZM ISAS106                     INP=0
0AC9E CC00056D (03314)         MOVZM ISAS107                     IFLG=0
0ACA0 80004B12 (03315)         JMP IOSSINT
0ACA2 CC00056C (03316) INS2F   MOVZM ISAS106                     INP=0
0ACA4 F0500503 (03317)         MOVMR R5,ISAS01
0ACA6 F050056D (03318)         MOVRM R5,ISAS107                  IFLG=1
0ACA8 80004B12 (03319)         JMP IOSSINT
                 (03320) *
```

```
                   (03321) *  LINE SIDE OUTPUT
      000049F2 (03322)            #L=D16SS1
049F2 90700000 (03323) D16ST1  MOVIR R7,0
049F4 8600AC62 (03324)            CALL R0,SENS1F
049F6 0E70     (03325)            RET
049F7 0800     (03326)            EVEN
049F8 90700001 (03327)            MOVIR R7,1
049FA 8600AC66 (03328)            CALL R0,SENS2F
049FC 0E70     (03329)            RET
049FD 0800     (03330)            EVEN
049FE 800049F2 (03331)            JMP D16ST1
               (03332) *
               (03333) *  LINE SIDE INPUT
      00004A00 (03334)            #L=D16SS2
04A00 90700010 (03335) D16ST2  MOVIR R7,16
04A02 8600AC70 (03336)            CALL R0,RCVRS1F
04A04 0E70     (03337)            RET
04A05 0800     (03338)            EVEN
04A06 90700011 (03339)            MOVIR R7,17
04A08 8600AC7E (03340)            CALL R0,RCVRS2F
04A0A 0E70     (03341)            RET
04A0B 0800     (03342)            EVEN
04A0C 80004A00 (03343)            JMP D16ST2
               (03344) *
               (03345) *  SPEECH SIDE OUTPUT
      00004ACA (03346)            #L=D22SS1
04ACA 9070000C (03347) D22ST1  MOVIR R7,12
04ACC 8600ACBE (03348)            CALL R0,OUTS1F
04ACE 0E70     (03349)            RET
04ACF 0800     (03350)            EVEN
04AD0 9070000D (03351)            MOVIR R7,13
04AD2 8600AC94 (03352)            CALL R0,OUTS2F
04AD4 0E70     (03353)            RET
04AD5 0800     (03354)            EVEN
04AD6 80004ACA (03355)            JMP D22ST1
               (03356) *
               (03357) *  SPEECH SIDE INPUT
      00004AEE (03358)            #L=D23SS1
04AEE 9070000E (03359) D23ST1  MOVIR R7,14
04AF0 8600AC9C (03360)            CALL R0,INS1F
04AF2 0E70     (03361)            RET
04AF3 0800     (03362)            EVEN
04AF4 9070000F (03363)            MOVIR R7,15
04AF6 8600ACA2 (03364)            CALL R0,INS2F
```

```
04AF8 0E70      (03365)         RET
04AF9 0800      (03366)         EVEN
04AFA E0004AFE  (03367)         JMP D23S11
                (03368) *
                (03369) *  PATCH EXECUTIVE TO CREATE BINARY VALUED I/O FLAGS
04AFC 80004B12  (03370)         JMP IOSSINT
                (03371)         EJECT
```

PAGE 90: PROG. CSFBSP.TXT

```
ACOTH:     0000A (01968) (02668)
AFDTSORG:  008FB (00010)
AFLGS:     00567 (00079) (01597) (01606) (01683)
ANAS:      00566 (00077) (00824)
APONFLS:   00580 (00099) (00697) (00700) (00706) (00709) (00715) (00718) (00723) (00726) (00733)
                 (00736) (00747) (00750) (00756) (00759) (00764) (00767) (00844) (00847) (00856)
                 (00859) (00876) (00880) (00908) (00912) (00936) (00940) (01014) (01018) (01086)
                 (01096) (01129) (01133) (01156) (01160) (01201) (01205) (01228) (01232) (01271)
                 (01275) (01303) (01307) (01348) (01352) (01391) (01395) (01439) (01443) (01479)
                 (01483) (01576) (01580) (01613) (01617) (01638) (01642) (01662) (01666) (01688)
                 (01692) (01712) (01716) (01736) (01740) (01804) (01808) (01828) (01832) (01852)
                 (01856) (01879) (01882) (01888) (01891) (01897) (01900) (01908) (01911) (01918)
                 (01921) (01927) (01930) (01936) (01939) (02015) (02019) (02162) (02166) (02185)
                 (02189) (02208) (02212) (02231) (02235) (02254) (02258) (02277) (02281) (02306)
                 (02310) (02341) (02345) (02368) (02372) (02435) (02439) (02586) (02590) (02618)
                 (02622) (02716) (02720) (02783) (02787) (02812) (02816) (02856) (02860) (02883)
                 (02887) (02927) (02931) (02954) (02958) (02998) (03002) (03028) (03032) (03073)
                 (03077) (03114) (03118) (03161) (03165) (03199) (03202) (03219) (03222) (03230)
                 (03233) (03245) (03248) (03258) (03261)
BARS:      09C95 (00223)
BCH24S:    0ABCF (02566) (02572)
BCH25S:    0ABFF (02556) (02560) (02572)
BCHD0S:    0A758 (02299) (02302)
BCHD1S:    0A75C (02306)
BCHD2S:    0A786 (02332) (02365)
BCHD3S:    0A792 (02333) (02341)
BCHD4S:    0AB7F (02502) (02509)
BCHD5S:    0AB6A (02489) (02494) (02509)
BCHDAS:    0AB94 (02393) (02522) (02544) (02573)
BCHDBS:    0ABAA (02418) (02546)
BCHDCS:    0AB80 (02480) (02549) (02550)
BCHDDS:    0ABA6 (02513) (02543)
BCHDSS:    0AB00 (02398) (02416)
BCHDTS:    0AB16 (02426) (02435)
BCHDUS:    0AB5B (02458) (02469) (02478)
BCHF0S:    0A2FF (01473) (01476)
BCHF1S:    0A2F0 (01478)
BCHF2S:    0A318 (01503) (01513)
BCHF3S:    0A320 (01504) (01510)
BF1S:      0BFB0 (00169) (01878) (01887) (01896) (01904) (01955) (02154)
BF2S:      0BFF1 (00171) (01800) (01803) (01827) (01851) (01959) (02157)
BF3S:      09162 (00173) (01907) (01917) (01926) (01935)
BIT0S:     0A0BB (00195) (00996)
BIT1S:     0A0B4 (00195) (00195) (00989)
```

```
BIT2S:     0A680 (00196) (00196) (00982)
BIT3S:     0A07C (00196) (00196) (00975)
BIT4S:     0A078 (00195) (00195) (00196) (00196) (00968)
BIT5S:     0A074 (00195) (00961)
BLTH:      000F6 (00062)
BLTH1:     0004F (00068) (00838) (00851) (00863) (03213) (03225) (03236) (03240) (03252) (03265)
BLTH12:    0007C (00069) (00850) (00852) (00862) (00864) (03227) (03238) (03251) (03253) (03264)
                 (03266)
BUS2:      0A654 (02113) (02145)
CLRSG0:    00004 (00015) (00742) (02715) (03272)
CLRSG2:    00006 (00016)
CLRSG3:    00007 (00017) (00704) (00867)
CPTP1S:    09F88 (00778) (01119) (01190) (01262) (01337) (02846) (02917) (02988) (03062)
CSPUSNOS:  021FC (00011)
CS*SP24:   09F1A (00107) (00689)
CS*SSTD6:  00405 (00014) (00689)
D16ST1:    049F2 (03324) (03331)
D16S12:    04A00 (03335) (03343)
D16SS1:    049F2 (03275) (03322)
D16SS2:    04A00 (03276) (03334)
D22ST1:    04ACA (03347) (03355)
D22SS1:    04ACA (03277) (03346)
D23ST1:    04AFF (03359) (03367)
D23SS1:    04AFF (03278) (03358)
DBIT0S:    0A242 (00198) (01301) (01344)
DBIT0S1:   0A278 (01383) (01421)
DBIT1S:    0A1DC (00198) (01227) (01270)
DBIT1S1:   0A208 (01299) (01332) (01339)
DBIT2S:    0A176 (00198) (01155) (01198)
DBIT2S1:   0A1A2 (01225) (01257) (01264)
DBIT3S:    0A110 (00198) (01083) (01127)
DBIT3S1:   0A13C (01153) (01185) (01192)
DBIT4S:    0A0D2 (00198) (01078)
DBIT4S1:   0A0D6 (01081) (01114) (01121)
DMYS:      00794 (00022)
DSGDS:     0A966 (02716) (02739)
DSGD1S:    0A9A0 (02755) (02759)
DSRBUFS:   0A8D4 (02543) (02578) (02581)
FBIT0S:    0A2B2 (01370) (01428)
FBIT0S1:   0A2BC (01436) (01465)
FIGTENS:   09C8E (00216) (00778) (01035) (01419) (01420) (02683) (03141) (03142)
FNDS0:     0A2AF (01413) (01418) (01421)
FNDDCS:    0A807 (02538) (02577)
FNDFCS:    0A35C (01565)
```

```
FNDSES:     0A2E8 (01068) (01128) (01200) (01293) (01346) (01423) (01467)
FERS:       09C91 (00219) (00785) (00789) (00799) (00803)
FERSREM:    09E9F (00801) (00804) (00806)
FERSRET:    09E92 (00787) (00791) (00793)
FRR1S:      09E8A (00785)
FRR2S:      09E8E (00789)
FRR3S:      09E96 (00799)
FRR4S:      09E9A (00803)
FRSYN:      0057C (00047) (02076) (02122) (02132)
EXITS:      0A0E6 (02774) (02855) (02926) (02997) (03071) (03145) (03186)
FRITOS:     0A94A (00200) (02671)
FRIT1S:     0A944 (00200) (00200) (02665)
FRIT2S:     0A93E (00201) (00201) (02659)
FRIT3S:     0A938 (00201) (00201) (02653)
FRIT4S:     0A932 (00200) (00200) (00201) (00201) (02647)
FRIT5S:     0A92C (00200) (02641)
FL11S:      0A530 (01797) (01877)
FL12S:      0A592 (01874) (01942)
FL16S:      0A662 (02153) (02157)
FL17S:      0A664 (02156) (02159)
FL20S:      0AC4C (03243) (03257)
FLRS:       09EE0 (00841) (00854)
FLR1S:      09EE2 (00853) (00865)
FL9S:       0A414 (01608) (01685)
FORFORS:    09C8F (00217) (01053) (01115) (01186) (01258) (01333) (01414) (02760) (02842) (02913)
                  (02984) (03058) (03136)
FRMLNGS:    09C88 (00213)
FRSTF:      09C84 (00205) (02081) (02754) (02757)
GRITOS:     0AB44 (00203) (03026) (03069)
GRIT0S1:    0AB78 (03107) (03143)
GRIT1S:     0AAGF (00203) (02953) (02996)
GRIT1S1:    0AB0A (03024) (03057) (03064)
GRIT2S:     0AA78 (00203) (02882) (02925)
GRIT2S1:    0AAA4 (02951) (02983) (02990)
GRIT3S:     0AA12 (00203) (02809) (02854)
GRIT3S1:    0AA3E (02880) (02912) (02919)
GRIT4S:     0A9B0 (00203) (02763)
GRIT4S1:    0A9DB (02807) (02841) (02848)
GENDS0:     0ABAF (03135) (03140) (03143)
GTIOENS:    09E0B (00675)
GTIOLSS:    0AC5C (00675) (03228) (03239) (03254) (03269)
GTIOSTS:    09E0A (00674)
HS:         00001 (00050) (01775)
HRITOS:     0AB02 (03095) (03150)
```

```
HGITOS1:    0AGGA (03157) (03184)
IGITS:      0866F (00163) (01056) (01078) (01081) (01150) (01153) (01222) (01225) (01294) (01299)
IDEVS24:    0001B (00012) (00110) (00122) (00123)
IDWSRD:     00001 (01966) (02036) (02066) (02136)
IFLGS:      0056D (00086) (00840)
ILVLS1:     00001 (00013) (00110) (00122) (00123)
INS1F:      0AC9C (03313) (03360)
INS2F:      0ACA2 (03316) (03364)
INDX1S:     09C40 (00195) (00957)
INDX2S:     09C58 (00198) (01071)
INDX4S:     09C62 (00200) (02639)
INDX5S:     09C7A (00203) (02777)
INFULL:     00570 (00095) (00691) (00722) (00829)
INITS:      00573 (00098) (00793) (00806)
INPS:       0056C (00085) (00712) (00833)
INPGS:      08D80 (00167) (00843) (00852) (00855) (00864)
IODSP1S:    09F21 (00693)
IODSPTS:    09F1C (00674) (00689) (00690)
IOSSINT:    04B12 (03279) (03284) (03288) (03296) (03305) (03308) (03312) (03315) (03319) (03370)
ISAS01:     00504 (00024) (03286) (03298) (03310) (03317)
ISAS100:    00566 (00025) (00077) (03282) (03285)
ISAS101:    00567 (00026) (00079) (03283) (03287)
ISAS102:    00568 (00027) (00081) (03289) (03297)
ISAS103:    00569 (00028) (00082) (03290) (03299)
ISAS104:    0056A (00029) (00083) (03306) (03309)
ISAS105:    0056B (00030) (00084) (03307) (03311)
ISAS106:    0056C (00031) (00085) (03313) (03316)
ISAS107:    0056D (00032) (00086) (03314) (03318)
ISAS108:    0056E (00033) (00093)
ISAS109:    0056F (00034) (00094)
ISAS110:    00570 (00035) (00095)
ISAS111:    00571 (00036) (00096)
ISAS112:    00572 (00037) (00097)
ISAS113:    00573 (00038) (00098)
ISAS127:    00581 (00048) (00087)
ISVTS:      00502 (00023) (00024) (00025) (00026) (00027) (00028) (00029) (00030) (00031) (00032)
                  (00033) (00034) (00035) (00036) (00037) (00038) (00039) (00040) (00041) (00042)
                  (00043) (00044) (00045) (00046) (00047) (00048)
KOUNTS:     00572 (00097)
LOBS:       0AC32 (03209) (03240)
LOBSRA1:    09F7E (00764) (03191)
LOBUSRA:    09F7C (00763) (03273)
LOBUFS:     0AFFB (00755) (03189)
LODATS:     0A35F (01566) (01571)
```

```
LOOES:     0A04C (00934) (01[illegible])
LOOPS1:    0A904 (02616) (02680)
LOZS:      0AC66 (03206) (03213)
LOZ1S:     0AC20 (03216) (03229)
LPARM:     0000D (00066) (00214) (00214)
LSEN:      00171 (00065) (00211) (01904) (01907)
LSEN1:     0003F (00070) (00898) (00901) (00902) (00904) (00905) (01605) (01635) (01660) (01710)
                 (01734) (02160) (02183) (02184) (02206) (02207) (02229) (02230) (02252) (02253)
                 (02275) (02276)
LSEN12:    0007C (00071) (01634) (01636) (01659) (01661) (01709) (01711) (01733) (01735)
LSEN2:     0003E (00072) (01795) (01826) (01850) (01866) (01895) (01916) (01925) (01934)
LSEN22:    0007A (00073) (01825) (01827) (01874) (01851) (01885) (01887) (01894) (01896) (01903)
                 (01924) (01926) (01933) (01935)
LSTER:     0057B (00046) (02083) (02119) (02124) (02131)
LSTOR:     00004 (01967) (02075) (02121)
LTH:       001C0 (00063) (00213) (02751)
LTH1:      00640 (00064)
MS:        00060 (00051)
MSS:       00000 (00052)
MASKS:     09CBE (00215)
MBLTH:     00171 (01969) (01980) (01993) (02011)
MOVSPA1:   09E4A (00723) (00830)
MOVES:     09EAA (00714) (00829)
MOVESP:    0A49C (01781)
MOVESPA:   09E4B (00722) (00868)
NESS:      09EFC (00659) (02304) (02482) (02492) (02508) (02511) (02516) (02518) (02520) (02522)
                 (02525) (02529) (02543) (02546) (02550)
NESTS:     09EFB (00658) (02395) (02491) (02504) (02507) (02512)
NEWMAX:    0A5EB (02044)
NEWSYN:    00579 (00044) (02079) (03208) (03211)
NIN1S:     00D2A (00131) (00854) (00862)
NIN2S:     00F20 (00133) (00842) (00F50)
NOMAX:     0A5EE (02033) (02053)
NOUT1S:    0100C (00135) (03229) (03238) (03257) (03266)
NOUT2S:    01102 (00137) (03217) (03227) (03244) (03253)
OFLGS:     00568 (00084) (03215) (03242)
OLSYN:     0057A (00045) (02088) (02115) (02138)
ONES:      09C90 (00218) (01382) (01435)
OUTS:      0056A (00083) (00753) (03193)
OUTS1F:    0AC8F (03306) (03348)
OUTS2F:    0AC94 (03309) (03352)
OUTBS:     09444 (00178) (03241) (03251) (03264)
OUTFUL:    00571 (00096) (00692) (00763) (03190)
PARMSP2:   09CBC (00214) (00933) (02613)
```

```
POST2S:     09CB5 (00207)
POST3S:     09CB6 (00208) (01117) (01186) (01260) (01335) (01369) (02844) (02915) (02986) (03060)
                  (03094)
POST3S1:    09CB7 (00209) (01416)
POST3S2:    09CB6 (00210) (03138)
POST4S:     09CB9 (00211) (01127) (01199) (01292) (01344) (01375) (01428) (02854) (02925) (02996)
                  (03069) (03100) (03150)
OPES:       08A00 (00153) (00934) (01036) (01041)
QTDCS:      08A09 (00156)
QTDCTS:     08A0F (00161) (01107) (01178) (01250) (01325) (01378) (01431)
QTGS:       08A0C (00159)
QTMS:       08A0A (00157)
QTPARS:     08A00 (00154)
QTVS:       08A0B (00158)
QIVAR:      08A08 (00155)
R1SAV:      09F0C (00676) (00881) (00890) (00913) (00922) (00941) (00950) (01019) (01028) (01091)
                  (01100) (01134) (01143) (01161) (01170) (01206) (01215) (01233) (01242) (01276)
                  (01285) (01308) (01317) (01353) (01362) (01396) (01405) (01444) (01453) (01484)
                  (01493) (01581) (01590) (01618) (01627) (01643) (01652) (01667) (01676) (01693)
                  (01702) (01717) (01726) (01741) (01750) (01809) (01818) (01833) (01842) (01857)
                  (01866) (02020) (02029) (02167) (02176) (02190) (02199) (02213) (02222) (02236)
                  (02245) (02259) (02268) (02282) (02291) (02311) (02320) (02346) (02355) (02373)
                  (02382) (02440) (02449) (02591) (02600) (02623) (02632) (02721) (02730) (02788)
                  (02797) (02817) (02826) (02861) (02870) (02888) (02897) (02932) (02941) (02959)
                  (02968) (03003) (03012) (03033) (03042) (03078) (03087) (03119) (03128) (03166)
                  (03175)
R2SAV:      09F0F (00677) (00882) (00891) (00914) (00923) (00942) (00951) (01020) (01029) (01092)
                  (01101) (01135) (01144) (01162) (01171) (01207) (01216) (01234) (01243) (01277)
                  (01286) (01309) (01318) (01354) (01363) (01397) (01406) (01445) (01454) (01485)
                  (01494) (01582) (01591) (01619) (01628) (01644) (01653) (01668) (01677) (01694)
                  (01703) (01718) (01727) (01742) (01751) (01810) (01819) (01834) (01843) (01858)
                  (01867) (02021) (02030) (02168) (02177) (02191) (02200) (02214) (02223) (02237)
                  (02246) (02260) (02269) (02283) (02292) (02312) (02321) (02347) (02356) (02374)
                  (02383) (02441) (02450) (02592) (02601) (02624) (02633) (02722) (02731) (02789)
                  (02798) (02818) (02827) (02862) (02871) (02889) (02898) (02933) (02942) (02960)
                  (02969) (03004) (03013) (03034) (03043) (03079) (03088) (03120) (03129) (03167)
                  (03176)
R3SAV:      09F10 (00678) (00883) (00892) (00915) (00924) (00943) (00952) (01021) (01030) (01093)
                  (01102) (01136) (01145) (01163) (01172) (01208) (01217) (01235) (01244) (01278)
                  (01287) (01310) (01319) (01355) (01364) (01398) (01407) (01446) (01455) (01486)
                  (01495) (01583) (01592) (01620) (01629) (01645) (01654) (01669) (01678) (01695)
                  (01704) (01719) (01728) (01743) (01752) (01811) (01820) (01835) (01844) (01859)
                  (01868) (02022) (02031) (02169) (02178) (02192) (02201) (02215) (02224) (02238)
                  (02247) (02261) (02270) (02284) (02293) (02313) (02322) (02348) (02357) (02375)
```

```
           (02384) (02442) (02451) (02593) (02602) (02625) (02634) (02723) (02732) (02790)
           (02799) (02819) (02826) (02863) (02872) (02890) (02899) (02934) (02943) (02961)
           (02970) (03005) (03014) (03035) (03044) (03080) (03089) (03121) (03130) (03168)
           (03177)
R4SAV: 09F12 (00879) (00881) (00893) (00916) (00925) (00944) (00953) (01022) (01031) (01094)
           (01103) (01137) (01146) (01164) (01173) (01209) (01218) (01236) (01245) (01279)
           (01288) (01311) (01320) (01356) (01365) (01399) (01408) (01447) (01456) (01487)
           (01496) (01584) (01593) (01621) (01630) (01646) (01655) (01670) (01679) (01696)
           (01705) (01720) (01729) (01744) (01753) (01812) (01821) (01836) (01845) (01860)
           (01869) (02023) (02032) (02170) (02179) (02193) (02202) (02216) (02225) (02239)
           (02248) (02262) (02271) (02285) (02294) (02314) (02323) (02349) (02358) (02376)
           (02385) (02443) (02452) (02594) (02603) (02626) (02635) (02724) (02733) (02750)
           (02759) (02791) (02800) (02820) (02829) (02864) (02873) (02891) (02900) (02935)
           (02941) (02962) (02971) (03006) (03015) (03036) (03045) (03081) (03090) (03122)
           (03131) (03169) (03178)
R5SAV: 09F11 (00880) (00885) (00894) (00917) (00926) (00945) (00954) (01023) (01032) (01095)
           (01101) (01138) (01147) (01165) (01174) (01210) (01219) (01237) (01246) (01280)
           (01289) (01312) (01321) (01357) (01366) (01400) (01409) (01448) (01457) (01488)
           (01497) (01585) (01594) (01622) (01631) (01647) (01656) (01671) (01680) (01697)
           (01706) (01721) (01730) (01745) (01754) (01813) (01822) (01837) (01846) (01861)
           (01870) (02024) (02033) (02171) (02180) (02194) (02203) (02217) (02226) (02240)
           (02249) (02263) (02272) (02286) (02295) (02315) (02324) (02350) (02359) (02377)
           (02386) (02444) (02453) (02595) (02604) (02627) (02636) (02725) (02734) (02792)
           (02801) (02821) (02830) (02865) (02874) (02892) (02901) (02936) (02945) (02963)
           (02972) (03007) (03016) (03037) (03046) (03082) (03091) (03123) (03132) (03170)
           (03179)
R6SAV: 09F16 (00881) (00886) (00895) (00918) (00927) (00946) (00955) (01024) (01033) (01096)
           (01105) (01139) (01148) (01166) (01175) (01211) (01220) (01238) (01247) (01281)
           (01290) (01313) (01322) (01358) (01367) (01401) .01410) (01449) (01458) (01489)
           (01498) (01586) (01595) (01623) (01632) (01648) (01657) (01672) (01681) (01698)
           (01707) (01722) (01731) (01746) (01755) (01814) (01823) (01838) (01847) (01862)
           (01871) (02025) (02034) (02172) (02181) (02195) (02204) (02218) (02227) (02241)
           (02250) (02264) (02273) (02287) (02296) (02316) (02325) (02351) (02360) (02378)
           (02387) (02445) (02454) (02596) (02605) (02628) (02637) (02726) (02735) (02793)
           (02802) (02822) (02831) (02866) (02875) (02893) (02902) (02937) (02946) (02964)
           (02973) (03008) (03017) (03038) (03047) (03083) (03092) (03124) (03133) (03171)
           (03180)
R7SAV: 09F18 (00882) (00887) (00896) (00919) (00928) (00947) (00956) (01025) (01034) (01097)
           (01106) (01140) (01149) (01167) (01176) (01212) (01221) (01239) (01248) (01282)
           (01291) (01314) (01323) (01359) (01368) (01402) (01411) (01450) (01459) (01490)
           (01499) (01567) (01596) (01624) (01633) (01649) (01658) (01673) (01682) (01699)
           (01708) (01723) (01732) (01747) (01756) (01815) (01824) (01839) (01848) (01863)
           (01872) (02026) (02035) (02173) (02182) (02196) (02205) (02219) (02228) (02242)
           (02251) (02265) (02274) (02288) (02297) (02317) (02326) (02352) (02361) (02379)
```

```
                 (02388) (02446) (02455) (02597) (02606) (02629) (02638) (02727) (02736) (02794)
                 (02803) (02823) (02832) (02867) (02876) (02894) (02903) (02938) (02947) (02965)
                 (02974) (03009) (03018) (03039) (03048) (03084) (03093) (03125) (03134) (03172)
                 (03181)
RCVOFF:    00561 (00087) (00746) (01775)
RCVFS1F:   0AC70 (03289) (03336)
RCVRS2F:   0AC7F (03297) (03340)
RDATAS:    0056F (00093) (01787) (01792) (01796) (01957) (02152)
RECVS:     09544 (00160) (00210) (02161) (02184) (02207) (02230) (02253) (02276) (02332) (02517)
                 (02519) (02521) (02559) (02571) (02641) (02647) (02653) (02659) (02665) (02671)
                 (02687) (02693) (02833) (02904) (02975) (03049) (03105) (03155)
RECVSRA:   09E60 (00741) (00807) (01974) (02089) (02094) (02758) (03187)
RECV1S:    011EB (00139) (01877) (01885) (01894) (01903) (01914) (01924) (01933) (03291)
RECV2S:    01369 (00141) (01799) (01825) (01849) (03300)
REGS:      09F01 (00662)
RGO:       00574 (00039) (00729) (01774)
RGO2:      00576 (00041) (02737) (02740)
RIRITS:    09AA6 (00192) (02763) (02804) (02807) (02877) (02880) (02948) (02951) (03019) (03024)
RMISIS:    0A5A4 (01963) (01971)
RMISSIN:   0A5A8 (01974)
RMISSR:    0A5AC (01964) (01975)
ROFHS:     096B6 (00182) (02616) (02642) (02648) (02654) (02660) (02666) (02672) (02686) (02688)
                 (02693)
ROTDCTS:   096C4 (00184) (02752) (02835) (02906) (02977) (03051) (03110) (03160)
RSSPF:     09935 (00190) (01965) (02038) (02060)
RSSSS:     097C4 (00187) (01991) (01992) (01994) (02040) (02042) (02044) (02059)
RSYN:      0057B (00043) (01960) (01972) (01989) (02072) (02092) (02142) (03205) (03292) (03301)
SISFS:     09F00 (00661) (02406) (02428)
SAS11:     00398 (00055)
SAVES:     09C92 (00220)
SCPS19:    00DFE (00118) (00124)
SCHKS:     0A594 (01955)
SCHKIS:    0A59C (01958) (01960)
SENS:      08C0F (00165) (00209) (00899) (00902) (00905) (00963) (00970) (00977) (00984) (00991)
                 (00998) (01037) (01043) (01108) (01179) (01251) (01326) (01380) (01433) (01503)
                 (01520) (01522) (01524) (01526) (01528) (01530) (01532) (01534) (01536) (01538)
                 (01540) (01542) (01544) (01546) (01548) (01550) (01552) (01554) (01604) (01610)
                 (01634) (01659) (01685) (01709) (01733)
SENS1F:    0AC62 (03282) (03324)
SENS2F:    0AC6B (03285) (03328)
[illegible]1S:     013DC (00113) (01687) (01711) (01735)
SEN2S:     0164F (00145) (01612) (01636) (01661)
SETSG0:    00024 (00018) (01781) (02741) (03194)
SETSG2:    00026 (00019)
```

```
SEISG3:    00027 (00026) (00823) (00F32)
SELGS:     00569 (00082) (01786)
STS6MO:    0A58A (01991)
STSLP:     0A561 (01984) (01988)
SRHOES:    09EEB (00826) (00871)
SSSALT:    0A5E2 (02040)
SSSLP:     0A5CB (02015) (02064)
SSSIPT:    0A604 (02050) (02053) (02056) (02060)
SSSNAL:    0A602 (02049) (02059)
SSSNO:     0A62A (02069) (02071) (02092)
START:     08A00 (00053)
SUSELE:    0A63E (02120) (02124)
SUSERR:    0A632 (02118) (02129)
SUSTS:     0A650 (02134) (02142)
SUSVAL:    0A64A (02126) (02136)
SUPDAT:    0A62E (02111) (03295) (03304)
SVTS:      00382 (00054) (00055)
SYNS:      0056B (00061) (01778)
SYNB1S:    097C4 (00186) (00187)
SYNB2S:    09935 (00189) (00190)
SYNCS:     00577 (00042) (02074) (02111) (02159)
SYNIN:     0A5AF (01973) (01980)
SYNSP:     0A5C4 (01975) (02011)
SYSELGS:   1FFCF (00021) (00704) (00742) (00823) (00832) (00867) (01781) (02715) (02741) (03194)
                 (03272)
TRAL1S:    09E38 (00465) (02494) (02560)
TRAL2S:    09E78 (00530) (02496) (02561)
TRAL3S:    09EBB (00594) (02497)
TRENCS:    09CA2 (00224) (01506)
TRPRCS:    09DB9 (00336) (02415) (02434) (02465) (02477)
TRPSMS:    09CEC (00271) (02335) (02338)
TRRPCS:    09DE8 (00400) (02399) (02427) (02459) (02461) (02470)
TRRSLDS:   09C94 (00222)
TRANSRA:   09E30 (00704) (00795) (01760)
TSTS0:     09E22 (00694) (00762) (00770)
TSTS1:     09E3A (00704) (00712)
TSTS2:     09E52 (00721) (00729)
TSTS3:     09E6E (00739) (00753)
UNLS:      09E0E (00843)
UNPEELS:   0A656 (01965) (02147)
WS:        00002 (00056) (01991) (01994)
XDATAS:    0056F (00094) (01598) (01603)
XGO:       00575 (00040) (00694) (00822)
XMITS:     09EA2 (00695) (00822)
```

```
PAGE    99:        PROG. CSPUSP.TXT

  XMTISE:     03490 (01684) (01757)
  XRECVS:     0A496 (00731) (01774)
  ZS:         00003 (00057)
  ZZ1S:       09C93 (00221)
  ZEROS:      09CBA (00212)
  ZROS:       092D3 (00175) (00897) (00900) (00903) (03214) (03226) (03237)

LINES WITH ERRORS:     0   (MAP VERSION 800101.10)    E=     0
```

3.6.3 Fortran Programs

The modules contained in this section are:

FCBGN
FF2R
VMOV1
VMOV2
VMOV3
MPFDVM
MPDCTM
MPQDPP
MPDQPP
MPMWGX
MPFSTV
MPSSRT
MPIDCM
MPASRT
MPFSTQ
MPFSTD
MPVDNM
MPBASP
MPSCAN
MPCDBA
MASTER
INIT
USER
MAPON
CREATE
DEFINE
SETUP
LOAD
IOSM
AOM
ADAM
SYSTMA
INPUT
DCVAR
DCTF

PITLPC
QDPARM
VPBS
BASIS
ASRT
SDCT
BITA
QDCT
CHANL
DPARM
SSRT
DDCT
DCTR
VARDC
OUTPUT
SNR
TAPE2
OPT1
OPT2
OPT3
OPT4
OPT5
OPT6
OPT7
OPT8
OPT8A (OPT8B and OPT8C are similar)
PREMAP
FILEN

```
0001          INTEGER FUNCTION FCBGN(FCBNO,Y,A,U,B,V,C,W,D,IS)
        C
        C TITLE: FCBGN
        C
        C
        C PURPOSE:
        C       GENERATE FCB AND START EXECUTION OF MAP
        C
        C F4 USAGE:
        C
        C    MAP = FCBGN(FCBNO,Y,A,U,B,V,C,W,D,IS)
        C
        C DESCRIPTION OF PARAMETERS
        C      FCBNO INTEGER VARIABLE SPECIFYING FCB NUMBER
        C      Y,U,V,W - INTEGER VARIABLES SPECIFYING MAP
        C                BUFFER IDENTIFIERS
        C      A,B,C,D - INTEGER VARIABLES SPECIFYING MAP
        C                SCALAR IDENTIFIERS
        C      IS      - POINTER TO LOGICAL ARRAY  SUPPLIED BY BLOCK DATA. K-TH
        C                ELEMENT IS .TRUE. IF K-TH ID ARGUMENT
        C                IS TO BE USED, OTHERWISE .FALSE.
        C      IER     - OPTIONAL ERROR INDICATOR
        C                MAP =0 INDICATES NO ERROR (IDENTIFIERS WITHIN PROPER BOUNDS).
        C                MAP =-K INDICATES K-TH ARGUMENT IN
        C                THE CALLING LIST IS
        C                OUTSIDE LEGAL BOUNDS. MAP EXECUTION ABORTED.
        C
        C
        C FUNCTIONS AND SUBROUTINES REQUIRED
        C      RUNMP(FCBG,FCBS) - INDICATES MAP EXECUTION.
        C
0002          INTEGER Y,A,U,B,V,C,W,D,FCBNO,BIDMN,BIDMX,SIDMN,SIDMX
0003          INTEGER FCBG, FCBS, FCBN2, RUNMP
0004          INTEGER FCBSZ,HWS,FCB,HRI
0005          LOGICAL ISARG
0006          DIMENSION ISARG(8,22)
        C
0007          COMMON/EPZZZ/FCBG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LEVEL
0008          DIMENSION FCBS(11)
        C
0009          DATA BIDMN/1/,BIDMX/63/,SIDMN/0/,SIDMX/191/
0010          DATA FCBS/0,10*8/
0011          DATA ISARG(1,1),ISARG(2,1),ISARG(3,1),ISARG(4,1),
             1     ISARG(5,1),ISARG(6,1),ISARG(7,1),ISARG(8,1)/
             2  .FALSE.,.TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,2 * .FALSE./
0012          DATA ISARG(1,2),ISARG(2,2),ISARG(3,2),ISARG(4,2),
             1     ISARG(5,2),ISARG(6,2),ISARG(7,2),ISARG(8,2)/
             2  4*.TRUE.,4*.FALSE./
0013          DATA ISARG(1,3),ISARG(2,3),ISARG(3,3),ISARG(4,3),
             1     ISARG(5,3),ISARG(6,3),ISARG(7,3),ISARG(8,3)/
             2  6*.TRUE.,2*.FALSE./
0014          DATA ISARG(1,4),ISARG(2,4),ISARG(3,4),ISARG(4,4),
             1     ISARG(5,4),ISARG(6,4),ISARG(7,4),ISARG(8,4)/
             2  5*.TRUE.,3*.FALSE./
0015          DATA ISARG(1,5),ISARG(2,5),ISARG(3,5),ISARG(4,5),
             1     ISARG(5,5),ISARG(6,5),ISARG(7,5),ISARG(8,5)/
```

```
               2  3*.TRUE.,5*.FALSE./
0016            DATA ISARG(1,6),ISARG(2,6),ISARG(3,6),ISARG(4,6),
               1     ISARG(5,6),ISARG(6,6),ISARG(7,6),ISARG(8,6)/
               2  .FALSE.,3*.TRUE.,4*.FALSE./
0017            DATA ISARG(1,7),ISARG(2,7),ISARG(3,7),ISARG(4,7),
               1     ISARG(5,7),ISARG(6,7),ISARG(7,7),ISARG(8,7)/
               2  .FALSE.,4*.TRUE.,3*.FALSE./
0018            DATA ISARG(1,8),ISARG(2,8),ISARG(3,8),ISARG(4,8),
               1     ISARG(5,8),ISARG(6,8),ISARG(7,8),ISARG(8,8)/
               2  .TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,3*.FALSE./
0019            DATA ISARG(1,9),ISARG(2,9),ISARG(3,9),ISARG(4,9),
               1     ISARG(5,9),ISARG(6,9),ISARG(7,9),ISARG(8,9)/
               2  3*.TRUE.,.FALSE.,.TRUE.,3*.FALSE./
0020            DATA ISARG(1,10),ISARG(2,10),ISARG(3,10),ISARG(4,10),
               1     ISARG(5,10),ISARG(6,10),ISARG(7,10),ISARG(8,10)/
               2  2*.FALSE.,.TRUE.,5*.FALSE./
0021            DATA ISARG(1,11),ISARG(2,11),ISARG(3,11),ISARG(4,11),
               1     ISARG(5,11),ISARG(6,11),ISARG(7,11),ISARG(8,11)/
               2  .TRUE.,.FALSE.,.TRUE.,5*.FALSE./
0022            DATA ISARG(1,12),ISARG(2,12),ISARG(3,12),ISARG(4,12),
               1     ISARG(5,12),ISARG(6,12),ISARG(7,12),ISARG(8,12)/
               2  .TRUE.,3*.FALSE.,.TRUE.,3*.FALSE./
0023            DATA ISARG(1,13),ISARG(2,13),ISARG(3,13),ISARG(4,13),
               1     ISARG(5,13),ISARG(6,13),ISARG(7,13),ISARG(8,13)/
               2  .TRUE.,7*.FALSE./
0024            DATA ISARG(1,14),ISARG(2,14),ISARG(3,14),ISARG(4,14),
               1     ISARG(5,14),ISARG(6,14),ISARG(7,14),ISARG(8,14)/
               2  8*.TRUE./
0025            DATA ISARG(1,15),ISARG(2,15),ISARG(3,15),ISARG(4,15),
               1     ISARG(5,15),ISARG(6,15),ISARG(7,15),ISARG(8,15)/
               2  2*.TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,2*.FALSE./
0026            DATA ISARG(1,16),ISARG(2,16),ISARG(3,16),ISARG(4,16),
               1     ISARG(5,16),ISARG(6,16),ISARG(7,16),ISARG(8,16)/
               2  6*.TRUE.,.FALSE.,.TRUE./
0027            DATA ISARG(1,17),ISARG(2,17),ISARG(3,17),ISARG(4,17),
               1     ISARG(5,17),ISARG(6,17),ISARG(7,17),ISARG(8,17)/
               2  3*.TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,.FALSE./
0028            DATA ISARG(1,18),ISARG(2,18),ISARG(3,18),ISARG(4,18),
               1     ISARG(5,18),ISARG(6,18),ISARG(7,18),ISARG(8,18)/
               2  .TRUE.,.FALSE.,.TRUE.,3*.FALSE.,.TRUE.,.FALSE./
0029            DATA ISARG(1,19),ISARG(2,19),ISARG(3,19),ISARG(4,19),
               1     ISARG(5,19),ISARG(6,19),ISARG(7,19),ISARG(8,19)/
               2  .FALSE.,.FALSE.,.FALSE.,3*.FALSE.,.FALSE.,.FALSE./
0030            DATA ISARG(1,20),ISARG(2,20),ISARG(3,20),ISARG(4,20),
               1     ISARG(5,20),ISARG(6,20),ISARG(7,20),ISARG(8,20)/
               2  .TRUE.,.TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,2*.FALSE./
0031            DATA ISARG(1,21),ISARG(2,21),ISARG(3,21),ISARG(4,21),
               1     ISARG(5,21),ISARG(6,21),ISARG(7,21),ISARG(8,21)/
               2  .TRUE.,.TRUE.,.FALSE.,.FALSE.,.FALSE.,.FALSE.,2*.FALSE./
0032            DATA ISARG(1,22),ISARG(2,22),ISARG(3,22),ISARG(4,22),
               1     ISARG(5,22),ISARG(6,22),ISARG(7,22),ISARG(8,22)/
               2  .TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,.FALSE.,.TRUE.,.FALSE./
          C
          C
          C WHEN ECHNO IS NEG, SPECIAL SUPPORT IS REQUIRED
0033            ISPSH=0
```

```
0034          IF(FCBN0.LT.0)ISPSH=1
0035          FCBN2=IABS(FCBN0)
        C
        C LOAD FCB VALUES
0036          FCBG(1) = 0
0037          FCBG(2) = FCBN2
0038          FCBG(3) = ISPSH
0039          FCBG(4) = Y
0040          FCBG(5) = A
0041          FCBG(6) = D
0042          FCBG(7) = B
0043          FCBG(8) = V
0044          FCBG(9) = C
0045          FCBG(10)= W
0046          FCBG(11)= 0
        C
        C VALIDATE ARGUMENT IF USED, SET TO ZERO IF NOT.
0047          K=0
0048          DO 400 I=4,10,2
0049          J=I+1
0050           IF(ISARG(I-3,IS)) GO TO 100
0051          FCBG(I)=0
0052          GOTO 200
0053      100 K=K+1
0054          IF(FCBG(I).LT.BIDMN.OR.FCBG(I).GT.BIDMX) GO TO 1000
0055    200   IF(ISARG(J-3,IS)) GO TO 300
0056          FCBG(J)=0
0057          GOTO 400
0058      300 K=K+1
0059          IF(FCBG(J).LT.SIDMN.OR.FCBG(J).GT.SIDMX) GO TO 1000
0060    400   CONTINUE
        C
        C START MAP AND RETURN
        C
0061          FCBGN = RUNMP(FCBG(1),FCBS(1))
0062          RETURN
        C
        C MAP EXECUTION ABORTED:  SET ERROR CODE AND RETURN
0063     1000 FCBGN=-K
0064          IF(FCBGN.NE.0)CALL MPERR(FCBGN)
0065          RETURN
0066          END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000436	143	RW,I,CON,LCL
$IDATA	000014	6	RW,D,CON,LCL
$VARS	000610	196	RW,D,CON,LCL
MPZZZ	000312	101	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 001574   446

```
FORTRAN IV-PLUS V02-51E          21:13:55     11-SEP-80          PAGE 4
FCMGN.FTN        /DE/WR

NO FPP INSTRUCTIONS GENERATED

FCMGN,LP/LI:1=FCMGN/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:37:15     09-AUG-80           PAGE 1
FF2R.FTN          /WR

0001              INTEGER FUNCTION FF2R (Y, SA, U, V, W)
0002              INTEGER FCBGN, Y, SA, U, V, W
0003              FF2R = FCBGN(-214, Y, SA, U, 0, V, 0, W, 0, 14)
0004              RETURN
0005              END


PROGRAM SECTIONS

 NAME        SIZE                 ATTRIBUTES

$CODE1  000054     22             RW,I,CON,LCL
$PDATA  000014      6             RW,D,CON,LCL
$IDATA  000030     12             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000120     40

NO FPP INSTRUCTIONS GENERATED

FF2R,LP/LI:1=FF2R/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:35:39     09-AUG-80            PAGE 1
VMOV1.FTN           /DE/WR

0001                INTEGER FUNCTION VMOV1(Y)
0002                INTEGER FCBGN,Y
0003                VMOV1=FCBGN(237,Y,0,0,0,0,0,0,0,13)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME        SIZE                     ATTRIBUTES

$CODE1  000024     10                 RW,I,CON,LCL
$PDATA  000014      6                 RW,D,CON,LCL
$IDATA  000030     12                 RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070     28

NO FPP INSTRUCTIONS GENERATED

VMOV1,LP/LI:1=VMOV1/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E          13:35:43     09-AUG-80            PAGE 1
VMOV2.FTN         /DE/WR

                  INTEGER FUNCTION VMOV2(Y)
                  INTEGER FCRGN,Y
                  VMOV2=FCRGN(238,Y,0,0,0,0,0,0,0,13)
                  RETURN
                  END

PROGRAM SECTIONS

 NAME        SIZE                    ATTRIBUTES

SCODE1  000024     10                RW,I,CON,LCL
SPDATA  000014      6                RW,D,CON,LCL
SIDATA  000030     12                RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070     28

NO FPP INSTRUCTIONS GENERATED

VMOV2,LP/LI:1=VMOV2/DE/NOTR
```

```
VMOV3.FTN         /DE/WR

           INTEGER FUNCTION VMOV3(Y)
           INTEGER FCHGN,Y
           VMOV3=FCHGN(239,Y,0,0,0,0,0,0,0,13)
           RETURN
           END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000024	10	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000030	12	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070   28

NO FPP INSTRUCTIONS GENERATED

VMOV3,LP/LI:1=VMOV3/DE/NOTR

```
FORTRAN IV-PLUS V02-51E          13:35:51     09-AUG-80           PAGE 1
MPEDVM.FTN        /DE/WR

0001              INTEGER FUNCTION MPEDVM(Y,U,V)
          C
          C       MAP USAGE:
          C
          C       IERR=MPEDVM(Y,U,V)
          C
0002              INTEGER ECBGN,Y,U,V
0003              MPEDVM=ECBGN(240,Y,0,U,0,V,0,0,0,8)
0004              RETURN
0005              END

PROGRAM SECTIONS

 NAME        SIZE                  ATTRIBUTES

$CODE1  000040     16              RW,I,CON,LCL
$PDATA  000014      6              RW,D,CON,LCL
$IDATA  000030     12              RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000104     34

NO FPP INSTRUCTIONS GENERATED

MPEDVM,LP/LI:1=MPEDVM/DE/NOTR
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

8 OF 9

AD
A091663

```
FORTRAN IV-PLUS V02-51E          13:35:56    09-AUG-80           PAGE 1
MPDCTM.FTN       /DE/WR

0001             INTEGER FUNCTION MPDCTM(Y,U,V,W)
        C
        C        MAP USAGE:
        C
        C        IERR=MPDCTM(Y,U,V,W)
        C
0002             INTEGER FCBGN,Y,U,V,W
0003             MPDCTM=FCBGN(241,Y,0,U,0,V,0,W,0,22)
0004             RETURN
0005             END

PROGRAM SECTIONS

 NAME       SIZE                 ATTRIBUTES

SCODE1  000046    19             RW,I,CON,LCL
$PDATA  000014     6             RW,D,CON,LCL
SIDATA  000030    12             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000112    37

NO FPP INSTRUCTIONS GENERATED

MPDCTM,LP/LI:1=MPDCTM/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E        13:36:01    09-AUG-80         PAGE 1
MPQDPP.FTN        /DE/WR

0001            INTEGER FUNCTION MPQDPP(Y,U,V)
          C
          C     MAP USAGE:
          C
          C     IERR=MPQDPP(Y,U,V)
          C
0002            INTEGER FCHGN,U,Y,V,W
0003            MPQDPP=FCHGN(242,Y,0,U,0,V,0,0,0,8)
0004            RETURN
0005            END

PROGRAM SECTIONS

 NAME       SIZE                 ATTRIBUTES

$CODE1  000040    16             RW,I,CON,LCL
$PDATA  000014     6             RW,D,CON,LCL
$IDATA  000030    12             RW,D,CON,LCL
$VARS   000007     1             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000106    35

NO FPP INSTRUCTIONS GENERATED

MPQDPP,LP/LI:1=MPQDPP/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:36:05     09-AUG-80            PAGE 1
MPDOPP.FTN          /DE/WR

0001                INTEGER FUNCTION MPDOPP(Y,U,V)
            C
            C       MAP USAGE:
            C
            C       IERR=MPDOPP(Y,U,V)
            C
0002                INTEGER FCBGN,U,Y,V,W
0003                MPDOPP=FCBGN(243,Y,0,U,0,V,0,0,0,H)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

SCODE1  000040    16              RW,I,CON,LCL
SPDATA  000014     6              RW,D,CON,LCL
SIDATA  000030    12              RW,D,CON,LCL
SVARS   000002     1              RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000106    35

NO FPP INSTRUCTIONS GENERATED

MPDOPP,LP/LI:1=MPDOPP/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E         13:36:10    09-AUG-80          PAGE 1
MPMWGX.FTN      /DE/WR

0001                INTEGER FUNCTION MPMWGX(Y,U,V,W)
         C
         C          MAP USAGE:
         C
         C          IERR=MPMWGX(Y,U,V,W)
         C
0002                INTEGER FCBGN,Y,U,V,W
0003                MPMWGX=FCBGN(244,Y,0,U,0,V,0,W,0,22)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1  000046     19           RW,I,CON,LCL
$PDATA  000014      6           RW,D,CON,LCL
$IDATA  000030     12           RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000112    37

NO FPP INSTRUCTIONS GENERATED

MPMWGX,LP/LI:1=MPMWGX/DE/NOTR
```

```
0001              INTEGER FUNCTION MPESTV(U)
        C
        C         MAP USAGE:
        C
        C         IERR=MPESTV(U)
        C
0002              INTEGER FCBGN,U
0003              MPESTV=FCBGN(245,0,0,U,0,0,0,0,0,10)
0004              RETURN
0005              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000024	10	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000030	12	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070   28

NO FPP INSTRUCTIONS GENERATED

MPESTV,LP/LI:1=MPESTV/DE/NOTR

```
FORTRAN IV-PLUS V02-51E        13:36:18    09-AUG-80          PAGE 1
MPSSRT.FTN       /DE/WR

0001            INTEGER FUNCTION MPSSRT(Y)
        C
        C       MAP USAGE:
        C
        C       IERR=MPSSRT(Y)
        C
0002            INTEGER FCBGN,Y
0003            MPSSRT=FCBGN(247,Y,0,0,0,0,0,0,0,13)
0004            RETURN
0005            END

PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

SCODE1  000024     10             RW,I,CON,LCL
SPDATA  000014      6             RW,D,CON,LCL
SIDATA  000030     12             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070    28

NO FPP INSTRUCTIONS GENERATED

MPSSRT,LP/LI:1=MPSSRT/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E          13:36:23     09-AUG-80            PAGE 1
MPIDCM.FTN          /DE/WR

0001                INTEGER FUNCTION MPIDCM(Y,U,V)
              C
              C     MAP USAGE:
              C
              C     IERR=MPIDCM(Y,U,V)
              C
0002                INTEGER FCBGN,U,Y,V,W
0003                MPIDCM=FCBGN(248,Y,0,U,0,V,0,0,0,8)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME        SIZE                 ATTRIBUTES

$CODE1   000040    16             RW,I,CON,LCL
$PDATA   000014     6             RW,D,CON,LCL
$IDATA   000030    12             RW,D,CON,LCL
$VARS    000002     1             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000106    35

NO FPP INSTRUCTIONS GENERATED

MPIDCM,LP/LI:1=MPIDCM/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E         13:36:28    09-AUG-80           PAGE 1
MPASRT.FTN          /DE/WR

0001                INTEGER FUNCTION MPASRT(Y)
        C
        C           MAP USAGE:
        C
        C           IERR=MPASRT(Y)
        C
0002                INTEGER FCHGN,Y
0003                MPASRT=FCHGN(249,Y,0,0,0,0,0,0,0,13)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

$CODE1   000024     10            RW,I,CON,LCL
$PDATA   000014      6            RW,D,CON,LCL
$IDATA   000030     12            RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000070     28

NO FPP INSTRUCTIONS GENERATED

MPASRT,LP/LI:1=MPASRT/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:36:33    09-AUG-80          PAGE 1
MPFSTQ.FTN        /DE/WR

0001                INTEGER FUNCTION MPFSTQ(Y,U,V,W)
          C
          C         MAP USAGE:
          C
          C         IERR=MPFSTQ(Y,U,V,W)
          C
0002                INTEGER FCBGN,Y,U,V,W
0003                MPFSTQ=FCBGN(250,Y,0,U,0,V,0,W,0,22)
0004                RETURN
0005                END

PROGRAM SECTIONS

 NAME       SIZE                 ATTRIBUTES

$CODE1  000046    19             RW,I,CON,LCL
$PDATA  000014     6             RW,D,CON,LCL
$IDATA  000030    12             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000112    37

NO FPP INSTRUCTIONS GENERATED

MPFSTQ,LP/LI:1=MPFSTQ/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:36:37    09-AUG-80          PAGE 1
MPFSTD.FTN       /DE/WR

0001             INTEGER FUNCTION MPFSTD(Y,U,V,W)
        C
        C        MAP USAGE:
        C
        C        IERR=MPFSTD(Y,U,V,W)
        C
0002             INTEGER FCBGN,Y,U,V,W
0003             MPFSTD=FCBGN(251,Y,0,U,0,V,0,W,0,22)
0004             RETURN
0005             END

PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

$CODE1  000046    19              RW,I,CON,LCL
$PDATA  000014     6              RW,D,CON,LCL
$IDATA  000030    12              RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000112    37

NO FPP INSTRUCTIONS GENERATED

MPFSTD,LP/LI:1=MPFSTD/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:36:42    09-AUG-80           PAGE 1
MPVDNM.FTN          /DE/WR

0001                INTEGER FUNCTION MPVDNM(Y,U,V)
          C
          C         MAP USAGE:
          C
          C         IERR=MPVDNM(Y,U,V)
          C
0002                INTEGER FCBGN,U,Y,V
0003                MPVDNM=FCBGN(252,Y,0,U,0,V,0,0,0,8)
0004                RETURN
0005                END


PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

$CODE1  000040    16              RW,I,CON,LCL
$PDATA  000014     6              RW,D,CON,LCL
$IDATA  000030    12              RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000104    34

NO FPP INSTRUCTIONS GENERATED

MPVDNM,LP/LI:1=MPVDNM/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51E          13:36:47     09-AUG-80          PAGE 1
MPRASP.FTN       /DE/WR

0001                 INTEGER FUNCTION MPRASP(Y,U,V,W)
           C
           C         MAP USAGE:
           C
           C         IERR=MPRASP(Y,U,V)
           C
0002                 INTEGER FCBGN,Y,U,V,W
0003                 MPRASP=FCBGN(253,Y,0,U,0,V,0,W,0,22)
0004                 RETURN
0005                 END

PROGRAM SECTIONS

 NAME       SIZE                    ATTRIBUTES

SCODE1  000046     19               RW,I,CON,LCL
SPDATA  000014      6               RW,D,CON,LCL
SIDATA  000030     12               RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000112     37

NO FPP INSTRUCTIONS GENERATED

MPRASP,LP/LI:1=MPRASP/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:36:52     09-AUG-80            PAGE 1
MPSCAN.FTN        /DE/WR

0001                INTEGER FUNCTION MPSCAN(Y)
            C
            C       MAP USAGE:
            C
            C       IERR=MPSCAN(Y)
            C
0002                INTEGER FCBGN,U,Y,V,W
0003                MPSCAN=FCBGN(254,Y,0,0,0,0,0,0,0,13)
0004                RETURN
0005                END


PROGRAM SECTIONS

 NAME        SIZE                  ATTRIBUTES

$CODE1  000024    10               RW,I,CON,LCL
$PDATA  000014     6               RW,D,CON,LCL
$IDATA  000030    12               RW,D,CON,LCL
$VARS   000006     3               RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000076    31

NO FPP INSTRUCTIONS GENERATED

MPSCAN,LP/LI:1=MPSCAN/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F         13:36:56    09-AUG-80          PAGE 1
MPCDBA.FTN         /DE/WR

0001            INTEGER FUNCTION MPCDBA(Y)
        C
        C       MAP USAGE:
        C
        C       IERR=MPCDBA(Y)
        C
0002            INTEGER FCBGN,Y,U,V
0003            MPCDBA=FCBGN(255,Y,0,0,0,0,0,0,0,13)
0004            RETURN
0005            END


PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1  000024     10           RW,I,CON,LCL
$PDATA  000014      6           RW,D,CON,LCL
$IDATA  000030     12           RW,D,CON,LCL
$VARS   000004      2           RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000074     30

NO FPP INSTRUCTIONS GENERATED

MPCDBA,LP/LI:1=MPCDBA/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          21:16:56     11-SEP-80            PAGE 1
MASTER.ATC       /DE/WR

         C        PROGRAM NAME:MASTER.ATC          ORIGINATED:05-FEB-79
         C                                                UPDATE:23-MAY-80
         C        THIS IS THE CONTROLLER FOR THE DCA ATC MAP-300 SYSTEM
         C
         C
         C
         C
0001              INTEGER BUS1,BUS2,BUS3
0002              INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                  1,BYTE4,BYTER,CNVYES,CNVNO,UNITY,STATUS,SHORT
0003              INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,F1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,OTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,DUIB
                  9,RECV,AROPB,ARQDCT,RIBIT
                  1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
         C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0004              INTEGER ORDR,START,SIZE,DSW,BTLTH
0005              INTEGER OTDCT,OTPAR,OTDC,OTVAR,OTM,OTG,OTV
0006              LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007              COMMON/SECTIO/MODDL
0008              COMMON/MTAPE0/NIN(300),NOUT(300)
0009              COMMON/MTAPE1/NSKIP,IST,NTOTI,NIUPS,NTOTO
0010              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011              COMMON/MTAPE3/NBF(1324),NBUF(1324)
0012              COMMON/MTAPE4/LST,IBEG
0013              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                  1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0014              COMMON/MTAPE6/APPEND
0015              COMMON/DELAY/M4,M3,M2,M1,IV4,IV3,IV2,IV1
0016              COMMON/CHANEL/IAQPB(14),IAQDCT(256)
0017              COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0018              COMMON/OVRO/Y(256),NP,XN
0019              COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0020              COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0021              COMMON/OVRP/K,G,VO,PWEI1(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0022              COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0023              COMMON/OVR2/IORDR(256),LTHM1
0024              COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0025              COMMON/OVRF/XR(512)
0026              COMMON/OVRO/CSN,START,SIZE,YES,NO
0027              COMMON/OVROT/OTDCT(256),OTPAR(8),OTDC,OTVAR,OTM,OTG,OTV
0028              COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0029              COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,OTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
```

```
                 8,SYNB1,SYNB2,BF1,BF2,DBIB
                 9,RECV,ARQPB,ARQPCT,FIBIT
                 1,DCTIT1,DCTIT2,TORDR1,TORDR2,FIBIT1,BIBIT2,MIBIT3,MIBIT4,OTDCT1
0030             DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0031             DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0032             DATA IOSPF,IEEOF,IOEOF/02440,-10,030000/
0033             DATA MASK/03777/
0034             DATA MT0/0,2048,0,0,0,0/
0035             DATA MT1/0,2048,0,0,0,0/
          C      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          C
          C      <<<<<<<<<<<<<<<<<<<<<<<<<<
          C      >>>>> INITIALIZATION <<<<<
          C      >>>>>>>>>>>>>>>>>>>>>>>>>>
          C
0036             CALL INIT
0037             CALL USER
0038             CALL MAPON
0039             CALL CREATE
0040             CALL DEFINE
0041             CALL SETUP
0042             IF(LIVE.EQ.YES)GO TO 4000
          C
          C      <<<<<<<<<<<<<<<<<<<<<<<
          C      >>>>> TRANSMITTER <<<<<
          C      >>>>>>>>>>>>>>>>>>>>>>>
          C
0043      1001   CALL INPUT
0044             MODUL=1
          C      SELECT APPROP. CDBA
0045             IF(NEND.NE.0)GO TO 5000
0046             IF(RETIM.EQ.YES)GO TO 4000
0047             CALL DCVAR
0048             CALL DCTF
0049             CALL PITLPC
0050             CALL QDPARM
0051             CALL VPBS
0052             CALL BASIS
0053             CALL ASRT
0054             CALL SDCT
0055             CALL BITA
0056             CALL QDCT
          C
          C      <<<<<<<<<<<<<<<<<<<<<<<
          C      >>>>>>> CHANNEL <<<<<<<
          C      >>>>>>>>>>>>>>>>>>>>>>>
          C
0057             CALL CHANL
          C
          C      <<<<<<<<<<<<<<<<<<<<<<<<
          C      >>>>>>> RECEIVER <<<<<<<
          C      >>>>>>>>>>>>>>>>>>>>>>>>
          C
          C      SELECT APPROP. CDBA
0058             MODUL=2
0059             CALL DPARM
```

```
0060                CALL VPBS
0061                CALL BASIS
0062                CALL SSRT
0063                CALL SDCT
0064                CALL RITA
0065                CALL DDCT
0066                CALL DCTR
0067                CALL VARDC
0068        4000    CALL SYSTEM
0069        5000    CALL OUTPUT
            C
            C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<
            C       >>>>> MONITOR RESULTS <<<<<
            C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>
            C
0070                CALL SNR
0071                LIVE=YES
0072                GO TO 4000
0073                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000436	143	RW,I,CON,LCL
SIDATA	000002	1	RW,D,CON,LCL
SVARS	000044	18	RW,D,CON,LCL
SECTIO	000002	1	RW,D,OVR,GBL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000004	2	RW,D,OVR,GBL
DELAY	000020	8	RW,D,OVR,GBL
CHANEL	001034	270	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
OVRQT	001032	269	RW,D,OVR,GBL
OVRDT	002124	554	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 044316   9319

NO FPP INSTRUCTIONS GENERATED

MASTER,LP/LI:1=MASTER.ATC/DE/NOTR

```
        C       PROGRAM NAME:INIT.FTN   ORIGINATED:05-FEB-79
        C                                       UPDATE:29-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE INIT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA YES,NO/'Y','N'/
        C
        C       DETERMINE CONSTANTS
        C
        C       DCT BIT ALLOCATION,BTLTH
0022            BTLTH=268
        C       FFT SIZES,M...
```

```
0023            NSHRT=8
0024            NLONG=9
          C     VERY ACCURATE VALUE FOR PI=3.14159...
0025            PI=4.0*ATAN(1.)
          C     LOGARITHM BASE 10 OF 2
0026            DLOG2=ALOG(2.0)
          C     TRANSMIT,RECEIVE SIZE,LSEN
0027            LSEN=369
          C     # OF SIDEBAND BITS/FRAME,LPARM
0028            LPARM=13
          C     FRAME SIZE,LTH
0029            LTH=256
0030            LTH2=2*LTH
0031            LTH3=3*LTH
0032            LTH4=4*LTH
0033            LTHM1=LTH-1
          C     LPC ORDER,LPCN
0034            LPCN=8
          C     OUTPUT INTEROLATING SIZE,NP
0035            NP=10
0036            XN=NP
          C     CLEAR CUMULATIVE SNR RATIO,CNS
0037            CNS=0.
          C     FRAME COUNTER,ICOUNT
0038            ICOUNT=0
          C     ARGUMENT FOR TRIG FUNCTIONS
0039            ARG=PI*(2.*LTH-1.)/(2.*LTH)
          C
          C     SET UP SPEECH I/O HANDLER
          C
0040            IST=1
0041            NSKIP=1
0042            NSKIPS=NSKIP
          C     INPUT RECORD SIZE,NTOTI
0043            NTOTI=LTH-NP
          C     INPUT UPDATE SIZE,NTUPS
0044            NTUPS=LTH-NP
          C     OUTPUT RECORD SIZE,NTOTO
0045            NTOTO=LTH-NP
          C     RESET ALL ARRAYS
0046            DO 1 I=1,NTOTI
0047      1     NIN(I)=0
0048            DO 2 I=1,NTOTO
0049      2     NOUT(I)=0
0050            RETURN
0051            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000366	123	RW,I,CON,LCL
$VARS	000060	24	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL

```
$TAPE2   000012     5          RW,D,OVR,GBL
OVR1     002020   520          RW,D,OVR,GBL
OVRD     002006   515          RW,D,OVR,GBL
OVRF     006016  1543          RW,D,OVR,GBL
OVRA     000152    53          RW,D,OVR,GBL
OVRP     002022   521          RW,D,OVR,GBL
OVRB     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRD     000012     5          RW,D,OVR,GBL
$APDFF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025450  5524

INIT,LP/LI:1=INIT/DE/NOTR
```

```
        C       PROGRAM NAME:USER.FTN    ORIGINATED:05-FEB-79
        C                                         UPDATE:12-JUN-80
        C
        C
        C
        C
0001            SUBROUTINE USER
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTER,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,ORDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPR,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPR,ARQDCT,RIBIT
                1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0011            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOFOF,IORLB,MT0(6),MT1(6),DSW
0012            COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0013            COMMON/OVRO/Y(256),NP,XN
0014            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0015            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0016            COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0017            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0018            COMMON/OVR2/IORDR(256),LTHB1
0019            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0020            COMMON/OVRE/XR(512)
0021            COMMON/OVRD/CSN,START,SIZE,YES,NO
0022            COMMON/MAPDEF/WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,ORDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPR,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPR,ARQDCT,RIBIT
                1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        C
        C
        C       USER INTERACTION
        C
```

```
      C       CALL ATQIO(IOATT,5,1,0,ISW,0,DSW)
              TYPE 106
      106     FORMAT(1H1,10X,'>>>>> GTE 9600 BPS ATC SYSTEM <<<<<'/)
      C       TYPE 103
      C103    FORMAT(1HS,'FRAME TIMING(Y/N)? ')
      C       READ(5,104)TIMING
      104     FORMAT(A1)
              RLTIM=YES
              LIVE=NO
              START=1
              IREC=1
      C       IF(TIMING.EQ.YES)GO TO 2
              TIMING=NO
              TYPE 107
      C107    FORMAT(1HS,'LIVE INPUT(Y/N)? ')
      107     FORMAT(1HS,'START-UP(Y/N)? ')
              ACCEPT 104,LIVE
              ITEMP=NO
              IF(LIVE.EQ.NO)ITEMP=YES
              LIVE=ITEMP
              IF(LIVE.EQ.YES)GO TO 1
      C       TYPE 102
      C102    FORMAT(1HS,'REAL TIME OPERATION(Y/N)? ')
      C       ACCEPT 104,RLTIM
              RLTIM=NO
      C2      TYPE 105
      C105    FORMAT(1HS,'STARTING FRAME # =')
      C       ACCEPT 101,START
              START=1
              IREC=1
      C       IF(TIMING.EQ.YES)GO TO 1
      C       TYPE 100
      C100    FORMAT(1HS,'NO. OF FRAMES= ')
      C       ACCEPT 101,IREC
      101     FORMAT(I6)
      C1      TYPE 110
      1       CONTINUE
      C110    FORMAT(//1X,'SYSTEM INITIALIZATION:')
              RETURN
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000220	72	RW,I,CON,LCL
SIDATA	000104	34	RW,D,CON,LCL
SVARS	000046	19	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL

```
OVRF     006016  1543     RW,D,OVR,GBL
OVRA     000152    53     RW,D,OVR,GBL
OVRP     002022   521     RW,D,OVR,GBL
OVRR     005001  1282     RW,D,OVR,GBL
OVR2     001002   257     RW,D,OVR,GBL
OVRL     000016     7     RW,D,OVR,GBL
OVRD     000012     5     RW,D,OVR,GBL
*APDF*   000200    64     RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 037740  8176

NO FPP INSTRUCTIONS GENERATED

USER,LP/LI:1=USER/DE/NOTR
```

```
          C         PROGRAM NAME:MAPON.FTN  ORIGINATED:05-FEB-79
          C                                             UPDATE:14-JUL-80
          C
          C
          C
          C
0001                SUBROUTINE MAPON
0002                INTEGER ZERO(738),PARITY,RCVOFF
0003                DIMENSION SCALAR(127)
0004                INTEGER BUS1,BUS2,BUS3
0005                INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                    1,BYTE4,BYTE8,CNVYES,CNVNO,ORITY,STATUS,SHORT
0006                INTEGER WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                    1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                    2,ORDCTM,YNM,ORPRM,ORDCTM
                    3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                    4,MIBIT,DCTM,OPRM,QTDCTM
                    5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                    6,RECV1,RECV2,SEN1,SEN2
                    7,AOPR,AOTDCT,AIBIT,SEN,INPB
                    8,SYNB1,SYNB2,BE1,BE2,OUTB
                    9,RECV,AROPR,AROOCT,RIBIT
                    1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
                    2,ZRO
          C         <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0007                INTEGER ORDR,START,SIZE,DSW,BTLTH
0008                LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0009                COMMON/MTAPE0/NIN(300),NOUT(300)
0010                COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTO
0011                COMMON/MTAPE2/NEND,NERR,NEIGE,NINS,NOUTS
0012                COMMON/OVR1/LTH,PCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0013                COMMON/OVRO/Y(256),NP,XN
0014                COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0015                COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0016                COMMON/OVRP/4,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0017                COMMON/OVR6/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0018                COMMON/OVR2/IORDR(256),LTHM1
0019                COMMON/OVRL/SLOG,LNGTH,CIEEP,ITOT,BTLTH
0020                COMMON/OVRE/XR(512)
0021                COMMON/OVRD/CSN,START,SIZE,YES,NO
0022                COMMON/MAPDEF/WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                    1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                    2,ORDCTM,YNM,ORPRM,ORDCTM
                    3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                    4,MIBIT,DCTM,OPRM,QTDCTM
                    5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                    6,RECV1,RECV2,SEN1,SEN2
                    7,AOPR,AOTDCT,AIBIT,SEN,INPB
                    8,SYNB1,SYNB2,BE1,BE2,OUTB
                    9,RECV,AROPR,AROOCT,RIBIT
                    1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
0023                EQUIVALENCE (SCALAR(1),Y(1)),(ZERO(1),XR(1))
0024                DATA ZERO/738*0/
          C         >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0025                DATA REAL,CMPLX,INTGR/0,1,2/
0026                DATA EVERY,SINGLE,DOUBLE/1,2,3/
```

```
      DATA BYTE2,BYTE4,BYTE8/2,4,8/
      DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
      DATA BUS1,BUS2,BUS3/64,128,192/
      DATA WORK,X1,X1R,X11,IORDRM,PWFEM,DCTM1/10,11,0,0,12,13,14/
      DATA DCTM2,R1,A1,KF,PARC,TMP1,TMP2/15,16,17,18,3,19,20/
      DATA DRDCTM,YNM/21,0/
      DATA VLONG,COSZ,VSHRT,YOM,XOM,MAPX/23,24,25,26,27,28/
      DATA MIBIT,DCTM,OPRM,OTDCTM/29,30,31,32/
      DATA NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2/33,34,35,36,37,38/
      DATA RECV1,RECV2,SEN1,SEN2/39,40,41,42/
      DATA AOP8,AOTDCT,AIBIT/33,44,45/
      DATA TMP3,TMP4,X2/46,47,48/
      DATA SEN,INPB/49,50/
      DATA SYNB1,BF1/62,63/
      DATA RECV,AROPR,ARODCT,RIBIT/56,57,58,2/
      DATA OUTB,ZRO,UNITY/59,1,63/
      DATA DCTIT1,DCTIT2,IORDR1,IORDR2,MIBIT1/51,52,53,54,55/
      DATA MIBIT2,MIBIT3,MIBIT4,OTDCT1/60,5,6,7/
      DATA PARITY,INIT,RCVOFF/114,113,127/
      DATA ORPRM,ORDCTM/8,9/
```

```
C
C                            K A P - 3 0 0
C                       B U F F E R    A R E A S
C---------------------------------------------------------------------
C BID      BID #   BUS        BA     BA+NS-1        NS    ST    SI    WS
C---------------------------------------------------------------------
C AOPB       43     1     35328.     35331.        14     I     E     L
C---------------------------------------------------------------------
C AQIOCT     44     1     35342.     35597.       256     I     E     L
C---------------------------------------------------------------------
C ATBIT      45     1     35598.     35853.       256     I     E     L
C---------------------------------------------------------------------
C SEG        49     1     35854.     36223.       370     I     E     L
C---------------------------------------------------------------------
C INPB       50     1     36224.     36479.       256     I     E     L
C---------------------------------------------------------------------
C BF1        63     1     36480.     36848.       369     I     E     L
C---------------------------------------------------------------------
C BF2               1     36849.     37217.       369     I     E     L
C---------------------------------------------------------------------
C BF3               1     37218.     37586.       369     I     E     L
C---------------------------------------------------------------------
C ZRO         1     1     37587.     37955.       369     I     E     L
C---------------------------------------------------------------------
C OUTB       59     1     37956.     38211.       256     I     E     L
C---------------------------------------------------------------------
C RECV       56     1     38212.     38581.       370     I     E     L
C---------------------------------------------------------------------
C AROEB      57     1     38582.     38595.        14     I     E     L
C---------------------------------------------------------------------
C ARQLCT     58     1     38596.     38851.       256     I     E     L
C---------------------------------------------------------------------
C SYNB1      62     1     38852.     39220.       369     I     E     L
C---------------------------------------------------------------------
C SYNB2             1     39221.     39589.       369     I     E     L
C---------------------------------------------------------------------
C RIBIT       2     1     39590.     36245.       256     I     E     L
C---------------------------------------------------------------------
C BIDBF1     63     1     10000.     14499.      4500     I     E     L
C---------------------------------------------------------------------
C BIDBF2            1      8800.      9999.      1200     I     E     L
C---------------------------------------------------------------------
```

```
C---------------------------------------------------------------------------------
C COSZ       24      2      1024.0      1535.0      256      R      E      L
C---------------------------------------------------------------------------------
C YOM        26      2      2048.0      2067.0       10      R      E      L
C---------------------------------------------------------------------------------
C XOM        27      2      2068.0      2087.0       10      R      E      L
C---------------------------------------------------------------------------------
C MAPX#      28      2      2068.0      2579.0      256      R      E      L
C---------------------------------------------------------------------------------
C MIBIT      29      2      2088.0      2343.0      256      I      E      L
C---------------------------------------------------------------------------------
C DCIM       30      2      2344.0      2855.0      256      R      E      L
C---------------------------------------------------------------------------------
C                    2      2856.0      2867.0       12      I      E      L
C---------------------------------------------------------------------------------
C OTDCIM     32      2      2868.0      3123.0      256      I      E      L
C---------------------------------------------------------------------------------
C NIVM       33      2      3124.0      3369.0      246      I      E      L
C---------------------------------------------------------------------------------
C NIQ1       34      2      3370.0      3615.0      246      I      E      L
C---------------------------------------------------------------------------------
C NII2       35      2      3616.0      3861.0      246      I      E      L
C---------------------------------------------------------------------------------
C NOUTM      36      2      3862.0      4107.0      246      I      E      L
C---------------------------------------------------------------------------------
C NOUT1      37      2      4108.0      4353.0      246      I      E      L
C---------------------------------------------------------------------------------
C NOUT2      38      2      4354.0      4599.0      246      I      E      L
C---------------------------------------------------------------------------------
C RECV1      39      2      4600.0      4968.0      369      I      E      L
C---------------------------------------------------------------------------------
C RECV2      40      2      4969.0      5337.0      369      I      E      L
C---------------------------------------------------------------------------------
C SEN1       41      2      5340.0      5709.0      370      I      E      L
C---------------------------------------------------------------------------------
C SEN2       42      2      5710.0      6079.0      370      I      E      L
C---------------------------------------------------------------------------------
C DCTIT1     51      2      9000.0      9511.0      256      R      E      L
C---------------------------------------------------------------------------------
C DCTIT2    52     2        9512.0     10023.0      256      R      E      L
C---------------------------------------------------------------------------------
C IORDR1     53      2     10024.0     10279.0      256      I      E      L
C---------------------------------------------------------------------------------
C IORDR2     54      2     10280.0     10535.0      256      I      E      L
C---------------------------------------------------------------------------------
C MIBIT1     55      2     10536.0     10791.0      256      I      E      L
C---------------------------------------------------------------------------------
C MIBIT2     60      2     10792.0     11047.0      256      I      E      L
C---------------------------------------------------------------------------------
C OTDCT1      7      2     11048.0     11303.0      256      I      E      L
C---------------------------------------------------------------------------------
C MIBIT3      5      2     11304.0     11559.0      256      I      E      L
C---------------------------------------------------------------------------------
C MIBIT4      6      2     11560.0     11815.0      256      I      E      L
C---------------------------------------------------------------------------------
```

```
C QPRK          31     2     11816.0     11829.0       14     I     E     L
C---------------------------------------------------------------------------
C QRPK4          8     2     11830.0     11843.0       14     I     E     L
C---------------------------------------------------------------------------
C QROCIM         9     2     11844.0     12099.0      256     I     E     L
C---------------------------------------------------------------------------
```

```
C-------------------------------------------------------------------------
C *ORF       10    3      0.0    2047.0    512    C    E    L
C-------------------------------------------------------------------------
C X1         11    3   2048.0    4095.0    512    C    E    L
C-------------------------------------------------------------------------
C X1P*             3                       512    R    E    L
C-------------------------------------------------------------------------
C X1I*             3                       512    R    E    L
C-------------------------------------------------------------------------
C X2         48    3   2048.0    3071.0    256    C    E    L
C-------------------------------------------------------------------------
C IORDR9*    12    3      0.0     255.0    256    I    E    L
C-------------------------------------------------------------------------
C P*FITT9*   13    3      0.0     255.0    256    I    E    L
C-------------------------------------------------------------------------
C DCTM1*     14    3    512.0    1023.0    256    R    E    L
C-------------------------------------------------------------------------
C DCTM2*     15    3   1024.0    1535.0    256    R    E    L
C-------------------------------------------------------------------------
C R1*        16    3      0.0      17.0      9    R    E    L
C-------------------------------------------------------------------------
C A1*        17    3     18.0      33.0      8    R    E    L
C-------------------------------------------------------------------------
C KF*        18    3     35.0      66.0     16    R    E    L
C-------------------------------------------------------------------------
C PARC*       3    3                          8    R    E    L
C-------------------------------------------------------------------------
C TMP1*      19    3   2048.0    2559.0    256    R    E    L
C-------------------------------------------------------------------------
C TMP2*      20    3   2560.0    3071.0    256    R    E    L
C-------------------------------------------------------------------------
C TMP3*      46    3   3584.0    4095.0    256    R    E    L
C-------------------------------------------------------------------------
C TMP4*      47    3   3072.0    3583.0    256    R    E    L
C-------------------------------------------------------------------------
C ORDCTM*    21    3   1024.0    1535.0    256    R    E    L
C-------------------------------------------------------------------------
C XNM*              3    512.0    1023.0    256    R    E    L
C-------------------------------------------------------------------------
C VLONG      23    2   4096.0    5119.0    512    R    E    L
C-------------------------------------------------------------------------
C VSHRT      25    2   5120.0    5631.0    256    R    E    L
C-------------------------------------------------------------------------
```

```
          C
          C         INITIAL MAP-300 SETUP
          C
0047      D         INSTR=0
0048                STATUS=MPOPN(3)
0049      D         IF(STATUS.NE.0)GO TO 10
0050      D         INSTR=-1
0051                STATUS=MPIBC(0)
0052      D         IF(STATUS.NE.0)GO TO 10
          C
          C         DEFINE PROGRAM CONSTANTS
          C
          C         LARGEST FFT SIZE,LMAX
          C         SMALLEST FFT SIZE,LMIN
0053                LMAX=2**MLONG
0054                LMIN=2**MSHRT
0055                FLMAX=FLOAT(LMAX)
0056                FLMIN=FLOAT(LMIN)
          C         FRAME PROCESSING SIZE,XLTH
0057                XLTH=FLOAT(LTH)
          C         TRANSMIT/RECEIVE BUFFER SIZE,LSEN
0058                XLSEN=FLOAT(LSEN)
          C         # OF SIDEBAND BITS/FRAME,LPARM
0059                XLPARM=FLOAT(LPARM)
0060                DARG=1.0/FLOAT(LTH4)
0061                XLOG16=ALOG10(16.0)
          C
          C         MAP SCALAR TABLE SETUP
          C
0062                DO 1 I=50,127
0063      1         SCALAR(I)=0.0
0064                SCALAR(50)=1.0/FLMAX
0065                SCALAR(51)=1.0/FLMIN
0066                SCALAR(54)=1.0/FLOAT(LTH)
0067                SCALAR(56)=DARG
0068                SCALAR(57)=-DARG
0069                SCALAR(74)=FLOAT(LPCN+1)
0070                SCALAR(75)=FLOAT(LPCN)
0071                SCALAR(76)=4.0
0072                SCALAR(77)=10.0E-10
0073                SCALAR(79)=FLOAT(BTLTH)
          C9100     TYPE 9101
          C9101     FORMAT(1HS,'BIT MAX.= ')
          C         READ(5,9002,END=9100,ERR=9100)BITMAX
0074                BITMAX=3.
0075                SCALAR(84)=BITMAX
          C9000     TYPE 9001
          C9001     FORMAT(1HS,'LPC THRESHOLD= ')
          C         READ(5,9002,END=9000,ERR=9000)THRSH
0076                THRSH=.995
0077      9002      FORMAT(F15.8)
0078                SCALAR(86)=THRSH
0079                SCALAR(87)=FLOAT(LTH)
0080                SCALAR(88)=0.62293E-01
0081                SCALAR(89)=0.44600E+01
```

```
0082            SCALAR(92)=FLOAT(LTH2)
0083            SCALAR(94)=20.0
0084            SCALAR(95)=XLOG16
0085            SCALAR(97)=FLOAT(LTH-1)
0086            SCALAR(98)=0.05
0087            SCALAR(100)=0.62293E-01
0088            SCALAR(101)=0.44600E+01
0089            SCALAR(102)=0.62293E-01
0090            SCALAR(103)=0.44600E+01
0091            SCALAR(PARITY)=0.0
0092      D     INSTR=40
0093            STATUS=MPWST(50,SCALAR(50),77,(NVYES)
0094      D     IF(STATUS.NE.0)GO TO 10
0095      D     INSTR=41
0096            STATUS=MPIST(INIT,0)
0097      D     IF(STATUS.NE.0)GO TO 10
0098      D     INSTR=42
0099            STATUS=MPIST(RCVOFF,-1)
0100      D     IF(STATUS.NE.0)GO TO 10
          C
          C
          C     MAP BUFFER SETUP
          C
          C>>>>>>>DEFINE FFT AREAS
0101      D     INSTR=1
0102            STATUS=MPCLB(BUS3+X1,2048.0,FLMAX,CMPLX,EVERY,LONG)
0103      D     IF(STATUS.NE.0)GO TO 10
0104      D     INSTR=4
0105            STATUS=MPCLB(BUS3+VLONG,4096.0,FLMAX,REAL,EVERY,LONG)
0106      D     IF(STATUS.NE.0)GO TO 10
0107      D     INSTR=5
0108            STATUS=MPCLB(BUS3+WORK,0.0,FLMAX,CMPLX,EVERY,LONG)
0109      D     IF(STATUS.NE.0)GO TO 10
0110      D     INSTR=6
0111            STATUS=MPDCV(UNITY,1,LMAX,REAL)
0112      D     IF(STATUS.NE.0)GO TO 10
0113      D     INSTR=7
0114            STATUS=VCOS(VLONG,0,UNITY,50,UNITY,0)
0115      D     IF(STATUS.NE.0)GO TO 10
0116      D     INSTR=8
0117            STATUS=MPDCV(UNITY,1,LTH,REAL)
0118      D     IF(STATUS.NE.0)GO TO 10
0119      D     INSTR=9
0120            STATUS=MPCLB(BUS2+COSZ,1024.0,XLTH,REAL,EVERY,LONG)
0121      D     IF(STATUS.NE.0)GO TO 10
0122      D     INSTR=10
0123            STATUS=VCOS(COSZ,0,UNITY,56,UNITY,0)
0124      D     IF(STATUS.NE.0)GO TO 10
0125      D     INSTR=11
0126            STATUS=MPCLB(BUS3+X2,2048.0,FLMIN,CMPLX,EVERY,LONG)
0127      D     IF(STATUS.NE.0)GO TO 10
0128      D     INSTR=12
0129            STATUS=MPCLB(BUS3+VSHRT,5120.0,FLMIN,REAL,EVERY,LONG)
0130      D     IF(STATUS.NE.0)GO TO 10
0131      D     INSTR=13
0132            STATUS=MPDCV(UNITY,1,LMIN,REAL)
```

```
0133    D       IF(STATUS.NE.0)GO TO 10
0134    D       INSTR=14
0135            STATUS=VCOS(VSHRT,0,UNITY,54,UNITY,0)
0136    D       IF(STATUS.NE.0)GO TO 10
        C>>>>>>>DEFINE BUFFERING AREAS
0137    D       INSTR=20
0138            STATUS=MPCLB(BUS2+MAPX,2068.0,XLTH,REAL,EVERY,LONG)
0139    D       IF(STATUS.NE.0)GO TO 10
0140    D       INSTR=21
0141            STATUS=MPCLB(BUS2+XOM,2068.0,FLOAT(NP),REAL,EVERY,LONG)
0142    D       IF(STATUS.NE.0)GO TO 10
0143    D       INSTR=23
0144            STATUS=MPCLB(BUS2+YOM,2048.0,FLOAT(NP),REAL,EVERY,LONG)
0145    D       IF(STATUS.NE.0)GO TO 10
        C>>>>>>>DEFINE LPC AREAS
0146    D       INSTR=30
0147            STATUS=MPCLB(BUS3+R1,0.0,FLOAT(LPCN+1),REAL,EVERY,LONG)
0148    D       IF(STATUS.NE.0)GO TO 10
0149    D       INSTR=31
0150            STATUS=MPCLB(BUS3+A1,18.0,FLOAT(LPCN),REAL,EVERY,LONG)
0151    D       IF(STATUS.NE.0)GO TO 10
0152    D       INSTR=32
0153            STATUS=MPCLB(BUS3+KE,35.0,FLOAT(2*LPCN),REAL,EVERY,LONG)
0154    D       IF(STATUS.NE.0)GO TO 10
0155    D       INSTR=33
0156            STATUS=MPFOR(PARC,KE)
0157    D       IF(STATUS.NE.0)GO TO 10
0158    D       INSTR=31
0159            STATUS=MPSBS(PARC,-LPCN)
0160    D       IF(STATUS.NE.0)GO TO 10
        C>>>>>>>DEFINE BASIS SPECTRUM AREAS
0161    D       INSTR=40
0162            STATUS=MPCLB(BUS3+DCTM1,512.0,XLTH,REAL,EVERY,LONG)
0163    D       IF(STATUS.NE.0)GO TO 10
0164    D       INSTR=41
0165            STATUS=MPCLB(BUS3+DCTM2,1024.0,XLTH,REAL,EVERY,LONG)
0166    D       IF(STATUS.NE.0)GO TO 10
0167    D       INSTR=42
0168            STATUS=MPCLB(BUS2+MIBIT,2088.0,XLTH,INTGR,EVERY,LONG)
0169    D       IF(STATUS.NE.0)GO TO 10
0170    D       INSTR=43
0171            STATUS=MPCLB(BUS3+IORDRM,0.0,XLTH,INTGR,EVERY,LONG)
0172    D       IF(STATUS.NE.0)GO TO 10
0173    D       INSTR=44
0174            STATUS=MPCLB(BUS3+PWFITM,0.0,XLTH,REAL,EVERY,LONG)
0175    D       IF(STATUS.NE.0)GO TO 10
0176    D       INSTR=45
0177            STATUS=MPCLB(BUS2+DCTM,2344.0,XLTH,REAL,EVERY,LONG)
0178    D       IF(STATUS.NE.0)GO TO 10
        C>>>>>>>DEFINE QUANTIZED PARAMETER AREAS
0179    D       INSTR=50
0180            STATUS=MPCLB(BUS2+QTDCTM,2868.0,XLTH,INTGR,EVERY,LONG)
0181    D       IF(STATUS.NE.0)GO TO 10
0182    D       INSTR=51
0183            STATUS=MPCLB(BUS2+QPPM,11616.0,XLPARM+1.,INTGR,EVERY,LONG)
0184    D       IF(STATUS.NE.0)GO TO 10
```

```
      D          INSTR=52
                 STATUS=MPCLB(BUS2+ORPEM,11830.0,XLPARM+1.,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=53
                 STATUS=MPCLB(BUS2+ORDCTM,11844.0,XLTH,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      C>>>>>>>>DEFINE DEQUANTIZED PARAMETER AREAS
      D          INSTR=60
                 STATUS=MPCLB(BUS3+DRDCTM,1024.0,XLTH,REAL,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      C>>>>>>>>DEFINE I/O AREAS
      D          INSTR=70
                 STATUS=MPCLB(BUS2+NINM,3124.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=71
                 STATUS=MPCLB(BUS2+NOUTM,3862.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=1022
                 STATUS=MPCLB(BUS2+RECV1,4600.0,XLSEN,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1024
                 STATUS=MPCLB(BUS2+RECV2,4969.0,XLSEN,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1026
                 STATUS=MPCLB(BUS2+SEN1,5340.0,XLSEN+1.,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1028
                 STATUS=MPCLB(BUS2+SEN2,5710.0,XLSEN+1.,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1030
                 STATUS=MPCLB(BUS2+NIN1,3370.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1032
                 STATUS=MPCLB(BUS2+NIN2,3616.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1034
                 STATUS=MPCLB(BUS2+NOUT1,4108.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1036
                 STATUS=MPCLB(BUS2+NOUT2,4354.0,FLOAT(NTUPS),INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1042
                 STATUS=MPCLB(BUS1+INPB,36224.0,XLTH,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1043
                 STATUS=MPCLB(BUS1+OUTB,37956.0,XLTH,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
      D          INSTR=1044
                 STATUS=MPCLB(BUS1+SYNB1,36480.0,XLSEN,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
                 INSTR=1055
                 STATUS=MPCLB(BUS1+ZRO,37587.0,XLSEN,INTGR,EVERY,LONG)
                 IF(STATUS.NE.0)GOTO 10
      D          INSTR=1054
                 STATUS=MPCLB(BUS1+BF1,37218.0,XLSEN,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GOTO 10
```

```
      D         INSTR=1045
                STATUS=MPCLR(BUS1+AIBIT,35598.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1046
                STATUS=MPCLR(BUS1+RIBIT,39590.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1048
                STATUS=MPCLR(BUS1+AOPR,35328.0,XLPARM+1.,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1049
                STATUS=MPCLR(BUS1+AOTDCT,35342.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1050
                STATUS=MPCLR(BUS1+AROPR,38582.0,XLPARM+1.,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1051
                STATUS=MPCLR(BUS1+ARODCT,38596.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1052
                STATUS=MPCLR(BUS1+SEN,35854.0,XLSEN+1.,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      D         INSTR=1056
                STATUS=MPCLR(BUS1+RECV,38212.0,XLSEN+1.,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GOTO 10
      C>>>>>>>>DEFINE GENERAL PURPOSE AREAS
      D         INSTR=80
                STATUS=MPCLR(BUS3+TMP1,2048.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=81
                STATUS=MPCLR(BUS3+TMP2,2560.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=82
                STATUS=MPCLR(BUS3+TMP3,3584.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=83
                STATUS=MPCLR(BUS3+TMP4,3072.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      C>>>>>>>>DEFINE "I/O REQUIRED STACK" AREAS
      D         INSTR=1200
                STATUS=MPCLR(BUS2+DCTIT1,9000.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=1201
                STATUS=MPCLR(BUS2+DCTIT2,9512.0,XLTH,REAL,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=1202
                STATUS=MPCLR(BUS2+IORDR1,10024.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=1203
                STATUS=MPCLR(BUS2+IORDR2,10280.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=1204
                STATUS=MPCLR(BUS2+MIBIT1,10536.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=1205
                STATUS=MPCLR(BUS2+MIBIT2,10792.0,XLTH,INTGR,EVERY,LONG)
      D         IF(STATUS.NE.0)GO TO 10
```

```
0293    D        INSTR=1206
0294             STATUS=MPCLB(BUS2+MIBIT3,11304,0,XLTH,INTGR,EVERY,LONG)
0295    D        IF(STATUS.NE.0)GO TO 10
0296    D        INSTR=1207
0297             STATUS=MPCLB(BUS2+MIBIT4,11560,0,XLTH,INTGR,EVERY,LONG)
0298    D        IF(STATUS.NE.0)GO TO 10
0299    D        INSTR=1208
0300             STATUS=MPCLB(BUS2+OTOCT1,11048,0,XLTH,INTGR,EVERY,LONG)
0301    D        IF(STATUS.NE.0)GO TO 10
        C9003    TYPE 9004
        C9004    FORMAT(1HS,'CLEAR ALL BUFFERS(Y/N)? ')
        C        READ(5,9005,END=9999,ERR=9003)ANSER
        C9005    FORMAT(A1)
        C        IF(ANSER.EQ.NO)GO TO 9999
        C>>>>>>>CLEAR BUFFERING AREAS
0302    D        INSTR=100
0303             STATUS=VCLR(YOM)
0304    D        IF(STATUS.NE.0)GO TO 10
0305    D        INSTR=101
0306             STATUS=VCLR(XOM)
0307    D        IF(STATUS.NE.0)GO TO 10
0308    D        INSTR=2053
0309             STATUS=MPWDB(NOUT1,ZERO(1),BYTE2,CNVYES,ZERO(2*NTUPS))
0310    D        IF(STATUS.NE.0)GOTO 10
0311    D        INSTR=2055
0312             STATUS=MPWDB(NIN1,ZERO(1),BYTE2,CNVYES,ZERO(2*NTUPS))
0313    D        IF(STATUS.NE.0)GOTO 10
0314    D        INSTR=2057
0315             STATUS=MPWDB(SEN1,ZERO(1),BYTE2,CNVYES,ZERO(2*LSEN))
0316    D        IF(STATUS.NE.0)GOTO 10
0317    D        INSTR=2059
0318             STATUS=MPWDB(RECV1,ZERO(1),BYTE2,CNVYES,ZERO(2*LSEN))
0319    D        IF(STATUS.NE.0)GOTO 10
        C???????????????????????????????????????????????????????????
        C        MAY NOT BE A GOOD IDEA TO SET AIBIT(1)=0
        CD       INSTR=2061
        C        STATUS=MPWDB(AIBIT,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
        CD       IF(STATUS.NE.0)GOTO 10
        C???????????????????????????????????????????????????????????
        C???????????????????????????????????????????????????????????
        C        MAY NOT BE A GOOD IDEA TO SET RIBIT(1)=0
        CD       INSTR=2062
        C        STATUS=MPWDB(RIBIT,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
        CD       IF(STATUS.NE.0)GOTO 10
        C???????????????????????????????????????????????????????????
0320    D        INSTR=2064
0321             STATUS=MPWDB(AUPB,ZERO(1),BYTE2,CNVNO,ZERO(LPARM))
0322    D        IF(STATUS.NE.0)GOTO 10
0323    D        INSTR=2065
0324             STATUS=MPWDB(AUTDCT,ZERO(1),BYTE2,CNVNO,ZERO(LTH))
0325    D        IF(STATUS.NE.0)GOTO 10
0326    D        INSTR=2066
0327             STATUS=MPWDB(BE1,ZERO(1),BYTE2,CNVYES,ZERO(2*LSEN))
0328    D        IF(STATUS.NE.0)GOTO 10
0329    D        INSTR=2067
0330             STATUS=MPWDB(SYNB1,ZERO(1),BYTE2,CNVYES,ZERO(2*LSEN))
```

```
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2068
          STATUS=VCLR(DCTIT1)
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2069
          STATUS=VCLR(DCTIT2)
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2077
          STATUS=MPWDB(OTDCT1,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2078
          STATUS=MPWDB(OPRM,ZERO(1),BYTE2,CNVYES,ZERO(LPARM+1))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2079
          STATUS=MPWDB(ORPRM,ZERO(1),BYTE2,CNVYES,ZERO(LPARM+1))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2080
          STATUS=MPWDB(NOUT1,ZERO(1),BYTE2,CNVYES,ZERO(NTUPS))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2081
          STATUS=MPWDB(NOUT2,ZERO(1),BYTE2,CNVYES,ZERO(NTUPS))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2082
          STATUS=MPWDB(ZRO,ZERO(1),BYTE2,CNVYES,ZERO(LSEN))
D         IF(STATUS.NE.0)GOTO 10
          DO 9006 I=1,256
9006      ZERO(I)=I-1
D         INSTR=2071
          STATUS=MPWDB(IORDR1,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2072
          STATUS=MPWDB(IORDR2,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
          DO 9007 I=1,134
9007      ZERO(I)=2
          DO 9008 I=135,256
9008      ZERO(I)=0
D         INSTR=2073
          STATUS=MPWDB(MIBIT1,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2074
          STATUS=MPWDB(MIBIT2,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2075
          STATUS=MPWDB(MIBIT3,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
D         INSTR=2076
          STATUS=MPWDB(MIBIT4,ZERO(1),BYTE2,CNVYES,ZERO(LTH))
D         IF(STATUS.NE.0)GOTO 10
          DO 9009 I=1,134
9009      ZERO(I)=0
9999      CONTINUE
          TYPE 101
101       FORMAT(1X,'      (1) MAP OPENED & BUFFERS DECLARED!')
          RETURN
C
```

```
        C       DIAGNOSTIC TRAP AREA
        C
0386    10      TYPE 100,INSTR,STATUS
0387            CALL EXIT
0388    100     FORMAT(1X,'***MAP ERROR*** "MAPON" INSTR=',I4,' HAS STATUS=',I6/)
0389            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	010564	2234	RW,I,CON,LCL
SPDATA	000374	126	RW,D,CON,LCL
SIDATA	002344	626	RW,D,CON,LCL
SVARS	000134	46	RW,D,CON,LCL
STEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 040666 8411

MAPON,LP/LI:1=MAPON/DE/NOTR

```
        C       PROGRAM NAME:CREATE.FTN ORIGINATED:13-FEB-80
        C                                             UPDATE:20-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE CREATE
0002            INTEGER BIDBF1,BIDBF2,BIDBF3
0003            INTEGER BUS1,BUS2,BUS3
0004            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0005            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCTIT1,DCTIT2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0006            INTEGER ORDR,START,SIZE,DSW,BTLTH
0007            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008            COMMON/MTAPE0/NIN(300),NOUT(300)
0009            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0010            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCTIT1,DCTIT2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 022            DATA REAL,CMPLX,INTGR/0,1,2/
0023            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026            DATA BUS1,BUS2,BUS3/64,128,192/
0027            DATA BIDBF1,BASE1,SIZE1/63,10000,0,4000,0/
```

```
                 DATA BIDBF2,BASE2,SIZE2/63,8800,0,1200,0/
                 DATA BIDBF3,BASE3,SIZE3/63,14000,0,2380,0/
      C
      C          CONFIGURE & BIND FIRST BUFFER REGION
      C
      D          INSTR=100
                 STATUS=MPCLB(BUS1+BIDBF1,BASE1,SIZE1,INTGR,EVERY,LONG)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=1
                 STATUS=MPCBF(BIDBF1,153)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=153
                 STATUS=MPFDVM(MAPX,NINM,X2)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=154
                 STATUS=MPDCTM(DCTM,X1,X2,COSZ)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=155
                 STATUS=MPMWGX(R1,NINM,KE,A1)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=156
                 STATUS=MPODPP(OPRM,PARC,A1)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=157
                 STATUS=MPFSTV(A1)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=158
                 STATUS=MPBASP(DCTM1,PWEITM,TMP2,X1)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=159
                 STATUS=MPASRT(IORDRM)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=160
                 STATUS=MPSCAN(DCTM2)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=161
                 STATUS=MPCDHA(MIBIT3)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=162
                 STATUS=MPFSTQ(OTDCTM,MIBIT3,TMP1,TMP2)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=163
                 STATUS=MPDQPP(PARC,ORPRM,A1)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=164
                 STATUS=MPFSTD(DRDCTM,ORDCTM,TMP1,IORDR2)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=165
                 STATUS=MPIDCM(X2,DRDCTM,COSZ)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=166
                 STATUS=MPVDNM(NOUTM,YOM,X2)
      D          IF(STATUS.NE.0)GO TO 10
      D          INSTR=167
                 STATUS=MPSSRT(IORDRM)
      D          IF(STATUS.NE.0)GO TO 10
```

```
0081    D       INSTR=2
0082            STATUS=MPTBF(0)
0083    D       IF(STATUS.NE.0)GO TO 10
        C
        C       CONFIGURE & BIND SECOND BUFFER REGION
        C
0084    D       INSTR=101
0085            STATUS=MPCLB(BUS1+BIDBF2,BASE2,SIZE2,INTGR,EVERY,LONG)
0086    D       IF(STATUS.NE.0)GO TO 10
0087    D       INSTR=3
0088            STATUS=MPCBF(BIDBF2,168)
0089    D       IF(STATUS.NE.0)GO TO 10
0090    D       INSTR=168
0091            STATUS=FFTN(X2,1,X2,VSHRT,WORK)
0092    D       IF(STATUS.NE.0)GO TO 10
0093    D       INSTR=169
0094            STATUS=FFTIN(X1,1,X1,VLONG,WORK)
0095    D       IF(STATUS.NE.0)GO TO 10
0096    D       INSTR=170
0097            STATUS=FF2R(X1,4,X1,VLONG,WORK)
0098    D       IF(STATUS.NE.0)GO TO 10
0099    D       INSTR=171
0100            STATUS=FFTIN(X2,1,X2,VSHRT,WORK)
0101    D       IF(STATUS.NE.0)GOTO 10
0102    D       INSTR=4
0103            STATUS=MPTBF(0)
0104    D       IF(STATUS.NE.0)GO TO 10
        C
        C       CONFIGURE & BIND THIRD BUFFER REGION
        C
0105    D       INSTR=102
0106            STATUS=MPCLB(BUS1+BIDBF3,BASE3,SIZE3,INTGR,EVERY,LONG)
0107    D       IF(STATUS.NE.0)GO TO 10
0108    D       INSTR=5
0109            STATUS=MPCBF(BIDBF3,172)
0110    D       IF(STATUS.NE.0)GO TO 10
0111    D       INSTR=172
0112            STATUS=VMOV1(AOPB)
0113    D       IF(STATUS.NE.0)GOTO 10
0114    D       INSTR=173
0115            STATUS=VMOV2(ORPPM)
0116    D       IF(STATUS.NE.0)GOTO 10
0117    D       INSTR=174
0118            STATUS=VMOV3(RIBIT)
0119    D       IF(STATUS.NE.0)GOTO 10
0120    D       INSTR=175
0121            STATUS=MPCDBA(MIBIT1)
0122    D       IF(STATUS.NE.0)GOTO 10
        C>>>>> SPECIAL "VMOV"S WHICH  FUNCTION AS NULL MODEM W/ 1 FRAME DELAY
        C  CHANGED NEXT PRE-BD-FCN FOR NO DELAY FOR DEMO ****************
0123    D       INSTR=176
0124            STATUS=VMOV(AROPB,AOPB)
0125    D       IF(STATUS.NE.0)GOTO 10
0126    D       INSTR=177
0127            STATUS=VMOV(SEN1,AOPB)
0128    D       IF(STATUS.NE.0)GOTO 10
```

```
0129     D          INSTR=178
         C CHANGED NEXT LINE FOR DEMO -- TOOK OUT DELAY******
0130                STATUS=VMOV(AQODCT,AQTDCT)
0131     D          IF(STATUS.NE.0)GOTO 10
0132     D          INSTR=179
0133                STATUS=VMOV(SEN2,AQTDCT)
0134     D          IF(STATUS.NE.0)GOTO 10
0135     D          INSTR=6
0136                STATUS=MPTHF(0)
0137     D          IF(STATUS.NE.0)GO TO 10
         C
0138                TYPE 101
0139     101        FORMAT(1X,'      (2) PRE-BOUND FUNCTIONS CREATED!')
0140                RETURN
         C
         C          DIAGNOSTIC TRAP AREA
         C
0141     10         TYPE 100,INSTR,STATUS
0142                CALL EXIT
0143     100        FORMAT(1X,'***MAP ERROR*** "CREATE" INSTR=',I4,' HAS STATUS=',I6/)
0144                END


PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1   002026   523           RW,I,CON,LCL
$PDATA   000030    12           RW,D,CON,LCL
$IDATA   000524   170           RW,D,CON,LCL
$VARS    000110    36           RW,D,CON,LCL
MTAPE0   002260   600           RW,D,OVR,GBL
MTAPE1   000012     5           RW,D,OVR,GBL
MTAPE2   000012     5           RW,D,OVR,GBL
OVRI     002020   520           RW,D,OVR,GBL
OVRO     002006   515           RW,D,OVR,GBL
OVRF     006016  1543           RW,D,OVR,GBL
OVRA     000152    53           RW,D,OVR,GBL
OVRP     002022   521           RW,D,OVR,GBL
OVRB     005004  1282           RW,D,OVR,GBL
OVR2     001002   257           RW,D,OVR,GBL
OVRL     000016     7           RW,D,OVR,GBL
OVRD     000012     5           RW,D,OVR,GBL
MAPDEF   000200    64           RW,D,OVR,GBL


TOTAL SPACE ALLOCATED = 027714  6118

CREATE,LP/LI:1=CREATE/DE/NOTR
```

```
        C       PROGRAM NAME:DEFINE.FTN ORIGINATED:13-FEB-80
        C                                         UPDATE:09-JUL-80
        C
        C
        C
        C
0001            SUBROUTINE DEFINE
0002            REAL IOSHA,IOSSA
0003            INTEGER EQ,GE,GT,SYNCNT,ANA,AFLG,SYN,SFLG,RDATA,XDATA
                1, SPIN,SPOUT,PARITY,RCVOFF,OUT,OFLG,CARD
                2,APDNFL,RGO,XGO,RGO2
                3,SYNC,RSYN,NEWSYN,OLSYN,LSTER,ERSYN
0004            INTEGER FL0,FL1,FL2,FL3,FL4,FL5,FL6,FL7
0005            INTEGER ATCANA,ATCSYN,WAIT,STRTUP,TIMER
0006            INTEGER AOM,ADAM,AOMPM,ADAMPM
0007            INTEGER BUS1,BUS2,BUS3
0008            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0009            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,AROOCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0010            INTEGER ORDR,START,SIZE,DSW,BTLTH
0011            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0012            COMMON/MTAPE0/NIN(300),NOUT(300)
0013            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0014            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0015            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0016            COMMON/OVRO/Y(256),NP,XN
0017            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0018            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0019            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0020            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0021            COMMON/OVR2/IORDR(256),LTHM1
0022            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0023            COMMON/OVRE/XR(512)
0024            COMMON/OVRD/CSN,START,SIZE,YES,NO
0025            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,AROOCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
```

```
          C         >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0026                DATA REAL,CMPLX,INTGR/0,1,2/
0027                DATA EVERY,SINGLE,DOUBLE/1,2,3/
0028                DATA BYTE2,BYTE4,BYTE8/2,4,8/
0029                DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0030                DATA BUS1,BUS2,BUS3/64,128,192/
0031                DATA FL0,FL1,FL2,FL3,FL4,FL5,FL6,FL7/0,1,2,3,4,5,6,7/
0032                DATA ATCANA,ATCSYN,WAIT,STRTUP,TIMER/1,2,3,4,7/
0033                DATA IOSPM,IOS,IOSBA,IOSSA/61,2,22100.,0./
0034                DATA AOMPM,AOMBA,AOMSA,AOMSZ/62,22700.,0.,512./
0035                DATA ADAMPM,ADAMBA,ADAMSA,ADAMSZ/63,23300.,0.,512./
0036                DATA IOSID,AOM,ADAM/16,22,23/
0037                DATA EQ,NE,GE,GT,LE,LT/0,1,2,3,4,5/
0038                DATA SYNCNT/100/
0039                DATA ANA,AFLG,SYN,SFLG/100,101,102,103/
0040                DATA INP,IFLG,OUT,OFLG/106,107,104,105/
0041                DATA RDATA,XDATA,SPIN,SPOUT,KOUNT/108,109,110,111,112/
0042                DATA INIT,PARITY,PCVOFF/113,114,127/
0043                DATA APDNFL,CARD,RGO,XGO,RGO2,SYNC/126,125,114,115,116,117/
0044                DATA RSYN,NEWSYN,OLSYN,LSTER,ERSYN/118,119,120,121,122/
          C
          C         DEFINE FUNCTION LIST SUPPORT
          C
          C         LIST "ATCANA": ATC ANALYZER W/ERROR CODING AND SERIALIZATION
0045                INSTR=1043
0046                STATUS=MPHFL(ATCANA)
0047                IF(STATUS.NE.0)GOTO 10
0048      D         INSTR=30
          C         HF(172)=VMOV1(AQPB)
0049                STATUS=MPXHF(172)
0050      D         IF(STATUS.NE.0)GO TO 10
0051      D         INSTR=31
          C         HF(153)=MPFDVM(MAPX,NINM,X2)
0052                STATUS=MPXHF(153)
0053      D         IF(STATUS.NE.0)GO TO 10
0054      D         INSTR=32
          C         HF(168)=FFTN(X2,1,X2,VSHRT,WORK)
0055                STATUS=MPXHF(168)
0056      D         IF(STATUS.NE.0)GO TO 10
0057      D         INSTR=33
          C         HF(154)=MPDCTM(DCTM,X1,X2,COSZ)
0058                STATUS=MPXHF(154)
0059      D         IF(STATUS.NE.0)GO TO 10
0060      D         INSTR=34
          C         HF(169)=FFTIN(X1,1,X1,VLONG,WORK)
0061                STATUS=MPXHF(169)
0062      D         IF(STATUS.NE.0)GO TO 10
0063      D         INSTR=37
          C         HF(155)=MPMWGX(R1,NINM,KE,A1)
0064                STATUS=MPXHF(155)
0065      D         IF(STATUS.NE.0)GO TO 10
0066      D         INSTR=38
          C         HF(156)=MPODPP(QPRM,PARC,A1)
0067                STATUS=MPXHF(156)
0068      D         IF(STATUS.NE.0)GO TO 10
0069      D         INSTR=40
```

```
         C        BF(157)=MPFSTV(A1)
0070                STATUS=MPXBF(157)
0071     D          IF(STATUS.NE.0)GO TO 10
0072     D          INSTR=41
         C          BF(170)=FF2R(X1,4,X1,VLONG,WORK)
0073                STATUS=MPXBF(170)
0074     D          IF(STATUS.NE.0)GO TO 10
0075     D          INSTR=42
         C          BF(158)=MPBASP(DCTM1,PWEITM,TMP2,X1)
0076                STATUS=MPXBF(158)
0077     D          IF(STATUS.NE.0)GO TO 10
0078     D          INSTR=43
         C          BF(159)=MPASRT(IORDRM)
0079                STATUS=MPXBF(159)
0080     D          IF(STATUS.NE.0)GO TO 10
0081     D          INSTR=44
         C          BF(160)=MPSCAN(DCTM2)
0082                STATUS=MPXBF(160)
0083     D          IF(STATUS.NE.0)GO TO 10
0084     D          INSTR=45
         C          BF(161)=MPCDBA(MIBIT3)
0085                STATUS=MPXBF(161)
0086     D          IF(STATUS.NE.0)GO TO 10
0087     D          INSTR=46
         C          BF(162)=MPFSTQ(OTDCTM,MIBIT,TMP1,TMP2)
0088                STATUS=MPXBF(162)
0089     D          IF(STATUS.NE.0)GO TO 10
0090     D          INSTR=48
         C          IF(TIMING.EQ.YES)STATUS=MPXFL(ATCSYN)
0091     D          IF(STATUS.NE.0)GO TO 10
0092                INSTR=1050
0093                STATUS=MPEFL(FL1)
0094                IF(STATUS.NE.0)GOTO 10
         C
         C          LIST "ATCSYN": ATC SYNTHESIZER W/ERROR DECODER & DESERIALIZATION
0095                INSTR=1055
0096                STATUS=MPBFL(ATCSYN)
0097                IF(STATUS .NE.0)GOTO 10
         C          ***********************************
         C          PATCHED OUT FOLLOWING LINE TO EXECUTE IN "DEMO" MODE IF NOT
         C          IN TIMING MODE.
         C          IF(LIVE.EQ.YES)GO TO 200
         C          IF(TIMING.EQ.YES)GO TO 200
         C>>>>> INTRODUCE "NULL" MODE*
0098     D          INSTR=580
         C          BF(176)=VMOV(ARQPB,AQPB)
         C          STATUS=MPXBF(176)
0099     D          IF(STATUS.NE.0)GO TO 10
         C*********DON'T NEED DELAY FOR DEMO***
         CD         INSTR=581
         CC         BF(177)=VMOV(RECV1,AQPB)
         C          STATUS=MPXBF(177)
         CD         IF(STATUS.NE.0)GO TO 10
         C****************************************
0100     D          INSTR=582
         C          BF(178)=VMOV(ARQDCT,AQTDCT)
```

```
             C         STATUS=MPXBF(178)
0101         D         IF(STATUS.NE.0)GO TO 10
             C********** REMOVED FOR DEMO *******
             CD        INSTR=583
             CC        BF(179)=VMOV(RECV2,AOTDCT)
             C         STATUS=MPXBF(179)
             CD        IF(STATUS.NE.0)GO TO 10
             C*******************
0102         200       CONTINUE
0103         D         INSTR=60
             C         BF(173)=VMOV2(ORPRM)
0104                   STATUS=MPXBF(173)
0105         D         IF(STATUS.NE.0)GO TO 10
0106         D         INSTR=61
             C         BF(163)=MPDOPP(PARC,ORPRM,A1)
0107                   STATUS=MPXBF(163)
0108         D         IF(STATUS.NE.0)GO TO 10
0109         D         INSTR=62
             C         BF(157)=MPFSTV(A1)
0110                   STATUS=MPXBF(157)
0111         D         IF(STATUS.NE.0)GO TO 10
0112         D         INSTR=63
             C         BF(170)=FF2R(X1,4,X1,VLONG,WORK)
0113                   STATUS=MPXBF(170)
0114         D         IF(STATUS.NE.0)GO TO 10
0115         D         INSTR=64
             C         BF(158)=MPBASP(DCTM1,PWFITM,TMP2,X1)
0116                   STATUS=MPXBF(158)
0117         D         IF(STATUS.NE.0)GO TO 10
0118         D         INSTR=65
             C         BF(167)=MPSSRT(IORDRM)
0119                   STATUS=MPXBF(167)
0120         D         IF(STATUS.NE.0)GO TO 10
0121         D         INSTR=66
             C         BF(160)=MPSCAN(DCTM2)
0122                   STATUS=MPXBF(160)
0123         D         IF(STATUS.NE.0)GO TO 10
0124         D         INSTR=67
             C         BF(175)=MPCDBA(MIBIT1)
0125                   STATUS=MPXBF(175)
0126         D         IF(STATUS.NE.0)GO TO 10
0127         D         INSTR=670
             C         BF(174)=VMOV3(RIBIT)
0128                   STATUS=MPXBF(174)
0129         D         IF(STATUS.NE.0)GO TO 10
0130         D         INSTR=68
             C         BF(164)=MPFSTD(DRDCTM,ORDCTM,TMP1,IORDR2)
0131                   STATUS=MPXBF(164)
0132         D         IF(STATUS.NE.0)GO TO 10
0133         D         INSTR=69
             C         BF(165)=MPIDCM(X2,DRDCTM,COSZ)
0134                   STATUS=MPXBF(165)
0135         D         IF(STATUS.NE.0)GO TO 10
0136         D         INSTR=70
             C         BF(171)=FFTIN(X2,1,X2,VSHRT,WORK)
0137                   STATUS=MPXBF(171)
```

```
0138     D        IF(STATUS.NE.0)GO TO 10
0139     D        INSTR=71
         C        BF(166)=MPVDNM(NOUTM,YOM,X2)
0140              STATUS=MPXBF(166)
0141     D        IF(STATUS.NE.0)GO TO 10
0142     D        INSTR=1070
0143              STATUS=MPEFL(FL2)
0144              IF(STATUS.NE.0)GOTO 10
         C
         C        LIST "WAIT": THE WAIT LOOP
0145     D        INSTR=1080
0146              STATUS=MPBFL(WAIT)
0147     D        IF(STATUS.NE.0)GOTO 10
0148     D        INSTR=1085
0149              STATUS=MPIIF(INIT,EQ,0,FL4,FL0)
0150     D        IF(STATUS.NE.0)GOTO 10
0151     D        INSTR=1090
0152              STATUS=MPIIF(SYN,EQ,0,ATCSYN,FL0)
0153     D        IF(STATUS.NE.0)GOTO 10
0154     D        INSTR=1095
0155              STATUS=MPIIF(ANA,EQ,0,ATCANA,FL0)
0156     D        IF(STATUS.NE.0)GOTO 10
0157     D        INSTR=1100
0158              STATUS=MPEFL(FL3)
0159     D        IF(STATUS.NE.0)GOTO 10
         C
         C        LIST "STRTUP":INITIALIZATION LOOP
0160     D        INSTR=1105
0161              STATUS=MPBFL(STRTUP)
0162     D        IF(STATUS.NE.0)GOTO 10
0163     D        INSTR=1106
0164              STATUS=MPIST(INIT,1)
0165     D        IF(STATUS.NE.0)GOTO 10
0166     D        INSTR=1111
0167              STATUS=MPIST(OUT,1)
0168     D        IF(STATUS.NE.0)GOTO 10
0169     D        INSTR=1112
0170              STATUS=MPIST(INP,1)
0171     D        IF(STATUS.NE.0)GOTO 10
0172     D        INSTR=1113
0173              STATUS=MPIST(AFLG,0)
0174     D        IF(STATUS.NE.0)GOTO 10
0175     D        INSTR=1114
0176              STATUS=MPIST(SFLG,0)
0177     D        IF(STATUS.NE.0)GOTO 10
0178     D        INSTR=1115
0179              STATUS=MPIST(IFLG,0)
0180     D        IF(STATUS.NE.0)GOTO 10
0181     D        INSTR=1116
0182              STATUS=MPIST(OFLG,0)
0183     D        IF(STATUS.NE.0)GOTO 10
0184     D        INSTR=1117
0185              STATUS=MPIST(SPIN,-1)
0186     D        IF(STATUS.NE.0)GOTO 10
0187     D        INSTR=1118
0188              STATUS=MPIST(SPOUT,-1)
```

```
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1122
         STATUS=MPIST(KOUNT,SYNCNT)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1123
         STATUS=MPIST(RDATA,-1)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1124
         STATUS=MPIST(XDATA,-1)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1125
         STATUS=MPIST(APDNFL,0)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1126
         STATUS=MPIST(RGO,0)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1127
         STATUS=MPIST(XGO,0)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1128
         STATUS=MPIST(RGO2,0)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1160
         STATUS=MPIST(SYNC,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1161
         STATUS=MPIST(RSYN,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1162
         STATUS=MPIST(NEWSYN,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1163
         STATUS=MPIST(OLSYN,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1164
         STATUS=MPIST(LSTEP,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1165
         STATUS=MPIST(ERSYN,0)
         IF(STATUS.NE.0)GOTO 10
D        INSTR=1137
         STATUS=MPLDS(AOM,IOS,AOMPM)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1138
         STATUS=MPRNS(AOM,IOS,AOMSA)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1139
         STATUS=MPLDS(ADAM,IOS,ADAMPM)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1140
         STATUS=MPRNS(ADAM,IOS,ADAMSA)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1135
         STATUS=MPLDS(IOSTD,IOS,IOSPM)
D        IF(STATUS.NE.0)GOTO 10
D        INSTR=1136
```

```
              STATUS=MPRNS(IOSID,IOS,IOSSA)
      D       IF(STATUS.NE.0)GOTO 10
      D       INSTR=1150
              STATUS=MPEFL(FL4)
      D       IF(STATUS.NE.0)GOTO 10
      C
      C       LIST "TIMER": THE ANALYZER/SYNTHESIZER TIMER LOOP
      D       INSTR=1280
              STATUS=MPBEL(TIMER)
      D       IF(STATUS.NE.0)GOTO 10
      D       INSTR=1285
              STATUS=MPIIF(0,EQ,0,ATCANA,FL0)
      D       IF(STATUS.NE.0)GOTO 10
      D       INSTR=1286
              STATUS=MPIIF(1,EQ,1,ATCSYN,FL0)
      D       IF(STATUS.NE.0)GOTO 10
      D       INSTR=1290
              STATUS=MPEFL(FL7)
      D       IF(STATUS.NE.0)GOTO 10
              TYPE 101
      101     FORMAT(1X,'     (3) FUNCTION LISTS DEFINED!')
              RETURN
      C
      C       DIAGNOSTIC TRAP AREA
      C
      10      TYPE 100,INSTR,STATUS
              CALL EXIT
      100     FORMAT(1X,'***MAP ERROR*** "DEFINE" INSTR=',I3,' HAS STATUS=',I6/)
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	003320	872	RW,I,CON,LCL
SPDATA	000150	52	RW,D,CON,LCL
SIDATA	000714	230	RW,D,CON,LCL
SVARS	000266	91	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 031674 6622

```
NO FPP INSTRUCTIONS GENERATED

DEFINE,LP/LI:1=DEFINE/DE/NOTR
```

```
        C       PROGRAM NAME:SETUP.FTN  ORIGINATED:28-APR-80
        C                                          UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE SETUP
0002            REAL IOSHA,IOSSA
0003            INTEGER IOSPMH(600),PROGRM(494),WORD
0004            LOGICAL*1 CWORD(4),HEXC(16)
0005            INTEGER AOMPM,ADAMPM
0006            INTEGER DUAL,EXT,DAFLTP,DAFIXP,EXTT,P2DIS,P2EN
                1 ,CC0FSR,CC0SSR,CC1FSR,CC1SSR,CC2FSR,CC2SSR
                2 ,CC3FSR,CC3SSR,CHANS1(2),DDFFLT,DDFFIX,OFFSET
0007            INTEGER BUS1,BUS2,BUS3
0008            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTER,CNVYES,CNVNO,UNITY,STATUS,SHORT
0009            INTEGER WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0010            INTEGER ORDR,START,SIZE,DSW,BTLTH
0011            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0012            COMMON/MTAPE0/NIN(300),NOUT(300)
0013            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0014            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0015            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0016            COMMON/OVRO/Y(256),NP,XN
0017            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0018            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0019            COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0020            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0021            COMMON/OVR2/IORDR(256),LTHM1
0022            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0023            COMMON/OVRF/XR(512)
0024            COMMON/OVRD/CSN,START,SIZE,YES,NO
0025            COMMON/MAPDEF/WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
0026              EQUIVALENCE (IOSPMB(1),XR(1)),(PROGRM(1),DCT2(1)),(MSGERR,ITOT)
0027              DATA REAL,CMPLX,INTGR/0,1,2/
0028              DATA EVERY,SINGLE,DOUBLE/1,2,3/
0029              DATA BYTE2,BYTE4,BYTE8/2,4,8/
0030              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0031              DATA BUS1,BUS2,BUS3/64,128,192/
0032              DATA MONO,DUAL,INT,EXT,DAFLTP,DAFIXP/0,1,0,2,0,4/
0033              DATA CC0FSR,CC0SSR,CC1FSR,CC1SSR/0,8,0,16/
0034              DATA CC2FSR,CC2SSR,CC3FSR,CC3SSR/0,32,0,64/
0035              DATA DDFFLT,DDFFIX/0,128/
0036              DATA OFFSET,INTT,EXTT/2,0,1/
0037              DATA P2DIS,P2EN/0,4/
0038              DATA CHANS1/1,-1/
0039              DATA IOSPMB,OVRHD/600*0,12./
0040              DATA IOSPM,IOS,IOSBA,IOSSA/61,2,22100.,0./
0041              DATA AOMPM,AOMBA,AOMSA,AOMSZ/62,22700.,0.,512./
0042              DATA ADAMPM,ADAMBA,ADAMSA,ADAMSZ/63,23300.,0.,512./
0043              DATA HEXC/'0','1','2','3','4','5','6','7','8','9',
                  1'A','B','C','D','E','F'/
0044              DATA CWORD/'E','C','C','4'/
          C
          C       SET UP IOS-2(MODEM SCROLL)
          C
0045              CALL ASSIGN(3,'SY:LINE.PBJ')
0046              CALL LOAD
0047              IF(MSGERR.EQ.0)GO TO 2000
0048              TYPE 2001
0049      2001    FORMAT(1X,'***FATAL: READ ERROR ON "LINE.PBJ" ***'//)
0050              STOP
0051      2000    CONTINUE

          C       IOSM(IOS-2 SCROLL)
0052              CALL IOSM
          C
          C       AOM(D/A SCROLL)
0053              CALL AOM
          C
          C       ADAM(A/D SCROLL)
0054              CALL ADAM
0055              RETURN
          C
          C       DIAGNOSTIC TRAP AREA
          C
0056      10      TYPE 100,INSTR,STATUS
0057              CALL EXIT
0058      100     FORMAT(1X,'***MAP ERROR*** "SETUP" INSTR=',I3,' HAS STATUS=',I6/)
0059              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000106	35	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000152	53	RW,D,CON,LCL
$VARS	000232	77	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025534 5550

NO FPP INSTRUCTIONS GENERATED

```
SETUP.LP/LI:1=SETUP/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F          13:28:58    09-AUG-80           PAGE 1
LOAD.FTN          /WR

         C       PROGRAM NAME:LOAD.FTN           ORIGINATED:18-APR-79
         C                                             UPDATED:02-MAY-80
         C       THIS PROGRAM READS THE MAP-300 ASSEMBLED OBJECT CODE,
         C       *.PBJ, AND RETAINS ONLY THE PM INSTRUCTIONS IN THE
         C       CALLING ARRAY.
         C
0001             SUBROUTINE LOAD
0002             INTEGER ORDR,START,SIZE,DSW,BTLTH
0003             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0004             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0005             COMMON/OVRD/CSN,START,SIZE,YES,NO
0006             LOGICAL*1 M,DDL(2),ZERO,H(68),N(2),K(78),L(6),F(2),REM(78)
                 1,PBJ(3),DMP(3),ASCII1,ASCII2,ASCII3,ASCII4,ONE,HEXC(16),S(4,5)
0007             INTEGER DT,DD(23),PM(1),ADDR,VALUE
                 1,APU,APS,HIM,IOS2,UNKN,PROC
0008             EQUIVALENCE (DT,DDL)
         C       EQUIVALENCE VARIABLES SUCH THAT PM->DCT2
         C                                       SIZE->TTY
         C                                       IERR->ITOT
0009             EQUIVALENCE (PM(1),DCT2(1)),(MSGERR,ITOT)
0010             DATA S(1,1),S(2,1),S(3,1),S(4,1)/'A','P','U',' '/
0011             DATA S(1,2),S(2,2),S(3,2),S(4,2)/'A','P','S',' '/
0012             DATA S(1,3),S(2,3),S(3,3),S(4,3)/'H','I','M',' '/
0013             DATA S(1,4),S(2,4),S(3,4),S(4,4)/'I','O','S','2'/
0014             DATA S(1,5),S(2,5),S(3,5),S(4,5)/'?','?','?','?'/
0015             DATA HEXC/'0','1','2','3','4','5','6','7','8'
                 1,'9','A','B','C','D','E','F'/
0016             DATA ASCII1,ASCII2,ASCII3,ASCII4/'1','2','3','4'/
0017             DATA APU,APS,HIM,IOS2,UNKN/1,2,3,4,5/
0018             DATA DD/'00','01','02','10'
                 1,'11','12','13','14','15','16','17','18','19'
                 2,'1A','1B','1C','1D','1E'
                 3,'1F','20','21','22','FF'/
0019             DATA DMP/'D','M','P'/
0020             DATA PBJ/'P','B','J'/
         C
         C       READ DATA DESIGNATOR,2 BYTES
         C
0021             ADDR=1
0022             IERR=0
0023             LINE=0
0024     5       READ(3,103,END=2000,ERR=2001)DDL(1),DDL(2)
0025             LINE=LINE+1
         C
         C       RECOGNIZE AND CLASSIFY
         C
0026             DO 10 I=1,23
0027             IF(DT.EQ.DD(I))GO TO 20
0028     10      CONTINUE
         D       TYPE 100,DT,LINE
0029             IERR=1
0030             GO TO 3000
0031     20      INDEX=1
0032             BACKSPACE 3
0033             GO TO (30,31,32,33,34,34,34,34,34,34,34,34
                 1,34,34,34,34,34,34,35,36,37,38,39)INDEX
```

```
C******
C       DD=00,NULL BYTE
C******
30      LEFT=78
        READ(3,116)DT,(REM(I),I=1,LEFT)
D       PRINT 113,LINE,DT
        GO TO 5
C******
C       DD=01,SET BUS BITS
C******
31      LEFT=76
        READ(3,116)DT,ZERO,M,(REM(I),I=1,LEFT)
D       PRINT 113,LINE,DT,ZERO,M
        GO TO 5
C******
C       DD=02,SET ADDRESS ON BUS
C******
32      LEFT=72
        READ(3,116)DT,ZERO,(H(I),I=1,5),(REM(I),I=1,LEFT)
D       PRINT 113,LINE,DT,ZERO,(H(I),I=1,5)
        LADDR=0
        DO 46 J=1,5
        DO 45 I=1,16
        IF(H(J).EQ.HEXC(I))GO TO 46
45      CONTINUE
D       TYPE 112,DT,H(J),LINE
        IERR=1
        GO TO 3000
46      LADDR=LADDR+(16.**(5-J))*(I-1)
        GO TO 5
C******
C       DD=10,PROGRAM HEADER
C******
33      LEFT=0
        READ(3,116)DT,(K(I),I=1,78)
D       PRINT 113,LINE,DT,(K(I),I=1,78)
DD      TYPE 104,(K(I),I=1,78)
        GO TO 5
C******
C       DD=12,MODULE HEADER
C******
34      LEFT=0
        READ(3,103)ONE,H(1),(L(I),I=1,6),(K(I),I=1,72)
D       PRINT 114,LINE,ONE,H(1),(L(I),I=1,6),(K(I),I=1,72)
        IF(H(1).EQ.ASCII1)PROC=APU
        IF(H(1).EQ.ASCII2)PROC=APS
        IF(H(1).EQ.ASCII3)PROC=HIM
        IF(H(1).EQ.ASCII4)PROC=IOS2
        IF(H(1).GT.ASCII4)PROC=UNKN
D       TYPE 106,(S(I,PROC),I=1,4)
D       TYPE 107,(L(I),I=1,6)
        GO TO 5
C******
C       DD=1F,MODULE END
C******
35      LEFT=0
```

```
0063               READ(3,117)DT,(H(I),I=1,4)
          D        PRINT 113,LINE,DT,(H(I),I=1,4)
          D        TYPE 108
0064               GO TO 5
          C*****
          C        DD=20,DATA OR INSTRUCTION
          C*****
0065      36       LEFT=0
0066               READ(3,105)DT,(N(I),I=1,2)
          D        PRINT 119,LINE,DT,(N(I),I=1,2)
0067               NUMWRD=0
0068               DO 41 J=1,2
0069               DO 40 I=1,16
0070               IF(N(J).EQ.HEXC(I))GO TO 41
0071      40       CONTINUE
          D        TYPE 112,DT,N(J),LINE
0072               GO TO 3000
0073      41       NUMWRD=NUMWRD+(16.**(2-J))*(I-1)
0074               NUMHEX=NUMWRD*4
0075               IF(NUMWRD.EQ.0)GO TO 5
0076               BACKSPACE 3
0077               READ(3,116)DT,(N(I),I=1,2),(H(I),I=1,NUMHEX)
0078               DO 43 II=1,NUMWRD
0079               XVALUE=0.0
0080               IBASE=(II-1)*4
0081               DO 44 J=1,4
0082               JJ=IBASE+J
0083               DO 42 I=1,16
0084               IF(H(JJ).EQ.HEXC(I))GO TO 44
0085      42       CONTINUE
          D        TYPE 112,DT,H(JJ),LINE
0086               IERR=1
0087               GO TO 3000
0088      44       XVALUE=XVALUE+(16.**(4-J))*(I-1)
0089               IF(XVALUE.LE.32767.0)GO TO 440
0090               XVALUE=XVALUE-65536.0
0091      440      VALUE=IFIX(XVALUE)
          D        PRINT 120,(S(I,PROC),I=1,4),ADDR,(H(IBASE+I),I=1,4),VALUE
0092               PM(ADDR)=VALUE
0093      43       ADDR=ADDR+1
0094               GO TO 5
          C*****
          C        DD=21,REPEATED 16 BIT DATA
          C*****
0095      37       LEFT=72
0096               READ(3,116)DT,(N(I),I=1,2),(H(I),I=1,4),(REM(I),I=1,LEFT)
          D        PRINT 113,LINE,DT,(N(I),I=1,2)
0097               GO TO 5
          C*****
          C        DD=22,REPEATED 32 BIT DATA
          C*****
0098      38       LEFT=68
0099               READ(3,116)DT,(N(I),I=1,2),(H(I),I=1,8),(REM(I),I=1,LEFT)
          D        PRINT 113,LINE,DT,(N(I),I=1,2)
0100               GO TO 5
          C*****
```

```
FORTRAN IV-PLUS V02-51E          13:28:58     09-AUG-80            PAGE 4
LOAD.FTN             /WR

        C        DD=FF,PROGRAM END
        C*****
0101    39       LEFT=76
0102             READ(3,116)DT,(F(I),I=1,2),(REM(I),I=1,LEFT)
        D        PRINT 113,LINE,DT,(F(I),I=1,2)
        D        TYPE 111,LINE,ADDR-1
0103             SIZE=ADDR-1
0104             GO TO 3000
        C
        C        FILE STRUCTURE PROBLEMS
        C
0105    2000     TYPE 109
0106             GO TO 3000
0107    2001     TYPE 110
0108             GO TO 3000
        C
        C        NORMAL EXIT
        C
0109    3000     CALL CLOSE(3)
0110             END FILE 6
0111             RETURN
        C
        C        FORMAT DECLARATIONS
        C
        D100     FORMAT(1X,'*** DD ERROR *** DT =',A2,' IN LINE ',I4/)
        D101     FORMAT(1H$,'"PRJ" FILE TO BE READ =')
        D102     FORMAT(1H$,'"DMP" FILE TO BE SAVED =')
0112    103      FORMAT(80A1)
        D104     FORMAT(//1X,'PROGRAM HEADER: ',80A1)
0113    105      FORMAT(A2,2A1)
        DD106    FORMAT(1X,'SUB-PROCESSOR TYPE: ',4A1)
        D107     FORMAT(1X,'SUB-PROCESSOR MODULE HEADER: ',6A1)
        D108     FORMAT(1X,'END OF SUB-PROCESSOR MODULE!')
0114    109      FORMAT(1X,'*** EOF BEFORE PROGRAM END FOUND ***')
0115    110      FORMAT(1X,'*** PRJ FILE STRUCTURE ERROR ***')
        D111     FORMAT(1X,'PROGRAM ENDS W/ LINE ',I4,' & LAST PC=',I8//)
        D112     FORMAT(1X,'***HEX CHAR. IS ?*** DT=',A2,' ?=',A1,' IN LINE ',I4/)
        D113     FORMAT(1X,'LINE=',I4,' DATA DESCRIPTOR=',1X,A2,' CODE=',80A1)
        D114     FORMAT(1X,'LINE=',I4,' DATA DESCRIPTOR=',1X,A1,A1,' CODE=',80A1)
0116    116      FORMAT(A2,78A1)
0117    117      FORMAT(A2,4A1)
        D119     FORMAT(1X,'LINE=',I4,' DATA DESCRIPTOR=',1X,A2,' LENGTH=',2A1)
        D120     FORMAT(1X,4A1,1X,'PM(',I3,')= H:',4A1,2X,'D:',I6)
0118             END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	003030	780	RW,I,CON,LCL
$PDATA	000070	28	RW,D,CON,LCL
$IDATA	000146	51	RW,D,CON,LCL
$VARS	000574	190	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
OVRB	005004	1282	RW,D,OVR,GBL

```
OVRL     000016      7              RW,D,OVR,GBL
OVRD     000016      7              RW,D,OVR,GBL


TOTAL SPACE ALLOCATED = 011124  2346

LOAD,LP/LI:1=LOAD/NOTR
```

```
        C        PROGRAM NAME:IOSM.FTN    ORIGINATE  28-APR-80
        C                                           UPDATE:20-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE IOSM
0002             REAL IOSBA,IOSSA
0003             INTEGER IOSPMB(600),PROGRM(494),WORD
0004             LOGICAL*1 CWORD(4),HEXC(16)
0005             INTEGER AOMPM,ADAMPM
0006             INTEGER DUAL,EXT,DAFLTP,DAFIXP,EXTI,P2DIS,P2EN
                 1 ,CCOFSR,CCOSSR,CC1FSR,CC1SSR,CC2FSR,CC2SSR
                 2 ,CC3FSR,CC3SSR,CHANS1(2),DDFFLT,DDFFIX,OFFSET
0007             INTEGER BUS1,BUS2,BUS3
0008             INTEGER REAL,CMPLX,INTGP,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0009             INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0010             INTEGER ORDR,START,SIZE,DSW,BTLTH
0011             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0012             COMMON/4TAPEO/NIN(300),NOUT(300)
0013             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0014             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0015             COMMON/OVR1/LIB,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0016             COMMON/OVRO/Y(256),NF,XN
0017             COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0018             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0019             COMMON/OVRP/M,G,V0,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0020             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0021             COMMON/OVR2/IORDR(256),LTHM1
0022             COMMON/OVRL/SLOG,LNGTH,CTEMP,LTOT,BTLTH
0023             COMMON/OVRE/XR(512)
0024             COMMON/OVRD/CSW,START,SIZE,YES,NO
0025             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
0026            EQUIVALENCE (IOSPMB(1),XR(1)),(PROGRM(1),DCT2(1)),(MSGERR,ITOT)
0027            DATA REAL,CMPLX,INTGR/0,1,2/
0028            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0029            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0030            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0031            DATA BUS1,BUS2,BUS3/64,128,192/
0032            DATA WORD,DUAL,INT,EXT,DAFLIP,DAFIXP/0,1,0,2,0,4/
0033            DATA CC0FSR,CC0SSR,CC1FSR,CC1SSR/0,8,0,16/
0034            DATA CC2FSR,CC2SSR,CC3FSR,CC3SSR/0,32,0,64/
0035            DATA DOFFLT,DOFFIX/0,128/
0036            DATA OFFSET,INIT,EXIT/2,0,1/
0037            DATA P2DIS,P2EN/0,4/
0038            DATA CHANS1/1,-1/
0039            DATA IOSPMB,OVRHD/600*0,12./
0040            DATA IOSPM,IOS,IOSBA,IOSSA/61,2,22100.,0./
0041            DATA ADMPM,ADMBA,ADMSA,ADMSZ/62,22700.,0.,512./
0042            DATA ADAMPM,ADAMBA,ADAMSA,ADAMSZ/63,23300.,0.,512./
0043            DATA HEXC/'0','1','2','3','4','5','6','7','8','9',
               1'A','B','C','D','E','F'/
0044            DATA CWORD/'E','C','C','4'/
          C
          C     IOS-2 HEADER BLOCK DEFINITION
          C
0045            PMSIZE=SIZE
0046            IOSPMB(1)=SIZE
0047            IOSPMB(2)=SEN2*256+SEN1
0048            DO 2002 WORD=1,SIZE
0049      2002  IOSPMB(WORD+2)=PROGRM(WORD)
          C     DATA TRANSFER
0050            IOSPMB(SIZE+2+1)=0
          C     CHANNELS + BUFFER SIZE
0051            IOSPMB(SIZE+2+2)=2
0052            IOSPMB(SIZE+2+3)=RECV2*256+RECV1
          C     # OF SCROLL REGISTER TO LOAD
0053            IOSPMB(SIZE+2+4)=0
          C     FIRST SCROLL REGISTER
0054            IOSPMB(SIZE+2+5)=0
          C     OFFSET IDENTIFIER
0055            IOSPMB(SIZE+2+6)=-1
          C     CHAIN ANCHORS
0056            IOSPMB(SIZE+2+7)=-1
0057            IOSPMB(SIZE+2+8)=-1
0058            IOSPMB(SIZE+2+9)=-1
0059            IOSPMB(SIZE+2+10)=-1
          C
          C     SET SPEECH SIDE & LINE SIDE INTERRUPT RATES
          C
0060            INSTR=1000
          C1000 TYPE 1001
          C1001 FORMAT(1HS,'HEX CODE FOR SPI= ')
          C     READ(5,1002,END=10,ERR=1000)CWORD
          C1002 FORMAT(4A1)
          C     DATUM=0.
          C     DO 2004 J=1,4
          C     DO 2005 I=1,16
          C     IF(CWORD(J).EQ.HEXC(I))GOTO 2006
```

```
              C2005    CONTINUE
              C        TYPE 2003
              C2003    FORMAT(1X,'***FATAL: ILLEGAL CHARACTER IN ICW ***'//)
              C        STOP
              C2006    DATUM=DATUM+(16.0**(4-J))*(I-1)
              C2004    CONTINUE
              C        IF(DATUM.GT.32767.)DATUM=DATUM-65536.
              C        IOSPMB(4)=IFIX(DATUM)
              C        HEX ECC4 TO CLOCKS
0061                   IOSPMB(4)=-4924
              C
              C        DEFINE IOS PROGRAM MEMORY BUFFER
              C
              C        IOS-2(MODEM SCROLL)
0062          D        INSTR=1000
0063                   STATUS=MPCLB(BUS1+IOSPM,IOSBA,PMSIZE+OVRHD,INTGR,EVERY,LONG)
0064          D        IF(STATUS.NE.0)GOTO 10
              C
              C        WRITE IOS PROGRAM INTO BUS1 AREAS
              C
              C        IOS-2(MODEM SCROLL)
0065          D        INSTR=2000
0066                   IOVRHD=IFIX(OVRHD)
0067                   STATUS=MP*DB(IOSPM,IOSPMB(1),BYTE2,CNVRO,IOSPMB(SIZE+IOVRHD))
0068          D        IF(STATUS.NE.0)GOTO 10
0069                   TYPE 101
0070          101      FORMAT(1X,'     (4) IOS-2 SETUP COMPLETE!')
0071                   RETURN
              C
              C        DIAGNOSTIC TRAP AREA
              C
0072          10       TYPE 100,INSTR,STATUS
0073                   CALL EXIT
0074          100      FORMAT(1X,'***MAP ERROR*** "IOSM" INSTR=',I4,' HAS STATUS=',I6/)
0075                   END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000622	201	RW,I,CON,LCL
$IDATA	000156	55	RW,D,CON,LCL
$VARS	000240	80	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL

```
IOSM.FTN        /DE/AR

OVRD     000012      5          RW,D,OVR,GBL
MAPDEF   000200     64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026244  5714

IOSM,LP/LI:1=IOSM/DE/NOTR
```

```
         C        PROGRAM NAME:AOM.FTN     ORIGINATED:28-APR-80
         C                                            UPDATE:20-MAY-80
         C
         C
         C
         C
0001              SUBROUTINE AOM
0002              REAL IOSBA,IOSSA
0003              INTEGER IOSPMB(600),PROGRM(494),WORD
0004              LOGICAL*1 CWORD(4),HEXC(16)
0005              INTEGER AOMPM,ADAMPM
0006              INTEGER DUAL,EXT,DAFLTP,DAFIXP,EXTT,P2DIS,P2EN
                  1 ,CC0FSR,CC0SSR,CC1FSR,CC1SSR,CC2FSR,CC2SSR
                  2 ,CC3FSR,CC3SSR,CHANS1(2),DDFFLT,DDFFIX,OFFSET
0007              INTEGER BUS1,BUS2,BUS3
0008              INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                  1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0009              INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,ORPRM,ORDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,OPRM,OTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AOPB,AOTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,AROOCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
         C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0010              INTEGER ORDR,START,SIZE,DSW,BTLTH
0011              LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0012              COMMON/MTAPE0/NIN(300),NOUT(300)
0013              COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0014              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0015              COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0016              COMMON/OVRO/Y(256),NP,XN
0017              COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0018              COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0019              COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0020              COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0021              COMMON/OVR2/IORDR(256),LTHM1
0022              COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0023              COMMON/OVRF/XR(512)
0024              COMMON/OVRD/CSN,START,SIZE,YES,NO
0025              COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,ORPRM,ORDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,OPRM,OTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AOPB,AOTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,AROOCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
         C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
0026              EQUIVALENCE (IOSPMB(1),XR(1)),(PROGRM(1),DCT2(1))
0027              DATA REAL,CMPLX,INTGR/0,1,2/
0028              DATA EVERY,SINGLE,DOUBLE/1,2,3/
0029              DATA BYTE2,BYTE4,BYTE8/2,4,8/
0030              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0031              DATA BUS1,BUS2,BUS3/64,128,192/
0032              DATA MONO,DUAL,INT,EXT,DAFLTP,DAFIXP/0,1,0,2,0,4/
0033              DATA CC0FSR,CC0SSR,CC1FSR,CC1SSR/0,8,0,16/
0034              DATA CC2FSR,CC2SSR,CC3FSR,CC3SSR/0,32,0,64/
0035              DATA DDFFLT,DDFFIX/0,128/
0036              DATA OFFSET,INTT,EXTT/2,0,1/
0037              DATA P2DIS,P2EN/0,4/
0038              DATA CHANS1/1,-1/
0039              DATA IOSPMB,OVRHD/600*0,12./
0040              DATA IOSPM,IOS,IOSBA,IOSSA/61,2,22100.,0./
0041              DATA AOMPM,AOMBA,AOMSA,AOMSZ/62,22700.,0.,512./
0042              DATA ADAMPM,ADAMBA,ADAMSA,ADAMSZ/63,23300.,0.,512./
0043              DATA HEXC/'0','1','2','3','4','5','6','7','8','9',
                 1'A','B','C','D','E','F'/
          C
          C       AOM(D/A SCROLL)
          D       INSTR=1001
0044              STATUS=MPCLB(BUS1+AOMPM,AOMBA,AOMSZ,INTGR,EVERY,LONG)
          D       IF(STATUS.NE.0)GOTO 10
          C
          C       AOM(D/A SCROLL)
0045              KNTRL=MONO+DAFIXP+EXT
0046              FREQ=1.0
          D       INSTR=2010
0047              STATUS=AOMID(AOMPM,FREQ,KNTRL,NOUT1,NOUT2,OFFSET)
          D       IF(STATUS.NE.0)GOTO 10
0048              TYPE 101
0049      101     FORMAT(1X,'     (5) AOM SETUP COMPLETE!')
0050              RETURN
          C
          C       DIAGNOSTIC TRAP AREA
          C
0051      10      TYPE 100,INSTR,STATUS
0052              CALL EXIT
0053      100     FORMAT(1X,'***MAP ERROR*** "AOM" INSTR=',I3,' HAS STATUS=',I6/)
0054              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000136	47	RW,I,CON,LCL
$IDATA	000156	55	RW,D,CON,LCL
$VARS	000240	80	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025556 5559

```
AOM.LP/LI:1=AOM/NOTR
```

```
        C        PROGRAM NAME:ADAM.FTN    ORIGINATED:28-APR-80
        C                                          UPDATE:20-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE ADAM
0002             REAL IOSBA,IOSSA
0003             INTEGER IOSPMB(600),PROGRM(494),WORD
0004             LOGICAL*1 CWORD(4),HEXC(16)
0005             INTEGER AOMPM,ADAMPM
0006             INTEGER DUAL,EXT,DAFLTP,DAFIXP,EXTT,P2DIS,P2EN
                 1 ,CCOFSR,CCOSSR,CC1FSR,CC1SSR,CC2FSR,CC2SSR
                 2 ,CC3FSR,CC3SSR,CHANS1(2),DDFFLT,DDFFIX,OFFSET
0007             INTEGER BUS1,BUS2,BUS3
0008             INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTER,CNVYES,CNVNO,UNITY,STATUS,SHORT
0009             INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AOPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,AROPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0010             INTEGER ORDR,START,SIZE,DSW,BTLTH
0011             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0012             COMMON/MTAPE0/NIN(300),NOUT(300)
0013             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0014             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0015             COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0016             COMMON/OVRO/Y(256),NP,XN
0017             COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0018             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0019             COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0020             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0021             COMMON/OVR2/IORDR(256),LTHM1
0022             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0023             COMMON/OVRF/XR(512)
0024             COMMON/OVRD/CSN,START,SIZE,YES,NO
0025             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AOPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,AROPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
0026               EQUIVALENCE (IOSPMB(1),XR(1)),(PROGRM(1),DCT2(1))
0027               DATA REAL,CMPLX,INTGR/0,1,2/
0028               DATA EVERY,SINGLE,DOUBLE/1,2,3/
0029               DATA BYTE2,BYTE4,BYTE8/2,4,8/
0030               DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0031               DATA BUS1,BUS2,BUS3/64,128,192/
0032               DATA MONO,DUAL,INT,EXT,DAFLTP,DAFIXP/0,1,0,2,0,4/
0033               DATA CC0FSR,CC0SSR,CC1FSR,CC1SSR/0,8,0,16/
0034               DATA CC2FSR,CC2SSR,CC3FSR,CC3SSR/0,32,0,64/
0035               DATA DDFFLT,DDFFIX/0,128/
0036               DATA OFFSET,INTT,EXTT/2,0,1/
0037               DATA P2DIS,P2EN/0,4/
0038               DATA CHANS1/1,-1/
0039               DATA IOSPMB,OVRHD/600*0,12./
0040               DATA IOSPM,IOS,IOSBA,IOSSA/61,2,22100.,0./
0041               DATA ADMPM,ADMBA,ADMSA,ADMSZ/62,22700.,0.,512./
0042               DATA ADAMPM,ADAMBA,ADAMSA,ADAMSZ/63,23300.,0.,512./
0043               DATA HEXC/'0','1','2','3','4','5','6','7','8','9',
                   1'A','B','C','D','E','F'/
           C
           C       ADAM(A/D SCROLL)
           D       INSTR=1002
0044               STATUS=MPCLB(BUS1+ADAMPM,ADAMBA,ADAMSZ,INTGR,EVERY,LONG)
           D       IF(STATUS.NE.0)GOTO 10
           C
           C       ADAM(A/D SCROLL)
0045               KNTRL=DDFFIX+CC0SSR+CC1FSR+CC2SSR+CC3SSR
0046               KNTRL=KNTRL+INTT+P2DIS
0047               FREQ=1.
0048               KNTRL=KNTRL+EXT
0049               FREQ1=FREQ
0050               FREQ2=FREQ
           D       INSTR=2005
0051               STATUS=ADMSD(ADAMPM,FREQ1,FREQ2,KNTRL,CHANS1,NIN1,NIN2)
           D       IF(STATUS.NE.0)GOTO 10
0052               TYPE 101
0053       101     FORMAT(1X,'      (6) ADAM SETUP COMPLETE!')
0054               RETURN
           C
           C       DIAGNOSTIC TRAP AREA
           C
0055       10      TYPE 100,INSTR,STATUS
0056               CALL EXIT
0057       100     FORMAT(1X,'***MAP ERROR*** "ADAM" INSTR=',I3,' HAS STATUS=',I6/)
0058               END
```

```
PROGRAM SECTIONS

 NAME       SIZE                    ATTRIBUTES

$CODE1  000176    63                RW,I,CON,LCL
$IDATA  000162    57                RW,D,CON,LCL
$VARS   000250    84                RW,D,CON,LCL
MTAPE0  002260   600                RW,D,OVR,GBL
MTAPE1  000012     5                RW,D,OVR,GBL
MTAPE2  000012     5                RW,D,OVR,GBL
OVRI    002020   520                RW,D,OVR,GBL
OVRO    002006   515                RW,D,OVR,GBL
OVRF    006016  1543                RW,D,OVR,GBL
OVRA    000152    53                RW,D,OVR,GBL
OVRP    002022   521                RW,D,OVR,GBL
OVRB    005004  1282                RW,D,OVR,GBL
OVR2    001002   257                RW,D,OVR,GBL
OVRL    000016     7                RW,D,OVR,GBL
OVRD    000012     5                RW,D,OVR,GBL
MAPDEF  000200    64                RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025632  5581
```

```
ADAM,LP/LI:1=ADAM/NOTR
```

```
      C        PROGRAM NAME:SYSTEM.FTN ORIGINATED:13-FEB-80
      C                                           UPDATE:22-MAY-80
      C
      C
      C
      C
0001           SUBROUTINE SYSTEM
0002           INTEGER ADM,ADAM,ADMPM,ADAMPM
0003           INTEGER EQ,GE,GT,SYNCNT,ANA,AFLG,SYN,SFLG,RDATA,XDATA
               1, SPIN,SPOUT,PARITY
0004           INTEGER ATCANA,ATCSYN,WAIT,STRTUP,TIMER
0005           INTEGER [illegible]S1,BUS2,BUS3
0006           INTEGER [illegible]L,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
               1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0007           INTEGER WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
               1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
               2,ORDCTM,YNM
               3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
               4,MIBIT,DCTM,OPRB,OTDCTM
               5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
               6,RECV1,RECV2,SEN1,SEN2
               7,AOPB,AOTDCT,AIBIT,SEN,INPB
               8,SYNB1,SYNB2,BF1,BF2,OUTB
               9,RECV,AROPB,AROUCT,RIBIT
               1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
0008           LOGICAL*1 BELL(5),ARRAY1(9),ARRAY2(8)
      C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0009           INTEGER ORDR,START,SIZE,DSW,BTLTH
0010           LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO,CR
0011           COMMON/MTAPE0/NIN(300),NOUT(300)
0012           COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTO
0013           COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0014           COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0015           COMMON/OVRO/Y(256),NP,XN
0016           COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0017           COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0018           COMMON/OVRP/M,G,VO,PVEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0019           COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0020           COMMON/OVP2/IORDR(256),LTHM1
0021           COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0022           COMMON/OVRE/XR(512)
0023           COMMON/OVRD/CSN,START,SIZE,YES,NO
0024           COMMON/MAPOFF/WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
               1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
               2,ORDCTM,YNM
               3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
               4,MIBIT,DCTM,OPRB,OTDCTM
               5,LINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
               6,RECV1,RECV2,SEN1,SEN2
               7,AOPB,AOTDCT,AIBIT,SEN,INPB
               8,SYNB1,SYNB2,BF1,BF2,OUTB
               9,RECV,AROPB,AROUCT,RIBIT
               1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
      C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0025           DATA REAL,CMPLX,INTGR/0,1,2/
0026           DATA EVERY,SINGLE,DOUBLE/1,2,3/
```

```
                DATA BYTE2,BYTE4,BYTE8/2,4,8/
                DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
                DATA BUS1,BUS2,BUS3/64,128,192/
                DATA ATCANA,ATCSYN,WAIT,STRTUP,TIMER/1,2,3,4,7/
                DATA IOSTD,AOM,ADAM/16,22,23/
                DATA EQ,NE,GE,GT,LE,LT/0,1,2,3,4,5/
                DATA INIT,PARITY/113,114/
                DATA CR,BELL/040,5*07/
      C
      C         EXECUTE PRE-BOUND FUNCTION LIST
      C
                IF(RETIM.EQ.NO.AND.LIVE.EQ.NO)RETURN
                IF(TIMING.EQ.NO)GO TO 1
      C>>>>>>> FOR TIMING PURPOSES ONLY!
      D         INSTR=0
                STATUS=MPWHL(PARITY,EQ,TIMER)
      D         IF(STATUS.NE.0)GO TO 10
                CALL DATE(ARRAY1)
                CALL TIME(ARRAY2)
      3         TYPE 20,BELL,ARRAY1,ARRAY2
      20        FORMAT(//1X,'PDP <==> MAP LINKAGE HAS BEEN SUSPENDED',5A1,/
                1,4X,9A1,' AT ',8A1/
                2,1X,'TYPE "RES ATCA" TO TERMINATE'//)
                CALL SUSPND
                GO TO 1000
      C>>>>>>> ANALYZER
      1         IF(LIVE.EQ.YES)GO TO 2
      D         INSTR=10
                STATUS=MPXEL(ATCANA)
      D         IF(STATUS.NE.0)GO TO 10
      C>>>>>>> SYNTHESIZER
      D         INSTR=11
                STATUS=MPXEL(ATCSYN)
      D         IF(STATUS.NE.0)GO TO 10
                RETURN
      C>>>>>>> REAL-TIME LIVE SYSTEM
      2         CONTINUE
      D         INSTR=2090
                STATUS=MPWHL(PARITY,EQ,WAIT)
      D         IF(STATUS.NE.0)GO TO 10
                CALL DATE(ARRAY1)
                CALL TIME(ARRAY2)
                TYPE 20,BELL,ARRAY1,ARRAY2
                CALL SUSPND
                INSTR=8002
                STATUS=MPSAA(AOM)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=8003
                STATUS=MPSAA(ADAM)
      D         IF(STATUS.NE.0)GO TO 10
      1000      CONTINUE
      D         INSTR=8004
                STATUS=MPDEL(ATCANA)
      D         IF(STATUS.NE.0)GO TO 10
      D         INSTR=8005
                STATUS=MPDEL(ATCSYN)
```

```
0074     D        IF(STATUS.NE.0)GO TO 10
0075     D        INSTR=8006
0076              STATUS=MPDEL(AAIT)
0077     D        IF(STATUS.NE.0)GO TO 10
0078     D        INSTR=8007
0079              STATUS=MPDEL(STRTUP)
0080     D        IF(STATUS.NE.0)GO TO 10
0081     D        INSTR=8008
0082              STATUS=MPCLS(0)
0083     D        IF(STATUS.NE.0)GO TO 10
0084              STOP
         C
         C        DIAGNOSTIC TRAP AREA
         C
0085     10       TYPE 100,INSTR,STATUS
0086              CALL EXIT
0087     100      FORMAT(1X,'***MAP ERROR*** "SYSTEM" INSTR=',I4,' HAS STATUS=',I6/)
0088              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000670	220	RW,I,CON,LCL
$PDATA	000004	2	RW,D,CON,LCL
$IDATA	000346	115	RW,D,CON,LCL
$VARS	000170	60	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000174	62	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026430   5772

NO FPP INSTRUCTIONS GENERATED

SYSTMA,LP/LI:1=SYSTMA/DE/NOTR

```
        C       PROGRAM NAME:INPUT.FTN  ORIGINATED:05-FEB-79
        C                                             UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE INPUT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,1ST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       DATA INPUT
```

```
             C
0026                 IF(ICOUNT)5012,5010,5012
0027         5010    NTOTI=NP
0028                 NTUPS=NP
0029                 CALL TAPE2(8)
0030                 CALL TAPE2(1)
0031         D       INSTR=1
0032                 STATUS=MPWDB(NINM,NIN(1),BYTE2,CNVYES,NIN(NP))
0033         D       IF(STATUS.NE.0)GO TO 10
0034         D       INSTR=2
0035                 STATUS=VFLT(XOM,39,NINM,0)
0036         D       IF(STATUS.NE.0)GO TO 10
0037                 NTOTI=LTH-NP
0038                 NTUPS=LTH-NP
0039         5012    CALL TAPE2(1)
0040                 IF(NEND.NE.0)GO TO 5000
0041                 IF(NERR.NE.0)GO TO 5000
0042                 ICOUNT=ICOUNT+1
0043                 IF(ICOUNT.LT.START)GO TO 5012
0044                 IF((ICOUNT-START+1).GT.IREC)GO TO 5000
0045         D       WRITE(6,899)ICOUNT
0046         D899    FORMAT(1H1,40(1H>),'TRANSMIT FRAME # ',I4,40(1H<))
0047         D       INSTR=11
0048                 STATUS=MPWDB(NINM,NIN(1),BYTE2,CNVYES,NIN(NTOTI))
0049         D       IF(STATUS.NE.0)GO TO 10
0050         D       INSTR=13
0051                 STATUS=VMOV(INPB,NINM)
0052         D       IF(STATUS.NE.0)GO TO 10
0053         D       INSTR=12
0054                 STATUS=MPRDB(NINM,NIN(1),BYTE2,CNVYES,NIN(NTOTI))
0055                 IF(STATUS.NE.0)GO TO 10
0056         D       WRITE(6,913)(I,NIN(I),I=1,NTOTI)
0057         D913    FORMAT(/1X,4(1X,'NIN(',I3,')=',I6,2X))
0058                 RETURN
             C
             C       EOF ON INPUT,WRITE EOF
             C
0059         5000    NEND=1
0060                 RETURN
             C
             C       DIAGNOSTIC TRAP AREA
             C
0061         10      TYPE 101,INSTR,STATUS
0062                 CALL EXIT
0063         101     FORMAT(1X,'***MAP ERROR*** "INPUT" INSTR=',I3,' HAS STATUS=',I6/)
0064                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000622	201	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000230	76	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL

```
MTAPE0   002260   600            RW,D,OVR,GBL
MTAPE1   000012     5            RW,D,OVR,GBL
MTAPE2   000012     5            RW,D,OVR,GBL
OVRI     002020   520            RW,D,OVR,GBL
OVRO     002006   515            RW,D,OVR,GBL
OVRF     006016  1543            RW,D,OVR,GBL
OVRA     000152    53            RW,D,OVR,GBL
OVRP     002022   521            RW,D,OVR,GBL
OVRB     005004  1282            RW,D,OVR,GBL
OVR2     001002   257            RW,D,OVR,GBL
OVRL     000016     7            RW,D,OVR,GBL
OVRD     000012     5            RW,D,OVR,GBL
MAPDEF   000200    64            RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026150  5684

INPUT.LP/LI:1=INPUT/DE/NOTR
```

```
        C       PROGRAM NAME:DCVAR.FTN  ORIGINATED:05-FEB-79
        C                                        UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE DCVAR
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            INTEGER QTDCT,QTPAR,QTDC,QTVAR,QTM,QTG,QTV
0007            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008            COMMON/MTAPE0/NIN(300),NOUT(300)
0009            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0010            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/OVRQT/QTDCT(256),QTPAR(8),QTDC,QTVAR,QTM,QTG,QTV
0022            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0023            DATA REAL,CMPLX,INTGR/0,1,2/
0024            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0025            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0026            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0027            DATA BUS1,BUS2,BUS3/64,128,192/
```

```
        C
        C       SERIALIZE SIDEBAND W/ 1 FRAME DELAY ...
        C       AND MAINBAND W/ 2 FRAMES DELAY
        C
0028    D       INSTR=0
0029            STATUS=VMOV1(AQPB)
0030    D       IF(STATUS.NE.0)GOTO 10
        C
        C       CALCULATE DC & VAR THEN NORMALIZE & COMPRESS
        C
0031    D       INSTR=1
0032            STATUS=MPFDVM(MAPX,NINM,X2)
0033    D       IF(STATUS.NE.0)GO TO 10
        C
        C       RESULTS
        C
0034    D       CALL MPRDB(NINM,NIN(1),BYTE2,CNVYES,NIN(NTOTI))
0035    D       WRITE(6,898)(I,NIN(I),I=1,NTOTI)
0036    D898    FORMAT(/1X,4(1X,'NIN(',I3,')=',I6,2X))
0037    D       CALL MPRST(52,DCBIAS,1,CNVYES)
0038    D       WRITE(6,900)DCBIAS
0039    D900    FORMAT(/1X,'DCBIAS=',F12.5)
0040    D       CALL MPRST(55,VAR,1,CNVYES)
0041    D       WRITE(6,901)VAR
0042    D901    FORMAT(/1X,'VAR=',F12.5)
0043    D       CALL MPRDB(X2,XR(1),BYTE4,CNVYES,XR(LTH))
0044    D       WRITE(6,920)(I,XR(I),I=1,LTH)
0045    D920    FORMAT(/1X,4(1X,'XR(',I3,')=',E15.8,2X))
0046    D       CALL MPRDB(COSZ,XR(1),BYTE4,CNVYES,XR(LTH))
0047    D       WRITE(6,922)(I,XR(I),I=1,LTH)
0048    D922    FORMAT(/1X,4(1X,'COS(',I3,')=',E15.8,2X))
0049    D       CALL MPRDB(AQPB,NOUT(1),BYTE2,CNVYES,NOUT(LPARM))
0050    D       WRITE(6,899)(I,NOUT(I),I=1,LPARM)
0051    D899    FORMAT(/1X,4(1X,'AQPB(',I3,')=',I6,2X))
0052    D       CALL MPRDB(AQTDCT,QTDCT(1),BYTE2,CNVYES,QTDCT(LTH))
0053    D       WRITE(6,910)(I,QTDCT(I),I=1,LTH)
0054    D910    FORMAT(/1X,4(1X,'AQTDCT(',I3,')=',I6,2X))
0055    D       CALL MPRDB(AIBIT,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0056    D       WRITE(6,911)(I,IBIT(I),I=1,LTH)
0057    D911    FORMAT(/1X,4(1X,'AIBIT(',I3,')=',I6,2X))
0058    D       CALL MPRDB(MIBIT4,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0059    D       WRITE(6,902)(I,IBIT(I),I=1,LTH)
0060    D902    FORMAT(/1X,4(1X,'MIBIT4(',I3,')=',I6,2X))
0061            RETURN
        C
        C       DIAGNOSTIC TRAP AREA
        C
0062    10      TYPE 101,INSTR,STATUS
0063            CALL EXIT
0064    101     FORMAT(1X,'***MAP ERROR*** "DCVAR" INSTR=',I4,' HAS STATUS=',I6/)
0065            END
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES

```
$CODE1   001334   366     RW,I,CON,LCL
$PDATA   000014     6     RW,D,CON,LCL
$IDATA   000576   191     RW,D,CON,LCL
$VARS    000054    22     RW,D,CON,LCL
MTAPE0   002260   600     RW,D,OVR,GBL
MTAPE1   000012     5     RW,D,OVR,GBL
MTAPE2   000012     5     RW,D,OVR,GBL
OVRI     002020   520     RW,D,OVR,GBL
OVRD     002006   515     RW,D,OVR,GBL
OVRF     006016  1543     RW,D,OVR,GBL
OVRA     000152    53     RW,D,OVR,GBL
OVRP     002022   521     RW,D,OVR,GBL
OVRB     005004  1282     RW,D,OVR,GBL
OVR2     001002   257     RW,D,OVR,GBL
OVRL     000016     7     RW,D,OVR,GBL
OVRD     000012     5     RW,D,OVR,GBL
OVROT    001032   269     RW,D,OVR,GBL
MAPDEF   000200    64     RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 030256  6231

DCVAR,LP/LI:1=DCVAR/DE/NOTR
```

```
        C       PROGRAM NAME:DCTF.FTN   ORIGINATED:05-FEB-79
        C                                         UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE DCTF
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                1,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,AROCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,V0,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,AROCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       DCT FORWARD N REAL--->N/2 COMPLEX
```

```
           C        APPLY POST WEIGHTING W/ HALF SAMPLE DELAY
           C
0026       D        INSTR=1
0027                STATUS=FFTN(X2,1,X2,VSHRT,WORK)
0028       D        IF(STATUS.NE.0)GO TO 10
0029       D        INSTR=2
0030                STATUS=MPDCTM(DCTM,X1,X2,COSZ)
0031       D        IF(STATUS.NE.0)GO TO 10
0032       D        INSTR=3
0033                STATUS=FFTIN(X1,1,X1,VLONG,WORK)
0034       D        IF(STATUS.NE.0)GO TO 10
           C
           C        RESULTS
           C
0035       D        CALL MPRDB(DCTM,DCT(1),BYTE4,CNVYES,DCT(LTH))
0036       D        WRITE(6,902)(I,DCT(I),I=1,LTH)
0037       D902     FORMAT(/1X,4(1X,'DCT(',I3,')=',E15.8,2X))
0038                RETURN
           C
           C        DIAGNOSTIC TRAP AREA
           C
0039       10       TYPE 100,INSTR,STATUS
0040                CALL EXIT
0041       100      FORMAT(1X,'***MAP ERROR*** "DCTF" INSTR=',I3,' HAS STATUS=',I6/)
0042                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000320	104	RW,I,CON,LCL
SPDATA	000004	2	RW,D,CON,LCL
SIDATA	000176	63	RW,D,CON,LCL
SVARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025600 5568

DCTF.LP/LI:1=DCTF/DE/NOTR

```
.....FILLC.FTN ORIGINATED:05-FEB-79
                           UPDATE:09-MAY-80


     .... FILLPC
     .. ...1,BUS2,BUS3
     ... REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
     ...4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
     ... WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
     ..2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
     ...TM,YNM,ORPRM,ORDCTM
     .....COSZ,VSHRT,YOM,XOM,MAPX
     . MIBIT,DCTM,OPRM,OTDCTM
     .NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
     . RECV1,RECV2,SEN1,SEN2
     ....,AOTDCT,AIBIT,SEN,INPB
     ..SYNB1,SYNB2,BF1,BF2,OUTB
     ..RECV,AROPB,AROOCT,RIBIT
     1..CT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     INTEGER ORDR,START,SIZE,DSW,BTLTH
     LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
     COMMON/MTAPE0/NIN(300),NOUT(300)
     COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
     COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
     COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
     COMMON/OVRO/Y(256),NP,XN
     COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
     COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
     COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
     COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
     COMMON/OVR2/IORDR(256),LTHM1
     COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
     COMMON/OVRF/XR(512)
     COMMON/OVRD/CSN,START,SIZE,YES,NO
     COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
     1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
     2,DRDCTM,YNM,ORPRM,ORDCTM
     3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
     4,MIBIT,DCTM,OPRM,OTDCTM
     5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
     6,RECV1,RECV2,SEN1,SEN2
     7,AOPB,AOTDCT,AIBIT,SEN,INPB
     8,SYNB1,SYNB2,BF1,BF2,OUTB
     9,RECV,AROPB,AROOCT,RIBIT
     1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
C    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     DATA REAL,CMPLX,INTGR/0,1,2/
     DATA EVERY,SINGLE,DOUBLE/1,2,3/
     DATA BYTE2,BYTE4,BYTE8/2,4,8/
     DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
     DATA BUS1,BUS2,BUS3/64,128,192/
C
C    GET PITCH & PITCH GAIN
```

```
             C
0026    D         INSTR=1
0027              STATUS=MPMWGX(R1,NINM,KE,A1)
0028    D         IF(STATUS.NE.0)GO TO 10
             C
             C    RESULTS
             C
0029    D         CALL MPRDB(R1,R(1),BYTE4,CNVYES,R(LPCN+1))
0030    D         WRITE(6,903)(I,R(I),I=1,LPCN+1)
0031    D903      FORMAT(/1X,5(1X,'R(',I2,')=',E15.8,2X))
0032    D         CALL MPRST(61,G,1,CNVYES)
0033              CALL MPRST(62,XM,1,CNVYES)
0034    D         CALL MPRST(63,VU,1,CNVYES)
0035              M=IFIX(XM)
0036              IVUV=IFIX(VU)
0037    D         WRITE(6,904)M,G,IVUV
0038    D904      FORMAT(/1X,'M=',I6,', G=',E15.8,' AND VUV=',I1)
0039    D         CALL MPRDB(A1,A(1),BYTE4,CNVYES,A(LPCN))
0040    D         WRITE(6,906)(I,A(I),I=1,LPCN)
0041    D906      FORMAT(/1X,4(1X,'A(',I2,')=',E15.8,2X))
0042              CALL MPRDB(KE,PARCOR(1),BYTE4,CNVYES,PARCOR(LPCN))
0043    D         WRITE(6,907)(I,PARCOR(I),I=1,LPCN)
0044    D907      FORMAT(/1X,4(1X,'K(',I2,')=',E15.8,2X))
0045              DO 1 I=2,LPCN
0046              IF(PARCOR(I).NE.0.0)GO TO 1
0047              WRITE(4,908)ICOUNT,PARCOR(1)
0048    2         TYPE 908,ICOUNT,PARCOR(1)
0049    908       FORMAT(//1X,'***WARNING*** BAD PARCORS IN FRAME ',I4
                  1,' W/ K(1)=',E15.8/)
0050              RETURN
0051    1         CONTINUE
0052              RETURN
             C
             C    DIAGNOSTIC TRAP AREA
             C
0053    10        TYPE 100,INSTR,STATUS
0054              CALL EXIT
0055    100       FORMAT(1X,'***MAP ERROR*** "PITLPC" INSTR=',I3,' HAS STATUS=',I6/)
0056              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001000	256	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000446	147	RW,D,CON,LCL
$VARS	000062	25	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL

```
PITLPC.FTN       /DE/WR

OVRA     000152    53          RW,D,OVR,GBL
OVRP     002027   521          RW,D,OVR,GBL
OVRB     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRD     000012     5          RW,D,OVR,GBL
$APDFF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026556  5815

PITLPC,LP/LI:1=PITLPC/DE/NOTR
```

```
      C       PROGRAM NAME:QDPARM.FTN ORIGINATED:10-SEP-79
      C                                         UPDATE:12-MAY-80
      C
      C
      C
      C
0001          SUBROUTINE QDPARM
0002          INTEGER BUS1,BUS2,BUS3
0003          INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
              1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004          INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
              1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
              2,DRDCTM,YNM,QRPRM,QRDCTM
              3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
              4,MIBIT,DCTM,QPRM,QTDCTM
              5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
              6,RECV1,RECV2,SEN1,SEN2
              7,AQPB,AQTDCT,AIBIT,SEN,INPB
              8,SYNB1,SYNB2,BF1,BF2,OUTB
              9,RECV,ARQPB,ARQDCT,RIBIT
              1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
      C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005          INTEGER ORDR,START,SIZE,DSW,BTLTH
0006          INTEGER QTDCT,QTPAR,QTDC,QTVAR,QTM,QTG,QTV
0007          LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008          COMMON/MTAPE0/NIN(300),NOUT(300)
0009          COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0010          COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011          COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IRFC,NSKIPS
0012          COMMON/OVRO/Y(256),NP,XN
0013          COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014          COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015          COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016          COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017          COMMON/OVR2/IORDR(256),LTHM1
0018          COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019          COMMON/OVRF/XR(512)
0020          COMMON/OVRD/CSN,START,SIZE,YES,NO
0021          COMMON/OVRQT/QTDCT(256),QTPAR(8),QTDC,QTVAR,QTM,QTG,QTV
0022          COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0023          COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
              1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
              2,DRDCTM,YNM,QRPRM,QRDCTM
              3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
              4,MIBIT,DCTM,QPRM,QTDCTM
              5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
              6,RECV1,RECV2,SEN1,SEN2
              7,AQPB,AQTDCT,AIBIT,SEN,INPB
              8,SYNB1,SYNB2,BF1,BF2,OUTB
              9,RECV,ARQPB,ARQDCT,RIBIT
              1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
      C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0024          DATA REAL,CMPLX,INTGR/0,1,2/
0025          DATA EVERY,SINGLE,DOUBLE/1,2,3/
0026          DATA BYTE2,BYTE4,BYTE8/2,4,8/
0027          DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
```

```
0028            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       QUANTIZE PARCOR(J),J=1,...,LPCN,DCBIAS,VAR,G,M
        C       DEQUANTIZE PARCOR(J),J=1,...,LPCN,G,M
        C
0029    D       INSTR=1
0030            STATUS=MPODPP(OPRM,PARC,A1)
0031    D       IF(STATUS.NE.0)GO TO 10
        C
        C       RESULTS
        C
0032    D       CALL MPRDB(OPRM,NOUT(1),BYTE2,CNVYES,NOUT(LPARM+1))
0033    D       DO 651 I=1,LPCN
0034    D651    QTPAR(I)=NOUT(I)
0035    D       WRITE(6,924)(J,QTPAR(J),J=1,LPCN)
0036    D924    FORMAT(/1X,4(1X,'QTK(',I2,')=',I6,2X))
0037    D       QTVAR=NOUT(LPCN+1)
0038    D       WRITE(6,926)QTVAR
0039    D926    FORMAT(/1X,'QTVAR=',I6)
0040    D       QTDC=NOUT(LPCN+2)
0041    D       WRITE(6,925)QTDC
0042    D925    FORMAT(/1X,'QTDC=',I6)
0043    D       QTM=NOUT(LPCN+3)
0044    D       QTV=NOUT(LPCN+4)
0045    D       QTG=NOUT(LPCN+5)
0046    D       WRITE(6,927)QTM,QTG,QTV
0047    D927    FORMAT(/1X,'QTM=',I6,', QTG=',I6,' AND QTV=',I6)
0048    D       CALL MPRDB(PARC,DTPAR(1),BYTE4,CNVYES,DTPAR(LPCN))
0049    D       WRITE(6,928)(J,DTPAR(J),J=1,LPCN)
0050    D928    FORMAT(/1X,4(1X,'DTK(',I2,')=',F15.8,2X))
0051    D       CALL MPRST(88,DTDC,1,CNVYES)
0052    D       CALL MPRST(89,DTVAR,1,CNVYES)
0053    D       CALL MPRST(90,DTG,1,CNVYES)
0054    D       CALL MPRST(91,DTM,1,CNVYES)
0055    D       CALL MPRST(104,DTV,1,CNVYES)
0056    D       WRITE(6,929)DTM,DTG,DTV
0057    D929    FORMAT(/1X,'DTM=',E15.8,' DTG=',F15.8,' AND DTV=',E15.8)
0058    D       CALL MPRDB(A1,DTA(1),BYTE4,CNVYES,DTA(LPCN))
0059    D       WRITE(6,931)(J,DTA(J),J=1,LPCN)
0060    D931    FORMAT(/1X,4(1X,'DTA(',I2,')=',E15.8,2X))
0061    D       CALL MPRST(60,ENG,1,CNVYES)
0062    D       WRITE(6,932)ENG
0063    D932    FORMAT(/1X,'ENG=',F15.8)
0064            RETURN
        C
        C       DIAGNOSTIC TRAP AREA
        C
0065    10      TYPE 100,INSTR,STATUS
0066            CALL EXIT
0067    100     FORMAT(1X,'***MAP ERROR*** "QDPARM" INSTR=',I3,' HAS STATUS=',I6/)
0068            END
```

PROGRAM SECTIONS

NAME SIZE ATTRIBUTES

```
$CODE1   001102   289          RW,I,CON,LCL
$PDATA   000034    14          RW,D,CON,LCL
$IDATA   000530   172          RW,D,CON,LCL
$VARS    000056    23          RW,D,CON,LCL
*TAPE0   002260   600          RW,D,OVR,GBL
*TAPE1   000012     5          RW,D,OVR,GBL
*TAPE2   000012     5          RW,D,OVR,GBL
OVRI     002020   520          RW,D,OVR,GBL
OVRD     002006   515          RW,D,OVR,GBL
OVRF     006016  1543          RW,D,OVR,GBL
OVRA     000152    53          RW,D,OVR,GBL
OVRP     002022   521          RW,D,OVR,GBL
OVRB     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRD     000012     5          RW,D,OVR,GBL
OVROT    001032   269          RW,D,OVR,GBL
OVRDT    002124   554          RW,D,OVR,GBL
*APDFF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 032124  6698

NO FPP INSTRUCTIONS GENERATED

ODPARM,LP/LI:1=ODPARM/DE/NOTR
```

```
        C       PROGRAM NAME:VPBS.FTN    ORIGINATED:05-FEB-79
        C                                  UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE VPBS
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTD
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       FORM VOCAL TRACT VPBS SPECTRUM
```

```
          C
0026      D         INSTR=1
0027                STATUS=MPFSTV(A1)
0028      D         IF(STATUS.NE.0)GO TO 10
0029      D         INSTR=2
0030                STATUS=FF2R(X1,4,X1,VLONG,WORK)
0031      D         IF(STATUS.NE.0)GO TO 10
          C
          C         RESULTS
          C
0032      D         CALL MPRST(73,FGR,1,CNVYES)
0033      D         WRITE(6,906)K,EGR
0034      D906      FORMAT(/1X,'K=',I3,' AND FGR=',E15.8)
0035                RETURN
          C
          C         DIAGNOSTIC TRAP AREA
          C
0036      10        TYPE 100,INSTR,STATUS
0037                CALL EXIT
0038      100       FORMAT(1X,'***MAP ERROR*** "VPBS" INSTR=',I3,' HAS STATUS=',I6/)
0039                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000206	67	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000150	52	RW,D,CON,LCL
$VARS	000060	24	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDFF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025454   5526

VPBS,LP/LI:1=VPBS/DE/NOTR

```
        C       PROGRAM NAME:BASIS.FTN  ORIGINATED:05-FEB-79
        C                                         UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE BASIS
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       GENERATE BASIS FROM VOCAL TRACT & PITCH SPECTRA
```

```
              C
0026    D           INSTR=1
0027                STATUS=MPBASP(DCTM1,PWEITM,TMP2,X1)
0028    D           IF(STATUS.NE.0)GO TO 10
              C
              C     RESULTS
              C
0029    D           CALL MPRDB(DCTM1,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0030    D           WRITE(6,908)(I,DCT1(I),I=1,LTH)
0031    D908        FORMAT(/1X,4(1X,'DCT1(',I3,')=',E15.8,2X))
0032    D           CALL MPRDB(TMP2,DCT2(1),BYTE4,CNVYES,DCT2(LTH))
0033    D           WRITE(6,909)(I,DCT2(I),I=1,LTH)
0034    D909        FORMAT(/1X,4(1X,'DCT2(',I3,')=',E15.8,2X))
0035                RETURN
              C
              C     DIAGNOSTIC TRAP AREA
              C
0036    10          TYPE 100,INSTR,STATUS
0037                CALL EXIT
0038    100         FORMAT(1X,'***MAP ERROR*** "BASIS" INSTR=',I3,' HAS STATUS=',I6/)
0039                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000350	116	RW,I,CON,LCL
$IDATA	000216	71	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDFF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025644   5586

NO FPP INSTRUCTIONS GENERATED

BASIS,LP/LI:1=BASIS/DE/NOTR

```
        C       PROGRAM NAME:ASRT.FTN    ORIGINATED:10-SEP-79
        C                                          UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE ASRT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWFITM,DCTM1
                1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWFITM,DCTM1
                1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       ORDER DCT1(.) MAX-TO-MIN W/ ORDERING INFO IN IORDR(.)
```

```
            C        SORT TMP2 TO DCTM2
            C        SORT DCTM TO TMP3
            C        SORT DCTM1 TO TMP4
            C
0026        D        INSTR=1
0027                 STATUS=MPASRT(IORDRM)
0028        D        IF(STATUS.NE.0)GO TO 10
            C
            C        RESULTS
            C
0029        D        CALL MPRDB(DCTM2,DCT2(1),BYTE4,CNVYES,DCT2(LTH))
0030        D        CALL MPRDB(IORDRM,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0031        D        WRITE(6,910)(I,IORDR(I),I=1,LTH)
0032        D910     FORMAT(/1X,6(1X,'IORDR(',I3,')=',I3,2X))
0033        D        WRITE(6,909)(I,DCT2(I),I=1,LTH)
0034        D909     FORMAT(/1X,4(1X,'DCT2(',I3,')=',E15.8,2X))
0035        D        CALL MPRDB(TMP4,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0036        D        WRITE(6,911)(I,DCT1(I),I=1,LTH)
0037        D911     FORMAT(/1X,4(1X,'DCT1T(',I3,')=',E15.8,2X))
0038        D        CALL MPRDB(TMP3,DCT(1),BYTE4,CNVYES,DCT(LTH))
0039        D        WRITE(6,912)(I,DCT(I),I=1,LTH)
0040        D912     FORMAT(/1X,4(1X,'DCTT(',I3,')=',E15.8,2X))
0041                 RETURN
            C
            C        DIAGNOSTIC TRAP AREA
            C
0042        10       TYPE 100,INSTR,STATUS
0043                 CALL EXIT
0044        100      FORMAT(1X,'***MAP ERROR*** "ASRT" INSTR=',I3,' HAS STATUS=',I6/)
0045                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000620	200	RW,I,CON,LCL
$IDATA	000324	106	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVR1	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026222 5705

```
NO FPP INSTRUCTIONS GENERATED

ASRT,LP/LI:1=ASRT/DE/NOTR
```

```
        C       PROGRAM NAME:SDCT.FTN    ORIGINATED:05-FEB-79
        C                                          UPDATE:24-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE SDCT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCTIT1,DCTIT2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCTIT1,DCTIT2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA TMP3,TMP4/46,47/
0022            DATA REAL,CMPLX,INTGR/0,1,2/
0023            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026            DATA BUS1,BUS2,BUS3/64,128,192/
        C
```

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

9 OF 9

AD
A091663

END
DATE
FILMED
12-80
DTIC

```
        C       SCAN DCT LIST FOR ALL SIGNIFICANT POINTS
        C       PRELIMINARY BIT ASSIGNMENT TO TMP1
        C
0027    D       INSTR=1
0028            STATUS=MPSCAN(DCTM2)
0029    D       IF(STATUS.NE.0)GO TO 10
        C
        C       RESULTS
        C
0030    D       CALL MPRST(R0,CTEMP,1,CNVYES)
0031    D       CALL MPRST(R2,BITSMN,1,CNVYES)
0032    D       CALL MPRST(R3,XITOT,1,CNVYES)
0033    D       ITOT=IFIX(XITOT)
0034    D       WRITE(6,930)BITSMN,CTEMP,ITOT
0035    D930    FORMAT(/1X,'BITS= ',F15.8,' W/ CTEMP=',F15.8,' @ ITOT= ',I3)
0036    D       CALL MPRDB(TMP1,XR(1),BYTE4,CNVYES,XR(LTH))
0037    D       WRITE(6,911)(I,IFIX(XR(I)),I=1,LTH)
0038    D911    FORMAT(/1X,6(1X,'IBIT(',I3,')=',I3,2X))
0039            RETURN
        C
        C       DIAGNOSTIC TRAP AREA
        C
0040    10      TYPE 100,INSTR,STATUS
0041            CALL EXIT
0042    100     FORMAT(1X,'***MAP ERROR*** "SDCT" INSTR=',I3,' HAS STATUS=',I6/)
0043            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000344	114	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000250	84	RW,D,CON,LCL
$VARS	000064	26	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025722 5609

SDCT,LP/LI:1=SDCT/DE/NOTR

```
FORTRAN IV-PLUS V02-51E          13:30:46     09-AUG-80          PAGE 1
BITA.FTN          /DE/WR

        C       PROGRAM NAME:BITA.FTN   ORIGINATED:05-FEB-79
        C                                       UPDATE:23-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE BITA
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/SECTIO/MODUL
0008            COMMON/MTAPE0/NIN(300),NOUT(300)
0009            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0010            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022            DATA TMP3,TMP4/46,47/
0023            DATA REAL,CMPLX,INTGR/0,1,2/
0024            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0025            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0026            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0027            DATA BUS1,BUS2,BUS3/64,128,192/
```

```
FORTRAN IV-PLUS V02-51E          13:30:46     09-AUG-80            PAGE 2
BITA.FTN            /DE/WR

         C
         C        ASSIGN BITS TO DCT ELEMENTS
         C
         C        ANALYSIS:MODUL=1,SYNTHESIS:MODUL=2
0028     D        INSTR=1
0029              IF(MODUL.EQ.1)STATUS=MPCDBA(MIBIT3)
0030              IF(MODUL.EQ.2)STATUS=MPCDBA(MIBIT1)
0031     D        IF(STATUS.NE.0)GO TO 10
         C
         C        RESULTS
         C
0032     D        IF(MODUL.EQ.1)CALL MPRDB(MIBIT3,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0033     D        IF(MODUL.EQ.2)CALL MPRDB(MIBIT1,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0034     D        IF(STATUS.NE.0)GO TO 10
0035     D        WRITE(6,911)(I,IBIT(I),I=1,LTH)
0036     D911     FORMAT(/1X,6(1X,'IBIT(',I3,')=',I3,2X))
0037              RETURN
         C
         C        DIAGNOSTIC TRAP AREA
         C
0038     10       TYPE 100,INSTR,STATUS
0039              CALL EXIT
0040     100      FORMAT(1X,'***MAP ERROR*** "BITA" INSTR=',I3,' HAS STATUS=',I6/)
0041              END

PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

SCODE1   000334   110           RW,I,CON,LCL
SIDATA   000160    56           RW,D,CON,LCL
SVARS    000054    22           RW,D,CON,LCL
SECTIO   000002     1           RW,D,OVR,GBL
MTAPE0   002260   600           RW,D,OVR,GBL
MTAPE1   000012     5           RW,D,OVR,GBL
MTAPE2   000012     5           RW,D,OVR,GBL
OVRI     002020   520           RW,D,OVR,GBL
OVRO     002006   515           RW,D,OVR,GBL
OVRF     006016  1543           RW,D,OVR,GBL
OVRA     000152    53           RW,D,OVR,GBL
OVRP     002022   521           RW,D,OVR,GBL
OVRR     005004  1282           RW,D,OVR,GBL
OVR2     001002   257           RW,D,OVR,GBL
OVRL     000016     7           RW,D,OVR,GBL
OVRD     000012     5           RW,D,OVR,GBL
MAPDEF   000200    64           RW,D,OVR,GBL

TOTAL SPACE AL  )CATED = 025574  5566

NO FPP INSTRUCTIONS GENERATED

BITA,LP/LI:1=BITA/DE/NOTR
```

```
        C       PROGRAM NAME:QDCT.FTN    ORIGINATED:10-SEP-79
        C                                      UPDATE:23-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE QDCT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            INTEGER QTDCT,QTPAR,QTDC,QTVAR,QTM,QTG,QTV
0007            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008            COMMON/MTAPE0/NIN(300),NOUT(300)
0009            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0010            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/OVRQT/QTDCT(256),QTPAR(8),QTDC,QTVAR,QTM,QTG,QTV
0022            COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0023            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0024            DATA REAL,CMPLX,INTGR/0,1,2/
0025            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0026            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0027            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
```

```
0028              DATA BUS1,BUS2,BUS3/64,128,192/
         C
         C        QUANTIZE DCT(.) W/ IBIT(.) ACCORDING TO IORDR(.)
         C
0029     D        INSTR=1
0030              STATUS=MPESTQ(QTDCTM,MIBIT3,TMP1,TMP2)
0031     D        IF(STATUS.NE.0)GO TO 10
         C
         C        RESULTS
         C
0032     D        DO 21 I=1,LTH
0033     D21      IORDR(I)=IORDR(I)+1
0034     D        CALL MPRDB(QTDCTM,NOUT(1),BYTE2,CNVYES,NOUT(LTH))
0035     D        DO 22 I=1,LTH
0036     D        J=IORDR(I)
0037     D        QTDCT(J)=NOUT(I)
0038     D22      NOUT(I)=0
0039     D        WRITE(6,912)(I,QTDCT(I),I=1,LTH)
0040     D912     FORMAT(/1X,4(1X,'QTDCT(',I3,')=',I3,2X))
0041     D        CALL MPRDB(MIBIT3,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0042     D        WRITE(6,913)(I,IBIT(I),I=1,LTH)
0043     D913     FORMAT(/1X,4(1X,'MIBIT3(',I3,')=',I3,2X))
0044              RETURN
         C
         C        DIAGNOSTIC TRAP AREA
         C
0045     10       TYPE 100,INSTR,STATUS
0046              CALL EXIT
0047     100      FORMAT(1X,'***MAP ERROR*** "QDCT" INSTR=',I3,' HAS STATUS=',I6/)
0048              END


PROGRAM SECTIONS

 NAME       SIZE               ATTRIBUTES

$CODE1   000446    147         RW,I,CON,LCL
$IDATA   000216     71         RW,D,CON,LCL
$VARS    000056     23         RW,D,CON,LCL
MTAPE0   002260    600         RW,D,OVR,GBL
MTAPE1   000012      5         RW,D,OVR,GBL
MTAPE2   000012      5         RW,D,OVR,GBL
OVRI     002020    520         RW,D,OVR,GBL
OVRO     002006    515         RW,D,OVR,GBL
OVRF     006016   1543         RW,D,OVR,GBL
OVRA     000152     53         RW,D,OVR,GBL
OVRP     002022    521         RW,D,OVR,GBL
OVRB     005004   1282         RW,D,OVR,GBL
OVR2     001002    257         RW,D,OVR,GBL
OVRL     000016      7         RW,D,OVR,GBL
OVRD     000012      5         RW,D,OVR,GBL
OVROT    001032    269         RW,D,OVR,GBL
OVRDT    002124    554         RW,D,OVR,GBL
MAPDEF   000200     64         RW,D,OVR,GBL
```

```
TOTAL SPACE ALLOCATED = 031122  6441

NO FPP INSTRUCTIONS GENERATED

QDCT,LP/LI:1=QDCT/DE/NOTR
```

```
        C       PROGRAM NAME:CHANL.FTN  ORIGINATED:10-SEP-79
        C                                         UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE CHANL
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/CHANEL/IAOPB(14),IAODCT(256)
0011            COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVR0/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022            DATA REAL,CMPLX,INTGR/0,1,2/
0023            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026            DATA BUS1,BUS2,BUS3/64,128,192/
        C
```

```
0027    D          WRITE(6,699)ICOUNT
0028    D699       FORMAT(1H1,40(1H>),'RECEIVE FRAME # ',I4,40(1H<))
        C
        C          PASS DATA W/ NO ERRORS
        C
0029    D          INSTR=1
0030               STATUS=MPWDB(AROOCT,IAQDCT(1),BYTE2,CNVYES,IAQDCT(LTH))
0031    D          IF(STATUS.NE.0)GO TO 10
0032    D          INSTR=2
0033               STATUS=MPWDB(AROPB,IAQPB(1),BYTE2,CNVYES,IAQPB(LPARM+1))
0034    D          IF(STATUS.NE.0)GO TO 10
0035    D          WRITE(6,899)
0036    D899       FORMAT(/1X,'*** MODEM OUTPUT ***')
0037    D          WRITE(6,900)(IAQPB(I),I=1,LPARM)
0038    D900       FORMAT(/1X,6(1X,'AROPB(',I2,')=',I6,2X))
0039    D          WRITE(6,901)(IAQDCT(I),I=1,LTH)
0040    D901       FORMAT(/1X,6(1X,'AROOCT(',I3,')=',I6,2X))
        C
        C          ADD TRANSMISSION DELAY
        C
0041    D          INSTR=3
0042               STATUS=MPRDB(AQTDCT,IAQDCT(1),BYTE2,CNVYES,IAQDCT(LTH))
0043    D          IF(STATUS.NE.0)GO TO 10
0044    D          INSTR=4
0045               STATUS=MPRDB(AQPB,IAQPB(1),BYTE2,CNVYES,IAQPB(LPARM+1))
0046    D          IF(STATUS.NE.0)GO TO 10
0047    D          WRITE(6,799)
0048    D799       FORMAT(/1X,'*** MODEM INPUT ***')
0049    D          WRITE(6,800)(IAQPB(I),I=1,LPARM)
0050    D800       FORMAT(/1X,6(1X,'AQPB(',I2,')=',I6,2X))
0051    D          WRITE(6,801)(IAQDCT(I),I=1,LTH)
0052    D801       FORMAT(/1X,6(1X,'AQTDCT(',I3,')=',I6,2X))
0053               RETURN
        C
        C          DIAGNOSTIC TRAP AREA
        C
0054    10         TYPE 100,INSTR,STATUS
0055               CALL EXIT
0056    100        FORMAT(1X,'***MAP ERROR*** "CHANL" INSTR=',I3,' HAS STATUS=',I6/)
0057               END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000644	210	RW,I,CON,LCL
$IDATA	000452	149	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
CHANEL	001034	270	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL

```
OVRA     000152     53          RW,D,OVR,GBL
OVRP     002022    521          RW,D,OVR,GBL
OVRB     005004   1282          RW,D,OVR,GBL
OVR2     001002    257          RW,D,OVR,GBL
OVRL     000016      7          RW,D,OVR,GBL
OVRO     000012      5          RW,D,OVR,GBL
NAPDEF   000200     64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 027430  6028

NO FPP INSTRUCTIONS GENERATED

CHANL,LP/LI:1=CHANL/DE/NOTR
```

```
          C        PROGRAM NAME:DPARM.FTN  ORIGINATED:10-SEP-79
          C                                         UPDATE:13-MAY-80
          C
          C
          C
          C
0001               SUBROUTINE DPARM
0002               INTEGER BUS1,BUS2,BUS3
0003               INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                   1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004               INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                   1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                   2,DRDCTM,YNM,QRPRM,QRDCTM
                   3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                   4,MIBIT,DCTM,OPRM,OTDCTM
                   5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                   6,RECV1,RECV2,SEN1,SEN2
                   7,AOPB,AOTDCT,AIBIT,SEN,INPB
                   8,SYNB1,SYNB2,BF1,BF2,OUTB
                   9,RECV,AROPB,ARODCT,RIBIT
                   1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
          C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005               INTEGER ORDR,START,SIZE,DSW,BTLTH
0006               INTEGER ORDCT,QRPAR,QRDC,QRVAR,QRM,QRG,QRV
0007               LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008               COMMON/MTAPE0/NIN(300),NOUT(300)
0009               COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0010               COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011               COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012               COMMON/OVRO/Y(256),NP,XN
0013               COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014               COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015               COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016               COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017               COMMON/OVR2/IORDR(256),LTHM1
0018               COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019               COMMON/OVRF/XR(512)
0020               COMMON/OVRD/CSN,START,SIZE,YES,NO
0021               COMMON/OVRDT/DRDCT(256),DRPAR(8),DRDC,DRVAR,DRM,DRG,DRV,DRA(8)
0022               COMMON/OVRQT/QRDCT(256),QRPAR(8),QRDC,QRVAR,QRM,QRG,QRV
0023               COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                   1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                   2,DRDCTM,YNM,QRPRM,QRDCTM
                   3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                   4,MIBIT,DCTM,OPRM,OTDCTM
                   5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                   6,RECV1,RECV2,SEN1,SEN2
                   7,AOPB,AOTDCT,AIBIT,SEN,INPB
                   8,SYNB1,SYNB2,BF1,BF2,OUTB
                   9,RECV,AROPB,ARODCT,RIBIT
                   1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
          C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0024               DATA REAL,CMPLX,INTGR/0,1,2/
0025               DATA EVERY,SINGLE,DOUBLE/1,2,3/
0026               DATA BYTE2,BYTE4,BYTE8/2,4,8/
0027               DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
```

```
0028                DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C           SIDEBAND DEQUANTIZATION LAGS SIDEBAND DESERIALIZATION
        C           BY 1 FRAME DUE TO CSPU IN BACKGROUND
        C
0029    D           INSTR=1
0030                STATUS=VMOV2(ORPRM)
0031    D           IF(STATUS.NE.0)GO TO 10
        C
        C           DEQUANTIZE PARCOR(J),J=1,...,LPCN,G,M,VUV
        C
0032    D           INSTR=2
0033                STATUS=MPDQPP(PARC,ORPRM,A1)
0034    D           IF(STATUS.NE.0)GO TO 10
        C
        C           RESULTS
        C
0035    D           CALL MPRDH(ORPRM,IBIT(1),BYTE2,CNVYES,IBIT(LPARM+1))
0036    D           WRITE(6,927)(J,IBIT(J),J=1,LPARM)
0037    D927        FORMAT(/1X,4(1X,'ORPRM(',I2,')=',I6,2X))
0038    D           CALL MPRDH(PARC,DRPAR(1),BYTE4,CNVYES,DRPAR(LPCN))
0039    D           WRITE(6,928)(J,DRPAR(J),J=1,LPCN)
0040    D928        FORMAT(/1X,4(1X,'DRK(',I2,')=',E15.8,2X))
0041    D           CALL MPRST(88,DRDC,1,CNVYES)
0042    D           CALL MPRST(89,DRVAR,1,CNVYES)
0043    D           CALL MPRST(90,DRG,1,CNVYES)
0044    D           CALL MPRST(91,DRM,1,CNVYES)
0045    D           CALL MPRST(104,DRV,1,CNVYES)
0046    D           WRITE(6,925)DRDC,DRVAR
0047    D925        FORMAT(1X,'DRDC(88)=',E15.8,' AND DRVAR(89)=',E15.8)
0048    D           WRITE(6,929)DRM,DRG,DRV
0049    D929        FORMAT(/1X,'DRM=',E15.8,', DRG=',E15.8,' AND DRV=',E15.8)
0050    D           CALL MPRDH(A1,DRA(1),BYTE4,CNVYES,DRA(LPCN))
0051    D           WRITE(6,931)(J,DRA(J),J=1,LPCN)
0052    D931        FORMAT(/1X,4(1X,'DRA(',I2,')=',E15.8,2X))
0053    D           CALL MPRST(60,ENG,1,CNVYES)
0054    D           WRITE(6,932)ENG
0055    D932        FORMAT(/1X,'ENG=',E15.8)
0056    D           CALL MPRDH(ORDCTM,ORDCT(1),BYTE2,CNVYES,ORDCT(LTH))
0057    D           WRITE(6,940)(I,ORDCT(I),I=1,LTH)
0058    D940        FORMAT(/1X,4(1X,'ORDCTM(',I3,')=',I6,2X))
0059    D           CALL MPRDH(MIBIT2,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0060    D           WRITE(6,941)(I,IBIT(I),I=1,LTH)
0061    D941        FORMAT(/1X,4(1X,'MIBIT2(',I3,')=',I6,2X))
0062    D           CALL MPRDH(IORDR2,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0063    D           WRITE(6,942)(I,IORDR(I),I=1,LTH)
0064    D942        FORMAT(/1X,4(1X,'IORDR2(',I3,')=',I6,2X))
0065    D           CALL MPRDH(DCT1T2,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0066    D           WRITE(6,943)(I,DCT1(I),I=1,LTH)
0067    D943        FORMAT(/1X,4(1X,'DCT1(',I3,')=',E15.8,2X))
0068    D           CALL MPRST(100,DRDC,1,CNVYES)
0069    D           CALL MPRST(101,DRVAR,1,CNVYES)
0070    D           WRITE(6,944)DRDC,DRVAR
0071    D944        FORMAT(1X,'DRDC(100)=',E15.8,' AND VAR(101)=',E15.8)
0072                RETURN
        C
```

```
        C       DIAGNOSTIC TRAP AREA
        C
0073    10      TYPE 100,INSTR,STATUS
0074            CALL EXIT
0075    100     FORMAT(1X,'***MAP ERROR*** "DPARM" INSTR=',I4,' HAS STATUS=',I6/)
0076            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001542	433	RW,I,CON,LCL
$PDATA	000044	18	RW,D,CON,LCL
$IDATA	001036	271	RW,D,CON,LCL
$VARS	000056	23	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
OVRDT	002124	554	RW,D,OVR,GBL
OVRQT	001032	269	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 033102 6945

DPARM,LP/LI:1=DPARM/DE/NOTR

```
        C       PROGRAM NAME:SSRT.FTN   ORIGINATED:10-SEP-79
        C                                    UPDATE:07-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE SSRT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AOTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       ORDER DCT1(.) MAX-TO-MIN W/ ORDERING INFO IN IORDR(.)
```

```
FORTRAN IV-PLUS V02-51E          13:31:17     09-AUG-80           PAGE 2
SSRT.FTN            /DE/WR

          C        SORT TMP2 TO DCTM2
          C        SORT DCTM1 TO TMP4
          C
0026      D        INSTR=1
0027               STATUS=MPSSRT(IORDRM)
0028      D        IF(STATUS.NE.0)GO TO 10
          C
          C        RESULTS
          C
0029      D        CALL MPRDB(DCTM2,DCT2(1),BYTE4,CNVYES,DCT2(LTH))
0030      D        CALL MPRDB(IORDRM,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0031      D        WRITE(6,910)(I,IORDR(I),I=1,LTH)
0032      D910     FORMAT(/1X,6(1X,'IORDR(',I3,')=',I3,2X))
0033      D        WRITE(6,909)(I,DCT2(I),I=1,LTH)
0034      D909     FORMAT(/1X,4(1X,'DCT2(',I3,')=',E15.8,2X))
0035      D        CALL MPRDB(TMP4,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0036      D        WRITE(6,911)(I,DCT1(I),I=1,LTH)
0037      D911     FORMAT(/1X,4(1X,'DCT1T(',I3,')=',E15.8,2X))
0038               RETURN
          C
          C        DIAGNOSTIC TRAP AREA
          C
0039      10       TYPE 100,INSTR,STATUS
0040               CALL EXIT
0041      100      FORMAT(1X,'***MAP ERROR*** "SSRT" INSTR=',I3,' HAS STATUS=',I6/)
0042               END

PROGRAM SECTIONS

 NAME       SIZE               ATTRIBUTES

$CODE1   000470   156          RW,I,CON,LCL
$IDATA   000256    87          RW,D,CON,LCL
$VARS    000054    22          RW,D,CON,LCL
MTAPE0   002260   600          RW,D,OVR,GBL
MTAPE1   000012     5          RW,D,OVR,GBL
MTAPE2   000012     5          RW,D,OVR,GBL
OVRI     002020   520          RW,D,OVR,GBL
OVRD     002006   515          RW,D,OVR,GBL
OVRF     006016  1543          RW,D,OVR,GBL
OVRA     000152    53          RW,D,OVR,GBL
OVRP     002022   521          RW,D,OVR,GBL
OVRB     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRD     000012     5          RW,D,OVR,GBL
MAPDEF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026024  5642

NO FPP INSTRUCTIONS GENERATED

SSRT,LP/LI:1=SSRT/DE/NOTR
```

```
        C       PROGRAM NAME:DDCT.FTN   ORIGINATED:05-FEB-79
        C                                        UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE DDCT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
               1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
               1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
               2,DRDCTM,YNM,QRPRM,QRDCTM
               3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
               4,MIBIT,DCTM,QPRM,QTDCTM
               5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
               6,RECV1,RECV2,SEN1,SEN2
               7,AQPB,AQTDCT,AIBIT,SEN,INPB
               8,SYNB1,SYNB2,BF1,BF2,OUTB
               9,RECV,ARQPB,ARQDCT,RIBIT
               1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VB,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0021            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
               1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
               2,DRDCTM,YNM,QRPRM,QRDCTM
               3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
               4,MIBIT,DCTM,QPRM,QTDCTM
               5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
               6,RECV1,RECV2,SEN1,SEN2
               7,AQPB,AQTDCT,AIBIT,SEN,INPB
               8,SYNB1,SYNB2,BF1,BF2,OUTB
               9,RECV,ARQPB,ARQDCT,RIBIT
               1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022            DATA REAL,CMPLX,INTGR/0,1,2/
0023            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026            DATA BUS1,BUS2,BUS3/64,128,192/
        C
```

```
          C         DEQUANTIZATION OF MAINBAND LAGS MAINBAND DESERIALIZATION
          C         BY 1 FRAME DUE TO CSPU IN THE BACKGROUND
          C
0027      D         INSTR=1
          C         GIVE CSPU BIT ASSIGNMENT FOR UNPACKING OF MAINBAND
0028                STATUS=VMOV3(RIBIT)
0029      D         IF(STATUS.NE.0)GO TO 10
          C
          C         DEQUANTIZE DCT(I)
          C
0030      D         INSTR=2
0031                STATUS=MPESTD(DRDCTM,QRDCTM,TMP1,IORDR2)
0032      D         IF(STATUS.NE.0)GO TO 10
          C
          C         RESULTS
          C
0033      D         CALL MPRDB(DRDCTM,DTDCT(1),BYTE4,CNVYES,DTDCT(LTH))
0034      D         WRITE(6,934)(I,DTDCT(I),I=1,LTH)
0035      D934      FORMAT(/1X,4(1X,'DRDCTM(',I3,')=',E15.8,2X))
0036      D         CALL MPRDB(RIBIT,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0037      D         WRITE(6,935)(I,IBIT(I),I=1,LTH)
0038      D935      FORMAT(/1X,4(1X,'RIBIT(',I3,')=',I6,2X))
0039      D         CALL MPRDB(DCT1T1,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0040      D         WRITE(6,936)(I,DCT1(I),I=1,LTH)
0041      D936      FORMAT(/1X,4(1X,'DCT1T1(',I3,')=',E15.8,2X))
0042      D         CALL MPRDB(IORDR1,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0043      D         WRITE(6,937)(I,IORDR(I),I=1,LTH)
0044      D937      FORMAT(/1X,4(1X,'IORDR1(',I3,')=',I6,2X))
0045      D         CALL MPRDB(MIBIT1,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0046      D         WRITE(6,938)(I,IBIT(I),I=1,LTH)
0047      D938      FORMAT(/1X,4(1X,'MIBIT1(',I3,')=',I6,2X))
0048      D         CALL MPRST(102,DRDC,1,CNVYES)
0049      D         CALL MPRST(103,DRVAR,1,CNVYES)
0050      D         WRITE(6,940)DRDC,DRVAR
0051      D940      FORMAT(1X,'DRDC(102)=',F15.8,2X,' AND DRVAR(103)=',E15.8)
0052                RETURN
          C
          C         DIAGNOSTIC TRAP AREA
          C
0053      10        TYPE 100,INSTR,STATUS
0054                CALL EXIT
0055      100       FORMAT(1X,'***MAP ERROR*** "DDCT" INSTR=',I4,' HAS STATUS=',I6/)
0056                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001056	279	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000506	163	RW,D,CON,LCL
$VARS	000064	26	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL

```
OVRI     002020   520          RW,D,OVR,GBL
OVRO     002006   515          RW,D,OVR,GBL
OVRF     006016  1543          RW,D,OVR,GBL
OVRA     000152    53          RW,D,OVR,GBL
OVRP     002022   521          RW,D,OVR,GBL
OVRR     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRD     000012     5          RW,D,OVR,GBL
OVRDT    002124   554          RW,D,OVR,GBL
MAPDEF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 031012  6405

DDCT,LP/LI:1=DDCT/DE/NOTR
```

```
      C         PROGRAM NAME:DCTR.FTN   ORIGINATED:05-FEB-79
      C                                            UPDATE:12-MAY-80
      C
      C
      C
      C
0001            SUBROUTINE DCTR
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
      C         <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
      C         >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
      C
      C         FORM INPHASE & QUADRATURE COMPONENTS
```

```
FORTRAN IV-PLUS V02-51F          13:31:32     09-AUG-80            PAGE 2
DCTR.FTN         /DE/WR

          C
0026      D          INSTR=1
0027                 STATUS=MPIDCM(X2,DRDCTM,COSZ)
0028      D          IF(STATUS.NE.0)GO TO 10
0029      D          CALL MPRDB(X2,XR(1),BYTE8,CNVYES,XR(LTH2))
0030      D          WRITE(6,914)(I,XR(I),I=1,LTH2)
0031      D          INSTR=2
0032                 STATUS=FFTIN(X2,1,X2,VSHRT,WORK)
0033      D          IF(STATUS.NE.0)GO TO 10
          C
          C          RESULTS
          C
0034      D          CALL MPRDB(X2,XR(1),BYTE4,CNVYES,XR(LTH))
0035      D          WRITE(6,914)(I,XR(+I),I=1,LTH)
0036      D914       FORMAT(/1X,4(1X,'X(',I3,')=',E15.8,2X))
0037                 RETURN
          C
          C          DIAGNOSTIC TRAP AREA
          C
0038      10         TYPE 100,INSTR,STATUS
0039                 CALL EXIT
0040      100        FORMAT(1X,'***MAP ERROR*** "DCTR" INSTR=',I3,' HAS STATUS=',I6/)
0041                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000406	131	RW,I,CON,LCL
$PDATA	000004	2	RW,D,CON,LCL
$IDATA	000172	61	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025662 5593

DCTR,LP/LI:1=DCTR/DE/NOTR

```
        C       PROGRAM NAME:VARDC.FTN           ORIGINATED:17-SEP-79
        C                                                    UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE VARDC
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRT/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRE/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AOPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
0026    D       INSTR=1
```

```
0027                STATUS=MPVDNM(NOUTM,YOM,X2)
0028        D       IF(STATUS.NE.0)GO TO 10
            C
            C       RESULTS
            C
0029        D       CALL MPRST(100,DCBIAS,1,CNVYES)
0030        D       WRITE(6,900)DCBIAS
0031        D900    FORMAT(/1X,'DCBIAS=',E15.8)
0032        D       CALL MPRST(101,VAR,1,CNVYES)
0033        D       WRITE(6,901)VAR
0034        D901    FORMAT(/1X,'VAR=',E15.8)
0035                RETURN
            C
            C       DIAGNOSTIC TRAP AREA
            C
0036        10      TYPE 101,INSTR,STATUS
0037                CALL EXIT
0038        101     FORMAT(1X,'***MAP ERROR*** "VARDC" INSTR=',I3,' HAS STATUS=',I6/)
0039                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000200	64	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000160	56	RW,D,CON,LCL
$VARS	000052	21	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025450   5524

NO FPP INSTRUCTIONS GENERATED

VARDC,LP/LI:1=VARDC/DE/NOTR

```
        C        PROGRAM NAME:OUTPUT.FTN ORIGINATED:05-FEB-79
        C                                             UPDATE:09-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE OUTPUT
0002             INTEGER BUS1,BUS2,BUS3
0003             INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004             INTEGER WORK,X1,X1R,X1I,IORDRM,PWFITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AOTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005             INTEGER ORDR,START,SIZE,DSN,BTLTH
0006             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007             COMMON/MTAPE0/NIN(300),NOUT(300)
0008             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0009             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010             COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011             COMMON/OVRO/Y(256),NP,XN
0012             COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014             COMMON/OVRP/M,G,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016             COMMON/OVR2/IORDR(256),LTHM1
0017             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018             COMMON/OVRF/XR(512)
0019             COMMON/OVRD/CSN,START,SIZE,YES,NO
0020             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWFITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AOTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021             DATA REAL,CMPLX,INTGR/0,1,2/
0022             DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023             DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024             DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025             DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C        TRANSFER TO TAPE OUTPUT VECTOR
```

```
             C
             C          CHECK FOR REQUEST FROM "INPUT"
0026                    IF(NEND.NE.0)GO TO 5000
0027         D          INSTR=1
0028                    STATUS=MP*T(1,1)
0029                    STATUS=MPRDR(NOUTR,NOUT(1),BYTE2,CNVYES,NOUT(NTOTO))
0030                    IF(STATUS.NE.0)GO TO 10
             CD         WRITE(6,913)(I,NOUT(I),I=1,NTOTO)
             CD913      FORMAT(/1X,4(1X,'NOUT(',I3,')=',I6,2X))
             C
             C          DATA OUTPUT
             C
0031                    CALL TAPE2(2)
0032                    RETURN
             C
             C          EOF ON INPUT,WRITE EOF
             C
0033         5000       ICOUNT=ICOUNT-1
0034                    DO 5001 I=1,NTUPS
0035         5001       NOUT(I)=0
0036                    DO 5002 I=1,32
0037         5002       CALL TAPE2(2)
0038                    NSKIP=NSKIPS
0039                    NEND=0
0040                    IST=1
0041                    CALL TAPE2(6)
0042                    DO 5005 I=1,ICOUNT
0043                    CALL TAPE2(1)
0044                    IF(NEND)5077,5076,5077
0045         5076       DO 5006 J=1,NTUPS
0046         5006       NOUT(J)=NIN(J)
0047         5005       CALL TAPE2(2)
0048         5077       DO 5007 I=1,NTUPS
0049         5007       NOUT(I)=0
0050                    DO 5008 I=1,8
0051         5008       CALL TAPE2(2)
0052                    CALL TAPE2(5)
             C          CALL MPCLS(0)
             C          TYPE 100,ICOUNT
             C100       FORMAT(1X,'"ATC" DONE AFTER ',I6,' FRAMES!'/)
             C          TYPE 100
             C100       FORMAT(1X,'ATC SYSTEM READY')
             C          CALL EXIT
0053                    RETURN
             C
             C          DIAGNOSTIC TRAP AREA
             C
0054         10         TYPE 101,INSTR,STATUS
0055                    CALL EXIT
0056         101        FORMAT(1X,'***MAP ERROR*** "OUTPUT" INSTR=',I3,' HAS STATUS=',I6/)
0057                    END
```

PROGRAM SECTIONS

NAME   SIZE   ATTRIBUTES

```
SCODE1   000466   155          RW,I,CON,LCL
SPDATA   000020     8          RW,D,CON,LCL
SIDATA   000132    45          RW,D,CON,LCL
SVARS    000056    23          RW,D,CON,LCL
STEMPS   000002     1          RW,D,CON,LCL
#TAPE0   002260   600          RW,D,OVR,GBL
#TAPE1   000012     5          RW,D,OVR,GBL
#TAPE2   000012     5          RW,D,OVR,GBL
OVRT     002020   520          RW,D,OVR,GBL
OVRD     002006   515          RW,D,OVR,GBL
OVRF     006016  1543          RW,D,OVR,GBL
OVRA     000152    53          RW,D,OVR,GBL
OVRP     002016   519          RW,D,OVR,GBL
OVRR     005004  1282          RW,D,OVR,GBL
OVR2     001002   257          RW,D,OVR,GBL
OVRL     000016     7          RW,D,OVR,GBL
OVRO     000012     5          RW,D,OVR,GBL
MAPDEF   000200    64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025716  5607

NO FPP INSTRUCTIONS GENERATED

OUTPUT,LP/LI:1=OUTPUT/DE/NOTR
```

```
        C        PROGRAM NAME:SNR.FTN     ORIGINATED:05-FEB-79
        C                                         UPDATE:13-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE SNR
0002             REAL NOISE
0003             INTEGER BUS1,BUS2,BUS3
0004             INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0005             INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0006             INTEGER ORDR,START,SIZE,DSW,BTLTH
0007             DIMENSION NIND3(246)
0008             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0009             COMMON/MTAPE0/NIN(300),NOUT(300)
0010             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0011             COMMON/MTAPE2/NEND,NEPR,NFILE,NINS,NOUTS
0012             COMMON/DELAY/M4,M3,M2,M1,IV4,IV3,IV2,IV1
0013             COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0014             COMMON/OVRO/Y(256),NP,XN
0015             COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0016             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0017             COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0018             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0019             COMMON/OVR2/IORDR(256),LTHM1
0020             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0021             COMMON/OVRF/XR(512)
0022             COMMON/OVRD/CSN,START,SIZE,YES,NO
0023             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
0024             EQUIVALENCE (NIND3(1),XR(1))
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0025             DATA REAL,CMPLX,INTGR/0,1,2/
0026             DATA EVERY,SINGLE,DOUBLE/1,2,3/
0027             DATA BYTE2,BYTE4,BYTE8/2,4,8/
```

```
0028              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0029              DATA BUS1,BUS2,BUS3/64,128,192/
          C
          C       ADD 3 FRAMES DELAY IN INPUT FOR SNR CALCULATION
          C
0030      D       INSTR=0
0031              STATUS=MPRDB(SEN,NIND3(1),BYTE2,CNVYES,NIND3(NTUPS))
0032              STATUS=MPRDB(NIN2,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
0033              STATUS=MPWDB(SEN,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
0034              STATUS=MPRDB(NIN1,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
0035              STATUS=MPWDB(NIN2,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
0036              STATUS=MPWDB(NIN1,NIN(1),BYTE2,CNVYES,NIN(NTUPS))
0037      D       IF(STATUS.NE.0)GO TO 10
          C       NOW DELAY M & VUV FOR CONSISTENCY
0038              M4=M3
0039              M3=M2
0040              M2=M1
0041              M1=M
0042              IV4=IV3
0043              IV3=IV2
0044              IV2=IV1
0045              IV1=IFIX(VU)
          C
          C
          C       COMPUTE S/N RATIO
          C
0046              SUM1=0.0
0047              SUM2=0.0
          C       NIN(1) MATCHES NOUT(NP+1), ERGO NTUPS-NP MATCHES!
0048              DO 3605 I=1,NTOTO-NP
0049              SOURCE=FLOAT(NIND3(I))
0050              SIGNAL=FLOAT(NOUT(NP+I))
0051              NOISE=SIGNAL-SOURCE
0052              SUM1=SUM1+SOURCE**2
0053      3605    SUM2=SUM2+NOISE**2
0054              SN=0.0
0055              FRNUM=ICOUNT-START+1
0056              IF(FRNUM.LT.4)GO TO 200
0057              SN=10.0*ALOG10(SUM1/SUM2)
0058              IF(FRNUM.EQ.4)CSN=SN
0059              CSN=((FRNUM-1.)/FRNUM)*CSN+(1./FRNUM)*SN
          C200    WRITE(4,255)ICOUNT,SN,CSN,M4,IV4
          C       WRITE(5,255)ICOUNT,SN,CSN,M4,IV4
0060      255     FORMAT(1X,'FRAME=',I3,3X,'SN=',F6.2,3X,'CSN=',F6.2,3X
                  1,'PITCH=',I3,'    &    VUV=',I2)
0061      200     CONTINUE
0062              RETURN
          C
          C       DIAGNOSTIC TRAP AREA
          C
0063      10      TYPE 100,INSTR,STATUS
0064              CALL EXIT
0065      100     FORMAT(1X,'***MAP ERROR*** "SNR" INSTR=',I3,' HAS STATUS=',I6/)
0066              END
```

```
FORTRAN IV-PLUS V02-51F          21:16:38     11-SEP-80          PAGE 3
SNR.FTN            /OF/WR

PROGRAM SECTIONS

 NAME        SIZE               ATTRIBUTES

$CODE1  000720   232            RW,I,CON,LCL
$IDATA  000162    57            RW,D,CON,LCL
$VARS   000110    36            RW,D,CON,LCL
$TEMPS  000002     1            RW,D,CON,LCL
$TAPE0  002260   600            RW,D,OVR,GBL
$TAPE1  000012     5            RW,D,OVR,GBL
$TAPE2  000012     5            RW,D,OVR,GBL
DELAY   000020     8            RW,D,OVR,GBL
OVRI    002020   520            RW,D,OVR,GBL
OVRO    002006   515            RW,D,OVR,GBL
OVRF    006016  1543            RW,D,OVR,GBL
OVRA    000152    53            RW,D,OVR,GBL
OVRP    002022   521            RW,D,OVR,GBL
OVRB    005004  1282            RW,D,OVR,GBL
OVR2    001002   257            RW,D,OVR,GBL
OVRL    000016     7            RW,D,OVR,GBL
OVRD    000012     5            RW,D,OVR,GBL
MAPDEF  000200    64            RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026236  5711

SNR,LP/LI:1=SNR/OF/NOTR
```

```
FORTRAN IV-PLUS V02-51E          11:41:28    10-AUG-80           PAGE 1
TAPE2.FTN         /WR

                SUBROUTINE TAPE2(IQ)
                IMPLICIT INTEGER(A-Z)
                LOGICAL*1 APPEND
                COMMON/MTAPE0/NIN(300),NOUT(300)
                COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
                COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
                COMMON/MTAPE3/NBF(1324),NBUF(1324)
                COMMON/MTAPE4/LST,IBEG
                COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
                COMMON/MTAPE6/APPEND
                DATA IOATT,IOSUC,IEALN/0001400,1,-34/
                DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
                DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
                DATA MASK/0377/
                DATA MT0/0,2048,0,0,0,0/
                DATA MT1/0,2048,0,0,0,0/
                GO TO (700,800,900,1100,1000,1200,1300,1400),IQ
          700   CALL OPT1
                RETURN
          800   CALL OPT2
                RETURN
          900   CALL OPT3
                RETURN
          1100  CALL OPT4
                RETURN
          1000  CALL OPT5
                RETURN
          1200  CALL OPT6
                RETURN
          1300  CALL OPT7
                RETURN
          1400  CALL OPT8
                CALL OPT3
                IF(APPEND)CALL OPT4
                RETURN
                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000226	75	RW,I,CON,LCL
$PDATA	000022	9	RW,D,CON,LCL
$IDATA	000002	1	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004		RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015130   3372

NO FPP INSTRUCTIONS GENERATED

TAPE2,LP/LI:1=TAPE2/NOTR

```
FORTRAN IV-PLUS V02-51E         11:41:33    10-AUG-80           PAGE 1
OPT1.FTN          /WR

0001            SUBROUTINE OPT1
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 APPEND
0004            COMMON/MTAPE0/NIN(300),NOUT(300)
0005            COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0006            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008            COMMON/MTAPE4/LST,IBEG
0009            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010            COMMON/MTAPE6/APPEND
0011            DIMENSION ICARD(64)
0012            DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0013            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0014            DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0015            DATA MASK/0377/
0016            DATA MT0/0,2048,0,0,0,0/
0017            DATA MT1/0,2048,0,0,0,0/
        C
        C       INPUT DATA(=1)
        C
0018    700     CONTINUE
0019            IST=IST+NTUPS
0020            IF(1324.GE.IST) GO TO 200
0021    100     IST=IST-1024
0022            NSKIP=NSKIP+1
0023    200     KOVER=LST+NTOT1-1325
0024            IF(KOVER.GT.0) GO TO 305
0025            DO 5000 I=1,NTOT1
0026    5000    NIN(I)=NBF(LST+I-1)
0027            LST=LST+NTUPS
0028            GO TO 6002
0029    305     IPEP=LST-1024
0030            DO 5001 I=1,300
0031    5001    NBF(IPEP+I-1)=NBF(LST+I-1)
0032            LST=IPEP
0033            IF(NINS.EQ.0) GO TO 2100
        C>>>>>>>DISK INPUT
0034            DO 5010 I=1,16
0035            READ(2,END=2001,ERR=6000)(ICARD(J),J=1,64)
0036            K=64*(I-1)+300
0037            DO 5010 J=1,64
0038    5010    NBF(K+J)=ICARD(J)
0039            IF(NERR.NE.0) GO TO 6000
0040            GO TO 200
0041    2001    NEND=1
0042            GO TO 200
        C>>>>>>>TAPE INPUT
0043    2100    NEND=0
0044            NERR=0
0045            CALL GETADR(MT0,NBF(301))
0046            CALL WTQIO(IORLB,2,1,0,ISW,MT0,DSW)
0047            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 200
0048            IF(IAND(IEEOF,MASK).EQ.IAND(MASK,ISW(1)))NEND=1
0049            IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
0050            GO TO 200
```

```
FORTRAN IV-PLUS V02-51E          11:41:33     10-AUG-80            PAGE 2
OPT1.FTN            /WR

0051     6000     TYPE 6001
0052     6001     FORMAT(1X,'***INPUT FILE ERROR***'/)
0053              NERR=1
0054     6002     RETURN
0055              END


PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1  000646   211            RW,I,CON,LCL
$PDATA  000014     6            RW,D,CON,LCL
$IDATA  000062    25            RW,D,CON,LCL
$VARS   000212    69            RW,D,CON,LCL
MTAPE0  002260   600            RW,D,OVR,GBL
MTAPE1  000012     5            RW,D,OVR,GBL
MTAPE2  000012     5            RW,D,OVR,GBL
MTAPE3  012260  2648            RW,D,OVR,GBL
MTAPE4  000004     2            RW,D,OVR,GBL
MTAPE5  000064    26            RW,D,OVR,GBL
MTAPE6  000002     1            RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016034  3598

NO FPP INSTRUCTIONS GENERATED

OPT1,LP/LI:1=OPT1/NOTR
```

```
              SUBROUTINE OPT2
              IMPLICIT INTEGER(A-Z)
              LOGICAL*1 APPEND
              COMMON/MTAPE0/NIN(300),NOUT(300)
              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/MTAPE3/NBF(1324),NBUF(1324)
              COMMON/MTAPE4/LST,IBEG
              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
              1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
              COMMON/MTAPE6/APPEND
              DIMENSION ICARD(64)
              DATA IOATT,IOSUC,IEALN/O001400,1,-34/
              DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/O2400,O400,-4,O2440,O1000     /
              DATA IOSPF,IEEOF,IOEOF/O2440,-10,O3000/
              DATA MASK/O377/
              DATA MT0/0,2048,0,0,0,0/
              DATA MT1/0,2048,0,0,0,0/
      C
      C       OUTPUT DATA(=2)
      C
      800     CONTINUE
              DO 5005 I=1,NTOTO
      5005    NBUF(IBEG+I-1)=NOUT(I)
              IBEG=IBEG+NTOTO
              KON=1025-IBEG
              IF(KON.GT.0) GO TO 6002
              IF(NOUTS.EQ.0) GO TO 2002
      C>>>>>>>DISK OUTPUT
              DO 6010 I=1,16
              K=64*(I-1)
              DO 6011 J=1,64
      6011    ICARD(J)=NBUF(K+J)
      6010    WRITE(3)(ICARD(J),J=1,64)
              GO TO 2003
      C>>>>>>>TAPE OUTPUT
      2002    NERR=0
              CALL GETADR(MT1,NBUF)
              CALL WTOIO(IOWLB,3,1,0,ISW(1),MT1,DSW)
              IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 2003
              IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
      2003    LRESID=-KON
              DO 5006 I=1,LRESID
      5006    NBUF(I)=NBUF(1024+I)
              IBEG=LRESID+1
              RETURN
      6000    TYPE 6001
      6001    FORMAT(1X,'***OUTPUT FILE ERROR***'/)
              NERR=1
      6002    RETURN
              END
```

PROGRAM SECTIONS

NAME   SIZE   ATTRIBUTES

```
SCODE1  000432   141           RW,I,CON,LCL
SPDATA  000014     6           RW,D,CON,LCL
SIDATA  000062    25           RW,D,CON,LCL
SVARS   000212    69           RW,D,CON,LCL
MTAPE0  002260   600           RW,D,OVR,GBL
MTAPE1  000012     5           RW,D,OVR,GBL
MTAPE2  000012     5           RW,D,OVR,GBL
MTAPE3  012260  2648           RW,D,OVR,GBL
MTAPE4  000004     2           RW,D,OVR,GBL
MTAPE5  000064    26           RW,D,OVR,GBL
MTAPE6  000002     1           RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015620  3528

NO FPP INSTRUCTIONS GENERATED

OPT2,LP/LI:1=OPT2/NOTR
```

```
0001            SUBROUTINE OPT3
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 APPEND
0004            COMMON/MTAPE0/NIN(300),NOUT(300)
0005            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0006            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008            COMMON/MTAPE4/LST,IBEG
0009            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
               1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MTO(6),MT1(6),DSW
0010            COMMON/MTAPE6/APPEND
0011            DIMENSION ICARD(64)
0012            DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0013            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0014            DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0015            DATA MASK/0377/
0016            DATA MTO/0,2048,0,0,0,0/
0017            DATA MT1/0,2048,0,0,0,0/
          C
          C     INITIALIZE(=3)
          C
0018      900   CONTINUE
0019            NEND=0
0020            NERR=0
0021            IF((NINS+NOUTS).GT.1)GO TO 901
0022            UNIT=0
0023            DO 902 LUN=2,3
0024            IF(NINS.NE.0.AND.LUN.EQ.2)GO TO 902
0025            IF(NOUTS.NE.0.AND.LUN.EQ.3)GO TO 902
          C>>>>>>>>MT: ATTACH
0026            CALL ASNLUN(LUN,'MT',UNIT,DSW)
0027            IF(DSW.NE.1)GO TO 6000
0028            CALL WTQIO(IOATT,LUN,1,0,ISW(1),0,DSW)
0029            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 902
0030            IF(IAND(IEALN,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
0031      902   UNIT=UNIT+1
0032      901   DO 903 LUN=2,3
0033            IF(NINS.NE.0.AND.LUN.EQ.2)GO TO 903
0034            IF(NOUTS.NE.0.AND.LUN.EQ.3)GO TO 903
          C>>>>>>>>MT: REWIND
0035            CALL WTQIO(IORWD,LUN,1,0,ISW(1),0,DSW)
0036            IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6001
0037      903   CONTINUE
0038            IF(NFILE.EQ.0)GO TO 995
0039            IF(NINS.NE.0)GO TO 913
          C>>>>>>>>MT: FILE SKIP
0040            DO 907 I=1,NFILE-1
0041            CALL GETADR(MTO,NBF(301))
0042            CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
0043            IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6003
0044            MTO(1)=1
0045      907   CALL WTQIO(IOSPF,2,1,0,ISW(1),MTO,DSW)
0046      913   DO 912 J=1,NSKIP
0047            IF(NINS.EQ.0)GO TO 910
          C>>>>>>>>DISK INPUT
0048            DO 914 I=1,16
```

```
                 READ(2,END=905,ERR=6002)(ICARD(JJ),JJ=1,64)
                 K=64*(I-1)+300
                 DO 914 JJ=1,64
         914     NBF(K+JJ)=ICARD(JJ)
                 GO TO 912
         C>>>>>>>TAPE INPUT
         910     CALL GETADR(MTO,NBF(301))
                 CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
                 IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 912
                 IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))GO TO 6002
                 IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 912
         905     NEND=1
         912     CONTINUE
         995     IBEG=1
                 LST=IST+300
                 IST=IST-NTUPS
                 RETURN
         6000    TYPE 100
                 GO TO 6010
         6001    TYPE 101
                 GO TO 6010
         6002    TYPE 102
                 GO TO 6010
         6003    TYPE 103
         6010    NERR=1
                 RETURN
         100     FORMAT(1X,'*** MT: ATTACH FAILURE! ***'/)
         101     FORMAT(1X,'*** MT: REWIND FAILURE! ***'/)
         102     FORMAT(1X,'*** INPUT FILE "NSKIP" FAILURE! ***'/)
         103     FORMAT(1X,'*** MT: "NFILE" FAILURE! ***'/)
                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001212	325	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000332	109	RW,D,CON,LCL
$VARS	000214	70	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016662 3801

NO FPP INSTRUCTIONS GENERATED

OPT3,LP/LI:1=OPT3/NOTR

```
            SUBROUTINE OPT4
            IMPLICIT INTEGER(A-Z)
            LOGICAL*1 APPEND
            COMMON/MTAPE0/NIN(300),NOUT(300)
            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
            COMMON/MTAPE3/NBF(1324),NBUF(1324)
            COMMON/MTAPE4/LST,IBEG
            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
            1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
            COMMON/MTAPE6/APPEND
            DIMENSION ICARD(64)
            DATA IOATT,IOSUC,IEALN/0001400,1,-34/
            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
            DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
            DATA MASK/0377/
            DATA MT0/0,2048,0,0,0,0/
            DATA MT1/0,2048,0,0,0,0/
      C
      C     OUTPUT FILE SKIP(=4)
      C
      1100  CONTINUE
            NERR=0
            NEND=0
            IBEG=1
      1     CALL GETADR(MT1(1),NBUF(1))
            CALL WTQIO(IORLB,3,1,0,ISW(1),MT1,DSW)
            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
            IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
            CALL GETADR(MT1(1),NBUF(1))
            CALL WTQIO(IORLB,3,1,0,ISW(1),MT1,DSW)
            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
            IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
            MT1(1)=-1
            CALL WTQIO(IOSPF,3,1,0,ISW(1),MT1,DSW)
            RETURN
      6000  TYPE 100
            RETURN
      100   FORMAT(1X,'*** OUTPUT "APPEND" FAILURE! ***'/)
            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000224	74	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000114	38	RW,D,CON,LCL
$VARS	000200	64	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

```
NTAPE6  000002     1          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015432  3469

NO FPP INSTRUCTIONS GENERATED

OPT4,LP/LI:1=OPT4/NOTR
```

```
              SUBROUTINE OPTS
              IMPLICIT INTEGER(A-Z)
              LOGICAL*1 APPEND
              COMMON/MTAPE0/NIN(300),NOUT(300)
              COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTO
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/MTAPE3/NBF(1324),NBUF(1324)
              COMMON/MTAPE4/LST,IBEG
              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
              1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
              COMMON/MTAPE6/APPEND
              DIMENSION ICARD(64)
              DATA IOATT,IOSUC,IEALN/0001400,1,-34/
              DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
              DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
              DATA MASK/0377/
              DATA MT0/0,2048,0,0,0,0/
              DATA MT1/0,2048,0,0,0,0/
C
C             END-OF-FILE(=5)
C
1000          CONTINUE
              NERR=0
              NEND=0
              IF(NOUTS.EQ.0)GO TO 1001
C>>>>>>>>DISK EOF
              CALL CLOSE(3)
              GO TO 1003
C>>>>>>>>TAPE EOF
1001          DO 1002 I=1,2
1002          CALL WTQIO(IOEOF,3,1,0,ISW(1))
              IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
              MT1(1)=-1
              CALL WTQIO(IOSPF,3,1,0,ISW(1),MT1,DSW)
              IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
1003          IBEG=1
              RETURN
6000          TYPE 100
              NERR=1
              RETURN
100           FORMAT(1X,'*** MT: EOF FAILURE! ***'/)
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000204	66	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000076	31	RW,D,CON,LCL
$VARS	000202	65	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL

```
MTAPE4  000004    2       RW,D,OVR,GBL
MTAPE5  000064   26       RW,D,OVR,GBL
MTAPE6  000002    1       RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015376  3455

NO FPP INSTRUCTIONS GENERATED

OPT5,LP/LI:1=OPT5/NOTR
```

```
      SUBROUTINE OPT6
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 APPEND
      COMMON/MTAPE0/NIN(300),NOUT(300)
      COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
      COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
      COMMON/MTAPE3/NBF(1324),NBUF(1324)
      COMMON/MTAPE4/LST,IBEG
      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
     1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MTO(6),MT1(6),DSW
      COMMON/MTAPE6/APPEND
      DIMENSION ICARD(64)
      DATA IOATT,IOSUC,IEALN/0001400,1,-34/
      DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
      DATA IOSPF, EEOF,IOEOF/02440,-10,03000/
      DATA MASK/0377/
      DATA MTO/0,2048,0,0,0,0/
      DATA MT1/0,2048,0,0,0,0/
C
C     REWIND/SEARCH INPUT ONLY(=6)
C
 1200 CONTINUE
      NEND=0
      NERR=0
      IF(NINS.EQ.0)GO TO 1222
C>>>>>>>>DISK REWIND
      REWIND 2
      GO TO 1203
C>>>>>>>>MT: REWIND
 1222 CALL WTQIO(IORWD,2,1,0,ISW(1),0,DSW)
      IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6002
 1203 IF(NFILE.LE.1)GO TO 1204
      DO 1205 I=1,NFILE-1
      CALL GETADR(MTO,NBF(301))
      CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
      IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
      MTO(1)=1
 1205 CALL WTQIO(IOSPF,2,1,0,ISW(1),MTO,DSW)
 1204 DO 1210 J=1,NSKIP
      IF(NINS.EQ.0)GO TO 1201
C>>>>>>>>DISK SEARCH
      DO 1207 I=1,16
      READ(2,END=1202,ERR=6001)(ICARD(JJ),JJ=1,64)
      K=64*(I-1)+300
      DO 1207 JJ=1,64
 1207 NBF(K+JJ)=ICARD(JJ)
      GO TO 1210
C>>>>>>>>TAPE SEARCH
 1201 CALL GETADR(MTO,NBF(301))
      CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1210
      IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))GO TO 6000
      IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 1210
 1202 NEND=1
      GO TO 6000
 1210 CONTINUE
```

```
0049                LST=IST+300
0050                IST=IST-NTUPS
0051                RETURN
0052      6000      TYPE 100
0053                GO TO 6010
0054      6001      TYPE 101
0055                GO TO 6010
0056      6002      TYPE 102
0057      6010      NERR=1
0058                RETURN
0059      100       FORMAT(1X,'*** INPUT FILE "NSKIP" FAILURE! ***'/)
0060      101       FORMAT(1X,'*** INPUT FILE "NFILE" FAILURE! ***'/)
0061      102       FORMAT(1X,'*** MT: REWIND FAILURE! ***'/)
0062                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000712	229	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000246	83	RW,D,CON,LCL
$VARS	000210	68	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016266 3675

NO FPP INSTRUCTIONS GENERATED

OPT6,LP/LI:1=OPT6/NOTR

```
FORTRAN IV-PLUS V02-51E         11:42:14     10-AUG-80           PAGE 1
OPT7.FTN          /WR

0001              SUBROUTINE OPT7
0002              IMPLICIT INTEGER(A-Z)
0003              LOGICAL*1 APPEND
0004              COMMON/MTAPE0/NIN(300),NOUT(300)
0005              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0006              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007              COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008              COMMON/MTAPE4/LST,IBEG
0009              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                  1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010              COMMON/MTAPE6/APPEND
0011              DIMENSION ICARD(64)
0012              DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0013              DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0014              DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0015              DATA MASK/0377/
0016              DATA MT0/0,2048,0,0,0,0/
0017              DATA MT1/0,2048,0,0,0,0/
          C
          C       CLOSE INPUT FILE
          C
0018      1300    CONTINUE
0019              NEND=0
0020              NERR=0
0021              IF(NINS.EQ.1)CALL CLOSE(2)
0022              RETURN
0023              END


PROGRAM SECTIONS

 NAME        SIZE                 ATTRIBUTES

SCODE1   000032     13            RW,I,CON,LCL
SPDATA   000004      2            RW,D,CON,LCL
SIDATA   000004      2            RW,D,CON,LCL
SVARS    000200     64            RW,D,CON,LCL
MTAPE0   002260    600            RW,D,OVR,GBL
MTAPE1   000012      5            RW,D,OVR,GBL
MTAPE2   000012      5            RW,D,OVR,GBL
MTAPE3   012260   2648            RW,D,OVR,GBL
MTAPE4   000004      2            RW,D,OVR,GBL
MTAPE5   000064     26            RW,D,OVR,GBL
MTAPE6   000002      1            RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015120   3368

NO FPP INSTRUCTIONS GENERATED

OPT7,LP/LI:1=OPT7/NOTR
```

```
0001            SUBROUTINE OPTR
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 EXTIN,EXTOUT,APPEND
0004            COMMON/MTAPE0/NIN(300),NOUT(300)
0005            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0006            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008            COMMON/MTAPE4/LST,IBEG
0009            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010            COMMON/MTAPE6/APPEND
0011            DIMENSION EXTIN(3),EXTOUT(3)
0012            DIMENSION ICARD(64)
0013            DATA YES,NO/'Y','N'/
0014            DATA EXTIN/'I','N','P'/
0015            DATA EXTOUT/'O','U','T'/
0016            DATA NEND,NERR,NFILE/0,0,0/
0017            DATA NSKIP,IST/1,1/
0018            DATA IOATT,IOSUC,IEALN/O001400,1,-34/
0019            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/O2400,O400,-4,O2440,O1000    /
0020            DATA IOSPF,IEEOF,IOEOF/O2440,-10,O3000/
0021            DATA MASK/O377/
0022            DATA MT0/0,2048,0,0,0,0/
0023            DATA MT1/0,2048,0,0,0,0/
          C
          C     GET FILE INFO(=8)
          C
0024            APPEND=.FALSE.
0025      900   TYPE 100
0026      100   FORMAT(//1H$,'IS THE INPUT ON MAG. TAPE(Y/N)? ')
0027            READ(5,102,END=999,ERR=900)ANSER
0028      102   FORMAT(A1)
0029            NINS=1
0030            IF(ANSER.EQ.NO)NINS=1
0031            NOUTS=1
0032      901   TYPE 103
0033      103   FORMAT(1H$'IS THE OUTPUT GOING TO MAG TAPE(Y/N)? ')
0034            READ(5,102,END=999,ERR=901)ANSER
0035            IF(ANSER.EQ.NO)NOUTS=1
0036            IF(NOUTS.EQ.1)GO TO 5
0037      902   TYPE 104
0038      104   FORMAT(1H$,'APPEND DATA? ')
0039            READ (5,102,END=999,ERR=902)ANSER
0040            IF(ANSER.EQ.YES)APPEND=.TRUE.
0041      5     IF(NOUTS.EQ.0)GO  TO 151
          C     RSX11  SUPPORTED FILE
0042      903   TYPE 105
0043      105   FORMAT(1H$,'OUTPUT FILE NAME= ')
0044            CALL FILEN(3,EXTOUT)
          C
          C     BEGINNING OF INPUT
          C
0045      151   IF(NINS.EQ.1)GO TO 155
0046      904   TYPE 106
0047      106   FORMAT(1H$'MT FILE NO.=(I3) ')
0048            READ(5,101,END=999,ERR=904)NFILE
```

```
101     FORMAT(I3)
        GO TO 14
155     TYPE 107
107     FORMAT(1H$,'INPUT FILE NAME= ')
        CALL FILEN(2,EXTIN)
        NFILE=1
14      RETURN
999     CALL EXIT
        END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000524	170	RW,I,CON,LCL
$PDATA	000010	4	RW,D,CON,LCL
$IDATA	000300	96	RW,D,CON,LCL
$VARS	000214	70	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016126 3627

NO FPP INSTRUCTIONS GENERATED

```
0001                  SUBROUTINE FILEN(UNIT,EXT)
              C       THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
              C       FORM THE TTY DEVICE 5
              C       DEFAULT DEVICE
              C       UNLESS SPECIFIED IN INPUT STRING
              C       UNIT=UNIT NUMBER
              C       EXT = LOGICAL*1 BUFFER OF EXTENSION
              C
0002                  IMPLICIT INTEGER(A-Z)
0003                  LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                  DIMENSION INSTR(40)
0005                  DIMENSION EXT(3)
0006                  DATA BLNK,DOT/' ','.'/
              C
              C       INPUT FILE
0007          152     READ (5,99,END=999,ERR=152)(INSTR(I),I=1,40)
0008          99      FORMAT(40A1)
              C       CHECK FOR END OF LINE
0009                  DO 1600 I=40,1,-1
0010                  J=I
0011          1600    IF(INSTR(I).NE.BLNK)GO TO 1601
0012                  TYPE 151
0013          151     FORMAT(1H$,'>')
0014                  GO TO 152
0015          1601    DO 1602 I=1,J
0016                  IF(INSTR(I).NE.BLNK) GO TO 1602
              C
              C
              C       BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
              C
0017                  DO 1603 K=I,J-1
0018          1603    INSTR(K)=INSTR(K+1)
0019                  INSTR(J)=BLNK
0020                  J=J-1
0021                  GO TO 1601
0022          1602    CONTINUE
0023                  DO 103 I=1,J
0024          103     IF(INSTR(I).EQ.DOT) GO TO 25
0025                  INSTR(J+1)=DOT
0026                  INSTR(J+2)=EXT(1)
0027                  INSTR(J+3)=EXT(2)
0028                  INSTR(J+4)=EXT(3)
0029                  J=J+4
0030          25      CALL SCAN(INSTR,J)
0031                  CALL ASSIGN(UNIT,INSTR,J)
0032                  RETURN
0033          999     CALL EXIT
0034                  END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000472	157	RW,I,CON,LCL
$IDATA	000044	18	RW,D,CON,LCL

```
$VARS   000060    24         RW,D,CON,LCL
$TEMPS  000004     2         RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000622   201

NO FPP INSTRUCTIONS GENERATED
```

```
      C
            SUBROUTINE SCAN(BUF,LTH)
            IMPLICIT INTEGER(A-Z)
            LOGICAL*1 BUF,DEVICE
            DIMENSION BUF(1),DEVICE(4)
            DATA DEVICE/'S','Y','0',':'/
            DO 1 I=1,LTH
      1     IF(BUF(I).EQ.DEVICE(4))RETURN
            LTH=LTH+4
            DO 2 I=LTH,5,-1
      2     BUF(I)=BUF(I-4)
            DO 3 I=1,4
      3     BUF(I)=DEVICE(I)
            RETURN
            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000172	61	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000006	3	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214   70

NO FPP INSTRUCTIONS GENERATED

OPTR,LP/LI:1=OPTR/NOTR

```
0001                  SUBROUTINE OPTR
0002                  IMPLICIT INTEGER(A-Z)
0003                  LOGICAL*1 EXTIN,EXTOUT,APPEND
0004                  COMMON/MTAPE0/NIN(300),NOUT(300)
0005                  COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0006                  COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007                  COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008                  COMMON/MTAPE4/LST,IBEG
0009                  COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                      1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010                  COMMON/MTAPE6/APPEND
0011                  DIMENSION EXTIN(3),EXTOUT(3)
0012                  DIMENSION ICARD(64)
0013                  DATA YES,NO/'Y','N'/
0014                  DATA EXTIN/'I','N','P'/
0015                  DATA EXTOUT/'O','U','T'/
0016                  DATA NEND,NERR,NFILE/0,0,0/
0017                  DATA NSKIP,IST/1,1/
0018                  DATA IOATT,IOSUC,IFALN/0001400,1,-34/
0019                  DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0020                  DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0021                  DATA MASK/0377/
0022                  DATA MT0/0,2048,0,0,0,0/
0023                  DATA MT1/0,2048,0,0,0,0/
              C
              C       GET FILE INFO(=8)
              C
0024                  APPEND=.FALSE.
              C900    TYPE 100
              C100    FORMAT(//1HS,'IS THE INPUT ON MAG. TAPE(Y/N)? ')
              C       READ(5,102,END=999,ERR=900)ANSER
0025                  ANSER=NO
0026          102     FORMAT(A1)
0027                  NINS=1
0028                  IF(ANSER.EQ.NO)NINS=1
0029                  NOUTS=1
              C901    TYPE 103
              C103    FORMAT(1HS'IS THE OUTPUT GOING TO MAG TAPE(Y/N)? ')
              C       READ(5,102,END=999,ERR=901)ANSER
0030                  ANSER=NO
0031                  IF(ANSER.EQ.NO)NOUTS=1
0032                  IF(NOUTS.EQ.1)GO TO 5
0033          902     TYPE 104
0034          104     FORMAT(1HS,'APPEND DATA? ')
0035                  READ (5,102,END=999,ERR=902)ANSER
0036                  IF(ANSER.EQ.YES)APPEND=.TRUE.
0037          5       IF(NOUTS.EQ.0)GO  TO 151
              C       RSX11  SUPPORTED FILE
              C903    TYPE 105
              C105    FORMAT(1HS,'OUTPUT FILE NAME= ')
              C       CALL FILEN(3,EXTOUT)
0038                  CALL ASSIGN(3,'NL:')
              C
              C       BEGINNING OF INPUT
              C
0039          151     IF(NINS.EQ.1)GO TO 155
```

```
      904     TYPE 106
      106     FORMAT(1H$'MT FILE NO.=(I3) ')
              READ(5,101,END=999,ERR=904)NFILE
      101     FORMAT(I3)
              GO TO 14
      155     CONTINUE
      C155    TYPE 107
      C107    FORMAT(1H$,'INPUT FILE NAME= ')
      C       CALL FILEN(2,EXTIN)
              CALL ASSIGN(2,'[5,100]VOICEA.BIN',17)
              NFILE=1
      14      RETURN
      999     CALL EXIT
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000330	108	RW,I,CON,LCL
$PDATA	000042	17	RW,D,CON,LCL
$IDATA	000100	32	RW,D,CON,LCL
$VARS	000214	70	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015564   3514

NO FPP INSTRUCTIONS GENERATED

```
0001                SUBROUTINE FILEN(UNIT,EXT)
        C           THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
        C           FORM THE TTY DEVICE 5
        C           DEFAULT DEVICE
        C           UNLESS SPECIFIED IN INPUT STRING
        C           UNIT=UNIT NUMBER
        C           EXT = LOGICAL*1 BUFFER OF EXTENSION
        C
0002                IMPLICIT INTEGER(A-Z)
0003                LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                DIMENSION INSTR(40)
0005                DIMENSION EXT(3)
0006                DATA BLNK,DOT/' ','.'/
        C
        C           INPUT FILE
0007    152         READ (5,99,END=999,ERR=152)(INSTR(I),I=1,40)
0008    99          FORMAT(40A1)
        C           CHECK FOR END OF LINE
0009                DO 1600 I=40,1,-1
0010                J=I
0011    1600        IF(INSTR(I).NE.BLNK)GO TO 1601
0012                TYPE 151
0013    151         FORMAT(1H$,'>')
0014                GO TO 152
0015    1601        DO 1602 I=1,J
0016                IF(INSTR(I).NE.BLNK) GO TO 1602
        C
        C
        C           BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
        C
0017                DO 1603 K=I,J-1
0018    1603        INSTR(K)=INSTR(K+1)
0019                INSTR(J)=BLNK
0020                J=J-1
0021                GO TO 1601
0022    1602        CONTINUE
0023                DO 103 I=1,J
0024    103         IF(INSTR(I).EQ.DOT) GO TO 25
0025                INSTR(J+1)=DOT
0026                INSTR(J+2)=EXT(1)
0027                INSTR(J+3)=EXT(2)
0028                INSTR(J+4)=EXT(3)
0029                J=J+4
0030    25          CALL SCAN(INSTR,J)
0031                CALL ASSIGN(UNIT,INSTR,J)
0032                RETURN
0033    999         CALL EXIT
0034                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000472	157	RW,I,CON,LCL
$IDATA	000044	18	RW,D,CON,LCL

```
$VARS   000060    24            RW,D,CON,LCL
$TEMPS  000004     2            RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000622   201

NO FPP INSTRUCTIONS GENERATED
```

```
FORTRAN IV-PLUS V02-51F          21:16:27    11-SEP-80           PAGE 5
OPTRA.FTN        /DE/AR

      C
0001            SUBROUTINE SCAN(BUF,LTH)
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 BUF,DEVICE
0004            DIMENSION BUF(1),DEVICE(4)
0005            DATA DEVICE/'S','Y','0',':'/
0006            DO 1 I=1,LTH
0007      1     IF(BUF(I).EQ.DEVICE(4))RETURN
0008            LTH=LTH+4
0009            DO 2 I=LTH,5,-1
0010      2     BUF(I)=BUF(I-4)
0011            DO 3 I=1,4
0012      3     BUF(I)=DEVICE(I)
0013            RETURN
0014            END

PROGRAM SECTIONS

 NAME       SIZE               ATTRIBUTES

SCODE1  000172    61           RW,I,CON,LCL
SIDATA  000012     5           RW,D,CON,LCL
SVARS   000006     3           RW,D,CON,LCL
STEMPS  000002     1           RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214    70

NO FPP INSTRUCTIONS GENERATED

OPTRA,LP/LI:1=OPTRA/DE/NOTR
```

```
        C       PREMAP.FTN      ORIGINATED:29-MAY-79
        C                       UPDATED:04-JUN-79
        C       THIS PROGRAM INTERPRETS ASCII TEXT FILES FOR THE MAP
        C       ASSEMBLER AND ALIGNS ALL FIELDS W/ SPACES AND REMOVES
        C       ALL TABS. ALL FILES ARE ASSUMED TO BE 120 COLUMNS WIDE.
        C       OUTPUT LINE FORMAT:
        C       COLUMNS 1,2...7                 LABEL
        C       COLUMN 8                        BLANK
        C       COLUMNS 9,10...55               SOURCE CODE
        C       COLUMN 56                       BLANK
        C       COLUMNS 57,58...120             COMMENTS
        C
0001            BYTE BLANK,TAB,HYPHEN,STAR,CR,DUMMY,SCOLON
0002            BYTE ALAST
0003            BYTE A(200),L(200)
0004            LOGICAL*1 IN(3),OUT(3),OUTPUT
0005            DATA IN/'T','X','T'/,OUT/'M','A','P'/
0006            DATA BLANK,TAB,HYPHEN,STAR,CR,SCOLON/040,011,047,052,015,073/
        C
0007    2       TYPE 100
0008            CALL FILEN(2,IN)
0009            TYPE 101
0010            CALL FILEN(3,OUT)
0011            LINE=0
        C
        C       READ EACH SOURCE "LINE"
        C
0012    1       LINE=LINE+1
0013            OUTPUT=.TRUE.
0014            READ(2,102,END=200,ERR=300)(A(I),I=1,120)
        C
        C       "BLANK" OUT OUTPUT LINE
        C
0015            DO 10 I=1,120
0016    10      L(I)=BLANK
        C
        C       SPECIAL LINE OPERATORS
        C
0017            IF(A(1).EQ.HYPHEN)GO TO 70
0018            IF(A(1).EQ.STAR)GO TO 70
0019            IF(A(1).EQ.SCOLON)GO TO 70
        C
        C       LABEL FIELD
        C
0020            I=1
0021    11      IF(A(I).EQ.TAB)GO TO 12
0022            L(I)=A(I)
0023            I=I+1
0024            IF(I.GT.120)GO TO 80
0025            GO TO 11
0026    12      NEXT=I+1
        C
        C       COMMAND FIELD
        C
0027            J=0
0028            I=NEXT
```

```
0029    21      IF(A(I).EQ.TAB)GO TO 22
0030            L(9+J)=A(I)
0031            J=J+1
0032            IF(J.GT.111)GO TO 80
0033            I=I+1
0034            IF(I.GT.120)GO TO 80
0035            GO TO 21
0036    22      NEXT=I+1
        C
        C       COMMENT FIELD
        C
0037    30      K=0
0038            I=NEXT
0039    31      IF(A(I).EQ.TAB)GO TO 80
0040            IF(ALAST.EQ.BLANK.AND.A(I).EQ.BLANK)GO TO 80
0041            IF(A(I).EQ.CR)GO TO 80
0042            ALAST=A(I)
0043            OUTPUT=.TRUE.
0044            L(57+K)=A(I)
0045            K=K+1
0046            IF(K.LE.23)GO TO 32
0047            WRITE(3,106)(L(II),II=1,120)
0048            DO 33 II=1,120
0049    33      L(II)=BLANK
0050            L(1)=SCOLON
0051            K=0
0052            OUTPUT=.FALSE.
0053            ALAST=BLANK
0054    32      I=I+1
0055            IF(I.GT.120)GO TO 80
0056            GO TO 31
        C
        C       COPY LINE "EN TOTO"
        C
0057    70      DO 71 I=1,120
0058    71      L(I)=A(I)
        C
        C       GENERATE CORRECTED OUTPUT LINE
        C
0059    80      IF(OUTPUT.EQ..TRUE.)WRITE(3,106)(L(I),I=1,120)
0060            GO TO 1
        C
        C       NORMAL EXIT
        C
0061    200     CALL CLOSE(2)
0062            CALL CLOSE(3)
0063            TYPE 103,LINE
0064            STOP
        C
        C       ABNORMAL EXIT
        C
0065    300     CALL CLOSE(2)
0066            CALL CLOSE(3)
0067            TYPE 104,LINE
0068            GO TO 2
        C
```

```
        C       LINE STRUCTURE ERROR
        C
0069    400     CALL CLOSE(2)
0070            CALL CLOSE(3)
0071            TYPE 105,LINE
0072            STOP
        C
        C       FORMAT DECLARATIONS
        C
0073    100     FORMAT(1HS,'INPUT FILENAME= ')
0074    101     FORMAT(1HS,'OUTPUT FILENAME= ')
0075    102     FORMAT(120A1)
0076    103     FORMAT(1X,'NORMAL EXIT AFTER ',I3,' LINES!'/)
0077    104     FORMAT(1X,'***WARNING*** FILE ERROR IN LINE ',I3/)
0078    105     FORMAT(1X,'***WARNING*** LINE STRUCTURE ERROR IN LINE ',I3/)
0079    106     FORMAT(120A1)
0080    107     FORMAT(1X,100S)
0081            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001374	382	RW,I,CON,LCL
$PDATA	000010	4	RW,D,CON,LCL
$IDATA	000304	98	RW,D,CON,LCL
$VARS	000654	214	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002564   698

NO FPP INSTRUCTIONS GENERATED

PREMAP,LP/LI:1=PREMAP

```
0001                SUBROUTINE FILEN(UNIT,EXT)
           C
           C        THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
           C        FORM THE TTY DEVICE 5
           C        DEFAULT DEVICE
           C        UNLESS SPECIFIED IN INPUT STRING
           C        UNIT=UNIT NUMBER
           C        EXT = LOGICAL*1 BUFFER OF EXTENSION
           C
0002                IMPLICIT INTEGER(A-Z)
0003                LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                DIMENSION INSTR(40)
0005                DIMENSION EXT(3)
0006                DATA BLNK,DOT/' ','.'/
           C
           C        INPUT FILE
0007       152      READ (5,99)(INSTR(I),I=1,40)
0008       99       FORMAT(40A1)
           C        CHECK FOR END OF LINE
0009                DO 1600 I=40,1,-1
0010                J=I
0011       1600     IF(INSTR(I).NE.BLNK)GO TO 1601
0012                TYPE 151
0013       151      FORMAT(1H$,'>')
0014                GO TO 152
0015       1601     DO 1602 I=1,J
0016                IF(INSTR(I).NE.BLNK) GO TO 1602
           C
           C
           C        BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
           C
0017                DO 1603 K=I,J-1
0018       1603     INSTR(K)=INSTR(K+1)
0019                INSTR(J)=BLNK
0020                J=J-1
0021                GO TO 1601
0022       1602     CONTINUE
0023                DO 103 I=1,J
0024       103      IF(INSTR(I).EQ.DOT) GO TO 25
0025                INSTR(J+1)=DOT
0026                INSTR(J+2)=EXT(1)
0027                INSTR(J+3)=EXT(2)
0028                INSTR(J+4)=EXT(3)
0029                J=J+4
0030       25       CALL SCAN(INSTR,J)
0031                CALL ASSIGN(UNIT,INSTR,J)
0032                RETURN
0033                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000444	146	RW,I,CON,LCL
$IDATA	000042	17	RW,D,CON,LCL

```
$VARS   000060    24          RW,D,CON,LCL
$TEMPS  000004     2          RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000572   189

NO FPP INSTRUCTIONS GENERATED
```

```
      C
                 SUBROUTINE SCAN(BUF,LTH)
                 IMPLICIT INTEGER(A-Z)
                 LOGICAL*1 BUF,DEVICE
                 DIMENSION BUF(1),DEVICE(4)
                 DATA DEVICE/'S','Y','0',':'/
                 DO 1 I=1,LTH
      1          IF(BUF(I).EQ.DEVICE(4))RETURN
                 LTH=LTH+4
                 DO 2 I=LTH,5,-1
      2          BUF(I)=BUF(I-4)
                 DO 3 I=1,4
      3          BUF(I)=DEVICE(I)
                 RETURN
                 END
```

```
PROGRAM SECTIONS

 NAME       SIZE                 ATTRIBUTES

$CODE1  000172    61             RW,I,CON,LCL
$IDATA  000012     5             RW,D,CON,LCL
$VARS   000006     3             RW,D,CON,LCL
$TEMPS  000002     1             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214    70

NO FPP INSTRUCTIONS GENERATED

FILEN,LP/LI:1=FILEN/NOTR
```

END

DATE

FILMED

12-80

DTIC

AD-A091 663 GTE PRODUCTS CORP NEEDHAM HEIGHTS MA COMMUNICATION S--ETC F/G 5/8
SPEECH OPTIMIZATION AT 9600 BITS/SECOND. VOLUME 2. REAL-TIME SO--ETC(U)
SEP 80 A J GOLDBERG, L COSELL, S KWON DCA100-78-C-0064

UNCLASSIFIED NL

9 OF 9

AD
A091663

END
DATE
FILMED
12-80
DTIC

```
          C        SCAN DCT LIST FOR ALL SIGNIFICANT POINTS
          C        PRELIMINARY BIT ASSIGNMENT TO TMP1
          C
0027      D        INSTR=1
0028               STATUS=MPSCAN(DCTM2)
0029      D        IF(STATUS.NE.0)GO TO 10
          C
          C        RESULTS
          C
0030      D        CALL MPRST(R0,CTEMP,1,CNVYES)
0031      D        CALL MPRST(R2,BITSMN,1,CNVYES)
0032      D        CALL MPRST(R3,XITOT,1,CNVYES)
0033      D        ITOT=IFIX(XITOT)
0034      D        WRITE(6,930)BITSMN,CTEMP,ITOT
0035      D930     FORMAT(/1X,'BITS= ',F15.8,' W/ CTEMP=',F15.8,' @ ITOT= ',I3)
0036      D        CALL MPRDB(TMP1,XR(1),BYTE4,CNVYES,XR(LTH))
0037      D        WRITE(6,911)(I,IFIX(XR(I)),I=1,LTH)
0038      D911     FORMAT(/1X,6(1X,'TBIT(',I3,')=',I3,2X))
0039               RETURN
          C
          C        DIAGNOSTIC TRAP AREA
          C
0040      10       TYPE 100,INSTR,STATUS
0041               CALL EXIT
0042      100      FORMAT(1X,'***MAP ERROR*** "SDCT" INSTR=',I3,' HAS STATUS=',I6/)
0043               END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000344	114	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000250	84	RW,D,CON,LCL
$VARS	000064	26	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025722 5609

SDCT,LP/LI:1=SDCT/DE/NOTR

```
           C       PROGRAM NAME:BITA.FTN    ORIGINATED:05-FEB-79
           C                                      UPDATE:23-MAY-80
           C
           C
           C
           C
0001               SUBROUTINE BITA
0002               INTEGER BUS1,BUS2,BUS3
0003               INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                   1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004               INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                   1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                   2,DRDCTM,YNM,QRPRM,QRDCTM
                   3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                   4,MIBIT,DCTM,QPRM,QTDCTM
                   5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                   6,RECV1,RECV2,SEN1,SEN2
                   7,AQPB,AQTDCT,AIBIT,SEN,INPB
                   8,SYNB1,SYNB2,BF1,BF2,OUTB
                   9,RECV,AROPB,AROÐCT,RIBIT
                   1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
           C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005               INTEGER ORDR,START,SIZE,DSW,BTLTH
0006               LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007               COMMON/SECTIO/MODUL
0008               COMMON/MTAPE0/NIN(300),NOUT(300)
0009               COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0010               COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011               COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012               COMMON/OVRO/Y(256),NP,XN
0013               COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014               COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015               COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016               COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017               COMMON/OVR2/IORDR(256),LTHM1
0018               COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019               COMMON/OVRF/XR(512)
0020               COMMON/OVRD/CSN,START,SIZE,YES,NO
0021               COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                   1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                   2,DRDCTM,YNM,QRPRM,QRDCTM
                   3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                   4,MIBIT,DCTM,QPRM,QTDCTM
                   5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                   6,RECV1,RECV2,SEN1,SEN2
                   7,AQPB,AQTDCT,AIBIT,SEN,INPB
                   8,SYNB1,SYNB2,BF1,BF2,OUTB
                   9,RECV,AROPB,AROÐCT,RIBIT
                   1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
           C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022               DATA TMP3,TMP4/46,47/
0023               DATA REAL,CMPLX,INTGR/0,1,2/
0024               DATA EVERY,SINGLE,DOUBLE/1,2,3/
0025               DATA BYTE2,BYTE4,BYTE8/2,4,8/
0026               DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0027               DATA BUS1,BUS2,BUS3/64,128,192/
```

```
FORTRAN IV-PLUS V02-51E          13:30:46     09-AUG-80            PAGE 2
BITA.FTN            /DE/WR

            C
            C          ASSIGN BITS TO DCT ELEMENTS
            C
            C          ANALYSIS:MODUL=1,SYNTHESIS:MODUL=2
0028        D          INSTR=1
0029                   IF(MODUL.EQ.1)STATUS=MPCDBA(MIBIT3)
0030                   IF(MODUL.EQ.2)STATUS=MPCDBA(MIBIT1)
0031        D          IF(STATUS.NE.0)GO TO 10
            C
            C          RESULTS
            C
0032        D          IF(MODUL.EQ.1)CALL MPRDB(MIBIT3,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0033        D          IF(MODUL.EQ.2)CALL MPRDB(MIBIT1,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0034        D          IF(STATUS.NE.0)GO TO 10
0035        D          WRITE(6,911)(I,IBIT(I),I=1,LTH)
0036        D911       FORMAT(/1X,6(1X,'IBIT(',I3,')=',I3,2X))
0037                   RETURN
            C
            C          DIAGNOSTIC TRAP AREA
            C
0038        10         TYPE 100,INSTR,STATUS
0039                   CALL EXIT
0040        100        FORMAT(1X,'***MAP ERROR*** "BITA" INSTR=',I3,' HAS STATUS=',I6/)
0041                   END

PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1   000334   110           RW,I,CON,LCL
$IDATA   000160    56           RW,D,CON,LCL
$VARS    000054    22           RW,D,CON,LCL
SECTIO   000002     1           RW,D,OVR,GBL
MTAPE0   002260   600           RW,D,OVR,GBL
MTAPE1   000012     5           RW,D,OVR,GBL
MTAPE2   000012     5           RW,D,OVR,GBL
OVRI     002020   520           RW,D,OVR,GBL
OVRO     002006   515           RW,D,OVR,GBL
OVRF     006016  1543           RW,D,OVR,GBL
OVRA     000152    53           RW,D,OVR,GBL
OVRP     002022   521           RW,D,OVR,GBL
OVRB     005004  1282           RW,D,OVR,GBL
OVR2     001002   257           RW,D,OVR,GBL
OVRL     000016     7           RW,D,OVR,GBL
OVRD     000012     5           RW,D,OVR,GBL
MAPDEF   000200    64           RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025574  5566

NO FPP INSTRUCTIONS GENERATED

BITA,LP/LI:1=BITA/DE/NOTR
```

```
        C         PROGRAM NAME:QDCT.FTN    ORIGINATED:10-SEP-79
        C                                        UPDATE:23-MAY-80
        C
        C
        C
        C
0001              SUBROUTINE QDCT
0002              INTEGER BUS1,BUS2,BUS3
0003              INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                  1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004              INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,QTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,ARQDCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C         <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005              INTEGER ORDR,START,SIZE,DSW,BTLTH
0006              INTEGER QTDCT,QTPAR,QTDC,QTVAR,QTM,QTG,QTV
0007              LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008              COMMON/MTAPE0/NIN(300),NOUT(300)
0009              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0010              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0011              COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012              COMMON/OVRO/Y(256),NP,XN
0013              COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014              COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015              COMMON/OVRP/M,G,V0,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016              COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017              COMMON/OVR2/IORDR(256),LTHM1
0018              COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019              COMMON/OVRF/XR(512)
0020              COMMON/OVRD/CSN,START,SIZE,YES,NO
0021              COMMON/OVRQT/QTDCT(256),QTPAR(8),QTDC,QTVAR,QTM,QTG,QTV
0022              COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0023              COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,QTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,ARQDCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C         >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0024              DATA REAL,CMPLX,INTGR/0,1,2/
0025              DATA EVERY,SINGLE,DOUBLE/1,2,3/
0026              DATA BYTE2,BYTE4,BYTE8/2,4,8/
0027              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
```

```
0028                DATA BUS1,BUS2,BUS3/64,128,192/
          C
          C         QUANTIZE DCT(.) W/ IBIT(.) ACCORDING TO IORDR(.)
          C
0029      D         INSTR=1
0030                STATUS=MPESTQ(QTDCTM,MIBIT3,TMP1,TMP2)
0031      D         IF(STATUS.NE.0)GO TO 10
          C
          C         RESULTS
          C
0032      D         DO 21 I=1,LTH
0033      D21       IORDR(I)=IORDR(I)+1
0034      D         CALL MPRDB(QTDCTM,NOUT(1),BYTE2,CNVYES,NOUT(LTH))
0035      D         DO 22 I=1,LTH
0036      D         J=IORDR(I)
0037      D         QTDCT(J)=NOUT(I)
0038      D22       NOUT(I)=0
0039      D         WRITE(6,912)(I,QTDCT(I),I=1,LTH)
0040      D912      FORMAT(/1X,4(1X,'QTDCT(',I3,')=',I3,2X))
0041      D         CALL MPRDB(MIBIT3,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0042      D         WRITE(6,913)(I,IBIT(I),I=1,LTH)
0043      D913      FORMAT(/1X,4(1X,'MIBIT3(',I3,')=',I3,2X))
0044                RETURN
          C
          C         DIAGNOSTIC TRAP AREA
          C
0045      10        TYPE 100,INSTR,STATUS
0046                CALL EXIT
0047      100       FORMAT(1X,'***MAP ERROR*** "ODCT" INSTR=',I3,' HAS STATUS=',I6/)
0048                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000446	147	RW,I,CON,LCL
$IDATA	000216	71	RW,D,CON,LCL
$VARS	000056	23	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
OVROT	001032	269	RW,D,OVR,GBL
OVRDT	002124	554	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

```
FORTRAN IV-PLUS V02-51E          13:30:52    09-AUG-80          PAGE 3
QDCT.FTN          /DE/WR

TOTAL SPACE ALLOCATED = 031122  6441

NO FPP INSTRUCTIONS GENERATED

QDCT,LP/LI:1=QDCT/DE/NOTR
```

```
        C       PROGRAM NAME:CHANL.FTN  ORIGINATED:10-SEP-79
        C                                          UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE CHANL
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/CHANEL/IAQPB(14),IAQDCT(256)
0011            COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRE/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,AROPB,ARODCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022            DATA REAL,CMPLX,INTGR/0,1,2/
0023            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026            DATA BUS1,BUS2,BUS3/64,128,192/
        C
```

```
        D           WRITE(6,699)ICOUNT
        D699        FORMAT(1H1,40(1H>),'RECEIVE FRAME # ',I4,40(1H<))
        C
        C           PASS DATA W/ NO ERRORS
        C
        D           INSTR=1
                    STATUS=MPWDB(ARQDCT,IAQDCT(1),BYTE2,CNVYES,IAQDCT(LTH))
        D           IF(STATUS.NE.0)GO TO 10
        D           INSTR=2
                    STATUS=MPWDB(ARQPB,IAQPB(1),BYTE2,CNVYES,IAQPB(LPARM+1))
        D           IF(STATUS.NE.0)GO TO 10
        D           WRITE(6,899)
        D899        FORMAT(/1X,'*** MODEM OUTPUT ***')
        D           WRITE(6,900)(IAQPB(I),I=1,LPARM)
        D900        FORMAT(/1X,6(1X,'ARQPB(',I2,')=',I6,2X))
        D           WRITE(6,901)(IAQDCT(I),I=1,LTH)
        D901        FORMAT(/1X,6(1X,'ARQDCT(',I3,')=',I6,2X))
        C
        C           ADD TRANSMISSION DELAY
        C
        D           INSTR=3
                    STATUS=MPRDB(AQTDCT,IAQDCT(1),BYTE2,CNVYES,IAQDCT(LTH))
        D           IF(STATUS.NE.0)GO TO 10
        D           INSTR=4
                    STATUS=MPRDB(AQPB,IAQPB(1),BYTE2,CNVYES,IAQPB(LPARM+1))
        D           IF(STATUS.NE.0)GO TO 10
        D           WRITE(6,799)
        D799        FORMAT(/1X,'*** MODEM INPUT ***')
        D           WRITE(6,800)(IAQPB(I),I=1,LPARM)
        D800        FORMAT(/1X,6(1X,'AQPB(',I2,')=',I6,2X))
        D           WRITE(6,801)(IAQDCT(I),I=1,LTH)
        D801        FORMAT(/1X,6(1X,'AQTDCT(',I3,')=',I6,2X))
                    RETURN
        C
        C           DIAGNOSTIC TRAP AREA
        C
        10          TYPE 100,INSTR,STATUS
                    CALL EXIT
        100         FORMAT(1X,'***MAP ERROR*** "CHANL" INSTR=',I3,' HAS STATUS=',I6/)
                    END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000644	210	RW,I,CON,LCL
$IDATA	000452	149	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
CHANEL	001034	270	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRD	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL

```
CHANL.FTN        /DE/WR

OVRA     000152    53           RW,D,OVR,GBL
OVRP     002022   521           RW,D,OVR,GBL
OVRB     005004  1282           RW,D,OVR,GBL
OVR2     001002   257           RW,D,OVR,GBL
OVRL     000016     7           RW,D,OVR,GBL
OVRD     000012     5           RW,D,OVR,GBL
MAPDEF   000200    64           RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 027430  6028

NO FPP INSTRUCTIONS GENERATED

CHANL.LP/LI:1=CHANL/DE/NOTR
```

```
        C       PROGRAM NAME:DPARM.FTN  ORIGINATED:10-SEP-79
        C                                      UPDATE:13-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE DPARM
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            INTEGER QRDCT,QRPAR,QRDC,QRVAR,QRM,QRG,QRV
0007            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0008            COMMON/MTAPEO/NIN(300),NOUT(300)
0009            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0010            COMMON/MTAPE2/NEND,NFRR,NFILE,NINS,NOUTS
0011            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0012            COMMON/OVRO/Y(256),NP,XN
0013            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0014            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0015            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0016            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0017            COMMON/OVR2/IORDR(256),LTHM1
0018            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0019            COMMON/OVRF/XR(512)
0020            COMMON/OVRD/CSN,START,SIZE,YES,NO
0021            COMMON/OVRDT/DRDCT(256),DRPAR(8),DRDC,DRVAR,DRM,DRG,DRV,DRA(8)
0022            COMMON/OVRQT/QRDCT(256),QRPAR(8),QRDC,QRVAR,QRM,QRG,QRV
0023            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,OPRM,OTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,OTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0024            DATA REAL,CMPLX,INTGR/0,1,2/
0025            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0026            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0027            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
```

```
0028                DATA BUS1,BUS2,BUS3/64,128,192/
          C
          C         SIDEBAND DEQUANTIZATION LAGS SIDEBAND DESERIALIZATION
          C         BY 1 FRAME DUE TO CSPU IN BACKGROUND
          C
0029      D         INSTR=1
0030                STATUS=VMOV2(ORPRM)
0031      D         IF(STATUS.NE.0)GO TO 10
          C
          C         DEQUANTIZE PARCOR(J),J=1,...,LPCN,G,M,VUV
          C
0032      D         INSTR=2
0033                STATUS=MPDQPP(PARC,ORPRM,A1)
0034      D         IF(STATUS.NE.0)GO TO 10
          C
          C         RESULTS
          C
0035      D         CALL MPRDB(ORPRM,IBIT(1),BYTE2,CNVYES,IBIT(LPARM+1))
0036      D         WRITE(6,927)(J,IBIT(J),J=1,LPARM)
0037      D927      FORMAT(/1X,4(1X,'ORPRM(',I2,')=',I6,2X))
0038      D         CALL MPRDB(PARC,DRPAR(1),BYTE4,CNVYES,DRPAR(LPCN))
0039      D         WRITE(6,928)(J,DRPAR(J),J=1,LPCN)
0040      D928      FORMAT(/1X,4(1X,'DRK(',I2,')=',E15.8,2X))
0041      D         CALL MPRST(88,DRDC,1,CNVYES)
0042      D         CALL MPRST(89,DRVAR,1,CNVYES)
0043      D         CALL MPRST(90,DRG,1,CNVYES)
0044      D         CALL MPRST(91,DRM,1,CNVYES)
0045      D         CALL MPRST(104,DRV,1,CNVYES)
0046      D         WRITE(6,925)DRDC,DRVAR
0047      D925      FORMAT(1X,'DRDC(88)=',E15.8,' AND DRVAR(89)=',E15.8)
0048      D         WRITE(6,929)DRM,DRG,DRV
0049      D929      FORMAT(/1X,'DRM=',E15.8,', DRG=',E15.8,' AND DRV=',E15.8)
0050      D         CALL MPRDB(A1,DRA(1),BYTE4,CNVYES,DRA(LPCN))
0051      D         WRITE(6,931)(J,DRA(J),J=1,LPCN)
0052      D931      FORMAT(/1X,4(1X,'DRA(',I2,')=',E15.8,2X))
0053      D         CALL MPRST(60,ENG,1,CNVYES)
0054      D         WRITE(6,932)ENG
0055      D932      FORMAT(/1X,'ENG=',E15.8)
0056      D         CALL MPRDB(ORDCTM,ORDCT(1),BYTE2,CNVYES,ORDCT(LTH))
0057      D         WRITE(6,940)(I,ORDCT(I),I=1,LTH)
0058      D940      FORMAT(/1X,4(1X,'ORDCTM(',I3,')=',I6,2X))
0059      D         CALL MPRDB(MIBIT2,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0060      D         WRITE(6,941)(I,IBIT(I),I=1,LTH)
0061      D941      FORMAT(/1X,4(1X,'MIBIT2(',I3,')=',I6,2X))
0062      D         CALL MPRDB(IORDR2,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0063      D         WRITE(6,942)(I,IORDR(I),I=1,LTH)
0064      D942      FORMAT(/1X,4(1X,'IORDR2(',I3,')=',I6,2X))
0065      D         CALL MPRDB(DCTIT2,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0066      D         WRITE(6,943)(I,DCT1(I),I=1,LTH)
0067      D943      FORMAT(/1X,4(1X,'DCT1(',I3,')=',E15.8,2X))
0068      D         CALL MPRST(100,DRDC,1,CNVYES)
0069      D         CALL MPRST(101,DRVAR,1,CNVYES)
0070      D         WRITE(6,944)DRDC,DRVAR
0071      D944      FORMAT(1X,'DRDC(100)=',E15.8,' AND VAR(101)=',E15.8)
0072                RETURN
          C
```

```
        C       DIAGNOSTIC TRAP AREA
        C
0073    10      TYPE 100,INSTR,STATUS
0074            CALL EXIT
0075    100     FORMAT(1X,'***MAP ERROR*** "DPARM" INSTR=',I4,' HAS STATUS=',I6/)
0076            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001542	433	RW,I,CON,LCL
$PDATA	000044	18	RW,D,CON,LCL
$IDATA	001036	271	RW,D,CON,LCL
$VARS	000056	23	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRT	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
OVRDT	002124	554	RW,D,OVR,GBL
OVRQT	001032	269	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 033102 6945

DPARM.LP/LI:1=DPARM/DE/NOTR

```
        C       PROGRAM NAME:SSRT.FTN   ORIGINATED:10-SEP-79
        C                                   UPDATE:07-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE SSRT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,ORPRM,ORDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       ORDER DCT1(,) MAX-TO-MIN W/ ORDERING INFO IN IORDR(,)
```

```
        C       SORT TMP2 TO DCTM2
        C       SORT DCTM1 TO TMP4
        C
0026    D       INSTR=1
0027            STATUS=MPSSRT(IORDRM)
0028    D       IF(STATUS.NE.0)GO TO 10
        C
        C       RESULTS
        C
0029    D       CALL MPRDB(DCTM2,DCT2(1),BYTE4,CNVYES,DCT2(LTH))
0030    D       CALL MPRDB(IORDRM,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0031    D       WRITE(6,910)(I,IORDR(I),I=1,LTH)
0032    D910    FORMAT(/1X,6(1X,'IORDR(',I3,')=',I3,2X))
0033    D       WRITE(6,909)(I,DCT2(I),I=1,LTH)
0034    D909    FORMAT(/1X,4(1X,'DCT2(',I3,')=',E15.8,2X))
0035    D       CALL MPRDB(TMP4,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0036    D       WRITE(6,911)(I,DCT1(I),I=1,LTH)
0037    D911    FORMAT(/1X,4(1X,'DCT1T(',I3,')=',E15.8,2X))
0038            RETURN
        C
        C       DIAGNOSTIC TRAP AREA
        C
0039    10      TYPE 100,INSTR,STATUS
0040            CALL EXIT
0041    100     FORMAT(1X,'***MAP ERROR*** "SSRT" INSTR=',I3,' HAS STATUS=',I6/)
0042            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000470	156	RW,I,CON,LCL
$IDATA	000256	87	RW,D,CON,LCL
$VARS	000054	22	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026024 5642

NO FPP INSTRUCTIONS GENERATED

SSRT,LP/LI:1=SSRT/DE/NOTR

```
        C         PROGRAM NAME:DDCT.FTN   ORIGINATED:05-FEB-79
        C                                           UPDATE:13-MAY-80
        C
        C
        C
        C
0001              SUBROUTINE DDCT
0002              INTEGER BUS1,BUS2,BUS3
0003              INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                  1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004              INTEGER WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,QTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,AROOCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C         <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005              INTEGER ORDR,START,SIZE,DSW,BTLTH
0006              LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007              COMMON/MTAPE0/NIN(300),NOUT(300)
0008              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010              COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011              COMMON/OVRO/Y(256),NP,XN
0012              COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013              COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014              COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015              COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016              COMMON/OVR2/IORDR(256),LTHM1
0017              COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018              COMMON/OVRF/XR(512)
0019              COMMON/OVRD/CSN,START,SIZE,YES,NO
0020              COMMON/OVRDT/DTDCT(256),DTPAR(8),DTDC,DTVAR,DTM,DTG,DTV,DTA(8)
0021              COMMON/MAPDEF/WORK,X1,X1R,X11,IORDRM,PWEITM,DCTM1
                  1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                  2,DRDCTM,YNM,QRPRM,QRDCTM
                  3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                  4,MIBIT,DCTM,QPRM,QTDCTM
                  5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                  6,RECV1,RECV2,SEN1,SEN2
                  7,AQPB,AQTDCT,AIBIT,SEN,INPB
                  8,SYNB1,SYNB2,BF1,BF2,OUTB
                  9,RECV,AROPB,AROOCT,RIBIT
                  1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C         >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0022              DATA REAL,CMPLX,INTGR/0,1,2/
0023              DATA EVERY,SINGLE,DOUBLE/1,2,3/
0024              DATA BYTE2,BYTE4,BYTE8/2,4,8/
0025              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0026              DATA BUS1,BUS2,BUS3/64,128,192/
        C
```

```
            C          DEQUANTIZATION OF MAINBAND LAGS MAINBAND DESERIALIZATION
            C          BY 1 FRAME DUE TO CSPU IN THE BACKGROUND
            C
0027        D          INSTR=1
            C          GIVE CSPU BIT ASSIGNMENT FOR UNPACKING OF MAINBAND
0028                   STATUS=VMOV3(RIBIT)
0029        D          IF(STATUS.NE.0)GO TO 10
            C
            C          DEQUANTIZE DCT(I)
            C
0030        D          INSTR=2
0031                   STATUS=MPESTD(DRDCTM,ORDCTM,TMP1,IORDR2)
0032        D          IF(STATUS.NE.0)GO TO 10
            C
            C          RESULTS
            C
0033        D          CALL MPRDB(DRDCTM,DTDCT(1),BYTE4,CNVYES,DTDCT(LTH))
0034        D          WRITE(6,934)(I,DTDCT(I),I=1,LTH)
0035        D934       FORMAT(/1X,4(1X,'DRDCTM(',I3,')=',E15.8,2X))
0036        D          CALL MPRDB(RIBIT,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0037        D          WRITE(6,935)(I,IBIT(I),I=1,LTH)
0038        D935       FORMAT(/1X,4(1X,'RIBIT(',I3,')=',I6,2X))
0039        D          CALL MPRDB(DCT1T1,DCT1(1),BYTE4,CNVYES,DCT1(LTH))
0040        D          WRITE(6,936)(I,DCT1(I),I=1,LTH)
0041        D936       FORMAT(/1X,4(1X,'DCT1T1(',I3,')=',E15.8,2X))
0042        D          CALL MPRDB(IORDR1,IORDR(1),BYTE2,CNVYES,IORDR(LTH))
0043        D          WRITE(6,937)(I,IORDR(I),I=1,LTH)
0044        D937       FORMAT(/1X,4(1X,'IORDR1(',I3,')=',I6,2X))
0045        D          CALL MPRDB(MIBIT1,IBIT(1),BYTE2,CNVYES,IBIT(LTH))
0046        D          WRITE(6,938)(I,IBIT(I),I=1,LTH)
0047        D938       FORMAT(/1X,4(1X,'MIBIT1(',I3,')=',I6,2X))
0048        D          CALL MPRST(102,DRDC,1,CNVYES)
0049        D          CALL MPRST(103,DRVAR,1,CNVYES)
0050        D          WRITE(6,940)DRDC,DRVAR
0051        D940       FORMAT(1X,'DRDC(102)=',F15.8,2X,' AND DRVAR(103)=',E15.8)
0052                   RETURN
            C
            C          DIAGNOSTIC TRAP AREA
            C
0053        10         TYPE 100,INSTR,STATUS
0054                   CALL EXIT
0055        100        FORMAT(1X,'***MAP ERROR*** "DDCT" INSTR=',I4,' HAS STATUS=',I6/)
0056                   END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	001056	279	RW,I,CON,LCL
SPDATA	000014	6	RW,D,CON,LCL
SIDATA	000506	163	RW,D,CON,LCL
SVARS	000064	26	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL

```
OVRI     002020    520          RW,D,OVR,GBL
OVRO     002006    515          RW,D,OVR,GBL
OVRF     006016   1543          RW,D,OVR,GBL
OVRA     000152     53          RW,D,OVR,GBL
OVRP     002022    521          RW,D,OVR,GBL
OVRR     005004   1282          RW,D,OVR,GBL
OVR2     001002    257          RW,D,OVR,GBL
OVRL     000016      7          RW,D,OVR,GBL
OVRD     000012      5          RW,D,OVR,GBL
OVRDT    002124    554          RW,D,OVR,GBL
MAPDEF   000200     64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 031012  6405

DDCT,LP/LI:1=DDCT/DE/NOTR
```

```
        C       PROGRAM NAME:DCTR.FTN    ORIGINATED:05-FEB-79
        C                                          UPDATE:12-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE DCTR
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DSW,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRU/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014            COMMON/OVRP/M,G,VU,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRF/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       FORM INPHASE & QUADRATURE COMPONENTS
```

```
            C
0026        D          INSTR=1
0027                   STATUS=MPIDCM(X2,DRDCTM,COSZ)
0028        D          IF(STATUS.NE.0)GO TO 10
0029        D          CALL MPRDB(X2,XR(1),BYTE8,CNVYES,XR(LTH2))
0030        D          WRITE(6,914)(I,XR(I),I=1,LTH2)
0031        D          INSTR=2
0032                   STATUS=FFTIN(X2,1,X2,VSHRT,WORK)
0033        D          IF(STATUS.NE.0)GO TO 10
            C
            C          RESULTS
            C
0034        D          CALL MPRDB(X2,XR(1),BYTE4,CNVYES,XR(LTH))
0035        D          WRITE(6,914)(I,XR(+I),I=1,LTH)
0036        D914       FORMAT(/1X,4(1X,'X(',I3,')=',E15.8,2X))
0037                   RETURN
            C
            C          DIAGNOSTIC TRAP AREA
            C
0038        10         TYPE 100,INSTR,STATUS
0039                   CALL EXIT
0040        100        FORMAT(1X,'***MAP ERROR*** "DCTR" INSTR=',I3,' HAS STATUS=',I6/)
0041                   END


PROGRAM SECTIONS

 NAME       SIZE                  ATTRIBUTES

SCODE1   000406    131            RW,I,CON,LCL
SPDATA   000004      2            RW,D,CON,LCL
SIDATA   000172     61            RW,D,CON,LCL
SVARS    000054     22            RW,D,CON,LCL
MTAPE0   002260    600            RW,D,OVR,GBL
MTAPE1   000012      5            RW,D,OVR,GBL
MTAPE2   000012      5            RW,D,OVR,GBL
OVRI     002020    520            RW,D,OVR,GBL
OVRO     002006    515            RW,D,OVR,GBL
OVRF     006016   1543            RW,D,OVR,GBL
OVRA     000152     53            RW,D,OVR,GBL
OVRP     002022    521            RW,D,OVR,GBL
OVRB     005004   1282            RW,D,OVR,GBL
OVR2     001002    257            RW,D,OVR,GBL
OVRL     000016      7            RW,D,OVR,GBL
OVRD     000012      5            RW,D,OVR,GBL
MAPDEF   000200     64            RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025662  5593

DCTR,LP/LI:1=DCTR/DE/NOTR
```

```
FORTRAN IV-PLUS V02-51F        13:31:39    09-AUG-80          PAGE 1
VARDC.FTN          /DE/WR

        C        PROGRAM NAME:VARDC.FTN          ORIGINATED:17-SEP-79
        C                                               UPDATE:09-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE VARDC
0002             INTEGER BUS1,BUS2,BUS3
0003             INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004             INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,ORPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,OPRM,OTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AOPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,AROPB,ARODCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005             INTEGER ORDR,START,SIZE,DSW,BTLTH
0006             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007             COMMON/MTAPE0/NIN(300),NOUT(300)
0008             COMMON/MTAPE1/NSKIP,IST,NTOT1,NTUPS,NTOTO
0009             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010             COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011             COMMON/OVRO/Y(256),NP,XN
0012             COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0014             COMMON/OVRP/M,G,VU,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016             COMMON/OVR2/IORDR(256),LTHM1
0017             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018             COMMON/OVRF/XR(512)
0019             COMMON/OVRD/CSN,START,SIZE,YES,NO
0020             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,OPRM,OTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AOPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,AROPB,ARODCT,RIBIT
                 1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021             DATA REAL,CMPLX,INTGR/0,1,2/
0022             DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023             DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024             DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025             DATA BUS1,BUS2,BUS3/64,128,192/
        C
0026    D        INSTR=1
```

```
0027                STATUS=MPVDNM(NOUTM,YOM,X2)
0028     D          IF(STATUS.NE.0)GO TO 10
         C
         C          RESULTS
         C
0029     D          CALL MPRST(100,DCBIAS,1,CNVYES)
0030     D          WRITE(6,900)DCBIAS
0031     D900       FORMAT(/1X,'DCBIAS=',F15.8)
0032     D          CALL MPRST(101,VAR,1,CNVYES)
0033     D          WRITE(6,901)VAR
0034     D901       FORMAT(/1X,'VAR=',E15.8)
0035                RETURN
         C
         C          DIAGNOSTIC TRAP AREA
         C
0036     10         TYPE 101,INSTR,STATUS
0037                CALL EXIT
0038     101        FORMAT(1X,'***MAP ERROR*** "VARDC" INSTR=',I3,' HAS STATUS=',I6/)
0039                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000200	64	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000160	56	RW,D,CON,LCL
$VARS	000052	21	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
OVRI	002020	520	RW,D,OVR,GBL
OVRO	002006	515	RW,D,OVR,GBL
OVRF	006016	1543	RW,D,OVR,GBL
OVRA	000152	53	RW,D,OVR,GBL
OVRP	002022	521	RW,D,OVR,GBL
OVRB	005004	1282	RW,D,OVR,GBL
OVR2	001002	257	RW,D,OVR,GBL
OVRL	000016	7	RW,D,OVR,GBL
OVRD	000012	5	RW,D,OVR,GBL
MAPDEF	000200	64	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025450   5524

NO FPP INSTRUCTIONS GENERATED

VARDC,LP/LI:1=VARDC/DE/NOTR

```
        C       PROGRAM NAME:OUTPUT.FEG ORIGINATED:05-FEB-79
        C                                           UPDATE:09-MAY-80
        C
        C
        C
        C
0001            SUBROUTINE OUTPUT
0002            INTEGER BUS1,BUS2,BUS3
0003            INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0004            INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0005            INTEGER ORDR,START,SIZE,DS*,BTLTH
0006            LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0007            COMMON/MTAPE0/NIN(300),NOUT(300)
0008            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0009            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0010            COMMON/OVRI/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0011            COMMON/OVRO/Y(256),NP,XN
0012            COMMON/OVRF/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0013            COMMON/OVRA/R(9),A(8),PARCOR(8),GPCN,ENG
0014            COMMON/OVRP/M,G,PWFIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0015            COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0016            COMMON/OVR2/IORDR(256),LTHM1
0017            COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0018            COMMON/OVRE/XR(512)
0019            COMMON/OVRD/CSN,START,SIZE,YES,NO
0020            COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                1,DCTM2,R1,A1,KE,PARC,TMP1,TMP2,TMP3,TMP4,X2
                2,DRDCTM,YNM,QRPRM,QRDCTM
                3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                4,MIBIT,DCTM,QPRM,QTDCTM
                5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                6,RECV1,RECV2,SEN1,SEN2
                7,AQPB,AQTDCT,AIBIT,SEN,INPB
                8,SYNB1,SYNB2,BF1,BF2,OUTB
                9,RECV,ARQPB,ARQDCT,RIBIT
                1,DCT1T1,DCT1T2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C       >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0021            DATA REAL,CMPLX,INTGR/0,1,2/
0022            DATA EVERY,SINGLE,DOUBLE/1,2,3/
0023            DATA BYTE2,BYTE4,BYTE8/2,4,8/
0024            DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
0025            DATA BUS1,BUS2,BUS3/64,128,192/
        C
        C       TRANSFER TO TAPE OUTPUT VECTOR
```

```
          C
          C       CHECK FOR REQUEST FROM "INPUT"
0026              IF(NEND.NE.0)GO TO 5000
0027      D       INSTR=1
0028              STATUS=MPAT(1,1)
0029              STATUS=MPRDB(NOUTR,NOUT(1),BYTE2,CNVYES,NOUT(NTOTO))
0030              IF(STATUS.NE.0)GO TO 10
          CD      WRITE(6,913)(I,NOUT(I),I=1,NTOTO)
          CD913   FORMAT(/1X,4(1X,'NOUT(',I3,')=',I6,2X))
          C
          C       DATA OUTPUT
          C
0031              CALL TAPE2(2)
0032              RETURN
          C
          C       EOF ON INPUT,WRITE EOF
          C
0033      5000    ICOUNT=ICOUNT-1
0034              DO 5001 I=1,NTOPS
0035      5001    NOUT(I)=0
0036              DO 5002 I=1,32
0037      5002    CALL TAPE2(2)
0038              NSKIP=NSKIPS
0039              NEND=0
0040              IST=1
0041              CALL TAPE2(6)
0042              DO 5005 I=1,ICOUNT
0043              CALL TAPE2(1)
0044              IF(NEND)5077,5076,5077
0045      5076    DO 5006 J=1,NTOPS
0046      5006    NOUT(J)=NIN(J)
0047      5005    CALL TAPE2(2)
0048      5077    DO 5007 I=1,NTOPS
0049      5007    NOUT(I)=0
0050              DO 5008 I=1,8
0051      5008    CALL TAPE2(2)
0052              CALL TAPE2(5)
          C       CALL MPCLS(0)
          C       TYPE 100,ICOUNT
          C100    FORMAT(1X,'"ATC" DONE AFTER ',I6,' FRAMES!'/)
          C       TYPE 100
          C100    FORMAT(1X,'ATC SYSTEM READY')
          C       CALL EXIT
0053              RETURN
          C
          C       DIAGNOSTIC TRAP AREA
          C
0054      10      TYPE 101,INSTR,STATUS
0055              CALL EXIT
0056      101     FORMAT(1X,'***MAP ERROR*** "OUTPUT" INSTR=',I3,' HAS STATUS=',I6/)
0057              END
```

PROGRAM SECTIONS

NAME	SIZE	ATTRIBUTES

```
SCODE1   000466   155      RW,I,CON,LCL
SPDATA   000020     8      RW,D,CON,LCL
SIDATA   000132    45      RW,D,CON,LCL
SVARS    000056    23      RW,D,CON,LCL
STEMPS   000002     1      RW,D,CON,LCL
ATAPE0   002260   600      RW,D,OVR,GBL
ATAPE1   000012     5      RW,D,OVR,GBL
MTAPE2   000012     5      RW,D,OVR,GBL
OVRI     002020   520      RW,D,OVR,GBL
OVRO     002006   515      RW,D,OVR,GBL
OVRF     006016  1543      RW,D,OVR,GBL
OVRA     000152    53      RW,D,OVR,GBL
OVRP     002016   519      RW,D,OVR,GBL
OVRB     005004  1282      RW,D,OVR,GBL
OVR2     001002   257      RW,D,OVR,GBL
OVRL     000016     7      RW,D,OVR,GBL
OVRD     000012     5      RW,D,OVR,GBL
MAPDEF   000200    64      RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 025716  5607

NO FPP INSTRUCTIONS GENERATED

OUTPUT,LP/LI:1=OUTPUT/DE/NOTR
```

```
        C        PROGRAM NAME:SNR.FTN     ORIGINATED:05-FEB-79
        C                                          UPDATE:13-MAY-80
        C
        C
        C
        C
0001             SUBROUTINE SNR
0002             REAL NOISE
0003             INTEGER BUS1,BUS2,BUS3
0004             INTEGER REAL,CMPLX,INTGR,EVERY,SINGLE,DOUBLE,BYTE2
                 1,BYTE4,BYTE8,CNVYES,CNVNO,UNITY,STATUS,SHORT
0005             INTEGER WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
        C        <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0006             INTEGER ORDR,START,SIZE,DSW,BTLTH
0007             DIMENSION NIND3(246)
0008             LOGICAL*1 TIMING,RLTIM,LIVE,ANSER,YES,NO
0009             COMMON/MTAPE0/NIN(300),NOUT(300)
0010             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0011             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0012             COMMON/DELAY/M4,M3,M2,M1,IV4,IV3,IV2,IV1
0013             COMMON/OVR1/LTH,DCBIAS,VAR,X(256),ICOUNT,IREC,NSKIPS
0014             COMMON/OVRO/Y(256),NP,XN
0015             COMMON/OVRE/DCT(256),LTH2,LTH3,LTH4,ARG,MLONG,MSHRT
0016             COMMON/OVRA/R(9),A(8),PARCOR(8),LPCN,ENG
0017             COMMON/OVRP/M,G,VB,PWEIT(256),TIMING,RLTIM,LIVE,ANSER,LSEN,LPARM
0018             COMMON/OVRB/DCT2(256),DLOG2,DCT1(256),IBIT(256)
0019             COMMON/OVR2/IORDR(256),LTHM1
0020             COMMON/OVRL/SLOG,LNGTH,CTEMP,ITOT,BTLTH
0021             COMMON/OVRF/XR(512)
0022             COMMON/OVRD/CSN,START,SIZE,YES,NO
0023             COMMON/MAPDEF/WORK,X1,X1R,X1I,IORDRM,PWEITM,DCTM1
                 1,DCTM2,R1,A1,KF,PARC,TMP1,TMP2,TMP3,TMP4,X2
                 2,DRDCTM,YNM,QRPRM,QRDCTM
                 3,VLONG,COSZ,VSHRT,YOM,XOM,MAPX
                 4,MIBIT,DCTM,QPRM,QTDCTM
                 5,NINM,NIN1,NIN2,NOUTM,NOUT1,NOUT2
                 6,RECV1,RECV2,SEN1,SEN2
                 7,AQPB,AQTDCT,AIBIT,SEN,INPB
                 8,SYNB1,SYNB2,BF1,BF2,OUTB
                 9,RECV,ARQPB,ARQDCT,RIBIT
                 1,DCT1I1,DCT1I2,IORDR1,IORDR2,MIBIT1,MIBIT2,MIBIT3,MIBIT4,QTDCT1
0024             EQUIVALENCE (NIND3(1),XR(1))
        C        >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0025             DATA REAL,CMPLX,INTGR/0,1,2/
0026             DATA EVERY,SINGLE,DOUBLE/1,2,3/
0027             DATA BYTE2,BYTE4,BYTE8/2,4,8/
```

```
              DATA LONG,SHORT,CNVYES,CNVNO/0,1,1,0/
              DATA BUS1,BUS2,BUS3/64,128,192/
      C
      C       ADD 3 FRAMES DELAY IN INPUT FOR SNR CALCULATION
      C
      D       INSTR=0
              STATUS=MPRDB(SEN,NIND3(1),BYTE2,CNVYES,NIND3(NTUPS))
              STATUS=MPRDB(NIN2,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
              STATUS=MPWDB(SEN,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
              STATUS=MPRDB(NIN1,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
              STATUS=MPWDB(NIN2,IORDR(1),BYTE2,CNVYES,IORDR(NTUPS))
              STATUS=MPWDB(NIN1,NIN(1),BYTE2,CNVYES,NIN(NTUPS))
      D       IF(STATUS.NE.0)GO TO 10
      C       NOW DELAY M & VDV FOR CONSISTENCY
              M4=M3
              M3=M2
              M2=M1
              M1=M
              IV4=IV3
              IV3=IV2
              IV2=IV1
              IV1=IFIX(VD)
      C
      C
      C       COMPUTE S/N RATIO
      C
              SUM1=0.0
              SUM2=0.0
      C       NIN(I) MATCHES NOUT(NP+I), ERGO NTUPS-NP MATCHES!
              DO 3605 I=1,NTOTO-NP
              SOURCE=FLOAT(NIND3(I))
              SIGNAL=FLOAT(NOUT(NP+I))
              NOISE=SIGNAL-SOURCE
              SUM1=SUM1+SOURCE**2
        3605  SUM2=SUM2+NOISE**2
              SN=0.0
              FRNUM=ICOUNT-START+1
              IF(FRNUM.LT.4)GO TO 200
              SN=10.0*ALOG10(SUM1/SUM2)
              IF(FRNUM.EQ.4)CSN=SN
              CSN=((FRNUM-1.)/FRNUM)*CSN+(1./FRNUM)*SN
      C200    WRITE(4,255)ICOUNT,SN,CSN,M4,IV4
      C       WRITE(5,255)ICOUNT,SN,CSN,M4,IV4
        255   FORMAT(1X,'FRAME=',I3,3X,'SN=',F6.2,3X,'CSN=',F6.2,3X
              1,'PITCH=',I3,'    &    VDV=',I2)
        200   CONTINUE
              RETURN
      C
      C       DIAGNOSTIC TRAP AREA
      C
        10    TYPE 100,INSTR,STATUS
              CALL EXIT
        100   FORMAT(1X,'***MAP ERROR*** "SNR" INSTR=',I3,' HAS STATUS=',I6/)
              END
```

```
PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

SCODE1   000720    232          RW,I,CON,LCL
$IDATA   000162     57          RW,D,CON,LCL
$VARS    000110     36          RW,D,CON,LCL
$TEMPS   000002      1          RW,D,CON,LCL
#TAPE0   002260    600          RW,D,OVR,GBL
#TAPE1   000012      5          RW,D,OVR,GBL
#TAPE2   000012      5          RW,D,OVR,GBL
DELAY    000020      8          RW,D,OVR,GBL
OVRI     002020    520          RW,D,OVR,GBL
OVRD     002006    515          RW,D,OVR,GBL
OVRF     006016   1543          RW,D,OVR,GBL
OVRA     000152     53          RW,D,OVR,GBL
OVRP     002022    521          RW,D,OVR,GBL
OVRR     005004   1282          RW,D,OVR,GBL
OVR2     001002    257          RW,D,OVR,GBL
OVRL     000016      7          RW,D,OVR,GBL
OVRD     000012      5          RW,D,OVR,GBL
MAPDEF   000200     64          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 026236  5711

SNR,LP/LI:1=SNR/OF/NOTR
```

```
FORTRAN IV-PLUS V02-51E          11:41:28     10-AUG-80            PAGE 1
TAPE2.FTN          /WR

0001             SUBROUTINE TAPE2(IQ)
0002             IMPLICIT INTEGER(A-Z)
0003             LOGICAL*1 APPEND
0004             COMMON/MTAPE0/NIN(300),NOUT(300)
0005             COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0006             COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007             COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008             COMMON/MTAPE4/LST,IBEG
0009             COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                 1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010             COMMON/MTAPE6/APPEND
0011             DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0012             DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0013             DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0014             DATA MASK/0377/
0015             DATA MT0/0,2048,0,0,0,0/
0016             DATA MT1/0,2048,0,0,0,0/
0017             GO TO (700,800,900,1100,1000,1200,1300,1400),IQ
0018      700    CALL OPT1
0019             RETURN
0020      800    CALL OPT2
0021             RETURN
0022      900    CALL OPT3
0023             RETURN
0024      1100   CALL OPT4
0025             RETURN
0026      1000   CALL OPT5
0027             RETURN
0028      1200   CALL OPT6
0029             RETURN
0030      1300   CALL OPT7
0031             RETURN
0032      1400   CALL OPT8
0033             CALL OPT3
0034             IF(APPEND)CALL OPT4
0035             RETURN
0036             END

PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1   000226     75          RW,I,CON,LCL
$PDATA   000022      9          RW,D,CON,LCL
$IDATA   000002      1          RW,D,CON,LCL
MTAPE0   002260    600          RW,D,OVR,GBL
MTAPE1   000012      5          RW,D,OVR,GBL
MTAPE2   000012      5          RW,D,OVR,GBL
MTAPE3   012260   2648          RW,D,OVR,GBL
MTAPE4   000004      2          RW,D,OVR,GBL
MTAPE5   000064     26          RW,D,OVR,GBL
MTAPE6   000002      1          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015130   3372
```

```
NO FPP INSTRUCTIONS GENERATED

TAPE2,LP/LI:1=TAPE2/NOTR
```

```
0001            SUBROUTINE OPT1
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 APPEND
0004            COMMON/MTAPE0/NIN(300),NOUT(300)
0005            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0006            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008            COMMON/MTAPE4/LST,IBEG
0009            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010            COMMON/MTAPE6/APPEND
0011            DIMENSION ICARD(64)
0012            DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0013            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0014            DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0015            DATA MASK/0377/
0016            DATA MT0/0,2048,0,0,0,0/
0017            DATA MT1/0,2048,0,0,0,0/
        C
        C       INPUT DATA(=1)
        C
0018    700     CONTINUE
0019            IST=IST+NTUPS
0020            IF(1324.GE.IST) GO TO 200
0021    100     IST=IST-1024
0022            NSKIP=NSKIP+1
0023    200     KOVER=LST+NTOTI-1325
0024            IF(KOVER.GT.0) GO TO 305
0025            DO 5000 I=1,NTOTI
0026    5000    NIN(I)=NBF(LST+I-1)
0027            LST=LST+NTUPS
0028            GO TO 6002
0029    305     IPEP=LST-1024
0030            DO 5001 I=1,300
0031    5001    NBF(IPEP+I-1)=NBF(LST+I-1)
0032            LST=IPEP
0033            IF(NINS.EQ.0) GO TO 2100
        C>>>>>>>>DISK INPUT
0034            DO 5010 I=1,16
0035            READ(2,END=2001,ERR=6000)(ICARD(J),J=1,64)
0036            K=64*(I-1)+300
0037            DO 5010 J=1,64
0038    5010    NBF(K+J)=ICARD(J)
0039            IF(NERR.NE.0) GO TO 6000
0040            GO TO 200
0041    2001    NEND=1
0042            GO TO 200
        C>>>>>>>>TAPE INPUT
0043    2100    NEND=0
0044            NERR=0
0045            CALL GETADR(MT0,NBF(301))
0046            CALL WTQIO(IORLB,2,1,0,ISW,MT0,DSW)
0047            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 200
0048            IF(IAND(IEEOF,MASK).EQ.IAND(MASK,ISW(1)))NEND=1
0049            IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
0050            GO TO 200
```

```
FORTRAN IV-PLUS V02-51E          11:41:33     10-AUG-80            PAGE 2
OPT1.FTN           /WR

0051     6000     TYPE 6001
0052     6001     FORMAT(1X,'***INPUT FILE ERROR***'/)
0053              NERR=1
0054     6002     RETURN
0055              END


PROGRAM SECTIONS

 NAME        SIZE                   ATTRIBUTES

SCODE1  000646   211               RW,I,CON,LCL
SPDATA  000014     6               RW,D,CON,LCL
SIDATA  000062    25               RW,D,CON,LCL
SVARS   000212    69               RW,D,CON,LCL
MTAPE0  002260   600               RW,D,OVR,GBL
MTAPE1  000012     5               RW,D,OVR,GBL
MTAPE2  000012     5               RW,D,OVR,GBL
MTAPE3  012260  2648               RW,D,OVR,GBL
MTAPE4  000004     2               RW,D,OVR,GBL
MTAPE5  000064    26               RW,D,OVR,GBL
MTAPE6  000002     1               RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016034  3598

NO FPP INSTRUCTIONS GENERATED

OPT1.LP/LI:1=OPT1/NOTR
```

```
      SUBROUTINE OPT2
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 APPEND
      COMMON/MTAPE0/NIN(300),NOUT(300)
      COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
      COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
      COMMON/MTAPE3/NBF(1324),NBUF(1324)
      COMMON/MTAPE4/LST,IBEG
      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
     1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
      COMMON/MTAPE6/APPEND
      DIMENSION ICARD(64)
      DATA IOATT,IOSUC,IEALN/O001400,1,-34/
      DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/O2400,O400,-4,O2440,O1000   /
      DATA IOSPF,IEEOF,IOEOF/O2440,-10,O3000/
      DATA MASK/O377/
      DATA MT0/0,2048,0,0,0,0/
      DATA MT1/0,2048,0,0,0,0/
C
C     OUTPUT DATA(=2)
C
800   CONTINUE
      DO 5005 I=1,NTOTO
5005  NBUF(IBEG+I-1)=NOUT(I)
      IBEG=IBEG+NTOTO
      KON=1025-IBEG
      IF(KON.GT.0) GO TO 6002
      IF(NOUTS.EQ.0) GO TO 2002
C>>>>>>>DISK OUTPUT
      DO 6010 I=1,16
      K=64*(I-1)
      DO 6011 J=1,64
6011  ICARD(J)=NBUF(K+J)
6010  WRITE(3)(ICARD(J),J=1,64)
      GO TO 2003
C>>>>>>>TAPE OUTPUT
2002  NERR=0
      CALL GETADR(MT1,NBUF)
      CALL WTOIO(IOWLB,3,1,0,ISW(1),MT1,DSW)
      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 2003
      IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))NERR=1
2003  LRESID=-KON
      DO 5006 I=1,LRESID
5006  NBUF(I)=NBUF(1024+I)
      IBEG=LRESID+1
      RETURN
6000  TYPE 6001
6001  FORMAT(1X,'***OUTPUT FILE ERROR***'/)
      NERR=1
6002  RETURN
      END
```

PROGRAM SECTIONS

NAME SIZE ATTRIBUTES

```
SCODE1  000432   141      RW,I,CON,LCL
SPDATA  000014     6      RW,D,CON,LCL
SIDATA  000062    25      RW,D,CON,LCL
SVARS   000212    69      RW,D,CON,LCL
NTAPE0  002260   600      RW,D,OVR,GBL
NTAPE1  000012     5      RW,D,OVR,GBL
NTAPE2  000012     5      RW,D,OVR,GBL
NTAPE3  012260  2648      RW,D,OVR,GBL
NTAPE4  000004     2      RW,D,OVR,GBL
NTAPE5  000064    26      RW,D,OVR,GBL
NTAPE6  000002     1      RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015620  3528

NO FPP INSTRUCTIONS GENERATED

OPT2.LP/LI:1=OPT2/NOTR
```

```
      SUBROUTINE OPT3
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 APPEND
      COMMON/MTAPE0/NIN(300),NOUT(300)
      COMMON/MTAPE1/NSKIP,IST,NTOT1,NTOPS,NTOTO
      COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
      COMMON/MTAPE3/NBF(1324),NBUF(1324)
      COMMON/MTAPE4/LST,IBEG
      COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
     1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MTO(6),MT1(6),DSW
      COMMON/MTAPE6/APPEND
      DIMENSION ICAPD(64)
      DATA IOATT,IOSUC,IEALN/0001400,1,-34/
      DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
      DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
      DATA MASK/0377/
      DATA MTO/0,2048,0,0,0,0/
      DATA MT1/0,2048,0,0,0,0/
C
C     INITIALIZE(=3)
C
900   CONTINUE
      NEND=0
      NERR=0
      IF((NINS+NOUTS).GT.1)GO TO 901
      UNIT=0
      DO 902 LUN=2,3
      IF(NINS.NE.0.AND.LUN.EQ.2)GO TO 902
      IF(NOUTS.NE.0.AND.LUN.EQ.3)GO TO 902
C>>>>>>>MT: ATTACH
      CALL ASNLUN(LUN,'MT',UNIT,DSW)
      IF(DSW.NE.1)GO TO 6000
      CALL WTQIO(IOATT,LUN,1,0,ISW(1),0,DSW)
      IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 902
      IF(IAND(IEALN,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
902   UNIT=UNIT+1
901   DO 903 LUN=2,3
      IF(NINS.NE.0.AND.LUN.EQ.2)GO TO 903
      IF(NOUTS.NE.0.AND.LUN.EQ.3)GO TO 903
C>>>>>>>MT: REWIND
      CALL WTQIO(IORWD,LUN,1,0,ISW(1),0,DSW)
      IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6001
903   CONTINUE
      IF(NFILE.EQ.0)GO TO 995
      IF(NINS.NE.0)GO TO 913
C>>>>>>>MT: FILE SKIP
      DO 907 I=1,NFILE-1
      CALL GETADR(MTO,NBF(301))
      CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
      IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6003
      MTO(1)=1
907   CALL WTQIO(IOSPF,2,1,0,ISW(1),MTO,DSW)
913   DO 912 J=1,NSKIP
      IF(NINS.EQ.0)GO TO 910
C>>>>>>>DISK INPUT
      DO 914 I=1,16
```

```
              READ(2,END=905,ERR=6002)(ICARD(JJ),JJ=1,64)
              K=64*(I-1)+300
              DO 914 JJ=1,64
      914     NBF(K+JJ)=ICARD(JJ)
              GO TO 912
C>>>>>>>>TAPE INPUT
      910     CALL GETADR(MTO,NBF(301))
              CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
              IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 912
              IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))GO TO 6002
              IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 912
      905     NEND=1
      912     CONTINUE
      995     IBEG=1
              LST=IST+300
              IST=IST-NTUPS
              RETURN
      6000    TYPE 100
              GO TO 6010
      6001    TYPE 101
              GO TO 6010
      6002    TYPE 102
              GO TO 6010
      6003    TYPE 103
      6010    NERR=1
              RETURN
      100     FORMAT(1X,'*** MT: ATTACH FAILURE! ***'/)
      101     FORMAT(1X,'*** MT: REWIND FAILURE! ***'/)
      102     FORMAT(1X,'*** INPUT FILE "NSKIP" FAILURE! ***'/)
      103     FORMAT(1X,'*** MT: "NFILE" FAILURE! ***'/)
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	001212	325	RW,I,CON,LCL
$PDATA	000020	8	RW,D,CON,LCL
$IDATA	000332	109	RW,D,CON,LCL
$VARS	000214	70	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016662 3801

NO FPP INSTRUCTIONS GENERATED

OPT3,LP/LI:1=OPT3/NOTR

```
              SUBROUTINE OPT4
              IMPLICIT INTEGER(A-Z)
              LOGICAL*1 APPEND
              COMMON/MTAPE0/NIN(300),NOUT(300)
              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/MTAPE3/NBF(1324),NBUF(1324)
              COMMON/MTAPE4/LST,IBEG
              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
              1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
              COMMON/MTAPE6/APPEND
              DIMENSION ICARD(64)
              DATA IOATT,IOSUC,IEALN/0001400,1,-34/
              DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
              DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
              DATA MASK/0377/
              DATA MT0/0,2048,0,0,0,0/
              DATA MT1/0,2048,0,0,0,0/
      C
      C       OUTPUT FILE SKIP(=4)
      C
      1100    CONTINUE
              NERR=0
              NEND=0
              IBEG=1
      1       CALL GETADR(MT1(1),NBUF(1))
              CALL WTQIO(IORLB,3,1,0,ISW(1),MT1,DSW)
              IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
              IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
              CALL GETADR(MT1(1),NBUF(1))
              CALL WTQIO(IORLB,3,1,0,ISW(1),MT1,DSW)
              IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1
              IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 6000
              MT1(1)=-1
              CALL WTQIO(IOSPF,3,1,0,ISW(1),MT1,DSW)
              RETURN
      6000    TYPE 100
              RETURN
      100     FORMAT(1X,'*** OUTPUT "APPEND" FAILURE! ***'/)
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000224	74	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000114	38	RW,D,CON,LCL
$VARS	000200	64	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL

```
$TAPE6  000002      1             RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015432  3469

NO FPP INSTRUCTIONS GENERATED

OPT4,LP/LI:1=OPT4/NOTR
```

```
                SUBROUTINE OPTS
                IMPLICIT INTEGER(A-Z)
                LOGICAL*1 APPEND
                COMMON/MTAPE0/NIN(300),NOUT(300)
                COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
                COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
                COMMON/MTAPE3/NBF(1324),NBUF(1324)
                COMMON/MTAPE4/LST,IBEG
                COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
                1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
                COMMON/MTAPE6/APPEND
                DIMENSION ICARD(64)
                DATA IOATT,IOSUC,IEALN/0001400,1,-34/
                DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
                DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
                DATA MASK/0377/
                DATA MT0/0,2048,0,0,0,0/
                DATA MT1/0,2048,0,0,0,0/
          C
          C     END-OF-FILE(=5)
          C
          1000  CONTINUE
                NERR=0
                NEND=0
                IF(NOUTS.EQ.0)GO TO 1001
          C>>>>>>>>DISK EOF
                CALL CLOSE(3)
                GO TO 1003
          C>>>>>>>>TAPE EOF
          1001  DO 1002 I=1,2
          1002  CALL WTQIO(IOEOF,3,1,0,ISW(1))
                IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
                MT1(1)=-1
                CALL WTQIO(IOSPF,3,1,0,ISW(1),MT1,DSW)
                IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
          1003  IBEG=1
                RETURN
          6000  TYPE 100
                NERR=1
                RETURN
          100   FORMAT(1X,'*** MT: EOF FAILURE! ***'/)
                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000204	66	RW,I,CON,LCL
SPDATA	000014	6	RW,D,CON,LCL
SIDATA	000076	31	RW,D,CON,LCL
$VARS	000202	65	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL

```
MTAPE4  000004     2          RW,D,OVR,GBL
MTAPE5  000064    26          RW,D,OVR,GBL
MTAPE6  000002     1          RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015376  3455

NO FPP INSTRUCTIONS GENERATED

OPT5,LP/LI:1=OPT5/NOTR
```

```
0001            SUBROUTINE OPT6
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 APPEND
0004            COMMON/MTAPE0/NIN(300),NOUT(300)
0005            COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0006            COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007            COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008            COMMON/MTAPE4/LST,IBEG
0009            COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
               1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MTO(6),MT1(6),DSW
0010            COMMON/MTAPE6/APPEND
0011            DIMENSION ICARD(64)
0012            DATA IOATT,IOSUC,IEALN/0001400,1,-34/
0013            DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0014            DATA IOSPF, EEOF,IOEOF/02440,-10,03000/
0015            DATA MASK/0377/
0016            DATA MTO/0,2048,0,0,0,0/
0017            DATA MT1/0,2048,0,0,0,0/
          C
          C     REWIND/SEARCH INPUT ONLY(=6)
          C
0018      1200  CONTINUE
0019            NEND=0
0020            NERR=0
0021            IF(NINS.EQ.0)GO TO 1222
          C>>>>>>>DISK REWIND
0022            REWIND 2
0023            GO TO 1203
          C>>>>>>>MT: REWIND
0024      1222  CALL WTQIO(IORWD,2,1,0,ISW(1),0,DSW)
0025            IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6002
0026      1203  IF(NFILE.LE.1)GO TO 1204
0027            DO 1205 I=1,NFILE-1
0028            CALL GETADR(MTO,NBF(301))
0029            CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
0030            IF(IOSUC.NE.IAND(MASK,ISW(1)))GO TO 6000
0031            MTO(1)=1
0032      1205  CALL WTQIO(IOSPF,2,1,0,ISW(1),MTO,DSW)
0033      1204  DO 1210 J=1,NSKIP
0034            IF(NINS.EQ.0)GO TO 1201
          C>>>>>>>DISK SEARCH
0035            DO 1207 I=1,16
0036            READ(2,END=1202,ERR=6001)(ICARD(JJ),JJ=1,64)
0037            K=64*(I-1)+300
0038            DO 1207 JJ=1,64
0039      1207  NBF(K+JJ)=ICARD(JJ)
0040            GO TO 1210
          C>>>>>>>TAPE SEARCH
0041      1201  CALL GETADR(MTO,NBF(301))
0042            CALL WTQIO(IORLB,2,1,0,ISW(1),MTO,DSW)
0043            IF(IOSUC.EQ.IAND(MASK,ISW(1)))GO TO 1210
0044            IF(IAND(IEVER,MASK).EQ.IAND(MASK,ISW(1)))GO TO 6000
0045            IF(IAND(IEEOF,MASK).NE.IAND(MASK,ISW(1)))GO TO 1210
0046      1202  NEND=1
0047            GO TO 6000
0048      1210  CONTINUE
```

```
              IST=IST+100
              IST=IST-NTUPS
              RETURN
      6000    TYPE 100
              GO TO 6010
      6001    TYPE 101
              GO TO 6010
      6002    TYPE 102
      6010    NERR=1
              RETURN
      100     FORMAT(1X,'*** INPUT FILE "NSKIP" FAILURE! ***'/)
      101     FORMAT(1X,'*** INPUT FILE "NFILE" FAILURE! ***'/)
      102     FORMAT(1X,'*** MT: REWIND FAILURE! ***'/)
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000712	229	RW,I,CON,LCL
$PDATA	000014	6	RW,D,CON,LCL
$IDATA	000246	83	RW,D,CON,LCL
$VARS	000210	68	RW,D,CON,LCL
$TEMPS	000004	2	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016266   3675

NO FPP INSTRUCTIONS GENERATED

OPT6,LP/LI:1=OPT6/NOTR

```
              SUBROUTINE OPT7
              IMPLICIT INTEGER(A-Z)
              LOGICAL*1 APPEND
              COMMON/MTAPE0/NIN(300),NOUT(300)
              COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
              COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
              COMMON/MTAPE3/NBF(1324),NBUF(1324)
              COMMON/MTAPE4/LST,IREG
              COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
              1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
              COMMON/MTAPE6/APPEND
              DIMENSION ICARD(64)
              DATA IOATT,IOSUC,IEALN/0001400,1,-34/
              DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
              DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
              DATA MASK/0377/
              DATA MT0/0,2048,0,0,0,0/
              DATA MT1/0,2048,0,0,0,0/
      C
      C       CLOSE INPUT FILE
      C
      1300    CONTINUE
              NEND=0
              NERR=0
              IF(NINS.EQ.1)CALL CLOSE(2)
              RETURN
              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000032	13	RW,I,CON,LCL
$PDATA	000004	2	RW,D,CON,LCL
$IDATA	000004	2	RW,D,CON,LCL
$VARS	000200	64	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015120   3368

NO FPP INSTRUCTIONS GENERATED

OPT7,LP/LI:1=OPT7/NOTR

```
0001          SUBROUTINE OPTR
0002          IMPLICIT INTEGER(A-Z)
0003          LOGICAL*1 EXTIN,EXTOUT,APPEND
0004          COMMON/MTAPE0/NIN(300),NOUT(300)
0005          COMMON/MTAPE1/NSKIP,IST,NTOTI,NTUPS,NTOTO
0006          COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007          COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008          COMMON/MTAPE4/LST,IBEG
0009          COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSUC,IEALN,IORWD
              1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010          COMMON/MTAPE6/APPEND
0011          DIMENSION EXTIN(3),EXTOUT(3)
0012          DIMENSION ICARD(64)
0013          DATA YES,NO/'Y','N'/
0014          DATA EXTIN/'I','N','P'/
0015          DATA EXTOUT/'O','U','T'/
0016          DATA NEND,NERR,NFILE/0,0,0/
0017          DATA NSKIP,IST/1,1/
0018          DATA IOATT,IOSUC,IEALN/O001400,1,-34/
0019          DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/O2400,O400,-4,O2440,O1000     /
0020          DATA IOSPF,IEEOF,IOEOF/O2440,-10,O3000/
0021          DATA MASK/O377/
0022          DATA MT0/0,2048,0,0,0,0/
0023          DATA MT1/0,2048,0,0,0,0/
        C
        C     GET FILE INFO(=8)
        C
0024          APPEND=.FALSE.
0025    900   TYPE 100
0026    100   FORMAT(//1HS,'IS THE INPUT ON MAG. TAPE(Y/N)? ')
0027          READ(5,102,END=999,ERR=900)ANSER
0028    102   FORMAT(A1)
0029          NINS=1
0030          IF(ANSER.EQ.NO)NINS=1
0031          NOUTS=1
0032    901   TYPE 103
0033    103   FORMAT(1HS'IS THE OUTPUT GOING TO MAG TAPE(Y/N)? ')
0034          READ(5,102,END=999,ERR=901)ANSER
0035          IF(ANSER.EQ.NO)NOUTS=1
0036          IF(NOUTS.EQ.1)GO TO 5
0037    902   TYPE 104
0038    104   FORMAT(1HS,'APPEND DATA? ')
0039          READ (5,102,END=999,ERR=902)ANSER
0040          IF(ANSER.EQ.YES)APPEND=.TRUE.
0041    5     IF(NOUTS.EQ.0)GO  TO 151
        C     RSX11  SUPPORTED FILE
0042    903   TYPE 105
0043    105   FORMAT(1HS,'OUTPUT FILE NAME= ')
0044          CALL FILEN(3,EXTOUT)
        C
        C     BEGINNING OF INPUT
        C
0045    151   IF(NINS.EQ.1)GO TO 155
0046    904   TYPE 106
0047    106   FORMAT(1HS'MT FILE NO.=(I3) ')
0048          READ(5,101,END=999,ERR=904)NFILE
```

```
0049     101      FORMAT(I3)
0050              GO TO 14
0051     155      TYPE 107
0052     107      FORMAT(1HS,'INPUT FILE NAME= ')
0053              CALL FILEN(2,EXTIN)
0054              NFILE=1
0055     14       RETURN
0056     999      CALL EXIT
0057              END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000524	170	RW,I,CON,LCL
$PDATA	000010	4	RW,D,CON,LCL
$IDATA	000300	96	RW,D,CON,LCL
$VARS	000214	70	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 016126   3627

NO FPP INSTRUCTIONS GENERATED

```
0001                SUBROUTINE FILEN(UNIT,EXT)
        C           THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
        C           FORM THE TTY DEVICE 5
        C           DEFAULT DEVICE
        C           UNLESS SPECIFIED IN INPUT STRING
        C           UNIT=UNIT NUMBER
        C           EXT = LOGICAL*1 BUFFER OF EXTENSION
        C
0002                IMPLICIT INTEGER(A-Z)
0003                LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                DIMENSION INSTR(40)
0005                DIMENSION EXT(3)
0006                DATA BLNK,DOT/' ','.'/
        C
        C           INPUT FILE
0007    152         READ (5,99,END=999,ERR=152)(INSTR(I),I=1,40)
0008    99          FORMAT(40A1)
        C           CHECK FOR END OF LINE
0009                DO 1600 I=40,1,-1
0010                J=I
0011    1600        IF(INSTR(I).NE.BLNK)GO TO 1601
0012                TYPE 151
0013    151         FORMAT(1H$,'>')
0014                GO TO 152
0015    1601        DO 1602 I=1,J
0016                IF(INSTR(I).NE.BLNK) GO TO 1602
        C
        C
        C           BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
        C
0017                DO 1603 K=I,J-1
0018    1603        INSTR(K)=INSTR(K+1)
0019                INSTR(J)=BLNK
0020                J=J-1
0021                GO TO 1601
0022    1602        CONTINUE
0023                DO 103 I=1,J
0024    103         IF(INSTR(I).EQ.DOT) GO TO 25
0025                INSTR(J+1)=DOT
0026                INSTR(J+2)=EXT(1)
0027                INSTR(J+3)=EXT(2)
0028                INSTR(J+4)=EXT(3)
0029                J=J+4
0030    25          CALL SCAN(INSTR,J)
0031                CALL ASSIGN(UNIT,INSTR,J)
0032                RETURN
0033    999         CALL EXIT
0034                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000472	157	RW,I,CON,LCL
$IDATA	000044	18	RW,D,CON,LCL

```
$VARS    000060     24           RW,D,CON,LCL
$TEMPS   000004      2           RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000622   201

NO FPP INSTRUCTIONS GENERATED
```

```
      C
                SUBROUTINE SCAN(BUF,LTH)
                IMPLICIT INTEGER(A-Z)
                LOGICAL*1 BUF,DEVICE
                DIMENSION BUF(1),DEVICE(4)
                DATA DEVICE/'S','Y','0',':'/
                DO 1 I=1,LTH
      1         IF(BUF(I).EQ.DEVICE(4))RETURN
                LTH=LTH+4
                DO 2 I=LTH,5,-1
      2         BUF(I)=BUF(I-4)
                DO 3 I=1,4
      3         BUF(I)=DEVICE(I)
                RETURN
                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000172	61	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000006	3	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214   70

NO FPP INSTRUCTIONS GENERATED

OPTR,LP/LI:1=OPTR/NOTR

```
FORTRAN IV-PLUS V02-51E          21:16:21     11-SEP-80           PAGE 1
OPTRA.FTN        /DE/WR

0001                SUBROUTINE OPTR
0002                IMPLICIT INTEGER(A-Z)
0003                LOGICAL*1 EXTIN,EXTOUT,APPEND
0004                COMMON/MTAPE0/NIN(300),NOUT(300)
0005                COMMON/MTAPE1/NSKIP,IST,NTOTI,NTOPS,NTOTO
0006                COMMON/MTAPE2/NEND,NERR,NFILE,NINS,NOUTS
0007                COMMON/MTAPE3/NBF(1324),NBUF(1324)
0008                COMMON/MTAPE4/LST,IBEG
0009                COMMON/MTAPE5/MASK,ISW(2),IOATT,IOSDC,IEALN,IORWD
                   1,IOWLB,IEVER,IOSPF,IEEOF,IOEOF,IORLB,MT0(6),MT1(6),DSW
0010                COMMON/MTAPE6/APPEND
0011                DIMENSION EXTIN(3),EXTOUT(3)
0012                DIMENSION ICARD(64)
0013                DATA YES,NO/'Y','N'/
0014                DATA EXTIN/'I','N','P'/
0015                DATA EXTOUT/'O','U','T'/
0016                DATA NEND,NERR,NFILE/0,0,0/
0017                DATA NSKIP,IST/1,1/
0018                DATA IOATT,IOSDC,IEALN/0001400,1,-34/
0019                DATA IORWD,IOWLB,IEVER,IOSPF,IORLB/02400,0400,-4,02440,01000     /
0020                DATA IOSPF,IEEOF,IOEOF/02440,-10,03000/
0021                DATA MASK/0377/
0022                DATA MT0/0,2048,0,0,0,0/
0023                DATA MT1/0,2048,0,0,0,0/
          C
          C         GET FILE INFO(=8)
          C
0024                APPEND=.FALSE.
          C900      TYPE 100
          C100      FORMAT(//1H$,'IS THE INPUT ON MAG. TAPE(Y/N)? ')
          C         READ(5,102,END=999,ERR=900)ANSER
0025                ANSER=NO
0026      102       FORMAT(A1)
0027                NINS=1
0028                IF(ANSER.EQ.NO)NINS=1
0029                NOUTS=1
          C901      TYPE 103
          C103      FORMAT(1H$'IS THE OUTPUT GOING TO MAG TAPE(Y/N)? ')
          C         READ(5,102,END=999,ERR=901)ANSER
0030                ANSER=NO
0031                IF(ANSER.EQ.NO)NOUTS=1
0032                IF(NOUTS.EQ.1)GO TO 5
0033      902       TYPE 104
0034      104       FORMAT(1H$,'APPEND DATA? ')
0035                READ (5,102,END=999,ERR=902)ANSER
0036                IF(ANSER.EQ.YES)APPEND=.TRUE.
0037      5         IF(NOUTS.EQ.0)GO  TO 151
          C         RSX11  SUPPORTED FILE
          C903      TYPE 105
          C105      FORMAT(1H$,'OUTPUT FILE NAME= ')
          C         CALL FILEN(3,EXTOUT)
0038                CALL ASSIGN(3,'NL:')
          C
          C         BEGINNING OF INPUT
          C
0039      151       IF(NINS.EQ.1)GO TO 155
```

```
0040    904     TYPE 106
0041    106     FORMAT(1HS'MI FILE NO.=(I3) ')
0042            READ(5,101,END=999,ERR=904)NFILE
0043    101     FORMAT(I3)
0044            GO TO 14
0045    155     CONTINUE
        C155    TYPE 107
        C107    FORMAT(1HS,'INPUT FILE NAME= ')
        C       CALL FILEN(2,EXTIN)
0046            CALL ASSIGN(2,'I5,100]VOICEA.BIN',17)
0047            NFILE=1
0048    14      RETURN
0049    999     CALL EXIT
0050            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000330	108	RW,I,CON,LCL
SPDATA	000042	17	RW,D,CON,LCL
SIDATA	000100	32	RW,D,CON,LCL
SVARS	000214	70	RW,D,CON,LCL
MTAPE0	002260	600	RW,D,OVR,GBL
MTAPE1	000012	5	RW,D,OVR,GBL
MTAPE2	000012	5	RW,D,OVR,GBL
MTAPE3	012260	2648	RW,D,OVR,GBL
MTAPE4	000004	2	RW,D,OVR,GBL
MTAPE5	000064	26	RW,D,OVR,GBL
MTAPE6	000002	1	RW,D,OVR,GBL

TOTAL SPACE ALLOCATED = 015564 3514

NO FPP INSTRUCTIONS GENERATED

```
0001                SUBROUTINE FILEN(UNIT,EXT)
            C       THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
            C       FORM THE TTY DEVICE 5
            C       DEFAULT DEVICE
            C       UNLESS SPECIFIED IN INPUT STRING
            C       UNIT=UNIT NUMBER
            C       EXT = LOGICAL*1 BUFFER OF EXTENSION
            C
0002                IMPLICIT INTEGER(A-Z)
0003                LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                DIMENSION INSTR(40)
0005                DIMENSION EXT(3)
0006                DATA BLNK,DOT/' ','.'/
            C
            C       INPUT FILE
0007        152     READ (5,99,END=999,ERR=152)(INSTR(I),I=1,40)
0008        99      FORMAT(40A1)
            C       CHECK FOR END OF LINE
0009                DO 1600 I=40,1,-1
0010                J=I
0011        1600    IF(INSTR(I).NE.BLNK)GO TO 1601
0012                TYPE 151
0013        151     FORMAT(1H$,'>')
0014                GO TO 152
0015        1601    DO 1602 I=1,J
0016                IF(INSTR(I).NE.BLNK) GO TO 1602
            C
            C
            C       BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
            C
0017                DO 1603 K=1,J-1
0018        1603    INSTR(K)=INSTR(K+1)
0019                INSTR(J)=BLNK
0020                J=J-1
0021                GO TO 1601
0022        1602    CONTINUE
0023                DO 103 I=1,J
0024        103     IF(INSTR(I).EQ.DOT) GO TO 25
0025                INSTR(J+1)=DOT
0026                INSTR(J+2)=EXT(1)
0027                INSTR(J+3)=EXT(2)
0028                INSTR(J+4)=EXT(3)
0029                J=J+4
0030        25      CALL SCAN(INSTR,J)
0031                CALL ASSIGN(UNIT,INSTR,J)
0032                RETURN
0033        999     CALL EXIT
0034                END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000472	157	RW,I,CON,LCL
SIDATA	000044	18	RW,D,CON,LCL

```
FORTRAN IV-PLUS V02-51F          21:16:25    11-SEP-80           PAGE 4
OPTRA.FTN       /DE/WR

$VARS   000060    24             RW,D,CON,LCL
$TEMPS  000004     2             RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000622   201

NO FPP INSTRUCTIONS GENERATED
```

```
        C
0001            SUBROUTINE SCAN(BUF,LTH)
0002            IMPLICIT INTEGER(A-Z)
0003            LOGICAL*1 BUF,DEVICE
0004            DIMENSION BUF(1),DEVICE(4)
0005            DATA DEVICE/'S','Y','0',':'/
0006            DO 1 I=1,LTH
0007      1     IF(BUF(I).EQ.DEVICE(4))RETURN
0008            LTH=LTH+4
0009            DO 2 I=LTH,5,-1
0010      2     BUF(I)=BUF(I-4)
0011            DO 3 I=1,4
0012      3     BUF(I)=DEVICE(I)
0013            RETURN
0014            END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
SCODE1	000172	61	RW,I,CON,LCL
SIDATA	000012	5	RW,D,CON,LCL
SVARS	000006	3	RW,D,CON,LCL
STEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214   70

NO FPP INSTRUCTIONS GENERATED

OPTRA,LP/LI:1=OPTRA/DE/NOTR

```
        C        PREMAP.FTN       ORIGINATED:29-MAY-79
        C                         UPDATED:04-JUN-79
        C        THIS PROGRAM INTERPRETS ASCII TEXT FILES FOR THE MAP
        C        ASSEMBLER AND ALIGNS ALL FIELDS W/ SPACES AND REMOVES
        C        ALL TABS. ALL FILES ARE ASSUMED TO BE 120 COLUMNS WIDE.
        C        OUTPUT LINE FORMAT:
        C        COLUMNS 1,2...7                  LABEL
        C        COLUMN 8                         BLANK
        C        COLUMNS 9,10...55                SOURCE CODE
        C        COLUMN 56                        BLANK
        C        COLUMNS 57,58...120              COMMENTS
        C
0001             BYTE BLANK,TAB,HYPHEN,STAR,CR,DUMMY,SCOLON
0002             BYTE ALAST
0003             BYTE A(200),L(200)
0004             LOGICAL*1 IN(3),OUT(3),OUTPUT
0005             DATA IN/'T','X','T'/,OUT/'M','A','P'/
0006             DATA BLANK,TAB,HYPHEN,STAR,CR,SCOLON/040,011,047,052,015,073/
        C
0007    2        TYPE 100
0008             CALL FILEN(2,IN)
0009             TYPE 101
0010             CALL FILEN(3,OUT)
0011             LINE=0
        C
        C        READ EACH SOURCE "LINE"
        C
0012    1        LINE=LINE+1
0013             OUTPUT=.TRUE.
0014             READ(2,102,END=200,ERR=300)(A(I),I=1,120)
        C
        C        "BLANK" OUT OUTPUT LINE
        C
0015             DO 10 I=1,120
0016    10       L(I)=BLANK
        C
        C        SPECIAL LINE OPERATORS
        C
0017             IF(A(1).EQ.HYPHEN)GO TO 70
0018             IF(A(1).EQ.STAR)GO TO 70
0019             IF(A(1).EQ.SCOLON)GO TO 70
        C
        C        LABEL FIELD
        C
0020             I=1
0021    11       IF(A(I).EQ.TAB)GO TO 12
0022             L(I)=A(I)
0023             I=I+1
0024             IF(I.GT.120)GO TO 80
0025             GO TO 11
0026    12       NEXT=I+1
        C
        C        COMMAND FIELD
        C
0027             J=0
0028             I=NEXT
```

```
0029      21      IF(A(I).EQ.TAB)GO TO 22
0030              L(9+J)=A(I)
0031              J=J+1
0032              IF(J.GT.111)GO TO 80
0033              I=I+1
0034              IF(I.GT.120)GO TO 80
0035              GO TO 21
0036      22      NEXT=I+1
          C
          C       COMMENT FIELD
          C
0037      30      K=0
0038              I=NEXT
0039      31      IF(A(I).EQ.TAB)GO TO 80
0040              IF(ALAST.EQ.BLANK.AND.A(I).EQ.BLANK)GO TO 80
0041              IF(A(I).EQ.CR)GO TO 80
0042              ALAST=A(I)
0043              OUTPUT=.TRUE.
0044              L(57+K)=A(I)
0045              K=K+1
0046              IF(K.LE.23)GO TO 32
0047              WRITE(3,106)(L(II),II=1,120)
0048              DO 33 II=1,120
0049      33      L(II)=BLANK
0050              L(1)=SCOLON
0051              K=0
0052              OUTPUT=.FALSE.
0053              ALAST=BLANK
0054      32      I=I+1
0055              IF(I.GT.120)GO TO 80
0056              GO TO 31
          C
          C       COPY LINE "EN TOTO"
          C
0057      70      DO 71 I=1,120
0058      71      L(I)=A(I)
          C
          C       GENERATE CORRECTED OUTPUT LINE
          C
0059      80      IF(OUTPUT.EQ..TRUE.)WRITE(3,106)(L(I),I=1,120)
0060              GO TO 1
          C
          C       NORMAL EXIT
          C
0061      200     CALL CLOSE(2)
0062              CALL CLOSE(3)
0063              TYPE 103,LINE
0064              STOP
          C
          C       ABNORMAL EXIT
          C
0065      300     CALL CLOSE(2)
0066              CALL CLOSE(3)
0067              TYPE 104,LINE
0068              GO TO 2
          C
```

```
FORTRAN IV-PLUS V02-51F          13:17:26     10-AUG-80            PAGE 3
PREMAP.FTN        /TR:BLOCKS/WR

          C         LINE STRUCTURE ERROR
          C
0069      400       CALL CLOSE(2)
0070                CALL CLOSE(3)
0071                TYPE 105,LINE
0072                STOP
          C
          C         FORMAT DECLARATIONS
          C
0073      100       FORMAT(1H$,'INPUT FILENAME= ')
0074      101       FORMAT(1H$,'OUTPUT FILENAME= ')
0075      102       FORMAT(120A1)
0076      103       FORMAT(1X,'NORMAL EXIT AFTER ',I3,' LINES!'/)
0077      104       FORMAT(1X,'***WARNING*** FILE ERROR IN LINE ',I3/)
0078      105       FORMAT(1X,'***WARNING*** LINE STRUCTURE ERROR IN LINE ',I3/)
0079      106       FORMAT(120A1)
0080      107       FORMAT(1X,100S)
0081                END

PROGRAM SECTIONS

 NAME        SIZE                 ATTRIBUTES

$CODE1  001374   382              RW,I,CON,LCL
$PDATA  000010     4              RW,D,CON,LCL
$IDATA  000304    98              RW,D,CON,LCL
$VARS   000654   214              RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 002564   698

NO FPP INSTRUCTIONS GENERATED

PREMAP,LP/LI:1=PREMAP
```

```
FORTRAN IV-PLUS V02-51E          13:28:11     10-AUG-80          PAGE 1
FILEN.FTN         /WR

0001                  SUBROUTINE FILEN(UNIT,EXT)
          C
          C           THIS SUBROUTINE ACCEPTS THE NAME OF THE INPUT OR OUTPUT FILE
          C           FORM THE TTY DEVICE 5
          C           DEFAULT DEVICE
          C           UNLESS SPECIFIED IN INPUT STRING
          C           UNIT=UNIT NUMBER
          C           EXT = LOGICAL*1 BUFFER OF EXTENSION
          C
0002                  IMPLICIT INTEGER(A-Z)
0003                  LOGICAL*1 INSTR,DOT,BLNK,EXT
0004                  DIMENSION INSTR(40)
0005                  DIMENSION EXT(3)
0006                  DATA BLNK,DOT/' ','.'/
          C
          C           INPUT FILE
0007      152         READ (5,99)(INSTR(I),I=1,40)
0008      99          FORMAT(40A1)
          C           CHECK FOR END OF LINE
0009                  DO 1600 I=40,1,-1
0010                  J=I
0011      1600        IF(INSTR(I).NE.BLNK)GO TO 1601
0012                  TYPE 151
0013      151         FORMAT(1H$,'>')
0014                  GO TO 152
0015      1601        DO 1602 I=1,J
0016                  IF(INSTR(I).NE.BLNK) GO TO 1602
          C
          C
          C           BLANK DISCOVERED-COLLAPSE LINE BY ONE AND DECREASE CHARACTER COUNT
          C
0017                  DO 1603 K=I,J-1
0018      1603        INSTR(K)=INSTR(K+1)
0019                  INSTR(J)=BLNK
0020                  J=J-1
0021                  GO TO 1601
0022      1602        CONTINUE
0023                  DO 103 I=1,J
0024      103         IF(INSTR(I).EQ.DOT) GO TO 25
0025                  INSTR(J+1)=DOT
0026                  INSTR(J+2)=EXT(1)
0027                  INSTR(J+3)=EXT(2)
0028                  INSTR(J+4)=EXT(3)
0029                  J=J+4
0030      25          CALL SCAN(INSTR,J)
0031                  CALL ASSIGN(UNIT,INSTR,J)
0032                  RETURN
0033                  END

PROGRAM SECTIONS

 NAME       SIZE                ATTRIBUTES

$CODE1   000444    146          RW,I,CON,LCL
$IDATA   000042     17          RW,D,CON,LCL
```

```
$VARS   000060    24           RW,D,CON,LCL
$TEMPS  000004     2           RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000572   189

NO FPP INSTRUCTIONS GENERATED
```

```
FILEN.FTN        /WR

      C
                 SUBROUTINE SCAN(BUF,LTH)
                 IMPLICIT INTEGER(A-Z)
                 LOGICAL*1 BUF,DEVICE
                 DIMENSION BUF(1),DEVICE(4)
                 DATA DEVICE/'S','Y','0',':'/
                 DO 1 I=1,LTH
      1          IF(BUF(I).EQ.DEVICE(4))RETURN
                 LTH=LTH+4
                 DO 2 I=LTH,5,-1
      2          BUF(I)=BUF(I-4)
                 DO 3 I=1,4
      3          BUF(I)=DEVICE(I)
                 RETURN
                 END
```

PROGRAM SECTIONS

NAME	SIZE		ATTRIBUTES
$CODE1	000172	61	RW,I,CON,LCL
$IDATA	000012	5	RW,D,CON,LCL
$VARS	000006	3	RW,D,CON,LCL
$TEMPS	000002	1	RW,D,CON,LCL

TOTAL SPACE ALLOCATED = 000214   70

NO FPP INSTRUCTIONS GENERATED

FILEN,LP/LI:1=FILEN/NOTR

END

DATE

FILMED

12-80

DTIC