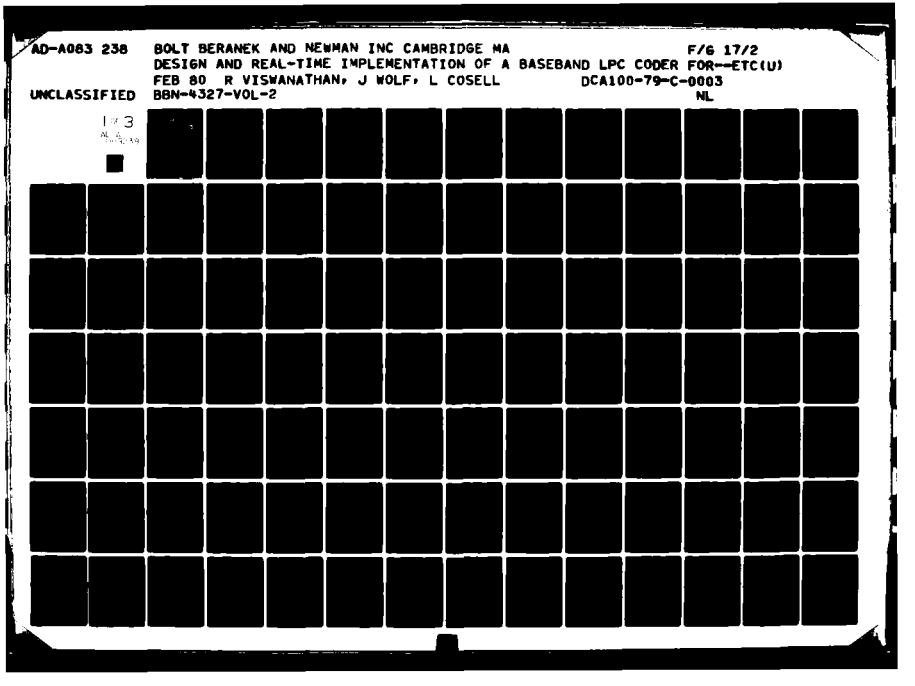
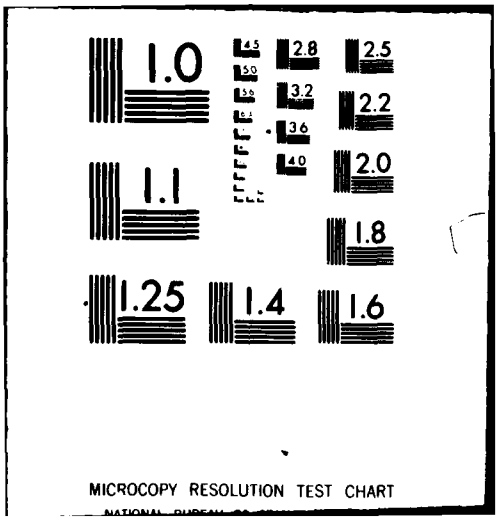

AD-A083 238
BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA
F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL
DCA100-79-C-0003
UNCLASSIFIED
BBN-4327-VOL-2
NL
1 OF 3


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

Bolt Beranek and Newman Inc.

LEVEL III

Report No. 4327 - Vol

Design and Real-Time Implementation of a Baseband LPC Coder for Speech Transmission Over 9600 Bps Noisy Channels

Volume II: Program Listings

Final Report

February 1980

Prepared for:
Defense Communications Agency

Unclassified.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: BBN Report No. 4327 – Vol 2	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR SPEECH TRANSMISSION OVER 9600 BPS NOISY CHANNELS - Volume II, Program Listings	5. TYPE OF REPORT & PERIOD COVERED: Final Report Nov 1978 – Feb 1980
	6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 4327
7. AUTHOR(s): R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins, and W. Russell	8. CONTRACT OR GRANT NUMBER(s): DCA100-79-C-0003
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238	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: February 1980
	13. NUMBER OF PAGES: 201
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, voice-excited coder, baseband coder, linear prediction, high-frequency regeneration, 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 baseband LPC speech coder that transmits high-quality speech over a 9600 bps synchronous channel with bit-error rates of up to 1%. Presented are the results of our investigation of a number of aspects of the baseband LPC coder with the goal of maximizing the quality of the transmitted speech. Important among these aspects are: baseband width, baseband coding, cont'd.

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

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (cont'd.)

high-frequency regeneration, and error-protection of important transmission parameters. The report also includes the system design, detailed documentation, and program listings of the MAP-300 real-time implementation of the optimized speech coder.

This report is bound in two volumes. Volume I contains the text of the report, and Volume II contains the program listings of the MAP-300 speech coder implementation.

Report No. 4327

DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR SPEECH TRANSMISSION OVER 9600 BPS NOISY CHANNELS

Final Report

Volume II: Program Listings

Authors: R. Viswanathan, J. Wolf, L. Cosell, K. Field,
A. Higgins, and W. Russell

February 1980

Prepared for:
The Defense Communications Agency

TABLE OF CONTENTS

(00001) ;[BBN-TENEXD]<MAP>BBN300.MSD.61, 30-Dec-79 16:14:24, Ed: KFIELD

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

```
                (00002) ;MAP-300 BBN FUNCTIONS FOR SNAPII REL. 3.5
                (00003) *
                (00004) *
                (00005) * DEFINE SYMBOLS FOR ARRAY FUNCTION ASSEMBLIES
                (00006) *
                (00007) *
                (00008) * FROM THE SNAP-II EXECUTIVE  ---  REL. 3.05  --- 5/22/79
                (00009) *
                (00010) *
000008E8 (00011)        AFDT$ORG = $8E8
0000F245 (00012)        AP$ASSS = $245
0000EEFA (00013)        AP$BNDR = $EFA
0000BF63 (00014)        AP$BNDR0 = $F63
0000BF2D (00015)        AP$BNDR1 = $F2D
0000024D (00016)        AP$BSL = $24D
00000248 (00017)        AP$CSSC = $248
0000BFB1 (00018)        AP$DONE = $FB1
0000BFBC (00019)        AP$DONER = $FBC
0000000C (00020)        AP$G0 = $C
0000000D (00021)        AP$G1 = $D
00000010 (00022)        AP$PBFF = $10
0000000A (00023)        AP$SAID = $A
00000008 (00024)        AP$SSCLR = $E
00000009 (00025)        AP$SSS = $9
00020004 (00026)        APS$BNO = $4
0001FFC0 (00027)        APS$LA = $1FFC0
0001FFC0 (00028)        APSR = $1FFC0
                (00029) *
00000604 (00030)        BCT$AD = $604
00000686 (00031)        BCT$AT = $686
00002502 (00032)        BCT$BA = $502
00000000 (00033)        BIT$G0 = $0
00000300 (00034)        BIT$RC = $F
                (00035) *
00001AD6 (00036)        CK4DB$ = $1AD6
00000792 (00037)        CLR$G0G1 = $792
00003190 (00038)        COS$ = $3190
000021FC (00039)        CSPU$NOS = $21FC
                (00040) *
00000794 (00041)        D4Y$ = $794
                (00042) *
00001AFA (00043)        ERROR$ = $1AFA
                (00044) *
00003110 (00045)        F11A$ = $3110
00000E00 (00046)        FDT$ = $7E0
00003040 (00047)        FEOS$ = $3040
0000079A (00048)        FFT$CBSZ = $79A
00000000 (00049)        FLG$CLR = $0
00000004 (00050)        FLG$G0 = $4
00000005 (00051)        FLG$G1 = $5
00000006 (00052)        FLG$G2 = $6
00000007 (00053)        FLG$G3 = $7
00000011 (00054)        FLG$RI = $11
```

```
                00000020 (00055)          FLG$SET = $20
                         (00056) *
                00001B3C (00057)          GATHER$ = $1B3C
                         (00058) *
                00000001 (00059)          HS = 1
                         (00060) *
                00000502 (00061)          ISVT$ = $502
                         (00062) *
                00001B7E (00063)          LOAD$AP = $1B7E
                00001B8F (00064)          LOAD$AP1 = $1B8F
                         (00065) *
                00000006 (00066)          MSK$BMB = $6
                FF00FF00 (00067)          MSK$LBYT = $FF00
                000FF0FF (00068)          MSK$RBYT = $00FF
                00000000 (00069)          MS$ = 0
                         (00070) *
                0000001B (00071)          OQE = $1B
                         (00072) *
                00035EA (00073)           S1011$=$35EA
                00007AE (00074)           SHFTL$R5 = $7AE
                00023FA (00075)           SINCOS$T = $23FA
                00003192 (00076)          SIN$ = $3192
                00000382 (00077)          SVT$ = $382
                00003CE (00078)           SVT$DN1 = $3CE
                0001FFCE (00079)          SYS$FLGS = $1FFCE
                         (00080) *
                00007B4 (00081)           TEM$0 = $7B4
                000021FE (00082)          TOE$ = $21FE
                00002288 (00083)          TOESPTR = $288
                         (00084) *
                00001C5A (00085)          VALBUF$ = $1C5A
                00002870 (00086)          VSMA2$=$2870
                         (00087) *
                00000002 (00088)          WS = $2
                         (00089) *
                0000192C (00090)          XFL$01 = $192C
                         (00091) *
                00007BA (00092)           ZERO = $7BA
                         (00093) *
                         (00094) *
                         (00095) *
                00002288 (00096) $L = TOESPTR
                         (00097) *
          00288 0F186EE2 (00098)          ADDR   TOE$CUR(,1)          UPDATE TOP OF EXEC POINTER
                         (00099) *
                         (00100) *
                00000003 (00101)          $M = 3
                         (00102) *
                         (00103) *
                         (00104) *
```

```
                    (00105) ;ARRAY FUNCTION DISPATCH TABLE PATCHES
                    (00106) *
                    (00107) *
                    (00108) *
          00000900  (00109)          #L = AFDT$ORG+3*W$*(132-128)    ;FCB 132 (VLTSY)
  0090    001E5EF2  (00110)          ADDR    VLTSY$(R7,1)
  0092    001E5FB6  (00111)          ADDR    V320@$(R7,1)
  0094    001021FC  (00112)          ADDR    CSPU$NOS(,1,0)
                    (00113) *
          00000906  (00114)          #L = AFDT$ORG+3*W$*(133-128)    ;FCB 133 (VKTOA)
  0096    001E6038  (00115)          ADDR    VKTOA$(R7,1)
  0098    001E60C6  (00116)          ADDR    V1100K$(R7,1)
  009A    001021FC  (00117)          ADDR    CSPU$NOS(,1,0)
                    (00118) *
          0000090C  (00119)          #L = AFDT$ORG+3*W$*(134-128)    ;FCB 134 (PRTR8)
  090C    0F1560F8  (00120)          ADDR    PRTR8$(R7,1)
  090E    001E618C  (00121)          ADDR    P2120$(R7,1)
  0910    001021FC  (00122)          ADDR    CSPU$NOS(,1,0)
                    (00123) *
          00000912  (00124)          #L = AFDT$ORG+3*W$*(135-128)    ;FCB 135 (MWLQ)
  0912    001E61D8  (00125)          ADDR    MWLF$APU(R7,1)
  0914    001E62EC  (00126)          ADDR    MWLF$APS(R7,1)
  0916    0110666C  (00127)          ADDR    MWLQ$SSM(,1,1)          ;(POST SUPPORT)
                    (00128) *
          0000096C  (00129)          #L = AFDT$ORG+3*W$*(150-128)    ;FCB 150  (VAPC)
  096C    001E6692  (00130)          ADDR    VAPC$(R7,1)
  096E    001E674C  (00131)          ADDR    AAPC$(R7,1)
  0970    001021FC  (00132)          ADDR CSPU$NOS(,1,0)
                    (00133) *
          000@0A5C  (00134)          #L = AFDT$ORG+3*W$*(190-128)    ;FCB 190  (DEAL)
  0A5C    001E67FE  (00135)          ADDR    DEALU$(R7,1)
  0A5E    001E6839  (00136)          ADDR    DEALS$(R7,1)
  0A60    001021FC  (00137)          ADDR    CSPU$NOS(,1,0)
                    (00138) *
          00000A80  (00139)          #L = AFDT$ORG+3*W$*(196-128)    ;FCB 196  (VAPCI)
  0A80    001E6876  (00140)          ADDR    APCIU$(R7,1)
  0A82    001E68CC  (00141)          ADDR    APCIS$(R7,1)
  0A84    001021FC  (00142)          ADDR    CSPU$NOS(,1,0)
                    (00143) *
          00000AE0  (00144)          #L = AFDT$ORG+3*W$*(212-128)    ; FCB #212 (PTAP)
  0AE0    001E694C  (00145)          ADDR    PTU$(R7,1)
  0AE2    001E6A38  (00146)          ADDR    PTS$(R7,1)
  0AE4    001021FC  (00147)          ADDR    CSPU$NOS(,1,0)
                    (00148) *
          000@0A92  (00149)          #L = AFDT$ORG+3*W$*(199-128)    ; (FCB #199 - ENRG)
  0A92    001E6AFE  (00150)          ADDR    ENU$(R7,1)
  0A94    001E6B7@  (00151)          ADDR    ENS$(R7,1)
  0A96    001021FC  (00152)          ADDR    CSPU$NOS(,1,0)
                    (00153) *
          00000A62  (00154)          #L = AFDT$ORG+3*W$*(191-128)    ;FCB 191 DCOR
  0A62    001E6D44  (00155)          ADDR    DCRU$(R7,1)
  0A64    001E6E62  (00156)          ADDR    DCRS$(R7,1)
  0A66    001021FC  (00157)          ADDR    CSPU$NOS(,1,0)
```

```
                (00158) *
```



```
                (00159) ;NON-ARRAY FUNCTION DISPATCH TABLE PATCHES
                (00160) *
                (00161) *
      00000BBA  (00162)         #L = FDT$ + (W$ * 105)          ;FCB 105  (MPIFF)
 0000BBA 0000667E (00163)       ADDR    MPIFF$
                (00164) *
      000009C6  (00165)         #L = FDT$ + (W$ * 111)          ;FCB 111 MPMBS
 00008C6 00006EBC (00166)       ADDR    MPMBS$
                (00167) *
```



```
                (00168) ;APU, APS, AND CSPU CODE FOR ADDED FUNCTIONS
                (00169) *
                (00170) *
                (00171) *
                (00172) * SET START ADDRESS FOR ADDED FUNCTIONS.
                (00173) * REL 3.5 LEAVES EMPTY SPACE (ON BUS 1) AT:
                (00174) *           $3B00 - $4000
                (00175) *           $4800 - $4A00
                (00176) *           $5D00 - $C000
                (00177) * PUT ADDED FUNCTIONS IN THIS SPACE.
                (00178) *
      00005D00  (00179)         #L = $5D00
                (00180) *
```

```
                    (00181) ; DECODING TABLES
                    (00182) *
05D00 989999C1 (00183) BBTAB   DATA -3.075
05D02 8D53F7C1 (00184)         DATA -1.666
05D04 EA3D70C0 (00185)         DATA -0.830
05D06 9DD2F1C0 (00186)         DATA -0.233
05D08 1DD2F1C0 (00187)         DATA 0.233
05D0A 6A3D70C0 (00188)         DATA 0.830
05D0C 0D53F7C1 (00189)         DATA 1.666
05D0E 189999C1 (00190)         DATA 3.075
                    (00191) DKTAB1:
05D10 FA700C40 (00192)         DATA   -0.9565597E+00
05D12 F995AE40 (00193)         DATA   -0.9498804E+00
05D14 F89A2740 (00194)         DATA   -0.9422044E+00
05D16 F7796B40 (00195)         DATA   -0.9333929E+00
05D18 F62E67C0 (00196)         DATA   -0.9232912E+00
05D1A F4B378C0 (00197)         DATA   -0.9117274E+00
05D1C F3027340 (00198)         DATA   -0.8985123E+00
05D1E F1148AC0 (00199)         DATA   -0.8834394E+00
05D20 EEE2724C (00200)         DATA   -0.8662856E+00
05D22 EC6458C0 (00201)         DATA   -0.8468127E+00
05D24 E9921CC0 (00202)         DATA   -0.8247701E+00
05D26 E6631B40 (00203)         DATA   -0.7998995E+00
05D28 E2CE5E40 (00204)         DATA   -0.7719400E+00
05D2A DECD3040 (00205)         DATA   -0.7406369E+00
05D2C DA560940 (00206)         DATA   -0.7057506E+00
05D2E D56287C0 (00207)         DATA   -0.6671694E+00
05D30 CFED16C0 (00208)         DATA   -0.6244229E+00
05D32 C9F1FCCC (00209)         DATA   -0.5776974E+00
05D34 C36FE2CC (00210)         DATA   -0.5268520E+00
05D36 BC685440 (00211)         DATA   -0.4719339E+00
05D38 B4E02F40 (00212)         DATA   -0.4130916E+00
05D3A ACDFFB40 (00213)         DATA   -0.3505854E+00
05D3C A4740C40 (00214)         DATA   -0.2847915E+00
05D3E 9BAC73C0 (00215)         DATA   -0.2162003E+00
05D40 929CAD40 (00216)         DATA   -0.1454064E+00
05D42 895B1140 (00217)         DATA   -0.7309163E-01
05D44 FFFFFF39 (00218)         DATA   -0.3725290E-08
05D46 095B1CC0 (00219)         DATA    0.7309162E-01
05D48 129CAD40 (00220)         DATA    0.1454064E+00
05D4A 1BAC73CC (00221)         DATA    0.2162003E+00
05D4C 24740C40 (00222)         DATA    0.2847915E+00
05D4E 2CDFFB40 (00223)         DATA    0.3505854E+00
                    (00224) DKTAB2:
05D50 B59B6040 (00225)         DATA   -0.4183042E+00
05D52 AD85174C (00226)         DATA   -0.3556241E+00
05D54 A4FE8740 (00227)         DATA   -0.2890176E+00
05D56 9C184240 (00228)         DATA   -0.2194903E+00
05D58 92E68240 (00229)         DATA   -0.1476596E+00
05D5A 89809640 (00230)         DATA   -0.7423668E-01
05D5C 97FFFFBA (00231)         DATA   -0.1117587E-07
05D5E 09809640 (00232)         DATA    0.7423665E-01
05D60 12E68240 (00233)         DATA    0.1476596E+00
```

```
05D62 1C184140 (00234)          DATA    0.2194902E+00
05D64 24FE8740 (00235)          DATA    0.2990176E+00
05D66 2D851740 (00236)          DATA    0.3556241E+00
05D68 359B6040 (00237)          DATA    0.4188042E+00
05D6A 3D34CF40 (00238)          DATA    0.4781741E+00
05D6C 4448DC40 (00239)          DATA    0.5334735E+00
05D6E 4AD2DA40 (00240)          DATA    0.5845597E+00
05D70 50D198C0 (00241)          DATA    0.6313964E+00
05D72 5646E8C0 (00242)          DATA    0.6740390E+00
05D74 5B3708C0 (00243)          DATA    0.7126174E+00
05D76 5FA827C0 (00244)          DATA    0.7473192E+00
05D78 63A1B940 (00245)          DATA    0.7783729E+00
05D7A 672C1E40 (00246)          DATA    0.8060339E+00
05D7C 6A5029C0 (00247)          DATA    0.8305714E+00
05D7E 6D16D3C0 (00248)          DATA    0.8522508E+00
05D80 6F80E4C0 (00249)          DATA    0.8713652E+00
05D82 71AEE540 (00250)          DATA    0.8881499E+00
05D84 7390DAC0 (00251)          DATA    0.9028581E+00
05D86 75364440 (00252)          DATA    0.9157186E+00
05D88 76A60840 (00253)          DATA    0.9269419E+00
05D8A 77E67F40 (00254)          DATA    0.9367199E+00
05D8C 78FD2EC0 (00255)          DATA    0.9452265E+00
05D8E 79EF5F40 (00256)          DATA    0.9526176E+00
              (00257)  DKTAB3:
05D9C F3C82EC0 (00258)          DATA   -0.9045466E+00
05D92 F1EAFAC0 (00259)          DATA   -0.8899835E+00
05D94 EFC9CFC0 (00260)          DATA   -0.8733463E+00
05D96 ED5C8CC0 (00261)          DATA   -0.8543869E+00
05D98 EA9A97C0 (00262)          DATA   -0.8328428E+00
05D9A E77AF8C0 (00263)          DATA   -0.8084403E+00
05D9C E3F492C0 (00264)          DATA   -0.7809013E+00
05D9E DFF561C0 (00265)          DATA   -0.7499506E+00
05DA0 DB8FD240 (00266)          DATA   -0.7153266E+00
05DA2 D6A12DC0 (00267)          DATA   -0.6767938E+00
05DA4 D12C1440 (00268)          DATA   -0.6341577E+00
05DA6 CB2C0A40 (00269)          DATA   -0.5872815E+00
05DA8 C49F0E40 (00270)          DATA   -0.5361040E+00
05DAA BD862A40 (00271)          DATA   -0.4806569E+00
05DAC B5E5F3C0 (00272)          DATA   -0.4210801E+00
05DAE ADC6F2C0 (00273)          DATA   -0.3576339E+00
05DB0 A535D240 (00274)          DATA   -0.2907050E+00
05DB2 9C4355C0 (00275)          DATA   -0.2208049E+00
05DB4 930407C0 (00276)          DATA   -0.1495605E+00
05DB6 898F9840 (00277)          DATA   -0.7469469E-01
05DB8 88000F3B (00278)          DATA   -0.5960465E-07
05DBA 898F9740 (00279)          DATA    0.7469457E-01
05DBC 13042640 (00280)          DATA    0.1485603E+00
05DBE 1C4354C0 (00281)          DATA    0.2208048E+00
05DC0 2535D4C0 (00282)          DATA    0.2907048E+00
05DC2 2DC6F1C0 (00283)          DATA    0.3576338E+00
05DC4 35E5F340 (00284)          DATA    0.4210800E+00
05DC6 3D962940 (00285)          DATA    0.4806568E+00
05DC8 449F0540 (00286)          DATA    0.5361040E+00
```

DECODING TABLES

```
05DCA 4B2C0940 (00287)          DATA     0.5872814E+00
05DCC 512C1340 (00288)          DATA     0.6341576E+00
05DCE 56A12DC0 (00289)          DATA     0.6767938E+00
                (00290)  DKTAB4:
05DD0 A2D27140 (00291)          DATA    -0.2720472E+00
05DD2 97081140 (00292)          DATA    -0.1839315E+00
05DD4 0BDF7640 (00293)          DATA    -0.9275703E-01
05DD6 FFFFFF39 (00294)          DATA    -0.3725290E-08
05DD8 0BDF7640 (00295)          DATA     0.9275702E-01
05DDA 17081140 (00296)          DATA     0.1839315E+00
05DDC 22D27140 (00297)          DATA     0.2720472E+00
05DDE 2D08AE40 (00298)          DATA     0.3550252E+00
05DE0 37957E40 (00299)          DATA     0.4342497E+00
05DE2 40D84D4E (00300)          DATA     0.5066010E+00
05DE4 49464C40 (00301)          DATA     0.5724578E+00
05DE6 50DAABC0 (00302)          DATA     0.6316733E+00
05DE8 57983F40 (00303)          DATA     0.6843337E+00
05DEA 5D87D540 (00304)          DATA     0.7307078E+00
05DEC 62B60140 (00305)          DATA     0.7711946E+00
05DEE 67340AC0 (00306)          DATA     0.8062757E+00
                (00307)  DKTAB5:
05DF0 C9C0E94E (00308)          DATA    -0.5761997E+00
05DF2 C26759C0 (00309)          DATA    -0.5187790E+00
05DF4 BA66CCC0 (00310)          DATA    -0.4562622E+00
05DF6 B1C73240 (00311)          DATA    -0.3888915E+00
05DF8 A8965640 (00312)          DATA    -0.3170879E+00
05DFA 9EE802C0 (00313)          DATA    -0.2414554E+00
05DFC 94D5AEC0 (00314)          DATA    -0.1627711E+00
05DFE 8A7DAE40 (00315)          DATA    -0.8196046E-01
05E00 9E2E71BD (00316)          DATA    -0.5756650E-04
05E02 0A79EEC0 (00317)          DATA     0.8184609E-01
05E04 14D202C0 (00318)          DATA     0.1626590E+00
05E06 1EE47540 (00319)          DATA     0.2413470E+00
05E08 2892F1C0 (00320)          DATA     0.3169844E+00
05E0A 31C3FEC0 (00321)          DATA     0.3887939E+00
05E0C 3A63CFC0 (00322)          DATA     0.4561710E+00
05E0E 42649740 (00323)          DATA     0.5186948E+00
                (00324)  DKTAB6:
05E10 A1972940 (00325)          DATA    -0.2624256E+00
05E12 96801940 (00326)          DATA    -0.1772491E+00
05E14 886F3940 (00327)          DATA    -0.8933180E-01
05E16 9000003A (00328)          DATA    -0.7450581E-08
05E18 886F3940 (00329)          DATA     0.8933177E-01
05E1A 16801940 (00330)          DATA     0.1772491E+00
05E1C 21972940 (00331)          DATA     0.2624256E+00
05E1E 28FE5D40 (00332)          DATA     0.3437001E+00
05E20 35C6E5C0 (00333)          DATA     0.4201324E+00
05E22 3EDA3CC0 (00334)          DATA     0.4910351E+00
05E24 472A5040 (00335)          DATA     0.5559788E+00
05E26 4EB100C0 (00336)          DATA     0.6147767E+00
05E28 556F1640 (00337)          DATA     0.6674526E+00
05E2A 5B6AEB40 (00338)          DATA     0.7142004E+00
05E2C 60AF01C0 (00339)          DATA     0.7553408E+00
```

```
05E2E 6548AB40 (00340)          DATA    0.7912802E+00
                (00341)  DKTAB7:
05E30 C03ABF40 (00342)          DATA   -0.5017928E+00
05E32 B2225640 (00343)          DATA   -0.3916729E+00
05E34 A2708BC0 (00344)          DATA   -0.2690596E+00
05E36 918814C0 (00345)          DATA   -0.1370569E+00
05E38 9FFFFFBA (00346)          DATA   -0.1490116E-07
05E3A 118813C0 (00347)          DATA    0.1370568E+00
05E3C 22708AC0 (00348)          DATA    0.2690595E+00
05E3E 32225640 (00349)          DATA    0.3916729E+00
                (00350)  DKTAB9:
05E40 9D9CC640 (00351)          DATA   -0.2313469E+00
05E42 8F835FC0 (00352)          DATA   -0.1172924E+00
05E44 9E2EF1BD (00353)          DATA   -0.5757022E-04
05E46 0EFFB7C0 (00354)          DATA    0.1171789E+00
05E48 1D993440 (00355)          DATA    0.2312379E+00
05E4A 2B6D6540 (00356)          DATA    0.3392760E+00
05E4C 3832BCC0 (00357)          DATA    0.4390484E+00
05E4E 43B7FEC0 (00358)          DATA    0.5290526E+00
                (00359)  EDTAB:
                (00360) **
                (00361) ** FIRST 10 VALUES CHANGED 12/79 KFIELD
                (00362) ** (MAKE SILENCE QUIETER...)
                (00363) **
                (00364) **      DATA    0.4983422E-03
                (00365) **      DATA    0.5489435E-03
                (00366) **      DATA    0.6046829E-03
                (00367) **      DATA    0.6660819E-03
                (00368) **      DATA    0.7337154E-03
                (00369) **      DATA    0.8082164E-03
                (00370) **      DATA    0.8902821E-03
                (00371) **      DATA    0.9806808E-03
                (00372) **      DATA    0.1080258E-02
                (00373) **      DATA    0.1189947E-02
                (00374) **
05E50 53E2D63C (00375)          DATA    0.10E-4
05E52 08E9B3BD (00376)          DATA    0.17E-4
05E54 0EAE18BD (00377)          DATA    0.29E-4
05E56 181E03BD (00378)          DATA    0.46E-4
05E58 2752543D (00379)          DATA    0.75E-4
05E5A 3EEA20BD (00380)          DATA    0.12E-3
05E5C 68DB8BBD (00381)          DATA    0.20E-3
05E5E 0A7C5ABE (00382)          DATA    0.32E-3
05E60 10B63FBE (00383)          DATA    0.51E-3
05E62 1ADEA8BE (00384)          DATA    0.82E-3
                (00385) **
05E64 2AF38FBE (00386)          DATA    0.1310773E-02
05E66 2F500CBE (00387)          DATA    0.1443869E-02
05E68 341D53E (00388)           DATA    0.1590478E-02
05E6A 39699F3E (00389)          DATA    0.1751974E-02
05E6C 3F3CE7BE (00390)          DATA    0.1929868E-02
05E6E 45A0B83E (00391)          DATA    0.2125826E-02
05E70 4C8871BE (00392)          DATA    0.2341681E-02
```

DECODING TABLES

```
05E72 5486053E (00393)          DATA    0.2579453E-02
05E74 5D1B213E (00394)          DATA    0.2841369E-02
05E76 668F563E (00395)          DATA    0.3129880E-02
05E78 78F9493E (00396)          DATA    0.3447686E-02
05E7A 7C71ED8E (00397)          DATA    0.3797761E-02
05E7C 08914C3F (00398)          DATA    0.4183383E-02
05E7E 097000BF (00399)          DATA    0.4608161E-02
05E80 0A6552BF (00400)          DATA    0.5076071E-02
05E82 0B738D3F (00401)          DATA    0.5591492E-02
05E84 0C9D383F (00402)          DATA    0.6159248E-02
05E86 0DE51CBF (00403)          DATA    0.6784653E-02
05E88 0F4E4C3F (00404)          DATA    0.7473562E-02
05E8A 10DC293F (00405)          DATA    0.8232423E-02
05E8C 12926BBF (00406)          DATA    0.9068338E-02
05E8E 14752E3F (00407)          DATA    0.9989130E-02
05E90 1688F63F (00408)          DATA    0.1100342E-01
05E92 18D2BCBF (00409)          DATA    0.1212070E-01
05E94 1B57FEBF (00410)          DATA    0.1335143E-01
05E96 1E1EC43F (00411)          DATA    0.1470712E-01
05E98 212DB63F (00412)          DATA    0.1620047E-01
05E9A 248C293F (00413)          DATA    0.1784546E-01
05E9C 28422CBF (00414)          DATA    0.1965747E-01
05E9E 2C58A8BF (00415)          DATA    0.2165348E-01
05EA0 30D966BF (00416)          DATA    0.2385216E-01
05EA2 35CF30BF (00417)          DATA    0.2627409E-01
05EA4 3B45E93F (00418)          DATA    0.2894194E-01
05EA6 414AA83F (00419)          DATA    0.3188068E-01
05EA8 47EBDB3F (00420)          DATA    0.3511783E-01
05EAA 4F3960BF (00421)          DATA    0.3868366E-01
05EAC 5744BD3F (00422)          DATA    0.4261158E-01
05EAE 6021338F (00423)          DATA    0.4693833E-01
05EB0 69E400BF (00424)          DATA    0.5170441E-01
05EB2 74A483BF (00425)          DATA    0.5695444E-01
05EB4 0007C8C0 (00426)          DATA    0.6273756E-01
05EB6 00D886C0 (00427)          DATA    0.6910788E-01
05EB8 09BE7740 (00428)          DATA    0.7612504E-01
05EBA 0ABBC940 (00429)          DATA    0.8385473E-01
05EBC 0BD2C1C0 (00430)          DATA    0.9236928E-01
05EBE 0D061740 (00431)          DATA    0.1017484E+00
05EC0 0E58A240 (00432)          DATA    0.1120799E+00
05EC2 0FCD9CC0 (00433)          DATA    0.1234604E+00
05EC4 11695440 (00434)          DATA    0.1359964E+00
05EC6 132CD2C0 (00435)          DATA    0.1498054E+00
05EC8 151F43C0 (00436)          DATA    0.1650166E+00
05ECA 17444FC0 (00437)          DATA    0.1817722E+00
05ECC 19A11D40 (00438)          DATA    0.2002293E+00
05ECE 1C3852C0 (00439)          DATA    0.2205604E+00
                (00440) DTQTAB:
05ED0 75C3258F (00441)          DATA    0.5750112E-01
05ED2 15E33D40 (00442)          DATA    0.1709973E+00
05ED4 23DB4CC0 (00443)          DATA    0.2801300E+00
05ED6 30F4DA40 (00444)          DATA    0.3824723E+00
05ED8 3CF486C0 (00445)          DATA    0.4762181E+00
```

DECODING TABLES

```
05EDA 47B692C0 (00446)          DATA    0.5602592E+00
05EDC 512C1440 (00447)          DATA    0.6341577E+00
05EDE 59596740 (00448)          DATA    0.6980409E+00
05EE0 60509040 (00449)          DATA    0.7524586E+00
05EE2 662C8DC0 (00450)          DATA    0.7982347E+00
05EE4 680D1140 (00451)          DATA    0.8363363E+00
05EE6 6F132640 (00452)          DATA    0.8677719E+00
05EE8 725ED9C0 (00453)          DATA    0.8935196E+00
05EEA 750DC8C0 (00454)          DATA    0.9144832E+00
05EEC 773A5F40 (00455)          DATA    0.9314609E+00
05EEE 78FB92C0 (00456)          DATA    0.9451774E+00
               (00457) *
               (00458) *
               (00459)
```

```
                (00460) ; APU3-VLTSY    VECTOR LATTICE SYNTHESIS FILTER
00000003        (00461) $N=3
                (00462) *
                (00463) *       CODED BY LCOSELL, 3/79
                (00464) *
                (00465) *       BINDS TO APS3-V3200
                (00466) *       PERFORMS LATTICE SYNTHESIS FILTER, USING REFL'N COEFFS
                (00467) *
                (00468) *       IMPLEMENTS THE FOLLOWING FORTRAN CODE:
                (00469) *
                (00470) *       DO 20 I=1,N
                (00471) *
                (00472) *       F(7)=W(I)-G(7)*K(8)
                (00473) *       DO 10 J=6,0,-1
                (00474) *       F(J)=F(J+1)-G(J)*K(J+1)
                (00475) *10     G(J+1)=G(J)+F(J)*K(J+1)
                (00476) *       G(0)=F(0)
                (00477) *20     Y(I)=F(0)
                (00478) *
                (00479) *
                (00480) * THERE ARE THREE INPUT VECTORS:
                (00481) *       K (LENGTH 8) REFLECTION COEFFICIENTS (V BID)
                (00482) *       G (LENGTH 9) FILTER MEMORY (U BID)
                (00483) *       W (LENGTH N) INPUT RESIDUAL SAMPLES (W BID)
                (00484) *THERE ARE TWO OUTPUT ARRAYS:
                (00485) *       G AS BEFORE
                (00486) *       Y OUTPUT SYNTHETIC SPEECH SAMPLES (LENGTH N) (Y BID)
                (00487) *
                (00488) *INPUT STREAM:
                (00489) *       G(7)
                (00490) *       G(6)
                (00491) *       ...,
                (00492) *       G(0)
                (00493) *       K(8)
                (00494) *       K(7)
                (00495) *       ...,
                (00496) *       K(1)
                (00497) *       W(1)
                (00498) *       W(2)
                (00499) *       ...,
                (00500) *       W(N).
                (00501) *
                (00502) *OUTPUT STREAM:
                (00503) *       Y(1)
                (00504) *       Y(2)
                (00505) *       ...,
                (00506) *       Y(N)
                (00507) *       G(7)
                (00508) *       G(6)
                (00509) *       ...,
                (00510) *       G(0).
                (00511) *
                (00512) *
```

```
                    (00513) *       REGISTER USAGE:
                    (00514) *LEFT           RIGHT
                    (00515) *       M0=G1           M0=G0
                    (00516) *       M1=G3           M1=G2
                    (00517) *       M2=G5           M2=G4
                    (00518) *       M3=G7           M3=G6
                    (00519) *NOTE: THESE ARE BEGINNING VALUES, ONLY.  AFTER EACH G IS USED FOR THE FINAL TI
                    (00520) *THE REGISTER IS THEN USED FOR TEMP STORAGE FOR F(J), AND THEN FOR
                    (00521) *THE G'S FOR THE NEXT INPUT SAMPLE
                    (00522) *
                    (00523) *       M4=K2           M4=K1
                    (00524) *       M5=K4           M5=K3
                    (00525) *       M6=K6           M6=K5
                    (00526) *       M7=K8           M7=K7
                    (00527) *
                    (00528) *       A0=INPUT,TEMP   A0=TEMP
                    (00529) *       A1=G1           A1=G0
                    (00530) *       A2=TEMP         A2=TEMP
                    (00531) *       A3=G3           A3=G2
                    (00532) *       A4=TEMP         A4=TEMP
                    (00533) *       A5=G5           A5=G4
                    (00534) *       A6=TEMP         A6=TEMP
                    (00535) *       A7=G7           A7=G6
                    (00536) *
                    (00537) *
                    (00538) *
                    (00539) ********************************************************************************
                    (00540) *
                    (00541)         EVEN
     05EF0 0000     (00542)         DATA    VLTSY$SA
     05EF1 05E2     (00543)         DATA    VLTSY$SZ
                    (00544) *
                    (00545) VLTSY$  BEGIN   APU(VLTSY)
           00000000 (00546)         #A=0
           00000000 (00547)         VLTSY$SA=#A
 A00 05EF2 08C80000 (00548) IN:     MOV(IQ,M3)\NOP  ;G7
 A01 05EF4 08F70000 (00549)         MOV(IQA,A7)\NOP ;G7
 A02 05EF6 0000C8CB (00550)         NOP\MOV(IQ,M3)  ;  ,G6
 A03 05EF8 00000BF7 (00551)         NOP\MOV(IQA,A7) ;  ,G6
 A04 05EFA 08CA0000 (00552)         MOV(IQ,M2)\NOP  ;G5
 A05 05EFC 08F50000 (00553)         MOV(IQA,A5)\NOP ;G5
 A06 05EFE 000008CA (00554)         NOP\MOV(IQ,M2)  ;  ,G4
 A07 05F00 00000BF5 (00555)         NOP\MOV(IQA,A5) ;  ,G4
 A08 05F02 08C90000 (00556)         MOV(IQ,M1)\NOP  ;G3
 A09 05F04 08F30000 (00557)         MOV(IQA,A3)\NOP ;G3
 A0A 05F06 000028C9 (00558)         NOP\MOV(IQ,M1)  ;  ,G2
 A0B 05F08 00000BF3 (00559)         NOP\MOV(IQA,A3) ;  ,G2
 A0C 05F0A 08C80000 (00560)         MOV(IQ,M0)\NOP  ;G1
 A0D 05F0C 08F10000 (00561)         MOV(IQA,A1)\NOP ;G1
 A0E 05F0E 00000BC8 (00562)         NOP\MOV(IQ,M0)  ;  ,G0
 A0F 05F10 00000BF1 (00563)         NOP\MOV(IQA,A1) ;  ,G0
 A10 05F12 08EF0000 (00564)         MOV(IQA,M7)\NOP ;K8
 A11 05F14 000008EF (00565)         NOP\MOV(IQA,M7) ;  ,K7
```

```
A12 05F16 08EE0000 (00566)          MOV(IQA,M6)\NOP ;K6
A13 05F18 00000BEE (00567)          NOP\MOV(IQA,M6) ;  ,K5
A14 05F1A 85E085E0 (00568)          MUL(M3,M7)\MUL(M3,M7)   ;G7*K8,G6*K7
A15 05F1C 08ED0000 (00569)          MOV(IQA,M5)\NOP ;K4
A16 05F1E 00000BED (00570)          NOP\MOV(IQA,M5) ;  ,K3
A17 05F20 08EC0000 (00571)          MOV(IQA,M4)\NOP ;K2
A18 05F22 00000BEC (00572)          NOP\MOV(IQA,M4) ;  ,K1
A19 05F24 08F00000 (00573)          MOV(IQA,A0)\NOP ;F8=INPUT
A1A 05F26 08B60000 (00574)          MOV(P,A6)\NOP   ;G7*K8
A1B 05F28 4E00C000 (00575)          SUB(A0,A6)\NOP  ;F8-G7*K8>F7
A1C 05F2A 85400556 (00576)          MUL(M2,M6)\MOV(A6),MUL(M2,M6)   ;G5*K6,G4*K5
A1D 05F2C 08980000 (00577) LOOP:    MOV(R,EXO)\NOP  ;F7
A1E 05F2E 0000B850 (00578)          NOP\MOV(EXI,A0) ;  ,F7
A1F 05F30 00004E00 (00579)          NOP\SUB(A0,A6)  ;  ,F7-G6*K7>F6
A20 05F32 0000B98B (00580)          NOP\MOV(R,M3)   ;  ,F6
A21 05F34 84B485F4 (00581)          MOV(A4),MUL(M1,M5)\MOV(A4),MUL(M3,M7)   ;(G5*K6)G3*K4;(G4K5)F6K7
A22 05F36 0C000898 (00582)          NOP\MOV(R,EXO)  ;  ,F6
A23 05F38 08560000 (00583)          MOV(EXI,A6)\NOP ;F5
A24 05F3A 4CC00200 (00584)          SUB(A6,A4)\NOP  ;F6-G5K6>F5
A25 05F3C 08980000 (00585)          MOV(R,EXO)\NOP  ;F5
A26 05F3E 088A0852 (00586)          MOV(R,M2)\MOV(EXI,A2)   ;F5,F5
A27 05F40 85520486 (00587)          MOV(A2),MUL(M2,M6)\MOV(A6),MUL(M1,M5)   ;(G3K4)F5K6,(F6K7)G2K3
A28 05F42 0000C40B (00588)          NOP\SUB(A2,A4)  ;  ,F5-G4K5>F4
A29 05F44 00000898 (00589)          NOP\MOV(R,EXO)  ;  ,F4
A2A 05F46 08540000 (00590)          MOV(EXI,A4)\NOP ;F4
A2B 05F48 4A8046EA (00591)          SUB(A4,A2)\MOV(M2),ADD(A7,A6)   ;F4-G3K4>F3,(F4)G6+F6K7>G7
A2C 05F4A 84169552 (00592)          MOV(A6),MUL(M2,M4)\MOV(A2),MUL(M2,M6)   ;(F5K6)G1K2,(G2K3)F4K5
A2D 05F4C 08980898 (00593)          MOV(R,EXO)\MOV(R,EXO)   ;F3,G7
A2E 05F4E 084B0854 (00594)          MOV(EXI,M3)\MOV(EXI,A4) ;G7(NEW),F3
A2F 05F50 46A94A80 (00595)          MOV(M1),ADD(A5,A6)\SUB(A4,A2)   ;(F3)G5+F5K6>G6,F3-G2K3>F2
A30 05F52 08570000 (00596)          MOV(EXI,A7)\NOP ;G7(NEW)
A31 05F54 84B08414 (00597)          MOV(A0),MUL(M1,M5)\MOV(A4),MUL(M0,M4)   ;(G1K2)F3K4,(F4K5)G0K1
A32 05F56 00000898 (00598)          NOP\MOV(R,EXO)  ;  ,F2
A33 05F58 08520000 (00599)          MOV(EXI,A2)\NOP ;F2
A34 05F5A 485844A9 (00600)          MOV(EXO),SUB(A2,A0)\MOV(M1),ADD(A5,A4)  ;(G6)F2-G1K2>F1,(F2)G4+
                    (00601) *                                               ;F4K5>G5
A35 05F5C 00000857 (00602)          NOP\MOV(EXI,A7) ;  ,G6(NEW)
A36 05F5E 0000084B (00603)          NOP\MOV(EXI,M3) ;  ,G6(NEW)
A37 05F60 08980898 (00604)          MOV(R,EXO)\MOV(R,EXO)   ;F1,G5
A38 05F62 08000852 (00605)          MOV(R,M0)\MOV(EXI,A2)   ;F1,F1
A39 05F64 841494B0 (00606)          MOV(A4),MUL(M0,M4)\MOV(A0),MUL(M1,M5)   ;(F3K4)F1K2,(G0K1)F2K3
A3A 05F66 446B4840 (00607)          ADD(A3,A4)\SUB(A2,A0)   ;G3+F3K4>G4,F1-G0K1>F0
A3B 05F68 0855000F (00608)          MOV(EXI,A5)\NOP ;G5(NEW)
A3C 05F6A 084A0000 (00609)          MOV(EXI,M2)\NOP ;G5(NEW)
A3D 05F6C 08980088 (00610)          MOV(R,EXO)\MOV(R,M0)    ;G4,F0=G0(NEW)
A3E 05F6E 08B28412 (00611)          MOV(P,A2)\MOV(A2),MUL(M0,M4)    ;F1K2,F0K1
A3F 05F70 422B4278 (00612)          ADD(A1,A2)\MOV(EXO),ADD(A3,A2)  ;G1+F1K2>G2,(F0)G2+F2K3>G3
A40 05F72 085C084A (00613)          MOV(EXI,00)\MOV(EXI,M2) ;F0 OUT,G4(NEW)
A41 05F74 08510855 (00614)          MOV(EXI,A1)\MOV(EXI,A5) ;F0=G0,G4(NEW)
A42 05F76 08980000 (00615)          MOV(R,EXO)\NOP  ;G2
A43 05F78 911D0B4F (00616)          JUMPS(LDONE,FI) ;END IF INPUT USED UP
A44 05F7A 08F00000 (00617)          MOV(IQA,A0)\NOP ;F0(NEW)=INPUT
A45 05F7C 85E085F0 (00618)          MUL(M3,M7)\MOV(A0),MUL(M3,M7)   ;G7K8,(F0K1)G6K7
```

```
A46 05F7E 022040038 (00619)          R(A1)\MOV(EXO),ADD(A1,A0)       ;F0=G0,(G3)G0+F0K1>G1
A47 05F80 00490049 (00620)          MOV(EXI,M1)\MOV(EXI,M1) ;G3(NEW),G2(NEW)
A48 05F82 00530053 (00621)          MOV(EXI,A3)\MOV(EXI,A3) ;G3(NEW),G2(NEW)
A49 05F84 00980098 (00622)          MOV(R,EXO)\MOV(R,EXO)   ;G0=F0,G1
A4A 05F86 00480000 (00623)          MOV(EXI,M0)\NOP ;G1(NEW)
A4B 05F88 00510051 (00624)          MOV(EXI,A1)\MOV(EXI,A1) ;G1(NEW),G0(NEW)
A4C 05F8A 85568556 (00625)          MOV(A6),MUL(M2,M6)\MOV(A6),MUL(M2,M6)   ;(G7K8)G5K6,(G6K7)G4K5
A4D 05F8C 4E00000F (00626)          SUB(A0,A6)\NOP  ;F8-G7K8>F7
A4E 05F8E 10000C1D (00627)          JUMP(LOOP)
                   (00628) *
                   (00629) *        FINISH LAST PART OF LAST LOOP, AND OUTPUT G'S
A4F 05F90 00000880 (00630) LDONE:   NOP\MOV(P,A0)   ;  ,F0K1
A50 05F92 02204038 (00631)          R(A1)\MOV(EXO),ADD(A1,A0)       ;F0=G0,(G3)G0+F0K1
A51 05F94 00530053 (00632)          MOV(EXI,A3)\MOV(EXI,A3) ;G3(NEW),G2(NEW)
A52 05F96 02F802F8 (00633)          MOV(EXO),R(A7)\MOV(EXO),R(A7)   ;(G0)G7,(G1)G6
A53 05F98 089C0000 (00634)          MOV(R,OQ)\NOP   ;G7 OUT
A54 05F9A 02A002BC (00635)          R(A5)\MOV(OQ),R(A5)     ;G5,(G6 OUT),G4
A55 05F9C 089C0000 (00636)          MOV(R,OQ)\NOP   ;G5 OUT
A56 05F9E 0260027C (00637)          R(A3)\MOV(OQ),R(A3)     ;G3,(G4 OUT)G2
A57 05FA0 089C0000 (00638)          MOV(R,OQ)\NOP   ;G3 OUT
A58 05FA2 0000099C (00639)          NOP\MOV(R,OQ)   ;  ,G2 OUT
A59 05FA4 085C0000 (00640)          MOV(EXI,OQ)\NOP ;G1 OUT
A5A 05FA6 0000085C (00641)          NOP\MOV(EXI,OQ) ;  ,G0 OUT
A5B 05FA8 20320000 (00642)          CLEAR(RA)\NOP    ;HALT
A5C 05FAA 00000000 (00643)          NOP
A5D 05FAC 10000000 (00644)          JUMP(0)
                   (00645) *
           0000005E (00646)          VLTSY$SZ=.-VLTSY$SA
     05FAE         (00647)          END     VLTSY$SZ
                   (00648)          EVEN
                   (00649) *
```

```
                    (00650) ; APS3-V3200
                    (00651) *
                    (00652) *       MAP-300 APS PROGRAM FOR VLTSY(V,U,V,W)
                    (00653) *       CODED BY KFIELD 3/79
                    (00654) *
                    (00655) *       INPUT STREAM: U(7),...,U(0),V(7),...,V(0),W(0),...,W(WBS-1),[FIN]
                    (00656) *
                    (00657) *       OUTPUT STREAM: Y(0),...,Y(WBS-1),U(7),...,U(0),[EOD]
                    (00658) *
                    (00659) *
                    (00660) *       HEADER BLOCK
                    (00661) *
                    (00662)         EVEN
    05FAE 0000601A  (00663)         ADDR    V3200$I         ;PTR TO CONSTR. INSTR. BLK.
    05FB0 0000000F  (00664)         ADDR    0               ;PTR TO SCALAR BLOCK (NO SCALARS)
    05FB2 000F      (00665)         DATA    0               ;NUMBER OF SCALARS
    05FB3 000F      (00666)         DATA    V3200$Z         ;MODULE SIZE
    05FB4 00006012  (00667)         ADDR    V3200$A         ;PTR TO CHAIN ANCHOR
                    (00668)         EVEN
                    (00669) *
                    (00670) *
                    (00671) V3200$  BEGIN   APS(V3200)
                    (00672) *
                    (00673) *
                    (00674) *       INPUT PROGRAM
                    (00675) *
                    (00676) *
A00 05FB6 002010400 (00677)         JSN(V3202$6,P2)         ;SET OUTPUT PC
A01 05FB8 023000300 (00678)         SET(RO)                 ;TURN ON OUTPUT GENERATION
                    (00679) *
                    (00680) *       INPUT PGM REGISTER USAGE:
                    (00681) *               BR0: VECTOR ELEMENT ADDRESSES
                    (00682) *               BR1: BUFFER SIZES
                    (00683) *               BR2: BUFFER SPACINGS
                    (00684) *
                    (00685) *
                    (00686) *       GENERATE "U" ADDRESSES
                    (00687) *
A02 05FBA 044010000 (00688)         LOAD(BR0,[1])           ;BR0 <= VECTOR U BASE ADDR
A03 05FBC F6500000  (00689)         LOAD(BR1,M$$)           ;DUMMY LOAD
A04 05FBE 086000000 (00690)         LOAD(BR2,M$$)           ;BR2 <= VECTOR U SPACING
A05 05FC0 0A290031  (00691)         SUBL(BR2,1)             ;BR2 <= SPACING-1
                    (00692) *
A06 05FC2 0C29003F  (00693) #1      ADDL(BR0,7)             ;GEN ADDR OF U(7)
A07 05FC4 0E2906B1  (00694)         SUBL(BR2,1),JUMPP(#1)   ;ADD (7 * SPACING)
                    (00695) *
A08 05FC6 100A900C  (00696)         ADD(BR0,[9])            ;BR0 <= ADDR(U7))+SPACING
                    (00697) *
A09 05FC8 12500007  (00698)         LOAD(BR1,7)             ;BR1 <= 7
A0A 05FCA 14029004  (00699) #2      SUB(BR0,[9],TF)         ;GEN ADDR (U(I))
A0B 05FCC 16190AB1  (00700)         SUBL(BR1,1),JUMPP(#2)   ;LOOP 8 TIMES
                    (00701) *
                    (00702) *
```

```
                    (00703) *       GENERATE "V" ADDRESSES
                    (00704) *
A0C 05FCE 18402004  (00705)         LOAD(BR0,[2])           ;BR0 <= VECTOR V BASE ADDR
A0D 05FD0 1A500000  (00706)         LOAD(BR1,M$$)           ;DUMMY LOAD
A0E 05FD2 1C600000  (00707)         LOAD(BR2,M$$)           ;BR2 <= VECTOR V SPACING
A0F 05FD4 1E290031  (00708)         SUBL(BR2,1)             ;BR2 <= SPACING-1
                    (00709) *
A10 05FD6 20090F3F  (00710) #3      ADDL(BR0,7)             ;GEN ADDR OF V(7)
A11 05FD8 22291EB1  (00711)         SUBL(BR2,1),JUMPP(#3)   ;ADD (7 * SPACING)
                    (00712) *
A12 05FDA 240AA00C  (00713)         ADD(BR0,[10])           ;BR0 <= ADDR(V(7)) + SPACING
                    (00714) *
A13 05FDC 26500007  (00715)         LOAD(BR1,7)             ;BR1 <= 7
A14 05FDE 2882A0F4  (00716) #4      SUB(BR0,[10],TF)        ;GEN ADDR(V(I))
A15 05FE0 2A1914B1  (00717)         SUBL(BR1,1),JUMPP(#4)   ;LOOP 8 TIMES
                    (00718) *
                    (00719) *
                    (00720) *       GENERATE "W" ADDRESSES
                    (00721) *
A16 05FE2 2C403004  (00722)         LOAD(BR0,[3])           ;BR0 <= VECTOR W BASE ADDR
A17 05FE4 2E500000  (00723)         LOAD(BR1,M$$)           ;BR1 <= WBS-1
A18 05FE6 30020000  (00724)         SUB(BR0,M$$)            ;BR2 <= W BASE MINUS SPACING
                    (00725) *
A19 05FE8 328AB0E6  (00726) #5      ADD(BR0,[11],TF)        ;GEN ADDR(W(I))
A1A 05FEA 34191981  (00727)         SUBL(BR1,1),JUMPP(#5)   ;LOOP WBS TIMES
                    (00728) *
                    (00729) *       WHEN DONE GENERATING INPUT ADDRESSES, CLEAR RI
                    (00730) *
A1B 05FEC 36200031  (00731)         CLEAR(RI)               ;HALT INPUT
A1C 05FEE 38000020  (00732)         NOP(0)
                    (00733) *
                    (00734) *
                    (00735) *       OUTPUT PROGRAM
                    (00736) *
                    (00737) *
                    (00738) *       OUTPUT PGM REG USAGE:
                    (00739) *               BW0: VECTOR ELEMENT ADDRESSES
                    (00740) *               BW1: BUFFER SIZES
                    (00741) *               BW2: BUFFER SPACINGS
                    (00742) *               BW3: SCRATCH REGISTER
                    (00743) *
A1D 05FF0 3A300032  (00744) V3200$6 SET(RA)                 ;TURN ON APU
                    (00745) *
                    (00746) *       GEN "Y" ADDRESSES
                    (00747) *
A1E 05FF2 3C40000A  (00748)         LOAD(BW0,[0])           ;BW0 <= Y BASE ADDR
A1F 05FF4 3E500000  (00749)         LOAD(BW1,M$$)           ;DUMMY LOAD
A20 05FF6 40020000  (00750)         SUB(BW0,M$$)            ;BW0 <= Y BASE MINUS SPACING
                    (00751) *
                    (00752) *       GET WBS-1  (GENERATE WBS Y'S)
                    (00753) *
A21 05FF8 42703006  (00754)         LOAD(BW3,[3])           ;SCRATCH <= W BASE
A22 05FFA 44500000  (00755)         LOAD(BW1,M$$)           ;BW1 <= WBS-1
```

```
A23 05FFC 46700000 (00756)          LOAD(BW3,MS$)            ;DUMMY LOAD
                   (00757) *
A24 05FFE 488A8006 (00758) #7       ADD(BW0,[8],TF)          ;GEN ADDR(Y(I))
A25 06000 4A1124B1 (00759)          SUBL(BW1,1),JUMPP(#7)    ;LOOP WBS TIMES
                   (00760) *
                   (00761) *       GENERATE "U" ADDRESSES
                   (00762) *
A26 06002 4C401F04 (00763)          LOAD(BW0,[1])            ;BW0 <= VECTOR U BASE ADDRESS
A27 06004 4E500000 (00764)          LOAD(BW1,MS$)            ;DUMMY LOAD
A28 06006 50600000 (00765)          LOAD(BW2,MS$)            ;BW2 <= VECTOR U SPACING
A29 06008 52210031 (00766)          SUBL(BW2,1)              ;BW2 <= SPACING-1
                   (00767) *
A2A 0600A 5401003F (00768) #8       ADDL(BW0,7)              ;GEN ADDR OF U(7)
A2B 0600C 56212AB1 (00769)          SUBL(BW2,1),JUMPP(#8)    ;ADD (7*SPACING)
                   (00770) *
A2C 0600E 580A900C (00771)          ADD(BW0,[9])             ;BW0 <= ADDR(U(7)) + SPACING
                   (00772) *
A2D 06010 5A500007 (00773)          LOAD(BW1,7)              ;BW1 <= 7
A2E 06012 5C029004 (00774) #9       SUB(BW0,[9],TF)          ;GEN ADDR(U(I))
A2F 06014 5E112EB1 (00775)          SUBL(BW1,1),JUMPP(#9)    ;LOOP 8 TIMES
                   (00776) *
                   (00777) *
                   (00778) *
A30 06016 60200030 (00779)          CLEAR(RO)                ;HALT OUTPUT
A31 06018 62000020 (00780)          NOP(0)
                   (00781) *
          00006012 (00782) V3200$A=#C                        ;ASSIGN VALUE TO CHAIN ANCHOR
    0601A          (00783)          END     #A-1            ;END OF MUDULE
                   (00784) *
                   (00785) *       CONSTR. INSTR. BLOCK
                   (00786) *
    0601A 00000000 (00787) V3200$I DATA    14F'0.0'
     ...
          00000080 (00788) V3200$Z=#L-V3200$                 ;MODULE SIZE
                   (00789) *
```

```
                    (00790) ; APU3-VKTOA    COMPUTE PREDICTOR COEFS. FROM REFL'N COEFS.
          00000003  (00791) #M=3
                    (00792) *
                    (00793) *       CODED BY LCOSELL 4/79
                    (00794) *
                    (00795) *       COMPUTES 9 LINEAR PREDICTION COEFS. FROM 8 REFLECTION COEFS.
                    (00796) *                       (A'S FROM K'S)
                    (00797) *
                    (00798) *       BINDS TO APS3-V1100K
                    (00799) *
                    (00800) *       EQ:   A[M](M)=K[M]
                    (00801) *             A[M](L)=A[M-1](L)+K[M]*A[M-1](M-L)
                    (00802) *                WHERE [.] INDICATES ITERATION, (.) INDEX
                    (00803) *       EQ:  SEE EQ. (3) FROM
                    (00804) *               J. MARKHOUL AND R. VISWANATHAN
                    (00805) *               "ADAPTIVE LATTICE METHODS FOR LINEAR PREDICTION",
                    (00806) *               ICASSP 1978 PROCEEDINGS, PG. 83-86.
                    (00807) *
                    (00808) *       INPUT STREAM:  K(1),K(2),...,K(8),[FI]
                    (00809) *
                    (00810) *       OUTPUT STREAM: A(0),A(1),...,A(8),[EO]
                    (00811) *
                    (00812) *       NUMBER OF INPUTS AND OUTPUTS FIXED AT 8 AND 9, RESPECT.
                    (00813) *       [FI] AND [EO] ARE NOT USED
                    (00814) *
                    (00815) *       REGISTER USAGE
                    (00816) *       M0  A[M-1](M-1)          A[M-1](1)
                    (00817) *       M1  A[M-1](M-2)          A[M-1](2) M>3
                    (00818) *       M2  A[M-1](M-3) M>5      A[M-1](3) M>5
                    (00819) *       M3  K[M]                 K[M]
                    (00820) *       M4  A[M-1](M-4) M>7      A[M-1](4) M>7
                    (00821) *       M5  UNUSED               UNUSED
                    (00822) *       M6  UNUSED               UNUSED
                    (00823) *       M7  K[M]                 K[M]
                    (00824) *
                    (00825) *       A0  K*A                  K*A
                    (00826) *       A1  A[M](1) M>0           A[M](M) M>0
                    (00827) *       A2  A[M](2) M>2           A[M](M-1) M>2
                    (00828) *       A3  A[M](3) M>4           A[M](M-2) M>4
                    (00829) *       A4  A[M](4) M>6           A[M](M-3) M>6
                    (00830) *       A5  UNUSED               UNUSED
                    (00831) *       A6  UNUSED               UNUSED
                    (00832) *       A7  UNUSED               UNUSED
                    (00833) *
                    (00834) *
                    (00835) ;*****************************************************************
                    (00836) *
                    (00837)         EVEN
          06P36 0000(00838)         DATA    VKTOA$SA
          06P37 0043(00839)         DATA    VKTOA$SZ
                    (00840) *
                    (00841) VKTOA$  BEGIN   APU(VKTOA)
          00000000  (00842)         #A=0
```

```
                00000000 (00843)          VKTOA$SA=.A
                         (00844) *
                         (00845) *
  A00 06038 F8F10000 (00846)          MOV(IQA,A1)\NOP          ;K(1)=A(1)
  A01 0603A 32200000 (00847)          NORM(A1)\NOP     FLOAT IT
  A02 0603C 00800000 (00848)          MOV(R,M0)\NOP
  A03 0603E 16801680 (00849)          K(+1)            GET 1.0 FOR A(0)
  A04 06040 09D008D0 (00850)          MOV(IQ,A0)       K2
  A05 06042 089C0000 (00851)          MOV(R,DQ)\NOP    A(0)
  A06 06044 32003200 (00852)          NORM(A0)         FLOAT IT
  A07 06046 080FC08F (00853)          MOV(R,M7)
  A08 06048 3000C032 (00854)          CALL(X23RD)      DO 2ND ITERATION (M=2)
  A09 0604A 30000032 (00855)          CALL(X23RD)      DO THIRD ITERATION (M=4)
  A0A 0604C 3000002D (00856)          CALL(X4TH)               M=4
  A0B 0604E 3000002E (00857)          CALL(X5TH)               M=5
  A0C 06050 00000260 (00858)          NOP\R(A3)        GET AC5](3) READY
  A0D 06052 30000027 (00859)          CALL(X6TH)       M=6
  A0E 06054 30000029 (00860)          CALL(X7TH)       M=7
  A0F 06056 00000280 (00861)          NOP\R(A4)        GET AC7](4) READY
  A10 06058 3000C023 (00862)          CALL(X8TH)
  A11 0605A 02200000 (00863)          R(A1)\NOP
  A12 0605C 089C0000 (00864)          MOV(R,DQ)\NOP    A(1) OUT
                         (00865) *
  A13 0605E 02400000 (00866)          R(A2)\NOP
  A14 06060 089C0000 (00867)          MOV(R,DQ)\NOP    A(2) OUT
                         (00868) *
  A15 06062 02600000 (00869)          R(A3)\NOP
  A16 06064 089C0000 (00870)          MOV(R,DQ)\NOP    A(3) OUT
                         (00871) *
  A17 06066 02800280 (00872)          R(A4)\R(A4)
  A18 06068 089C0000 (00873)          MOV(R,DQ)\NOP    A(4) OUT
                         (00874) *
  A19 0606A 0000099C (00875)          NOP\MOV(R,DQ)    A(5) OUT
  A1A 0606C 00000260 (00876)          NOP\R(A3)
  A1B 0606E 0000089C (00877)          NOP\MOV(R,DQ)    A(6) OUT
                         (00878) *
  A1C 06070 00000240 (00879)          NOP\R(A2)
  A1D 06072 0000089C (00880)          NOP\MOV(R,DQ)    A(7) OUT
                         (00881) *
  A1E 06074 00000220 (00882)          NOP\R(A1)
  A1F 06076 0000089C (00883)          NOP\MOV(R,DQ)    A(8) OUT
                         (00884) *
                         (00885) *
                         (00886) *
  A20 06078 20322032 (00887)          CLEAR(RA)                ;HALT
  A21 0607A 00000000 (00888)          NOP
  A22 0607C 10000000 (00889)          JUMP(0)
                         (00890) *
                         (00891) *
                         (00892) *
                         (00893) *
  A23 0607E 00000898 (00894) X8TH:    NOP\MOV(R,EXO)   ;\AC7](4)
  A24 06080 08542854 (00895)          MOV(EXI,A4)      AC7](4)
```

```
A25 0608Z 85800000 (00896)          MUL(M3,M4)\NOP           ACM-1]{M-4)*KCM] ((M=8))
                   (00897) *
A26 06084 10000029 (00898)          JUMP(X7TH)
A27 06086 00000898 (00899) X6TH:    NOP\MOV(R,EXO)  AC5](3)
A28 06088 08530000 (00900)          MOV(EXI,A3)\NOP AC5](3)
A29 0608A 857Z8560 (00901) X7TH:    MOV(AZ),MUL(M2,M7)\MUL(M2,M7)    ACM-1](M-3)*KCM]\ACM-1](3)*KCM]
A2A 0609C 44000000 (00902)          ADD(A2,A4)\NOP  SUM FOR ACM](4)
A2B 0608E C8940000 (00903)          MOV(R,A4)\NOP   STORE ACM](4)
                   (00904) *
A2C 06090 1002002E (00905)          JUMP(X5TH)
A2D 06092 08520000 (00906) X4TH:    MOV(EXI,A2)\NOP AC3](2)
A2E 06094 84F084FZ (00907) X5TH:    MOV(AZ),MUL(M1,M7)\MOV(AZ),MUL(M1,M7)    ACM-1](M-2)*KCM]\
                   (00908) *                                                 ACM-1](2)*KCM]
A2F 06096 43004300 (00909)          ADD(AZ,A3)\ADD(AZ,A3)    SUM FOR ACM](3)\ACM](M-3)
A30 06098 08980898 (00910)          MOV(R,EXO)\MOV(R,EXO)
A31 0609A 084C084A (00911)          MOV(EXI,M4)\MOV(EXI,M2) STORE ACM](M-3)\ACM](3)
                   (00912) *
A32 0609C 847Z8470 (00913) X23RD:   MOV(AZ),MUL(MZ,M7)\MOV(AZ),MUL(MZ,M7)   ACM-1](M-1)*KCM]\
                   (00914) *                                                 ACM-1](1)*KCM]
A33 0609E 42134214 (00915)          MOV(A3),ADD(AZ,A2)\MOV(A4),ADD(AZ,A2)   STORE ACM](3),
                   (00916) *                                                 SUM FOR ACM](2)\
                   (00917) *                                        ST ACM](M-3),SUM ACM](M-2)
                   (00918)
A34 060A0 08980898 (00919)          MOV(R,EXO)\MOV(R,EXO)
A35 060A2 084A0849 (00920)          MOV(EXI,M2)\MOV(EXI,M1) STORE ACM](M-2)\ACM](2)
A36 060A4 08B008B0 (00921)          MOV(P,AZ)\MOV(P,AZ)
A37 060A6 41124113 (00922)          MOV(A2),ADD(AZ,A1)\MOV(A3),ADD(AZ,A1)   STORE ACM](2),
                   (00923) *                                                 SUM FOR ACM](1)\
                   (00924) *                                        STORE ACM](M-2),SUM ACM](M-1)
                   (00925) *
A38 060A8 08980898 (00926)          MOV(R,EXO)\MOV(R,EXO)
A39 060AA 08490848 (00927)          MOV(EXI,M1)\MOV(EXI,MZ)          ACM](M-1)\ACM](1)
A3A 060AC 08910892 (00928)          MOV(R,A1)\MOV(R,A2)              ACM](1)\ACM](M-1)
A3B 060AE 08F708F7 (00929)          MOV(IQA,A7)      ACM](M),FIXED
A3C 060B0 32E032E0 (00930)          NORM(A7)
A3D 060B2 08880891 (00931)          MOV(R,MZ)\MOV(R,A1)      ACM](M),FLOATING
A3E 060B4 09D709D7 (00932)          MOV(IQ,A7)       KCM+1],FIXED
A3F 060B6 32E032E0 (00933)          NORM(A7)         FLOAT IT
A40 060B8 088B0888 (00934)          MOV(R,M3)        KCM+1],FLOATING
A41 060BA 089F098F (00935)          MOV(R,M7)        KCM+1],FLOATING
A42 060BC A000A000 (00936)          RETURN
                   (00937) *
                   (00938) *
                   (00939) *
          00000043 (00940)          VKTOA$SZ=#A-VKTOA$SA
    060BE          (00941)          END     VKTOA$SZ
                   (00942)          EVEN
                   (00943) *
                   (00944) *
```

```
                    (00945) * APS3-V1100K
                    (00946) *
                    (00947) *       MAP-300 APS PROGRAM FOR VKTOA(Y,U)
                    (00948) *       [WHERE Y'S ARE A'S; U'S ARE K'S]
                    (00949) *
                    (00950) *       CODED BY KFIELD 4/79
                    (00951) *
                    (00952) *       INPUT STREAM: U(0),U(1),...,U(7),[EI]
                    (00953) *       INPUT IS FIXED, LONG (TE) FORMAT
                    (00954) *
                    (00955) *       OUTPUT STREAM: Y(0),Y(1),...,Y(8),[EO]
                    (00956) *       OUTPUT IS FLOAT, LONG (TF) FORMAT
                    (00957) *
                    (00958) *       BUFFER SIZES ARE IGNORED:
                    (00959) *        8 U'S ARE INPUT,
                    (00960) *        9 Y'S ARE OUTPUT
                    (00961) *
                    (00962) *
                    (00963) *       HEADER BLOCK
                    (00964) *
                    (00965)         EVEN
  060BE 00060EE (00966)         ADDR    V1100K$I                ;PTR TO CONSTR. INSTR. BLK.
  060C0 00000000 (00967)        ADDR    0               ;PTR TO SCALAR BLOCK (NO SCALARS)
  060C2 0000     (00968)        DATA    0               ;NUMBER OF SCALARS
  060C3 0030     (00969)        DATA    V1100K$Z                ;MODULE SIZE
  060C4 00060E6 (00970)         ADDR    V1100K$A                ;PTR TO CHAIN ANCHOR
                    (00971)         EVEN
                    (00972) *
                    (00973) *
                    (00974) V1100K$ BEGIN   APS(V1100K)
                    (00975) *
                    (00976) *
                    (00977) *       INPUT PROGRAM
                    (00978) *
A00 060C6 00200A40 (00979)          JSN(V1100K$2,P2)                ;SET OUTPUT PC
A01 060C8 02300030 (00980)          SET(R0)         ;TURN ON OUTPUT GENERATION
                    (00981) *
                    (00982) *       INPUT PGM REGISTER USAGE:
                    (00983) *               BR0: VECTOR ELEMENT ADDRESSES
                    (00984) *               BR1: BUFFER SIZES MINUS ONE (FIXED AT 7)
                    (00985) *
                    (00986) *
                    (00987) *       GENERATE 'U' ADDRESSES
                    (00988) *
A02 060CA 04401000 (00989)          LOAD(BR0,[I])           ;BR0 <= VECTOR U BASE ADDR
A03 060CC 06500000 (00990)          LOAD(BR1,M$$)           ;DUMMY LOAD
A04 060CE 08020000 (00991)          SUB(BR0,M$$)            ;BR0 <= U BASE MINUS SPACING
A05 060D0 0A500007 (00992)          LOAD(BR1,7)             ;SET SIZE-1 TO 7
                    (00993) *
                    (00994) * INPUT ADDRESS GENERATION LOOP
                    (00995) *
A06 060D2 0D0A9003 (00996) #1       ADD(BR0,[9],TE)         ;GEN U-ELEMENT ADDR (FXD,LNG FORMAT)
A07 060D4 0E1906B1 (00997)          SUBL(BR1,1),JUMPP(#1)    ;CONTINUE UNTIL LAST ELEM
```

```
                    (00998) *
  A08 060D6 10200031 (00999)          CLEAR(RI)                ;HALT INPUT
  A09 060D8 12000020 (01000)          NOP(0)
                    (01001) *
                    (01002) *
                    (01003) *         OUTPUT PROGRAM
                    (01004) *         OUTPUT PGM REGISTER USAGE:
                    (01005) *                 BW0: VECTOR ELEMENT ADDRESSES
                    (01006) *                 BW1: BUFFER SIZES MINUS ONE (FIXED AT 8)
                    (01007) *
  A0A 060DA 1430FF32 (01008) V1100K$2         SET(RA)                  ;TURN ON APU
  A0B 060DC 1640000A (01009)          LOAD(BW0,[0])            ;LOAD 'Y' BASE ADDRESS
  A0C 060DE 18500000 (01010)          LOAD(BW1,M$$)            ;DUMMY LOAD
  A0D 060E0 1AE2000F (01011)          SUB(BW0,M$$)             ;SET BW0 TO Y BASE MINUS SPACING
  A0E 060E2 1C500007 (01012)          LOAD(BW1,7)              ;SET SIZE-1 TO 7
  A0F 060E4 1E1A0001 (01013)          ADD(BW1,1)               ;SET SIZE-1 TO 8
                    (01014) *
                    (01015) *
                    (01016) *         OUTPUT ADDR GEN LOOP
                    (01017) *
  A10 060E6 200A800A (01018) #3       ADD(BW0,[8],TF)          ;GEN Y-ELEMENT ADDR
  A11 060E8 22111081 (01019)          SUBL(BW1,1),JUMPP(#3)     ;CONTINUE UNTIL LAST ELEM
                    (01020) *
                    (01021) *
  A12 060EA 24200030 (01022)          CLEAR(RO)                ;HALT OUTPUT
  A13 060EC 26000020 (01023)          NOP(0)
                    (01024) *
                    (01025) *
            000F67E6 (01026) V1100K$A=#C                       ;ASSIGN VALUE TO CHAIN ANCHOR
                    (01027) *
      060EE         (01028)          END      #A-1            ;END OF MODULE
                    (01029) *
                    (01030) *
                    (01031) * STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                    (01032) *
      060EE 00000000 (01033) V1100K$I         DATA     4F'0.0'
       ...
                    (01034) *
                    (01035) *
            00000030 (01036) V1100K$Z=#L-V1100K$               ;COMPUTE MODULE SIZE
                    (01037) *
                    (01038) *
                    (01039)          EVEN
```

```
                        (01040) * APU3 - PRTRB(Y,A,U,B,V)   UPSAMPLE(3:1) WITH PERTURBATION
                00000003 (01041) #M=3
                        (01042) *
                        (01043) * BINDS TO APS3 - P2120
                        (01044) *
                        (01045) * EQ.:   (FOR I=0,(UBS-1)):
                        (01046) *        Y(3I),Y(3I+1),Y(3I+2) =
                        (01047) *
                        (01048) *        0,U(I),0    IF -.5 <= (V(I)*(SA + (SB)*ABS(U(I))) < .5
                        (01049) *        U(I),0,0    IF         (V(I)*(SA + (SB)*ABS(U(I))) < -.5
                        (01050) *        0,0,U(I)    IF         (V(I)*(SA + (SB)*ABS(U(I))) >= .5
                        (01051) *
                        (01052) * WHERE Y IS PERTURBED UPSAMPLED OUTPUT VECTOR
                        (01053) *       U IS DOWNSAMPLED INPUT VECTOR
                        (01054) *       V IS VECTOR OF RANDOM NUMBERS
                        (01055) *       SA,SB ARE EXTERNAL CONSTANTS
                        (01056) *
                        (01057) * INPUT STREAM: SA,SB,U(0),V(0),U(1),V(1),...,U(UBS-1),V(UBS-1),[EFI]
                        (01058) *
                        (01059) * OUTPUT STREAM: Y(0),Y(1),Y(2),...,Y((3*UBS)-1),[EOI]
                        (01060) *
                        (01061) *
                        (01062)         EVEN
     060F6 0000         (01063)         DATA    PRTRB$SA
     060F7 0046         (01064)         DATA    PRTRB$SZ
                        (01065) *
                        (01066) PRTRB$  BEGIN   APU(PRTRB)
                00000000 (01067)        #A=0
                        (01068) *
                00000000 (01069) PRTRB$SA=#A
                        (01070) *
A00 060F8 166C1660      (01071) #1      K(.5)           ; R <= .5
A01 060FA 08F008F0      (01072)         MOV(IQA,A0)     ; A0 <= SA
A02 060FC 08E808E8      (01073)         MOV(IQA,M0)     ; M0 <= SB
A03 060FE 06910691      (01074)         MOV(A1),K(-1.)  ; A1 <= .5 , R <= -1.
A04 06100 08920892      (01075)         MOV(R,A2)       ; A2 <= -1.
A05 06102 08130813      (01076)         MOV(ZERO,A3)    ; A3 <= 0.
                        (01077) *
A06 06104 08D40000      (01078) #2      MOV(IQ,A4)\NOP  ; LEFT:A4 <= U(I)
A07 06106 08EC0000      (01079)         MOV(IQA,M4)\NOP ; LEFT:M4 <= U(I)
A08 06108 08ED0000      (01080)         MOV(IQA,M5)\NOP ; LEFT:M5 <= V(I)
                        (01081) *
A09 0610A 000008D4      (01082)         NOP\MOV(IQ,A4)  ; RIGHT:A4 <= U(I+1)
A0A 0610C 000008EC      (01083)         NOP\MOV(IQA,M4) ; RIGHT:M4 <= U(I+1)
                        (01084) *
A0B 0610E 94009400      (01085)         MULABS(M0,M4)   ; P <= SB * ABS(U(I))
                        (01086) *
A0C 06110 000008ED      (01087)         NOP\MOV(IQA,M5) ; RIGHT:M5 <= V(I+1)
A0D 06112 08B608B6      (01088)         MOV(P,A6)       ; A6<= SB * ABS(U(I))
                        (01089) *
A0E 06114 46004600      (01090)         ADD(A0,A6)      ; R <= SA + (SB * ABS(U(I)))
A0F 06116 08960896      (01091)         MOV(R,A6)       ; MAKE SURE SUM IS .GE. 0
A10 06118 66606660      (01092)         MAX(A3,A6)      ; (A3 = 0)
```

```
A11 0611A 08890889 (01093)          MOV(R,M1)          ; M1<= SA + (SB * ABS(U(I)))
A12 0611C 84AF84A0 (01094)          MUL(M1,M5)         ; P <= V(I) * (SA + (SB ABS(U(I))))
A13 0611E 08B508B5 (01095)          MOV(P,A5)          ; A5 <= P
                   (01096) *
                   (01097) * DO PERTURB ON LEFT BOARD DATA
                   (01098) *
A14 06120 45200000 (01099) LEFT:    ADD(A1,A5)\NOP     ; SEE IF THIS IS <-.5 (ADD .5, TEST FOR <0)
                   (01100) *                                ; (DON'T WORRY ABOUT BOUNDARY CONDITION)
A15 06122 08800980 (01101)          MOV(R,NULL)            ; DON'T DO JUMPS TIL ADD IS DONE
A16 06124 911E002C (01102)          JUMPS(LCASE1,T1)       ; DO LEFT BOARD CASE 1: <-.5
A17 06126 45550000 (01103)          MOV(A5),ADD(A2,A5)\NOP ; SEE IF IT WAS >.5 (SUB .5+.5(TO CANCEL
                   (01104) *                                ;  PREV ADD OF .5), TEST FOR >0)
A18 06128 08800880 (01105)          MOV(R,NULL)            ; DON'T DO JUMPC TIL ADD IS DONE
A19 0612A 921E0031 (01106)          JUMPC(LCASE2,T1)       ; DO LEFT BOARD CASE 2: >.5
                   (01107) *                                ; ELSE DO LEFT BOARD CASE 3: >-.5 & <+.5
A1A 0612C 02800000 (01108) LCASE3:  R(A4)\NOP      ; R <= U(I)
A1B 0612E 081C0000 (01109)          MOV(ZERO,OQ)\NOP       ; DON'T PERTURB
A1C 06130 089C0000 (0111E)          MOV(R,OQ)\NOP
A1D 06132 081C0000 (01111)          MOV(ZERO,OQ)\NOP       ; GO TEST RIGHT BOARD DATA
                   (01112) *
                   (01113) *
                   (01114) *
A1E 06134 00004520 (01115) RIGHT:   NOP\ADD(A1,A5)  ; DO PERTURBATION ON RIGHT BOARD DATA
A1F 06136 00E00880 (01116)          NOP\MOV(R,NULL)
A20 06138 911F0036 (01117)          JUMPS(RCASE1,T2)
A21 0613A 00004555 (01118)          NOP\MOV(A5),ADD(A2,A5)
A22 0613C 00000880 (01119)          NOP\MOV(R,NULL)
A23 0613E 901F003E (01120)          JUMPC(RCASE2,T2)
                   (01121) *
A24 06140 00090280 (01122) RCASE3:  NOP\R(A4)
A25 06142 0000081C (01123)          NOP\MOV(ZERO,OQ)
A26 06144 0000089C (01124)          NOP\MOV(R,OQ)
A27 06146 0000081C (01125)          NOP\MOV(ZERO,OQ)
A28 06148 901D0006 (01126)          JUMPC($2,F1)
A29 0614A 00000000 (01127)          NOP
A2A 0614C 00000000 (01128)          NOP
A2B 0614E 10030043 (01129)          JUMP(PRDONE)
A2C 06150 02800000 (01130) LCASE1:  R(A4)\NOP      ; <-.5 , PERTURB BY -1
A2D 06152 089C0000 (01131)          MOV(R,OQ)\NOP
A2E 06154 081C0000 (01132)          MOV(ZERO,OQ)\NOP
A2F 06156 081C0000 (01133)          MOV(ZERO,OQ)\NOP
A30 06158 1000001E (01134)          JUMP(RIGHT)    ; GO TEST RIGHT BOARD DATA
                   (01135) *
A31 0615A 02800000 (01136) LCASE2:  R(A4)\NOP      ; >.5 , PERTURB BY +1
A32 0615C 081C0000 (01137)          MOV(ZERO,OQ)\NOP
A33 0615E 081C0000 (01138)          MOV(ZERO,OQ)\NOP
A34 06160 089C0000 (01139)          MOV(R,OQ)\NOP
A35 06162 1000001E (01140)          JUMP(RIGHT)    ; GO TEST RIGHT BOARD DATA
                   (01141) *
A36 06164 00000280 (01142) RCASE1:  NOP\R(A4)
A37 06166 0000089C (01143)          NOP\MOV(R,OQ)
A38 06168 0000081C (01144)          NOP\MOV(ZERO,OQ)
A39 0616A 0000081C (01145)          NOP\MOV(ZERO,OQ)
```

```
  A3A 0616C 9C1D0006 (01146)          JUMPC(#2,FI)
  A3B 0616E 00C00000 (01147)          NOP
  A3C 06170 00000000 (01148)          NOP
  A3D 06172 10000043 (01149)          JUMP(PRDONE)
                     (01150) *
  A3E 06174 00000280 (01151) RCASE2:  NOP\R(A4)
  A3F 06176 0000081C (01152)          NOP\MOV(ZERO,OQ)
  A40 06178 0000081C (01153)          NOP\MOV(ZERO,OQ)
  A41 0617A 00000A9C (01154)          NOP\MOV(R,OQ)
  A42 0617C 9C1D0006 (01155)          JUMPC(#2,FI)
                     (01156) *
                     (01157) *
  A43 0617E 2C322032 (01158) PRDONE:  CLEAR(RA)
  A44 06180 00CE0000 (01159)          NOP
  A45 06182 10000000 (01160)          JUMP(0)
                     (01161) *
            00000046 (01162)          PRTRB$SZ=#A-PRTRB$SA
      06184          (01163)          END     PRTRB$SZ
                     (01164)          EVEN
                     (01165) *
```

```
                    (01166) ; APS3 - P2120   APS PROGRAM FOR PRTRB(Y,A,U,B,V)
                    (01167) *
                    (01168) *
                    (01169) * INPUT STREAM: SA,SB,U(0),V(0),U(1),V(1),...,U(UBS-1),V(UBS-1),[EI]
                    (01170) *
                    (01171) * OUTPUT STREAM: Y(0),Y(1),Y(2),...,Y((3*UBS)-1),[EO]
                    (01172) *
                    (01173) *
                    (01174) *
                    (01175)         EVEN
 06184 000061C8     (01176)         ADDR    P2120$I         ; CONSTR INSTR BLK
 06186 00006190     (01177)         ADDR    P2120$+2*P2120$S        ; SCALAR BLK
 06188 0002         (01178)         DATA    2               ; NUMBER OF SCALARS
 06189 0041         (01179)         DATA    P2120$Z         ; MODULE SIZE
 0618A 000061C0     (01180)         ADDR    P2120$A         ; PTR TO CHAIN ANCHOR
                    (01181)         EVEN
                    (01182) *
                    (01183) *
                    (01184) P2120$  BEGIN   APS(P2120)
                    (01185) *
                    (01186) *
                    (01187) * INPUT PROGRAM
                    (01188) *       REGISTER USAGE:
                    (01189) *        BR0: SCALAR AND BUFFER "U" ELEMENT ADDRESSES
                    (01190) *        BR1: BUFFER "U" SIZE MINUS ONE
                    (01191) *        BR2: BUFFER "V" ELEMENT ADDRESSES
                    (01192) *        BR3: SCRATCH
                    (01193) *
A00 0618C 00200F40  (01194)         JSN(P2120$2,P2)         ; SET OUTPUT PC
A01 0618E 02300030  (01195)         SET(R0)                 ; TURN ON OUTPUT GENERATION
                    (01196) *
A02 06190 04C20000  (01197) P2120$S LOAD(BR0,M$$(1),TF)     ; GEN SA ADDR
A03 06192 06C20000  (01198)         LOAD(BR0,M$$(1),TF)     ; GEN SB ADDR
                    (01199) *
A04 06194 08401000  (01200)         LOAD(BR0,[1])           ; LOAD U BASE ADDR
A05 06196 0A500002  (01201)         LOAD(BR1,M$$)           ; LOAD U SIZE MINUS 1
A06 06198 0C020000  (01202)         SUB(BR0,M$$)            ; SET BR0 TO U BASE MINUS SPACING
                    (01203) *
A07 0619A 0E602006  (01204)         LOAD(BR2,[2])           ; LOAD V BASE ADDR
A08 0619C 10700000  (01205)         LOAD(BR3,M$$)           ; DUMMY LOAD
A09 0619E 12220002  (01206)         SUB(BR2,M$$)            ; SET BR2 TO V BASE MINUS SPACING
                    (01207) *
A0A 061A0 148A9006  (01208) $1      ADD(BR0,[9],TF)         ; GEN U-ELEMENT ADDR
A0B 061A2 16AAA002  (01209)         ADD(BR2,[10],TF)        ; GEN V-ELEMENT ADDR
A0C 061A4 18190AB1  (01210)         SUBL(BR1,1),JUMPP($1)   ; CONTINUE TIL UBS ELEMENTS
                    (01211) *
A0D 061A6 1A200031  (01212)         CLEAR(RI)               ; HALT INPUT
A0E 061A8 1C000020  (01213)         NOP(0)
                    (01214) *
                    (01215) *
                    (01216) *
                    (01217) * OUTPUT PGM
                    (01218) *       REGISTER USAGE:
```

```
                    (01219) *         BW0: BUFFER 'Y' ELEMENT ADDRESSES
                    (01220) *         BW1: (3*UBS)-1
                    (01221) *         BW2: SCRATCH
                    (01222) *
  A0F 061AA 1E300032 (01223) P2120$2 SET(RA)                    ; TURN ON APU
  A10 061AC 20400020 (01224)         LOAD(BW0,[0])              ; LOAD Y BASE ADDR
  A11 061AE 22600000 (01225)         LOAD(BW2,M$$)              ; DUMMY LOAD
  A12 061B0 24020000 (01226)         SUB(BW0,M$$)               ; SET BW0 TO Y BASE MINUS SPACING
                    (01227) *
  A13 061B2 26601006 (01228)         LOAD(BW2,[1])              ; DUMMY LOAD
  A14 061B4 285F0000 (01229)         LOAD(BW1,M$$)              ; LOAD U SEIZE MINUS ONE
  A15 061B6 2A600000 (01230)         LOAD(BW2,M$$)              ; DUMMY LOAD
  A16 061B8 2C210012 (01231)         MOVB(BW2,BW1)              ; BW2 = BW1 = UBS-1
  A17 061BA 2E11002A (01232)         ADDB(BW1,BW1)              ; BW1 <= 2UBS-2
  A18 061BC 3011002C (01233)         ADDB(BW1,BW2)              ; BW1 <= 3UBS-3
  A19 061BE 321A0002 (01234)         ADD(BW1,2)                 ; BW1 <= 3UBS-1
                    (01235) *
  A1A 061C0 348A800E (01236) #3      ADD(BW0,[8],TF)            ; GEN Y-ELEMENT ADDR
  A1B 061C2 36111AB1 (01237)         SUBL(BW1,1),JUMPP(#3)      ; CONTINUE TIL UBS*3 ELEMENTS
                    (01238) *
                    (01239) *
  A1C 061C4 38200030 (01240)         CLEAR(R0)                  ; HALT OUTPUT
  A1D 061C6 3A000020 (01241)         NOP(0)
                    (01242) *
                    (01243) *
            000061C0 (01244) P2120$A=#C                         ; CHAIN ANCHOR
                    (01245) *
      061C8         (01246)         END     #A-1
                    (01247) *
                    (01248) * STORAGE BLK FOR CONSTR INSTRS
                    (01249) *
      061C8 00000000 (01250) P2120$I DATA    7F'0.0'
       ...
            0000004A (01251) P2120$Z=#L-P2120$                  ; MODULE SIZE
```

```
                    (01252) ; MWLF(Y,A,U,V)   WEINER-LEVINSON MATRIX SOL'N WITH FXD PT OUTPUT
                    (01253) *
                    (01254) *
                    (01255) *
                    (01256) !       MWLF-APU PROGRAM
                    (01257) *
                    (01258) *
                    (01259) * WEINER LEVINSON DURBAN INVERSE
                    (01260) *
                    (01261) * MATHEMATICS
                    (01262) * SEE J. MAKHOUL,LINEAR PREDICTION REVIEW
                    (01263) * PROC. IEEE, VOL 63, PP566, APR 1975
                    (01264) *
                    (01265) * BUFFER DEFINITIONS
                    (01266) *
                    (01267) * V(K), CORRELATION COEFFICIENTS, V(K)=R(K)
                    (01268) * V(K) MUST BE COMPACT 32 BIT FLOATING POINT
                    (01269) * VSIZE NOT USED IN COMPUTATION, BUT
                    (01270) * VSIZE > USIZE FOR VALID RESULTS.
                    (01271) *
                    (01272) * U(K),A(K) COEFFICIENTS
                    (01273) * U(K),MUST BE COMPACT 32 BIT FLOATING POINT
                    (01274) * U(K)=A(K-1),IE.A(0)=1 NOT IN BUFFER
                    (01275) *
                    (01276) * Y(K), PARTIAL CORRELATIONS AND ERROR
                    (01277) * Y(K) MUST BE COMPACT 32 BIT FLOATING POINT
                    (01278) * YSIZE=2*USIZE
                    (01279) * Y(K)=P(K-1), 0<=K<USIZE
                    (01280) * Y(K+USIZE)=E(K-1), USIZE<=K<YSIZE
                    (01281) *CHANGED FROM MWLD SUCH THAT A ARRAY HOLDS TWO MULTIPLEXED LONG FIXED
                    (01282) *ARRAYS:  THE CODED K'S AND THE QUANTIZED K'S
                    (01283) *
                    (01284) * A,STABILITY TEST PARAMETER
                    (01285) * IF /P(N)/> CONTENTS OF (A), THEN
                    (01286) * FOR J>0
                    (01287) * A(N+J)=P(N+J)=0, E(N+J)=E(N)
                    (01288) *
                    (01289) *
                    (01290) * MWLF-APU INITIALIZATION
                    (01291) *
                    (01292) *
                    (01293)         EVEN                            START ON WORD BOUNDRY
 061D6 0000         (01294)         DATA    MWLF$SA                 START ADDRESS
 061D7 0086         (01295)         DATA    MWLF$SZ                 SIZE
                    (01296) *
                    (01297) *
                    (01298) MWLF$APU  BEGIN APU(MWLF)
 00000000           (01299) #A=0
                    (01300) *
A00 061D8 08F70000  (01301) MWLF$SA MOV(IQA,A7)\NOP                 A7=STABILITY TEST VALUE
A01 061DA 08C080C   (01302)         MOV(ZERO,M4)                    M4=P(0)=0
A02 061DC 08CD0000  (01303)         MOV(IQ,M5)\NOP
A03 061DE 08F50000  (01304)         MOV(IQA,A5)\NOP                 M5=A5=E(0)=R(0)
```

```
                    (01305) *
                    (01306) * MWLF-APU CALCULATION OF S(N)
                    (01307) * S(N)=R(N) + SUM[1,N-1] OF R(N-J)A(N-1,J)
                    (01308) * REGISTER CONTENTS AT START
                    (01309) * M4=P(N-1), M5=A5=E(N-1), A6=1, A7=TEST
                    (01310) *
 A04 061E0 2EA02EA0 (01311) #1      RCP(A5)
 A05 061E2 08F20000 (01312)         MOV(IQA,A2)\NOP              A2=R(N)
 A06 061E4 09080808 (01313)         MOV(ZERO,M0)
 A07 061E6 84008400 (01314)         MUL(M0,M4)                    SETS P = 0
 A08 061E8 02490249 (01315)         MOV(M1), R(A2)               M1=RCP(E(N-1))
 A09 061EA 900A0051 (01316)         JUMPC(MWLFRW,AF2)            FIRST TIME
                    (01317) *
 A0A 061EC 904B0C10 (01318)         JUMPC(MWLFSE,AF3),SET        EVEN OR ODD N
                    (01319) *
 A0B 061EE 202B202B (01320)         CLEAR(AF3)
                    (01321) *
 A0C 061F0 08EA0000 (01322) #2      MOV(IQA,M2)\NOP              M2=A(N-1,J)
 A0D 061F2 08EE0000 (01323)         MOV(IQA,M6)\NOP              M6=R(N-J)
 A0E 061F4 85508550 (01324)         MOV(A0), MUL(M2,M6)
 A0F 061F6 42124212 (01325)         MOV(A2), ADD(A0,A2)          S+A(N-1,J+1)R(N-J-1)
                    (01326) *
 A10 061F8 08EB0000 (01327) MWLFSE  MOV(IQA,M3)\NOP              M3=A(N-1,J+1)
 A11 061FA 08EF0000 (01328)         MOV(IQA,M7)\NOP              M7=R(N-J-1)
 A12 061FC 85F185F1 (01329)         MOV(A1), MUL(M3,M7)
 A13 061FE 42324232 (01330)         MOV(A2), ADD(A1,A2)          S+A(N-1,J)R(N-J)
                    (01331) *
 A14 06200 9016000C (01332)         JUMPC(#2,FWI)
                    (01333) *
 A15 06202 08B008B0 (01334)         MOV(P,A0)
 A16 06204 42124212 (01335)         MOV(A2), ADD(A0,A2)
                    (01336) *
 A17 06206 20372037 (01337) MWLFRE  CLEAR(WI)
                    (01338) *
                    (01339) *
                    (01340) * MWLF-APU CALCULATION OF P(N)
                    (01341) * P(N)=-S(N)/E(N-1)
                    (01342) * REGISTER CONTENTS AT START
                    (01343) * M4=P(N-1), A5=M5=E(N-1), A6=1, A7=TEST
                    (01344) * M1=RCP[E(N-1)], R=S(N)
                    (01345) *
                    (01346) *
 A18 06208 84A084A0 (01347)         MUL(M1,M5)                   E* RCP(E)
 A19 0620A 16AA16AA (01348)         MOV(M2), K(2)                M2=S(N)
 A1A 0620C 08B008B0 (01349)         MOV(P,A0)                    A0=E* RCP(E)
 A1B 0620E 48314831 (01350)         MOV(A1), SUB(A1,A0)          A1=2
 A1C 06210 168E168E (01351)         MOV(M6), K(1)                M6=2-E* RCP(E)
 A1D 06212 84C084C0 (01352)         MUL(M1,M6)
 A1E 06214 08A908A9 (01353)         MOV(P,M1)                    M1=A4=E1R,
 A1F 06216 84B484B4 (01354)         MOV(A4), MUL(M1,M5)          E1R APPRO(1/E) TO 2^(-11)
 A20 06218 08B008B0 (01355)         MOV(P,A0)
 A21 0621A 48D648D6 (01356)         MOV(A6), SUB(A6,A0)          A6=1
 A22 0621C 088E098E (01357)         MOV(R,M6)                    M6=1-E1R* E
```

```
A23 0621E 84C084C0 (01358)          MUL(M1,M6)
A24 06220 08B008B0 (01359)          MOV(P,A0)                    A0=E1R(1-E* E1R)
A25 06222 44004400 (01360)          ADD(A0,A4)
A26 06224 088E088E (01361)          MOV(R,M6)                    R=1/E(N-1), FULL PRECISION
A27 06226 A540A540 (01362)          MULN(M2,M6)
A28 06228 08B408B4 (01363)          MOV(P,A4)
A29 0622A 02800280 (01364)          R(A4)     P(N)
A2A 0622C 08A808A8 (01365)          MOV(P,M0)                    M0=M4=A4=P(N)
A2B 0622E 08AC08AC (01366)          MOV(P,M4)
A2C 06230 0B9C0000 (01367)          MOV(R,OQ)\NOP                OQ = P(N) (FIXED) = A(N,N)
                   (01368) *
                   (01369) *
                   (01370) *
                   (01371) * MWLF-APU CALCULATION OF A(N,J)
                   (01372) * A(N-1,J)=A(J)
                   (01373) * A(N,J)=A(J)+P(N)A(N-J)
                   (01374) * A(N,N-J)=A(N-J)+P(N)A(J)
                   (01375) * LOOP COMPUTES BOTH A(N,J) AND A(N,N-J)
                   (01376) * FOR N=2M, A(N,M) COMPUTED TWICE
                   (01377) *
                   (01378) * REGISTER CONTENTS AT START
                   (01379) * M0=M4=A4=P=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                   (01380) *
                   (01381) * DATA SEQUENCE
                   (01382) * INPT, [A(N-J+1),A(N-J),A(J)] FOR [1,N/2],WI, A(N/2+1)
                   (01383) * OTPT, DUMP, DUMP [A(N,J), A(N,N-J)]
                   (01384) *
                   (01385) *
A2D 06232 9004A0053 (01386)          JUMPC(MWLFAW,AF2), SET       FIRST TIME AF2 CLEAR
                   (01387) *
A2E 06234 08F30000 (01388)          MOV(IQA,A3)\NOP              A3=A(N-J+1)
                   (01389) *
A2F 06236 08EA0000 (01390) #3       MOV(IQA,M2)\NOP              M2=A(N-J)
A30 06238 85118511 (01391)          MOV(A1), MUL(M2,M4)
A31 0623A 433C0000 (01392)          MOV(OQ), ADD(A1,A3)\NOP      OQ=A(J-1)+P(N)A(N-J+1)
                   (01393) *                                      A(N-J+1)+P(N)A(J-1)
A32 0623C 08D20000 (01394)          MOV(IQ,A2)\NOP               A2=M3=A(J)
A33 0623E 08EB0000 (01395)          MOV(IQA,M3)\NOP
A34 06240 859C859C (01396)          MOV(A0), MUL(M3,M4)
A35 06242 421C000F (01397)          MOV(OQ), ADD(A0,A2)\NOP      OQ=A(N-J+1)+P(N)A(J-1)
A36 06244 08F30000 (01398)          MOV(IQA,A3)\NOP
A37 06246 00000000 (01399)          NOP
A38 06248 90160002F (01400)         JUMPC(#3,FWI)
                   (01401) *
A39 0624A 2C372037 (01402) MWLFAE   CLEAR(WI)
A3A 0624C 84118411 (01403)          MOV(A1), MUL(M0,M4)
A3B 0624E 433C0000 (01404)          MOV(OQ), ADD(A1,A3)\NOP
                   (01405) *
                   (01406) *
                   (01407) * MWLF-APU CALCULATION OF E(N)
                   (01408) * E(N)=[1-P(N)P(N)]E(N-1)
                   (01409) *
                   (01410) * REGISTER CONTENTS AT START
```

```
                    (01411) *
                    (01412) * M0=M4=A4=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                    (01413) * P=P(K)P(N),R=P(N)=A(N,N)
                    (01414) * DATA SEQUENCE
                    (01415) * OTPT  P(K), E(N)
A3C 06250 08B808B0  (01416)         MOV(P,A0)                       A0=P(N)P(N)
A3D 06252 40DC000F  (01417)         MOV(OQ), SUB(A6,A0)\NOP         OQ=A(N,N)
A3E 06254 028A828A  (01418)         MOV(M2), R(A4)                  M2=1-P(N)P(N)
A3F 06256 85208520  (01419)         MUL(M2,M5)                      E(N-1)* [1-P(N)P(N)]
A40 06258 74FC0000  (01420)         MOV(OQ), MAXABS(A7,A4)\NOP        OQ=P(N)
A41 0625A 08800888  (01421)         MOV(R,NULL)                     T1 SET IF /P(N)/>TEST
A42 0625C 08BC0000  (01422)         MOV(P,OQ)\NOP                   OQ=E(N)
A43 0625E 20092089  (01423)         SETTG(AF1)                      AF1 SET IF /P(N)/>TEST
A44 06260 08B50885  (01424)         MOV(P,A5)
A45 06262 08AD08AD  (01425)         MOV(P,M5)
A46 06264 20402049  (01426)         SET(AF0)                        LET APS/OTPT GO
A47 06266 911D004D  (01427)         JUMPS(MWLFD,FI)                    GO BACK UNLESS DONE
                    (01428) *
A48 06268 901E0004  (01429)         JUMPC(F1,T1)
                    (01430) *
                    (01431) *
                    (01432) * MWLF-APU ABORT ROUTINE
                    (01433) * P REGISTER=E(N)
                    (01434) *
                    (01435) *
A49 0626A 081C000C  (01436) #4      MOV(ZERO,OQ)\NOP                A(N+J)=0
A4A 0626C 081C000C  (01437)         MOV(ZERO,OQ)\NOP                P(N+J)=0
A4B 0626E 08BC0000  (01438)         MOV(P,OQ)\NOP                   E(N+J)=E(N)
A4C 06270 901C0049  (01439)         JUMPC(#4,E0)
                    (01440) *
A4D 06272 20322232  (01441) MWLFD   CLEAR(RA)
A4E 06274 00000000  (01442)         NOP
A4F 06276 10000055  (01443)         JUMP(KQUAN)
A50 06278 00000000  (01444)         NOP
A51 0627A 90160051  (01445) MWLFRW  JUMPC(MWLFRW,FWI)
A52 0627C 10000017  (01446)         JUMP(MWLFRE)
A53 0627E 00E00000  (01447) MWLFAW  MOV(IQA,NULL)\NOP
A54 06280 10000039  (01448)         JUMP(MWLFAE)
                    (01449) *
A55 06282 168008F1  (01450) KQUAN   K(+1)\MOV(IQA,A1)       ;1 ;2**-15
A56 06284 16861686  (01451)         MOV(A6),K(+2)
A57 06286 22B522B5  (01452)         MOV(A5),INCRE(A5)       ;32
A58 06288 08970899  (01453)         MOV(R,A7)\MOV(R,EXO)
A59 0628A 30000070  (01454)         CALL DOQ        ;K1
A5A 0628C 00570000  (01455)         MOV(EXI,A7)\NOP         ;32
A5B 0628E 30000070  (01456)         CALL DOQ        ;K2
A5C 06290 00570000  (01457)         MOV(EXI,A7)\NOP         ;32
A5D 06292 30000070  (01458)         CALL DOQ        ;K3
A5E 06294 16001600  (01459)         K(+1)
A5F 06296 22F722F7  (01460)         MOV(A7),INCRE(A7)       ;GET 16
A60 06298 08970899  (01461)         MOV(R,A7)\MOV(R,EXO)
A61 0629A 30000070  (01462)         CALL DOQ        ;K4
A62 0629C 00570000  (01463)         MOV(EXI,A7)\NOP         ;16
```

```
A63 0629E 30000070 (01464)          CALL DOQ          ;K5
A64 062A0 08570000 (01465)          MOV(EXI,A7)\NOP           ;16
A65 062A2 30000070 (01466)          CALL DOQ          ;K6
A66 062A4 16E016E0 (01467)          K(+8)
A67 062A6 08970898 (01468)          MOV(R,A7)\MOV(R,EXO)
A68 062A8 3000007E (01469)          CALL DOQ          ;K7
A69 062AA 08570000 (01470)          MOV(EXI,A7)\NOP
A6A 062AC 30000070 (01471)          CALL DOQ          ;K8
A6B 062AE 00000000 (01472)          NOP
A6C 062B0 20322E32 (01473)          CLEAR(RA)
A6D 062B2 00000000 (01474)          NOP
A6E 062B4 10000000 (01475)          JUMP(0)
A6F 062B6 00000000 (01476)          NOP
A70 062B8 46E00000 (01477) DOQ:     ADD(A7,A6)\NOP  ;N+1
A71 062BA 08F00810 (01478)          MOV(IQA,A0)\MOV(ZERO,A0)        ;K
A72 062BC 4EF74900 (01479)          MOV(A7),SUB(A7,A6)\SUB(A0,A1)   ;N ;-2**-15
A73 062BE 08970890 (01480) LPQ1:    MOV(R,A7)\MOV(R,A0)
A74 062C0 911E0081 (01481)          JUMPS(T1,MWDONE)          ;JUMP IFNO MORE THRESHOLDS
A75 062C2 08F10000 (01482)          MOV(IQA,A1)\NOP           ;THRESH (N)
A76 062C4 49004100 (01483)          SUB(A0,A1)\ADD(A0,A1)   ;K-THRESH;LAST CODE + 2**-15
A77 062C6 03F20000 (01484)          MOV(IQA,A2)\NOP           ;VALUE (N)
A78 062C8 4EE00890 (01485)          SUB(A7,A6)\MOV(R,A0)    ;M-(N+2);NEW CODE
A79 062CA 901E0073 (01486)          JUMPC(T1,LPQ1)  ;JUMP IF K BIGGER
                   (01487) *
A7A 062CC 08970800 (01488)          MOV(R,A7)\NOP   M-(N+2)
A7B 062CE 911E0081 (01489)          JUMPS(T1,MWDONE)          ;JUMP IF NO MORE THRESHOLDS
A7C 062D0 4EE00000 (01490) MWDONE:          SUB(A7,A6)\NOP
A7D 062D2 08E00000 (01491)          MOV(IQA,NULL)\NOP         ;THRESH(N)
A7E 062D4 08E00300 (01492)          MOV(IQA,NULL)\NOP         ;VALUE (N)
A7F 062D6 08970E00 (01493)          MOV(R,A7)\NOP
A80 062D8 901E007C (01494)          JUMPC(T1,MWDONE)          ;JUMP IF MORE THRESHOLDS
                   (01495) *
A81 062DA 3A4E3A00 (01496) MWDONE:  ALIGN(A2)\ALIGN(A0)       ;VALUE;CODE
A82 062DC 000C089C (01497)          NOP\MOV(R,0Q)   ;CODE
A83 062DE 089C000F (01498)          MOV(R,0Q)\NOP
A84 062E0 A00FA000 (01499)          RETURN
A85 062E2 00000000 (01500)          NOP
          00000086 (01501) MWLF$SZ=$A-MWLF$SA
                   (01502) *
    062E4          (01503)          END
```

```
                    (01504) *       MWLF-APS PROGRAM
                    (01505) *
                    (01506) *
                    (01507)           EVEN              START ON WORD BOUNDRY
    062E4 000063D5  (01508)           ADDR MWLF$I       POINTER TO CONSTRUCTED INSTRUCTION BLOCK
    062E6 000062F0  (01509)           ADDR MWLF$APS+2* MWLF$S
    062E8 0001      (01510)           DATA 1
    062E9 0100      (01511)           DATA MWLF$Z       MODULE SIZE
    062EA 000063CA  (01512)           ADDR MWLF$A       SUBSTITUTABLE ARGUMENTS CHAIN ANCHOR
                    (01513)           EVEN              END ON WORD BOUNDRY
                    (01514) *
                    (01515) *
                    (01516) *
                    (01517) *
                    (01518) *
                    (01519) * WEINER - APS INITIALIZATION ROUTINES
                    (01520) *
                    (01521) MWLF$APS  BEGIN APS(MWLF)
                    (01522) *
  A00 062EC 00000260 (01523)          JUMP(MWLF$S)
  A01 062EE 02004E60 (01524)          JUMP(KQ$I)
  A02 062F0 04E20000 (01525) MWLF$S   LOAD(BR2,M$$(1),L,TF)          INPUT STABILITY TEST
  A03 062F2 06300032 (01526)          SET(RA)
                    (01527) *
  A04 062F4 00C02000 (01528)          LOAD(BR0,[2],L,TF)             INPT R(0)
  A05 062F6 0A500000 (01529)          LOAD(BR1,M$$)                  NOT USED
  A06 062F8 0C0A0000 (01530)          ADD(BR0,M$$)                   BR0=R(1)
  A07 062FA 0E500000 (01531)          LOAD(BR1,0)
  A08 062FC 10010013 (01532)          MOVB(BW0,BR1)                  BW0=0=2(N-1)
  A09 062FE 12700100 (01533)          LOAD(BR3,[1],L)                BR3=A(1)
  A0A 06300 14500000 (01534)          LOAD(BR1,M$$)                  BR1=ASIZE-1, NOT USED
  A0B 06302 16320000 (01535)          SUB(BR3,M$$)                   BR3=A(0)
                    (01536) *
                    (01537) *
                    (01538) * MWLF - APS S(N) INPUT
                    (01539) * REGISTER CONTENTS AT START
                    (01540) * BR0=R(1), BW0=2(N-1), BR3=A(0)=AB-2
                    (01541) *
                    (01542) *
  A0C 06304 18890028 (01543) $1       ADDB(BR0,BW2,TF)               INPT R(N)
  A0D 06306 1A19001C (01544)          MOVB(BR1,BW0)                  BR1=2(N-1)
  A0E 06308 1C290017 (01545)          MOVB(BR2,BR3)
  A0F 0630A 1E1915F2 (01546)          SUBL(BR1,2),JUMPN(MWLSC)
  A10 0630C 20001260 (01547)          JUMP(MWLSE)
                    (01548) *
  A11 0630E 22020002 (01549) $2       SUB(BR0,2,TF)                  INPT R(N-J)
                    (01550) *
  A12 06310 24AA0002 (01551) MWLSE    ADD(BR2,2,TF)                 INPT A(J)
  A13 06312 26191182 (01552)          SUBL(BR1,2), JUMPP($2)
                    (01553) *
  A14 06314 28091572 (01554)          SUBL(BR0,2,TF),JUMP(MWLSC)    INPT R(1)
                    (01555) *
  A15 06316 2A300037 (01556) MWLSC    SET(WI)
```

```
A16 06318 2C000020 (01557)          NOP(0)                    ;!!!ADDED BY KFIELD 11/79!!!
A17 0631A 2E01003A (01558)          ADDL(BW0,2)                   BW0=2N
                   (01559) *
                   (01560) *
                   (01561) *
                   (01562) * MWLF = APS  A(N,J)  CALCULATION INPT
                   (01563) * REGISTER CONTENTS AT START
                   (01564) * BR0=R(1), BW0=2N
                   (01565) * BR2=A(N-1), BR3=A(0)=AB-2
                   (01566) *
                   (01567) *
A18 0631C 302A0002 (01568)          ADD(BR2,2)                    BR2=A(N)
A19 0631E 32210F15 (01569)          MOVB(BW2,BR2)
A1A 06320 34202A4F (01570)          JSN(MWLOT,P2)                 SET UP OUTPUT
                   (01571) *
A1B 06322 36191F50 (01572)          MOVB(BR1,BW0), JUMP(MWLAE)    BR1=2N
                   (01573) *
A1C 06324 39AA0000 (01574) #3       ADD(BR2,0,TF)                 A(N-1,N-J+1)INPT
A1D 06326 3AA20002 (01575)          SUB(BR2,2,TF)                 A(N-1,N-J)INPT
A1E 06328 3CBA0002 (01576)          ADD(BR3,2,TF)                 A(N-1,J)INPT
                   (01577) *
A1F 0632A 3E191CB4 (01578) MWLAE    SUBL(BR1,4), JUMPP(#3)
                   (01579) *
A20 0632C 4070102E (01580) MWLT     LOAD(BR3,[1],L)               BR3=A(1)=AB
A21 0632E 42500000 (01581)          LOAD(BR1,M$$)                 ASIZE=N,DONE
A22 06330 44320000 (01582)          SUB(BR3,M$$)                  BR3=A(0)=AB-2
A23 06332 46190028 (01583)          ADDB(BR1,BR1)                 BR1=2N-2, MWDONE
A24 06334 48A92578 (01584)          ADDL(BR2,0,TF), JUMP(#A+1)                CLEANUP OF A(N) CALC*
A25 06336 4A300037 (01585)          SET(WI)
A26 06338 4C000020 (01586)          NOP(0)
A27 0633A 4E190CA0 (01587)          SUBB(BR1,BW0), JUMPP(#1)
A28 0633C 502026F1 (01588)          JUMPS(#A-2,RI),CLEAR
A29 0633E 52000020 (01589)          NOP(0)                    ;!!!ADDED BY KFIELD 11/79!!!
                   (01590) *
                   (01591) *
                   (01592) * MWLF - APS OUTPUT
                   (01593) *
                   (01594) * REGISTER CONTENTS AT START
                   (01595) * BW0=2N, BW2=A(N)
                   (01596) *
                   (01597) *
A2A 06340 54300030 (01598) MWLOT    SET(RO)
A2B 06342 56B10C14 (01599)          MOVB(BW3,BW2,TF)              BW3=A(N), OUTPUT
A2C 06344 58210020 (01600)          SUBB(BW2,BW0)                 BW2=A(0)
A2D 06346 5AD20794 (01601)          LOAD(BW1,DMY$(1),TF)        OTPT DUMP
A2E 06348 5C910038 (01602)          ADDL(BW1,0,TF)                OTPT DUMP
A2F 0634A 5E113250 (01603)          MOVB(BW1,BW0), JUMP(MWLOE)    BW1=2N
                   (01604) *
A30 0634C 60AA0002 (01605) #4       ADD(BW2,2,TF)                 OTPT A(N,J)
A31 0634E 62B20002 (01606)          SUB(BW3,2,TF)                 OTPT A(N,N-J)
                   (01607) *
A32 06350 64113084 (01608) MWLOE    SUBL(BW1,4),JUMPP(#4)
A33 06352 666F0026 (01609)          LOAD(BW2,[0],L)               BW2=P(1)=PB
```

```
A34 06354 685000000 (01610)          LOAD(BW1,M$$)
A35 06356 6A220000 (01611)           SUB(BW2,M$$)                    BW2=P(0)=PB-2
A36 06358 6C1A0001 (01612)           ADD(BW1,1)                      BW1=PSIZE=2ASIZE
A37 0635A 6EA10020 (01613)           ADDB(BW2,BW0,TF)                OTPT P(N)
A38 0635C 70A1002A (01614)           ADDB(BW2,BW1,TF)                OTPT E(N)
                   (01615) *
A39 0635E 72000020 (01616) $5        NOP(0)
A3A 06360 748039A8 (01617)           JUMPC(#5,AF0)                   WAIT ON APU
                   (01618) *
A3B 06362 76203C68 (01619)           JUMP(#A+1,AF0),CLEAR
A3C 06364 78004CA9 (01620)           JUMPC(MWLOD,AF1)                JUMP TO MWDONE IF NO ABORT
                   (01621) *
A3D 06366 7A200031 (01622)           CLEAR(RI)
A3E 06368 7C000C20 (01623)           NOP(0)                  ;!!!ADDED BY KFIELD 11/79!!!
A3F 0636A 7E390010 (01624)           MOVB(BR3,BW0)                   BR3 = 2N
A40 0636C 80290010 (01625)           MOVB(BR2,BW0)                   BR2 = 2N
A41 0636E 82390022 (01626)           SUBB(BR3,BW1)                   BR3=2N-ASIZE
A42 06370 8439C031 (01627)           SUBL(BR3,1)                     BR3=2N-ASIZE-1
                   (01628) *
A43 06372 86401020 (01629)           LOAD(BW0,C1J)                   BW0 = A(1)
A44 06374 88700000 (0163C)           LOAD(BW3,M$$)                   NOT USED
A45 06376 8A020000 (01631)           SUB(BW0,M$$)                    BW0 = A(0)
A46 06378 8C014B6D (01632)           ADDB(BW0,BR2),JUMP(MWLOC)       BW0 = A(N)
                   (01633) *
A47 0637A 8E0A0002 (01634) $6        ADD(BW0,2,TF)                   OTPT A(N+J)=0
A48 0637C 90210022 (01635)           SUBB(BW2,BW1)                   BW2=P(N+J-1)
A49 0637E 92AA0002 (01636)           ADD(BW2,2,TF)                   OTPT P(N+J)=0
A4A 06380 94A1002A (01637)           ADDB(BW2,BW1,TF)                OTPT E(N+J)=E(N)
A4B 06382 963947FA (01638) MWLOC     ADDL(BR3,2), JUMPN(#6)
                   (01639) *
A4C 06384 98200030 (01640) MWLOD     CLEAR(RO)
A4D 06386 9A00002C (01641)           NOP(0)
A4E 06388 9C206E40 (01642) KQSI:     JSN(KQSO,P2)
A4F 0638A 9E300030 (01643)           SET(RO)
                   (01644) *
A50 0638C A0C203CE (01645)           LOAD(BR0,SVTS+2*30(1),TF)       ;2**-15
A51 0638E A2400010 (01646)           LOAD(BR0,C01,)  ;UNQUANT K'S
A52 06390 A4700000 (01647)           LOAD(BR3,M$$)   ;UNUSED
A53 06392 A6700000 (01648)           LOAD(BR3,M$$)   ;UNUSED
A54 06394 A8090032 (01649)           SUBL(BR0,2)     ;PREPARE TO INCREMENT LATER
A55 06396 AA5263EC (01650)           LOAD(BR1,KQTAB(1))      ;THRESH & QANT TABLE
A56 06398 AC190032 (01651)           SUBL(BR1,2)     ;PREPARE TO INCR LATER
A57 0639A AE600302 (01652)           LOAD(BR2,2)     ;COUNT NUMBER OF LENGTH 32 TABLES (3)
A58 0639C B070001F (01653) K123:     LOAD(BR3,31)    ;TABLE LENGTH
A59 0639E B2090030 (01654)           ADDL(BR0,2,TF)  ;K
A5A 063A0 B4990030 (01655) K1LP:     ADDL(BR1,2,TF)  ;THRESH(N)
A5B 063A2 B6990030 (01656)           ADDL(BR1,2,TF)  ;VALUE(N)
A5C 063A4 B8395AB1 (01657)           SUBL(BR3,1),JUMPP(K1LP)         ;DONE WITH THIS TABLE?
A5D 063A6 BA2958B1 (01658)           SUBL(BR2,1),JUMPP(K123)         ;JUMP IF NOT DONE K3
                   (01659) *
A5E 063A8 BC600002 (01660)           LOAD(BR2,2)     ;NUMBER OF LENGTH 16 TABLES (-1)
A5F 063AA BE70000F (01661) K456:     LOAD(BR3,15)    TABLE LENGTH
A60 063AC C0090030 (01662)           ADDL(BR0,2,TF)  ;K4,K5 OR K6
```

```
A61 063AE C2990C3A (01663) K4LP:   ADDL(BR1,2,TF)  ;THRESH
A62 063B0 C4990C3A (01664)         ADDL(BR1,2,TF)  ;VALUE
A63 063B2 C63961B1 (01665)         SUBL(BR3,1),JUMPP(K4LP)        ;DONE WITH THIS TABLE?
A64 063B4 C8295FB1 (01666)         SUBL(BR2,1),JUMPP(K456)        ;YES, DONE WITH K6?
                   (01667) *
A65 063B6 CA600001 (01668)         LOAD(BR2,1)     ;YES, NUMBER OF LENGTH 8 TABLES -1
                   (01669)
A66 063B8 CC700007 (01670) K78:    LOAD(BR3,7)     ;TABLE LENGTH
A67 063BA CE89003A (01671)         ADDL(BR0,2,TF)  ;K7 OR K8
A68 063BC D099003A (01672) K7LP:   ADDL(BR1,2,TF)  ;THRESH
A69 063BE D299003A (01673)         ADDL(BR1,2,TF)  ;VALUE
A6A 063C0 D43968B1 (01674)         SUBL(BR3,1),JUMPP(K7LP)
A6B 063C2 D62966B1 (01675)         SUBL(BR2,1),JUMPP(K78)
A6C 063C4 D8200031 (01676)         CLEAR(RI)
A6D 063C6 DA000020 (01677)         NOP(0)
                   (01678) *
                   (01679) *OUTPUT PROGRAM
A6E 063C8 DC300032 (01680) KQ$O:   SET(RA)
A6F 063CA DE40103C (01681)         LOAD(BW0,[1])   ;OUTPUT ARRAY (U)
A70 063CC E0500000 (01682)         LOAD(BW1,M$$)   UNUSED
A71 063CE E2500000 (01683)         LOAD(BW1,M$$)   ;UNUSED
A72 063D0 E4500007 (01684)         LOAD(BW1,7)     ;NUMBER OF DOUBLE OUTPUTS -1
A73 063D2 E6010031 (01685)         SUBL(BW0,1)     ;PREPARE TO INCREMENT
A74 063D4 E9210039 (01686) KQOLP:  ADDL(BW0,1,TE)  ;KCODE(N)
A75 063D6 EB010039 (01687)         ADDL(BW0,1,TE)  ;KQUANT(N)
A76 063D8 EC1174B1 (01688)         SUBL(BW1,1),JUMPP(KQOLP)       ;DONE?
A77 063DA EE200030 (01689)         CLEAR(RD)       ;YES
A78 063DC F0000020 (01690)         NOP(0)
                   (01691) *
                   (01692) *
          000063CA (01693) MWLF$A=#C
                   (01694) *
     063DE         (01695)         END
                   (01696) *
                   (01697) *          STORAGE FOR CONSTRUCTED INSTRUCTIONS
                   (01698) *
     063DE 00000000 (01699) MWLF$I    DATA 7F'0.0'
      ...
                   (01700) *
          00000100 (01701) MWLF$Z=#L-MWLF$APS
                   (01702) *
     063EC FA06EE40 (01703) KQTAB:       DATA   -0.9533365E+00
     063EE FA708C40 (01704)         DATA   -0.9565597E+00
     063F0 F91C45C0 (01705)         DATA   -0.9461753E+00
     063F2 F995AE40 (01706)         DATA   -0.9498804E+00
     063F4 F80EBDC0 (01707)         DATA   -0.9379499E+00
     063F6 F89A2740 (01708)         DATA   -0.9422044E+00
     063F8 F6D98940 (01709)         DATA   -0.9285137E+00
     063FA F7796B40 (01710)         DATA   -0.9333929E+00
     063FC F5774EC0 (01711)         DATA   -0.9177035E+00
     063FE F62E67C0 (01712)         DATA   -0.9232912E+00
     06400 F3E22340 (01713)         DATA   -0.9053387E+00
     06402 F4B37BC0 (01714)         DATA   -0.9117274E+00
```

```
06404 F2138CC0 (01715)          DATA   -0.8912216E+00
06406 F3027340 (01716)          DATA   -0.8985123E+00
06408 F0047EC0 (01717)          DATA   -0.8751372E+00
0640A F1148AC0 (01718)          DATA   -0.8834394E+00
0640C EDAD67C0 (01719)          DATA   -0.8568544E+00
0640E EEE27240 (01720)          DATA   -0.8662856E+00
06410 EB0642C0 (01721)          DATA   -0.8361286E+00
06412 EC6458C0 (01722)          DATA   -0.8468127E+00
06414 E8B6B440 (01723)          DATA   -0.8127046E+00
06416 E99210C0 (01724)          DATA   -0.8247701E+00
06418 E4A632C0 (01725)          DATA   -0.7863220E+00
0641A E6631B40 (01726)          DATA   -0.7998995E+00
0641C E0DC4340 (01727)          DATA   -0.7567219E+00
0641E E2CEEE40 (01728)          DATA   -0.7719400E+00
06420 DCA0BFC0 (01729)          DATA   -0.7236557E+00
06422 DECD3040 (01730)          DATA   -0.7406369E+00
06424 D7EC36C0 (01731)          DATA   -0.6868962E+00
06426 DA560940 (01732)          DATA   -0.7057506E+00
06428 D2B85540 (01733)          DATA   -0.6462504E+00
0642A D56287C0 (01734)          DATA   -0.6570694E+00
0642C CD0064C0 (01735)          DATA   -0.6015745E+00
0642E CFED16C0 (01736)          DATA   -0.6244229E+00
06430 C6C1CA40 (01737)          DATA   -0.5527890E+00
06432 C9F1FCC0 (01738)          DATA   -0.5776974E+00
06434 BFFC91C0 (01739)          DATA   -0.4998953E+00
06436 C36FE2C0 (01740)          DATA   -0.5260520E+00
06438 B8B3E2C0 (01741)          DATA   -0.4429897E+00
0643A BC685440 (01742)          DATA   -0.4719339E+00
0643C B0EE68C0 (01743)          DATA   -0.3822757E+00
0643E B4E02F40 (01744)          DATA   -0.4130916E+00
06440 A8B69040 (01745)          DATA   -0.3180714E+00
06442 ACDFFB40 (01746)          DATA   -0.3505854E+00
06444 A01A90C0 (01747)          DATA   -0.2508107E+00
06446 A4740C40 (01748)          DATA   -0.2847915E+00
06448 972C3FC0 (01749)          DATA   -0.1810379E+00
0644A 9BAC73C0 (01750)          DATA   -0.2162003E+00
0644C 8E009DC0 (01751)          DATA   -0.1093938E+00
0644E 929CAD40 (01752)          DATA   -0.1454064E+00
06450 CAF2313F (01753)          DATA   -0.3559476E-01
06452 895B1140 (01754)          DATA   -0.7309163E-01
06454 4AF22EBF (01755)          DATA    0.3659474E-01
06456 FFFFFF39 (01756)          DATA   -0.3725290E-08
06458 0E009DC0 (01757)          DATA    0.1093938E+00
0645A 095B10C0 (01758)          DATA    0.7309162E-01
0645C 172C3FC0 (01759)          DATA    0.1810379E+00
0645E 129CAD40 (01760)          DATA    0.1454064E+00
06460 201A90C0 (01761)          DATA    0.2508107E+00
06462 1BAC73C0 (01762)          DATA    0.2162003E+00
06464 28B69040 (01763)          DATA    0.3180714E+00
06466 24740C40 (01764)          DATA    0.2847915E+00
06468 30EE68C0 (01765)          DATA    0.3822757E+00
0646A 2CDFFB40 (01766)          DATA    0.3505854E+00
0646C B19F26C0 (01767)          DATA   -0.3876694E+00
```

```
06468 B59B6040 (01768)        DATA    -0.4188042E+00
0646A A94EE640 (01769)        DATA    -0.3227203E+00
0646C AD85174C (01770)        DATA    -0.3556241E+00
0646E A0962C40 (01771)        DATA    -0.2545829E+00
06470 A4FE874C (01772)        DATA    -0.2890176E+00
06472 97876C40 (01773)        DATA    -0.1838203E+00
06474 9C184240 (01774)        DATA    -0.2194903E+00
06476 8E3883C0 (01775)        DATA    -0.1111997E+00
06478 92E68240 (01776)        DATA    -0.1476596E+00
0647A CC1F973F (01777)        DATA    -0.3716963E-01
0647C 89B09640 (01778)        DATA    -0.7423668E-01
0647E 4C1F933F (01779)        DATA     0.3716960E-01
06480 97FFFFBA (01780)        DATA    -0.1117587E-07
06482 8E3883C0 (01781)        DATA     0.1111997E+00
06484 89B09640 (01782)        DATA     0.7423665E-01
06486 17876C40 (01783)        DATA     0.1838203E+00
06488 12E68240 (01784)        DATA     0.1476596E+00
0648A 20962C40 (01785)        DATA     0.2545829E+00
0648C 1C184140 (01786)        DATA     0.2194902E+00
0648E 294EE640 (01787)        DATA     0.3227203E+00
06490 24FE8740 (01788)        DATA     0.2890176E+00
06492 319F25C0 (01789)        DATA     0.3876693E+00
06494 2D85174C (01790)        DATA     0.3556241E+00
06496 397852C0 (01791)        DATA     0.4489845E+00
06498 359B6040 (01792)        DATA     0.4188042E+00
0649A 40CFE2C0 (01793)        DATA     0.5063442E+00
0649C 3D34CF40 (01794)        DATA     0.4781741E+00
0649E 479F43C0 (01795)        DATA     0.5595479E+00
064A0 4448DC40 (01796)        DATA     0.5334735E+00
064A2 4DE39440 (01797)        DATA     0.6085077E+00
064A4 4AD2DA40 (01798)        DATA     0.5845597E+00
064A6 539D37C0 (01799)        DATA     0.6532354E+00
064A8 50D198C0 (01800)        DATA     0.6313964E+00
064AA 58CF47C0 (01801)        DATA     0.6938257E+00
064AC 5646E8C0 (01802)        DATA     0.6748390E+00
064AE 5D7F09C0 (01803)        DATA     0.7304394E+00
064B0 5B3708C0 (01804)        DATA     0.7126174E+00
064B2 61B3604C (01805)        DATA     0.7632866E+00
064B4 5FA827C0 (01806)        DATA     0.7473192E+00
064B6 6574454C (01807)        DATA     0.7926148E+00
064B8 63A1B940 (01808)        DATA     0.7783729E+00
064BA 68CA604C (01809)        DATA     0.8186760E+00
064BC 672C1E40 (01810)        DATA     0.8060339E+00
064BE 6BBE98C0 (01811)        DATA     0.8417544E+00
064C0 6A5029C0 (01812)        DATA     0.8305714E+00
064C2 6E59E2C0 (01813)        DATA     0.8621181E+00
064C4 6D16D0C0 (01814)        DATA     0.8522588E+00
064C6 70A4E340 (01815)        DATA     0.8900320E+00
064C8 6F88E4C0 (01816)        DATA     0.8713652E+00
064CA 72A7E640 (01817)        DATA     0.8957489E+00
064CC 71AEE54C (01818)        DATA     0.8881499E+00
064CE 746AAFC0 (01819)        DATA     0.9095858E+00
064D0 7390DAC0 (01820)        DATA     0.9028581E+00
```

```
064D8 75F47240 (01821)        DATA     0.9215224E+00
064DA 75364440 (01822)        DATA     0.9157106E+00
064DC 774BC940 (01823)        DATA     0.9320003E+00
064DE 76A60840 (01824)        DATA     0.9269419E+00
064E0 7876AF40 (01825)        DATA     0.9411220E+00
064E2 77E67040 (01826)        DATA     0.9367199E+00
064E4 797A8C40 (01827)        DATA     0.9490524E+00
064E6 78FD2EC0 (01828)        DATA     0.9452265E+00
064E8 7A5C34C0 (01829)        DATA     0.9559389E+00
064EA 79EF5F40 (01830)        DATA     0.9526176E+00
064EC F2E19640 (01831)        DATA    -0.8975094E+00
064EE F3C82EC0 (01832)        DATA    -0.9045466E+00
064F0 F0E36240 (01833)        DATA    -0.8819392E+00
064F2 F1EAFAC0 (01834)        DATA    -0.8899835E+00
064F4 EE9D3840 (01835)        DATA    -0.8641730E+00
064F6 EFC9CFC0 (01836)        DATA    -0.8733463E+00
064F8 EC06B6C0 (01837)        DATA    -0.8439549E+00
064FA ED5C8CC0 (01838)        DATA    -0.8543869E+00
064FC E9170EC0 (01839)        DATA    -0.8210162E+00
064FE EA9A97C0 (01840)        DATA    -0.8328420E+00
06500 E5C53040 (01841)        DATA    -0.7950002E+00
06502 E77AF8C0 (01842)        DATA    -0.8004403E+00
06504 E2000040 (01843)        DATA    -0.7658692E+00
06506 E3F492C0 (01844)        DATA    -0.7809013E+00
06508 DDD6AB40 (01845)        DATA    -0.7331133E+00
0650A DFFE61C0 (01846)        DATA    -0.7499506E+00
0650C D928F240 (01847)        DATA    -0.6965621E+00
0650E DB0FD240 (01848)        DATA    -0.7153266E+00
06510 D3F7C4C0 (01849)        DATA    -0.6559088E+00
06512 D6A12DC0 (01850)        DATA    -0.6767938E+00
06514 CE3D9DC0 (01851)        DATA    -0.6112554E+00
06516 D12C1440 (01852)        DATA    -0.6341577E+00
06518 C7F72DC0 (01853)        DATA    -0.5622300E+00
0651A CB2C0A40 (01854)        DATA    -0.5872015E+00
0651C C123E640 (01855)        DATA    -0.5089081E+00
0651E C49F0E40 (01856)        DATA    -0.5361040E+00
06520 B9C608C0 (01857)        DATA    -0.4513713E+00
06522 BD862A40 (01858)        DATA    -0.4806569E+00
06524 B1E59AC0 (01859)        DATA    -0.3898195E+00
06526 B5E5F3C0 (01860)        DATA    -0.4210001E+00
06528 A9BBB140 (01861)        DATA    -0.3245756E+00
0652A ADC6F2C0 (01862)        DATA    -0.3576339E+00
0652C A0C70C40 (01863)        DATA    -0.2560897E+00
0652E A535D240 (01864)        DATA    -0.2907050E+00
06530 97A80E40 (01865)        DATA    -0.1849325E+00
06532 9C4355C0 (01866)        DATA    -0.2200049E+00
06534 8E4EDF40 (01867)        DATA    -0.1117020E+00
06536 930407C0 (01868)        DATA    -0.1485605E+00
06538 CC982BBF (01869)        DATA    -0.3739962E-01
0653A 898F9840 (01870)        DATA    -0.7469469E-01
0653C 4C981A3F (01871)        DATA     0.3739949E-01
0653E 80000038 (01872)        DATA    -0.5960465E-07
06540 8E4EDEC0 (01873)        DATA     0.1117019E+00
```

```
06542 098F9740 (01874)          DATA    0.7469457E-01
06544 17ABDD40 (01875)          DATA    0.1849324E+00
06546 13040640 (01876)          DATA    0.1485603E+00
06548 20C78A40 (01877)          DATA    0.2560895E+00
0654A 1C4354C0 (01878)          DATA    0.2208048E+00
0654C 298BB0C0 (01879)          DATA    0.3245755E+00
0654E 2535D0C0 (01880)          DATA    0.2907048E+00
06550 31E59A40 (01881)          DATA    0.3898194E+00
06552 2DC6F1C0 (01882)          DATA    0.3576338E+00
06554 39C68840 (01883)          DATA    0.4513712E+00
06556 35E5F340 (01884)          DATA    0.4210800E+00
06558 4123E5C0 (01885)          DATA    0.5089080E+00
0655A 3D862940 (01886)          DATA    0.4806560E+00
0655C 47F72DC0 (01887)          DATA    0.5622308E+00
0655E 449F0E40 (01888)          DATA    0.5361040E+00
06560 4E3D9D40 (01889)          DATA    0.6112553E+00
06562 4B2C0940 (01890)          DATA    0.5872814E+00
06564 53F7C3C0 (01891)          DATA    0.6559987E+00
06566 512C1340 (01892)          DATA    0.6341576E+00
06568 5928F240 (01893)          DATA    0.6965621E+00
0656A 56A12DC0 (01894)          DATA    0.6767938E+00
0656C 9D3E15C0 (01895)          DATA   -0.2284572E+00
0656E A2D27140 (01896)          DATA   -0.2720472E+00
06570 918EE3C0 (01897)          DATA   -0.1386380E+00
06572 97BB1140 (01898)          DATA   -0.1839315E+00
06574 DF303ABF (01899)          DATA   -0.4547871E-01
06576 8BDF7640 (01900)          DATA   -0.9275703E-01
06578 5F30393F (01901)          DATA    0.4647870E-01
0657A FFFFFF39 (01902)          DATA   -0.3725290E-09
0657C 11BEE3C0 (01903)          DATA    0.1386380E+00
0657E 0BDF7640 (01904)          DATA    0.9275702E-01
06580 1D3E15C0 (01905)          DATA    0.2284572E+00
06582 17BB1140 (01906)          DATA    0.1839315E+00
06584 284320C0 (01907)          DATA    0.3145486E+00
06586 22D27140 (01908)          DATA    0.2720472E+00
06588 32AB3940 (01909)          DATA    0.3957588E+00
0658A 2D8BAE40 (01910)          DATA    0.3558252E+00
0658C 3C50DCC0 (01911)          DATA    0.4712177E+00
0658E 37957E40 (01912)          DATA    0.4342497E+00
06590 452A65C0 (01913)          DATA    0.5403564E+00
06592 40D84D40 (01914)          DATA    0.5066010E+00
06594 4D2BAB40 (01915)          DATA    0.6028952E+00
06596 49464C40 (01916)          DATA    0.5724578E+00
06598 5453E140 (01917)          DATA    0.6588098E+00
0659A 50DAA8C0 (01918)          DATA    0.6316733E+00
0659C 5AA90DC0 (01919)          DATA    0.7082841E+00
0659E 57983F40 (01920)          DATA    0.6843337E+00
065A0 603657C0 (01921)          DATA    0.7516584E+00
065A2 5D87D540 (01922)          DATA    0.7307078E+00
065A4 650A5C40 (01923)          DATA    0.7893787E+00
065A6 62B68140 (01924)          DATA    0.7711946E+00
065A8 6935B940 (01925)          DATA    0.8219520E+00
065AA 6734DAC0 (01926)          DATA    0.8062757E+00
```

```
065AC C62948C0 (01927)        DATA    -0.5481349E+00
065AE C9C0E940 (01928)        DATA    -0.5761997E+00
065B0 BE7B95C0 (01929)        DATA    -0.4881465E+00
065B2 C26759C0 (01930)        DATA    -0.5187790E+00
065B4 B62A27C0 (01931)        DATA    -0.4231615E+00
065B6 BA66CCC0 (01932)        DATA    -0.4562622E+00
065B8 AD3FD340 (01933)        DATA    -0.3535103E+00
065BA B1C73240 (01934)        DATA    -0.3888915E+00
065BC A3CD64C0 (01935)        DATA    -0.2797856E+00
065BE A8965640 (01936)        DATA    -0.3170879E+00
065C0 99E9BCC0 (01937)        DATA    -0.2024399E+00
065C2 9EE802C0 (01938)        DATA    -0.2414554E+00
065C4 8FB80540 (01939)        DATA    -0.1225609E+00
065C6 94D5AEC0 (01940)        DATA    -0.1627711E+00
065C8 D420B63F (01941)        DATA    -0.4107802E-01
065CA 8A7DAE40 (01942)        DATA    -0.8196046E-01
065CC 53E4723F (01943)        DATA     0.4096307E-01
065CE 9E2E710D (01944)        DATA    -0.5756650E-04
065D0 0FAC9F40 (01945)        DATA     0.1224555E+00
065D2 0A79EEC0 (01946)        DATA     0.8194609E-01
065D4 19E5EEC0 (01947)        DATA     0.2023295E+00
065D6 14D202C0 (01948)        DATA     0.1626590E+00
065D8 23C9E9C0 (01949)        DATA     0.2795994E+00
065DA 1EE47540 (01950)        DATA     0.2413470E+00
065DC 2D3C86C0 (01951)        DATA     0.3534096E+00
065DE 2892F1C0 (01952)        DATA     0.3169844E+00
065E0 36270F40 (01953)        DATA     0.4230670E+00
065E2 31C3FEC0 (01954)        DATA     0.3887938E+00
065E4 3E78B5C0 (01955)        DATA     0.4880588E+00
065E6 3A63CFC0 (01956)        DATA     0.4561710E+00
065E8 4626A4C0 (01957)        DATA     0.5480543E+00
065EA 42649740 (01958)        DATA     0.5186948E+00
065EC 9C316340 (01959)        DATA    -0.2202572E+00
065EE A1972940 (01960)        DATA    -0.2624256E+00
065F0 91184540 (01961)        DATA    -0.1335532E+00
065F2 96B01940 (01962)        DATA    -0.1772491E+00
065F4 DBA8B43F (01963)        DATA    -0.4475537E-01
065F6 8B6F3940 (01964)        DATA    -0.8933180E-01
065F8 5BA8B13F (01965)        DATA     0.4475535E-01
065FA 90000003A (01966)       DATA    -0.7450581E-08
065FC 11184540 (01967)        DATA     0.1335532E+00
065FE 8B6F3940 (01968)        DATA     0.8933177E-01
06600 1C316340 (01969)        DATA     0.2202572E+00
06602 16801940 (01970)        DATA     0.1772491E+00
06604 26DCDC40 (01971)        DATA     0.3936151E+00
06606 21972940 (01972)        DATA     0.2624256E+00
06608 30F812C0 (01973)        DATA     0.3825706E+00
0660A 2BFE5D40 (01974)        DATA     0.3437001E+00
0660C 3A6841A0 (01975)        DATA     0.4563077E+00
0660E 35C6E5C0 (01976)        DATA     0.4201324E+00
06610 43183D40 (01977)        DATA     0.5242688E+00
06612 3EDA3CC0 (01978)        DATA     0.4910351E+00
06614 4B06EE40 (01979)        DATA     0.5861490E+00
```

```
06616 472A5040 (01980)          DATA     0.5559708E+00
06618 522806C0 (01981)          DATA     0.6418713E+00
0661A 4EB10FC0 (01982)          DATA     0.6147767E+00
0661C 5884B4C0 (01983)          DATA     0.6915499E+00
0661E 556F1640 (01984)          DATA     0.6674526E+00
06620 5E232640 (01985)          DATA     0.7354477E+00
06622 5B6AEB40 (01986)          DATA     0.7142004E+00
06624 63103C40 (01987)          DATA     0.7739330E+00
06626 60AF01C0 (01988)          DATA     0.7553408E+00
06628 675A3940 (01989)          DATA     0.8074409E+00
0662A 6548AB40 (01990)          DATA     0.7912802E+00
0662C B9663FC0 (01991)          DATA    -0.4484329E+00
0662E C03ABF40 (01992)          DATA    -0.5017928E+00
06630 AA7754C0 (01993)          DATA    -0.3317667E+00
06632 B2225640 (01994)          DATA    -0.3916729E+00
06634 9A1C2FC0 (01995)          DATA    -0.2039852E+00
06636 A270BBC0 (01996)          DATA    -0.2690596E+00
06638 80D02FC0 (01997)          DATA    -0.6885332E-01
0663A 918B14C0 (01998)          DATA    -0.1370569E+00
0663C 80D02F40 (01999)          DATA     0.6885329E-01
0663E 9FFFFFBA (02000)          DATA    -0.1490116E-07
06640 1A1C2FC0 (02001)          DATA     0.2039852E+00
06642 118B13C0 (02002)          DATA     0.1370568E+00
06644 2A7753C0 (02003)          DATA     0.3317666E+00
06646 2270BAC0 (02004)          DATA     0.2690595E+00
06648 39663FC0 (02005)          DATA     0.4484329E+00
0664A 32225640 (02006)          DATA     0.3916729E+00
0664C 96635440 (02007)          DATA    -0.1749063E+00
0664E 9D9CC640 (02008)          DATA    -0.2313469E+00
06650 F895C80F (02009)          DATA    -0.5987801E-01
06652 8F036FC0 (0201F)          DATA    -0.1172924E+00
06654 7858DF3F (02011)          DATA     0.5876326E-01
06656 9E2EF1BD (02012)          DATA    -0.5757022E-04
06658 165FAC40 (02013)          DATA     0.1747947E+00
0665A 0EFFB7C0 (02014)          DATA     0.1171789E+00
0665C 24A11840 (02015)          DATA     0.2861662E+00
0665E 1D993440 (02016)          DATA     0.2312379E+00
06660 31F598C0 (02017)          DATA     0.3903075E+00
06662 2B6D6540 (02018)          DATA     0.3392760E+00
06664 3E1F76C0 (02019)          DATA     0.4953352E+00
06666 3832BCC0 (02020)          DATA     0.4390484E+00
06668 48FA0F40 (02021)          DATA     0.5701312E+00
0666A 43B7FEC0 (02022)          DATA     0.5290526E+00
```

```
                    (02023) ; MWLQSSSM - SPECIAL INTERMEDIATE SUPPORT
                    (02024) ;
0666C 866C6671      (02025) MWLQSSSM        CALL    R6,MWSSS1
0666E 0F70          (02026)         RET
0665F 8000C0FB1     (02027)         JMP APSDONE
                    (02028) ;
                    (02029) ;
06671 906C6678      (02030) MWSSS1: MOVIR R6,INST-2
06673 0D70          (02031)         CLR R7
06674 D56FFFC0      (02032)         LPROCL R6,R7,APS$LA     ;LOAD NOP INTO 1ST WD APS
06676 FB63FFCE      (02033)         MOVLM $31,SYS$FLGS      ;SET RI
06678 0E70          (02034)         RETURN
06679 0C00          (02035)         EVEN
0667A 0000          (02036) INST:   DATA $0         ;NOP(2)
0667B 0020          (02037)         DATA $20
                    (02038)
                    (02039) ;
                    (02040) ;
0667C 0E70          (02041)         RETURN
```

```
                    (02042) * MPIFF$   MODULE TO PROCESS THE MPIFF(ISA,ISB,FLID) FCB
                    (02043) *
                    (02044) *       EXECUTES FUNCTION LIST "FLID" IF (ISA .NE. 0) .AND.
                    (02045) *                                                (ISB .EQ. 0)
                    (02046) *
                    (02047) *       FCB FORMAT (16 BIT FORMAT SHOWN)
                    (02048) *
                    (02049) *      WORD     LEFT BYTE          RIGHT BYTE
                    (02050) *      ----     ---------          ----------
                    (02051) *       0:       PTR TO NEXT FCB AND FUNCTION LIST FLAG (LSB)
                    (02052) *       1:         105                 ISA
                    (02053) *       2:           0                 ISB
                    (02054) *       3:           0                FLID
                    (02055) *       4:           0                   0
                    (02056) *       5:           0                   0
                    (02057) *
                    (02058) *
                    (02059) *       PROGRAM REGISTER USAGE:
                    (02060) *       R1:     PTR TO WORD 1 OF FCB ENTRY (SET ON ENTRY)
                    (02061) *       R2:     FUNCTION LIST ID
                    (02062) *       R4:     SCALAR A ID
                    (02063) *       R5:     SCALAR B ID
                    (02064) *
                    (02065) *
0667D 0800          (02066)         EVEN
0667E 704200FF      (02067) MPIFF$  MOVWK   R4,R1,MSK$RBYT           ;EXTRACT SCALAR A ID
06680 F0520001      (02068)         MOVMR   R5,H$(R1)               ;GET SCALAR B ID
06682 F0220002      (02069)         MOVMR   R2,W$(R1)               ;GET FUNCTION LIST ID
                    (02070) *
06684 E2000502      (02071)         CMPMZ   ISVT$(P4)               ;SEE IF SCALAR A=0
06686 81106684      (02072)         JMP     IFF$1,NEZ               ;IF A .NE. 0, TEST B
06688 0E70          (02073)         RETURN                          ;IF A .EQ. 0, RETURN
06689 0800          (02074)         EVEN
                    (02075) *
0668A E20A0502      (02076) IFF$1   CMPMZ   ISVT$(R5)               ;SEE IF SCALAR B=0
0668C 8010192C      (02077)         JMP     XFL$01,EQZ              ;IF B .EQ. 0, GO TO
                    (02078) *                                       ; "EXECUTE FUNCTION LIST"
                    (02079) *                                       ; (FLID IS IN R2).
0668E 0E70          (02080)         RETURN                          ;IF B .NE. 0, RETURN
                    (02081) *
                    (02082) *
0668F 0800          (02083)         EVEN
```

```
                    (02094) * APU3-VAPC
                    (02095) *THIS ROUTINE PERFORMS THE APC (ADAPTIVE PREDICTIVE CODING) FUNCTION
                    (02096) *
                    (02097) * NOTE: THIS ROUTINE MODIFIES ITS APS ROUTINE CODE.
                    (02098) * PITCH IS INPUT, CONVERTED TO INTEGER, AND WRITTEN TO THE
                    (02099) * BUS 1 COPY OF THE APS INSTRUCTION TO BE MODIFIED.
                    (02100) * WHEN THE WRITE HAS COMPLETED, THE BUS1 COPY IS READ
                    (02101) * AND WRITTEN TO APS PSEUDO MEMORY.
                    (02102) *
                    (02103) * VAPC COPIES CODED PITCH, CODED TAP, CODED GAIN AND CODED REFL'N
                    (02104) * COEFFS TO OUTPUT BUFFER (Y ARRAY).  THEN THE MAIN LOOP COMPUTES
                    (02105) * CODED BASEBAND RESIDUAL SAMPLES AS FOLLOWS, WHERE X(N) IS THE INPUT
                    (02106) * AND Y(N) IS THE NORMALIZED (I.E. DIVIDED BY GAIN) BASEBAND
                    (02107) * RESIDUAL SAMPLE:
                    (02108) *
                    (02109) *  YG(N)=G*Y(N)=X(N)-(TAP*D(N-PITCH))
                    (02110) *  Y(N)=YG(N)/G
                    (02111) *  YQ(N)=QUANT(Y(N))
                    (02112) *  YCODE(N)=CODE(Y(N))
                    (02113) *  D(N)=D(N-FRAMESIZE)=G*YQ(N)-YG(N)
                    (02114) *
                    (02115) *  MEMORY USAGE FOR MAIN LOOP:
                    (02116) *       A0  YCODE                 Q0
                    (02117) *       A1  T1                    Q1
                    (02118) *       A2  T2                    Q3
                    (02119) *       A3  T3                    Q3
                    (02120) *       A4  Y                     Y
                    (02121) *       A5  [YCODE]               2**-15
                    (02122) *       A6  4                     [YQ]
                    (02123) *       A7  T*D(N-P)/YG [YCODE]
                    (02124) *
                    (02125) *       M0  TAP                   UNUSED
                    (02126) *       M1  1/G                   G
                    (02127) *       M2  UNUSED                UNUSED
                    (02128) *       M3  UNUSED                UNUSED
                    (02129) *       M4  D(N-PITCH)            UNUSED
                    (02130) *       M5  X-T*D                 YQ
                    (02131) *       M6  UNUSED                UNUSED
                    (02132) *       M7  UNUSED                UNUSED
                    (02133)         EVEN
 06690 00FC         (02134)         DATA  VAPC$SA
 06691 0059         (02135)         DATA  VAPC$SZ
                    (02136) VAPCS   BEGIN APU(VAPC)
       00000000     (02137)         #A=0
                    (02138) *
                    (02139) *
                    (02140) *       CHANGE PITCH TO INTEGER:
                    (02141) *       MULT BY (2**-15), ALIGN.
                    (02142) *
                    (02143) VAPC$SA:
A00 06692 08E90000  (02144)         MOV(IOA,M1) \ NOP       ; (2**-15)
A01 06694 08ED0000  (02145)         MOV(IOA,M5) \ NOP       ; FLT PT PITCH
A02 06696 04A00003  (02146)         MUL(M1,M5) \ NOP
```

```
A03 06698 08B60000 (02137)          MOV(P,A6) \ NOP          ; A6 <= PITCH*(2**-15)
A04 0669A 3AC00000 (02138)          ALIGN(A6) \ NOP          ; R <= FXD PITCH
                   (02139) *
                   (02140) * SEND FXD PITCH TO BUS1 SO AS TO
                   (02141) * MODIFY BUS1 COPY OF APS INSTR.
                   (02142) *
A05 0669C 089C0000 (02143)          MOV(R,OQ) \ NOP          ; OQ <= FXD PITCH
                   (02144) *
                   (02145) * WAIT TIL BUS1 COPY IS MODIFIED,
                   (02146) * THEN READ IT AND WRITE TO APS PSEUDO MEM.
                   (02147) * (ALSO CLR WI SO APS INPUT PGM WILL CONTINUE)
                   (02148) * (IT WAS WAITING FOR WRITE TO COMPLETE.)
                   (02149) *
A06 0669E 0C0F2000 (02150) VAPC$W1 NOP
A07 066A0 901B2006 (02151)          JUMPC(VAPC$W1,OQE)       ; LOOP WHILE OQNE
A08 066A2 00000000 (02152)          NOP
A09 066A4 00000000 (02153)          NOP
A0A 066A6 00000000 (02154)          NOP
A0B 066A8 00000000 (02155)          NOP
A0C 066AA 00000000 (02156)          NOP
A0D 066AC 20372037 (02157)          CLEAR(WI)
A0E 066AE 00000000 (02158)          NOP
                   (02159) *
A0F 066B0 08FC0000 (02160)          MOV(IQA,OQ) \ NOP        ; READ MODIFIED VERSION
                   (02161) *                                 ; AND WRITE TO APS PSEUDO MEM
                   (02162) *
                   (02163) * WAIT TIL APS COPY WRITTEN
                   (02164) *
A10 066B2 0000000F (02165) VAPC$W2 NOP
A11 066B4 901B0010 (02166)          JUMPC(VAPC$W2,OQE)
                   (02167) *
                   (02168) * SEND FXD PITCH TO OQ AGAIN
                   (02169) * (FOR WRITE TO OUTPUT BUFFER)
                   (02170) *
A12 066B6 089C0000 (02171)          MOV(R,OQ) \ NOP          ; INTEGER PITCH
A13 066B8 08E80200 (02172)          MOV(IQA,M0)\NOP          ;QUANTIZED TAP (REAL)
A14 066BA 000E08E9 (02173)          NOP\MOV(IQA,M1)          ;QUANTIZED GAIN(REAL)
A15 066BC 08E90000 (02174)          MOV(IQA,M1)\NOP          ;INVERSE QUANTIZED GAIN(REAL)
A16 066BE 08FC0000 (02175)          MOV(IQA,OQ)\NOP          ;CODED TAP
A17 066C0 08FC0000 (02176)          MOV(IQA,OQ)\NOP          ;CODED GAIN(INT)
A18 066C2 08FC0200 (02177)          MOV(IQA,OQ)\NOP          ;8 CODED REFL'N COEFFS (INT'S)
A19 066C4 08FC0000 (02178)          MOV(IQA,OQ)\NOP
A1A 066C6 08FC0000 (02179)          MOV(IQA,OQ)\NOP
A1B 066C8 08FC0000 (02180)          MOV(IQA,OQ)\NOP
A1C 066CA 08FC0000 (02181)          MOV(IQA,OQ)\NOP
A1D 066CC 08FC0000 (02182)          MOV(IQA,OQ)\NOP
A1E 066CE 08FC0000 (02183)          MOV(IQA,OQ)\NOP
A1F 066D0 08FC0000 (02184)          MOV(IQA,OQ)\NOP
A20 066D2 08F10000 (02185)          MOV(IQA,A1)\NOP          ;T1=.531
A21 066D4 08F22000 (02186)          MOV(IQA,A2)\NOP          ;T2=1.249
A22 066D6 08F32000 (02187)          MOV(IQA,A3)\NOP          ;T3=2.371
A23 066D8 000039F0 (02188)          NOP\MOV(IQA,A0)          ;YQ0=.233
A24 066DA 000008F1 (02189)          NOP\MOV(IQA,A1)          ;YQ1=.938
```

```
A25 066DC 000008F2 (02190)          NOP\MOV(IQA,A2)          ;YQ2=1.666
A26 066DE 000008F3 (02191)          NOP\MOV(IQA,A3)          ;YQ3=3.875
                   (02192) *
                   (02193) *
                   (02194)
                   (02195) *  DO MAIN APCLP
A27 06650 08F80000 (02196) APCLP    MOV(IQA,A0)\NOP          ;X(N)
A28 066E2 08EC0008 (02197)          MOV(IQA,M4)\NOP          ;D(N-PITCH)
A29 066E4 84000008 (02198)          MUL(M0,M4)\NOP           ;TAP*D(N-PITCH)
A2A 066E6 08F71680 (02199)          MOV(IQA,A7)\K(+1)        ;TWO FAKE INS SO INPUT DOESN'T
                   (02200) *                          GET TOO FAR AHEAD OF OUTPUT\GET 1.0
A2B 066E8 08E00002 (02201)          MOV(IQA,NULL)\NOP
A2C 066EA 08870000 (02202)          MOV(P,A7)\NOP            ;STORE TAP*D(N-PITCH)
A2D 066EC 4F000995 (02203)          SUB(A0,A7)\MOV(R,A5)     ;X(N)-T*D(N-PITCH)\STORE 1.
A2E 066EE 088D2000 (02204)          MOV(R,M5)\NOP            ;Y(N)*G
A2F 066F0 84A084A0 (02205)          MUL(M1,M5)               ;Y(N)*G)*(1/G)
A30 066F2 08972897 (02206)          MOV(R,A7)                ;SAVE Y(N)*G
A31 066F4 08B40002 (02207)          MOV(P,A4)\NOP            ;SAVE Y(N)
                   (02208) *  QUANTIZE AND CODE Y(N)
A32 066F6 16C00817 (02209)          K(+4)\MOV(ZERO,A7)       ;GET CONSTANT 4, FOR LATER\CODE 0
A33 066F8 74360200 (02210)          MOV(A6),MAXABS(A1,A4)\R(A0)      ;Y.GE.BIN1?\GET YQ0
A34 066FA 08B82818 (02211)          MOV(P,EXO)\MOV(ZERO,EXO)         ;TRANSFER Y(N)\CODE=
A35 066FC 00002854 (02212)          NOP\MOV(EXI,A4)          ;WILL NEED Y(N) EVENTUALLY
A36 066FE 08804786 (02213)          MOV(R,NULL)\MOV(A6),ADD(A5,A7)           ;MAKE SURE MAXABS
                   (02214) *                                  IS DONE\GEN CODE 1
A37 06700 901E0042 (02215)          JUMPC(QOUT,T1)           ;JUMP IF BIN1 GE Y
A38 06702 00002898 (02216)          NOP\MOV(R,EXO)           ;CODE1
A39 06704 74400237 (02217)          MAXABS(A2,A4)\MOV(A7),R(A1)      ;TEST BIN2>Y\GET YQ1
A3A 06706 08804786 (02218)          MOV(R,NULL)\MOV(A6),ADD(A5,A7)           ;MAKE SURE MAXABS DONE
                   (02219) *                                  ;GEN CODE 2
A3B 06708 901E0042 (02220)          JUMPC(QOUT,T1)           ;JUMP IF BIN2 GE Y
A3C 0670A 00002898 (02221)          NOP\MOV(R,EXO)           ;CODE2
A3D 0670C 74602257 (02222)          MAXABS(A3,A4)\MOV(A7),R(A2)      ;BIN3\GET YQ2,
                   (02223) *                                  GEN CODE3
A3E 0670E 08804786 (02224)          MOV(R,NULL)\MOV(A6),ADD(A5,A7)   ;MAKE SURE MAXABS DONE
                   (02225) *                                  ;GEN CODE 3
A3F 06710 901E0042 (02226)          JUMPC(QOUT,T1)           ;JUMPIF BIN3 GE Y
A40 06712 00002278 (02227)          NOP\MOV(EXO),R(A3)       ;\EXO=CODE=3,GET YQ3
A41 06714 00002896 (02228)          NOP\MOV(R,A6)            ;\A6=YQ=YQ3*
A42 06716 12800000 (02229) QOUT     ABS(A4)\NOP              ;SIGN(Y)
A43 06718 08550817 (02230)          MOV(EXI,A5)\MOV(ZERO,A7)         ;CODE\0
A44 0671A 08900300 (02231)          MOV(R,NULL)\NOP          ;MAKE SURE ABS DONE
A45 0671C 55C01280 (02232)          ADDST(A6,A5)\ABS(A4)     ;4+ OR - CODE\GET SIGN Y
A46 0671E 06950008 (02233)          MOV(A5),K(-1)\NOP        ;NEED -1 SOON
A47 06720 12960000 (02234)          MOV(A6),ABS(A4)\NOP      ;STORE -1,GET SIGN Y
A48 06722 08800980 (02235)          MOV(R,NULL)\MOV(P,NULL)  ;MAKE SURE ADD OPS ARE DONE
A49 06724 55A056E0 (02236)          ADDLT(A5,A6)\ADDST(A7,A6)        ;DECR IF Y NEG\SIGN(YQ)=SIGN(Y)
                   (02237) *R=CODE (4 - 7)\R=YQ
                   (02238) * FIND DIFFERENCE OF QUANTIZED Y AND ACTUAL Y
A4A 06726 2A100880 (02239)          MOV(A0),DECR8(A0)\MOV(R,M5)      ;START TO RT SHFT CODE\STORE YQ
A4B 06728 2A1084A0 (02240)          MOV(A0),DECR8(A0)\MUL(M1,M5)     ;8 BITS RT\G*YQ
A4C 0672A 2A100000 (02241)          MOV(A0),DECR8(A0)\NOP            ;*2**-12
A4D 0672C 2A100000 (02242)          MOV(A0),DECR8(A0)\NOP            ;16 BITS RT
```

```
A4E 06728 40100000 (02243)          MOV(A0),ADD(A0,A0)\NOP          ;NOW * 2**-15
A4F 06730 3A100BB8 (02244)          MOV(A0),ALIGN(A0)\MOV(P,EXD)    ;FIX CODE\XFER G*YQ
A50 06732 089C0000 (02245)          MOV(R,0Q)\NOP                   ;CODE OUT
A51 06734 08540000 (02246)          MOV(EXI,A4)\NOP                 ;G*YQ(N)
A52 06736 4F800000 (02247)          SUB(A4,A7)\NOP                  ;G*YQ-G*Y=D
A53 06738 089C0000 (02248)          MOV(R,0Q)\NOP                   ;D(N) OUT
A54 0673A 089C0000 (02249)          MOV(R,0Q)\NOP                   ;D(N-FRAMESIZE) OUT
A55 0673C 901D0027 (02250)          JUMPC(APCLP,FI)
                   (02251) *
A56 0673E 20322032 (02252)          CLEAR(RA)
A57 06740 10000000 (02253)          JUMP(0)
A58 06742 00000000 (02254)          NOP
                   (02255) *
    06744          (02256)          END
           20002059 (02257)          VAPC$SZ=.A-VAPC$SA
```

```
                    (02258) ; APS3-AAPC  (VAPC)
                    (02259) ;APS PROGRAM FOR APC
                    (02260) ;INPUT STREAM: (2**-15),PITCH,'MODIFIED INSTR',TQTAP,TQG,TQGI,TCTAP,
                    (02261) ;TCGAIN,TCK1... TCK8(INT),B.B.T1...B.B.T3(REAL),BBQ0...BBQ3(REAL),
                    (02262) ;LOOP           X(1),D(1-PITCH),DUMMY,DUMMY,X(2),D(2-PITCH)...REAL
                    (02263) ;OUTPUT STREAM - FXD PTCH,MODIFIED INSTR,PITCH(INT)=TSINK(1),
                    (02264) ;                TCTAP=TSINK(2),TCG=TSINK(3),
                    (02265) ;                TCK1...TCK8=TSINK(4)...TSINK(11),
                    (02266) ;LOOP           BBCODE(1)(INT)=TSINK(12),D(1),D(1-FRSIZ),BBCODE(2)....
                    (02267) ; CALLED APC(Y,PITCH,D,QTAP,X,CTAP,BBT1)
                    (02269) ;
                    (02269) ; NOTE: THIS ROUTINE MODIFIES ITS APS ROUTINE CODE.
                    (02270) ; PITCH IS INPUT, CONVERTED TO INTEGER, AND WRITTEN TO THE
                    (02271) ; BUS 1 COPY OF THE APS INSTRUCTION TO BE MODIFIED.
                    (02272) ; WHEN THE WRITE HAS COMPLETED, THE BUS1 COPY IS READ
                    (02273) ; AND WRITTEN TO APS PSEUDO MEMORY.
                    (02274) ;
                    (02275) ;
                    (02276)         EVEN
 06744 000067F2     (02277)         ADDR AAPC$I
 06746 00006754     (02278)         ADDR AAPC$ + 2*AAPC$S
 06748 0004         (02279)         DATA 4          ;SCALAR BLOCK
 06749 0002         (02280)         DATA AAPC$Z
 0674A 000067C9     (02281)         ADDR AAPC$A
                    (02282)         EVEN
                    (02283) AAPC$   BEGIN APS(AAPC)
                    (02284) ;
                    (02285) ; LEAVE NON-FUNCTIONAL INSTR IN SO DINK WON'T
                    (02286) ; HURT US (WON'T HELP EITHER).
                    (02287) ;
A00 0674C 0040C000  (02288)         LOAD(BR0,2)
A01 0674E 02203240  (02289)         JSN(AAPC$3,P2)          ;START OUTPUT
A02 06750 04300030  (02290)         SET(R0)
                    (02291) ;
                    (02292) ;       GEN (2**-15)
                    (02293) ;
A03 06752 06F203CE  (02294)         LOAD(BR3,SYI$UN1(1),TF)
                    (02295) ;
A04 06754 08720000  (02296) AAPC$S  LOAD(BR3,M$$(1))        ;LOAD PITCH
A05 06756 0A420003  (02297)         LOAD(BR0,M$$(1))        ;LOAD QTAP
A06 06758 0C520002  (02298)         LOAD(BR1,M$$(1))        ;LOAD CTAP
A07 0675A 0E620F00  (02299)         LOAD(BR2,M$$(1))        ;LOAD QUANT INFO
                    (02300) ;
                    (02301) ;
                    (02302) ;
A08 0675C 1FB90017  (02303)         MOVB(BR3,BR3,TF)        ;GEN PITCH
                    (02304) ;
                    (02305) ;       WAIT FOR APU PGM TO SIGNAL THAT BUS1 COPY OF
                    (02306) ;       MODIFIED INSTR HAS BEEN WRITTEN
                    (02307) ;
A09 0675E 12300037  (02308)         SET(WI)
A0A 06760 14000020  (02309)         NOP(0)
A0B 06762 16000020  (02310)         NOP(0)
```

```
                    (02311) *
                    (02312) *       GEN MODIFIED INSTR ADDRESS
                    (02313) *
A0C 06764 10F26785  (02314)         LOAD(BR3,AAPC$M(1),TF)
                    (02315) *
A0D 06766 1A890011  (02316)         MOVB(BR0,BR0,TF)        ;GEN QTAP
A0E 06768 1C89003A  (02317)         ADDL(BR0,2,TF)          ;QUANTIZED GAIN
A0F 0676A 1E89003A  (02318)         ADDL(BR0,2,TF)          ;INVERSE QUANTIZED GAIN
A10 0676C 21290013  (02319)         MOVB(BR0,BR1,TE)        ;CODED TAP
A11 0676E 2309003A  (02320)         ADDL(BR0,2,TE)          ;CODED GAIN
A12 06770 2509003A  (02321)         ADDL(BR0,2,TE)          ;CK1
A13 06772 27090C3A  (02322)         ADDL(BR0,2,TE)          ;CK2
A14 06774 29090F3A  (02323)         ADDL(BR0,2,TE)          ;CK3
A15 06776 2B090C3A  (02324)         ADDL(BR0,2,TE)          ;CK4
A16 06778 2D090F3A  (02325)         ADDL(BR0,2,TE)          ;CK5
A17 0677A 2F090C3A  (02326)         ADDL(BR0,2,TE)          ;CK6
A18 0677C 31090C3A  (02327)         ADDL(BR0,2,TE)          ;CK7
A19 0677E 33090F3A  (02328)         ADDL(BR0,2,TE)          ;CK8
                    (02329) *BASEBAND QUANTIZATION CONSTANTS
A1A 06780 34890015  (02330)         MOVB(BR0,BR2,TF)        ;THRESH 1
A1B 06782 36890C3A  (02331)         ADDL(BR0,2,TF)          ;TH 2
A1C 06784 38890F3A  (02332)         ADDL(BR0,2,TF)          ;TH 3
A1D 06786 3A890C3A  (02333)         ADDL(BR0,2,TF)          ;QUANT VAL 0
A1E 06788 3C890F3A  (02334)         ADDL(BR0,2,TF)          ;QV 1
A1F 0678A 3E890C3A  (02335)         ADDL(BR0,2,TF)          ;QV 2
A20 0678C 40890F3A  (02336)         ADDL(BR0,2,TF)          ;QV 3
                    (02337) *
                    (02338) *       SET UP D(N-PITCH) ADDR
                    (02339) *
                    (02340) *!!!!NEXT INSTR MODIFIED BY OUTPUT PGM!!!!!!!!
                    (02341) *
            0000678E (02342) %APC$M = #L
A21 0678E 42480000  (02343)         LOAD(BR0,M$$)           ;LOAD PITCH
                    (02344) *
A22 06790 44090029  (02345)         ADDB(BR0,BR0)           ;DOUBLE FOR HWORD ADDR
A23 06792 46701000  (02346)         LOAD(BR3,D1)            ;UBA = D(0)
A24 06794 48500000  (02347)         LOAD(BR1,M$$)           ;UBS-1
A25 06796 4A500000  (02348)         LOAD(BR1,M$$)           ;SPACING
A26 06798 4C390021  (02349)         SUBB(BR3,BR0)           ;D(0-PITCH)
A27 0679A 4E390032  (02350)         SUBL(BR3,2)             ;SUBTR SPACING
                    (02351) *
                    (02352) *SET UP FOR LOOP
A28 0679C 50402000  (02353)         LOAD(BR0,D2)            ;BASE ADDR OF INPUT X =V BID
A29 0679E 52500000  (02354)         LOAD(BR1,M$$)           ;X LENGTH-1
A2A 067A0 54020000  (02355)         SUB(BR0,M$$)            ;INIT FOR INCREMENT INSTR IN LOOP
                    (02356) * INPUT LOOP
A2B 067A2 568AA006  (02357) #1      ADD(BR0,D10),TF)        ;X(N)
A2C 067A4 58890F3A  (02358)         ADDL(BR3,2,TF)          ;D(N-PITCH)
A2D 067A6 5AB90017  (02359)         MOVB(BR3,BR3,TF)        ;DUMMY 1
A2E 067A8 5CB90017  (02360)         MOVB(BR3,BR3,TF)        ;DUMMY 2
A2F 067AA 5E192BB1  (02361)         SUBL(BR1,1),JUMPP(#1)   ;COUNT X
A30 067AC 60200031  (02362)         CLEAR(RI)
A31 067AE 62000020  (02363)         NOP(P)
```

```
                    (@2364) *
                    (@2365) *
                    (@2366) *OUTPUT PROGRAM
                    (@2367) *
A32 @67B@ 643@@@32  (@2368) AAPC$3  SET(RA)
                    (@2369) *
                    (@2370) *       GEN BUS1 ADDR OF INPUT PGM INSTR TO
                    (@2371) *       BE MODIFIED (LEFT HWORD)
                    (@2372) *
A33 @67B2 6772678F  (@2373)         LOAD(BW3,AAPC$M+1(1),TE)
                    (@2374) *
                    (@2375) *       WRITE MODIFIED INSTR TO PSEUDO MEM
                    (@2376) *
A34 @67B4 68F3FFC@  (@2377)         LOAD(BW3,APS$LA(1),TF)
                    (@2378) *
                    (@2379) *
A35 @67B6 6A6@1@14  (@2380)         LOAD(BW2,[1])           ;D VECTOR=U BID
A36 @67B8 6C7@@@@F  (@2381)         LOAD(BW3,M$$)           ;SIZE D -1
A37 @67BA 6E4@@@@@  (@2382)         LOAD(BW@,M$$)           ;D SPACING
A38 @67BC 7@11@@14  (@2383)         MOVB(BW1,BW2)           ;SAVE D ADDR
A39 @67BE 72310@39  (@2384)         ADDL(BW3,1)             ;MAKE IT SIZE OF D
A3A @67C@ 74110026  (@2385)         SUBB(BW1,BW3)           ;D-DSIZE
A3B @67C2 76110026  (@2386)         SUBB(BW1,BW3)           ;D-2*DSIZE
A3C @67C4 78110032  (@2387)         SUBL(BW1,2)             ;INIT D-FRSIZ FOR INCREMENT
A3D @67C6 7A210@32  (@2388)         SUBL(BW2,2)             ;INIT D FOR INCREMENT
                    (@2389) *NOW OUTPUT CODED PARAMS
A3E @67C8 7D4@@@12  (@2390)         LOAD(BW@,[@],TE)        ;PITCH TO Y
A3F @67CA 7E7@@@@@  (@2391)         LOAD(BW3,M$$)           ;Y SIZE
A40 @67CC 81@@@@@@  (@2392)         ADD(BW@,M$$,TE)         ;TCTAP
A41 @67CE 83@1@@39  (@2393)         ADDL(BW@,1,TE)          ;TCGAIN
A42 @67D@ 8431@@33  (@2394)         SUBL(BW3,3)             ;COUNT PITCH,TAP,GAIN
A43 @67D2 87@1@@39  (@2395)         ADDL(BW@,1,TE)          ;CK1
A44 @67D4 89@1@@39  (@2396)         ADDL(BW@,1,TE)          ;CK2
A45 @67D6 8B@1@@39  (@2397)         ADDL(BW@,1,TE)          ;CK3
A46 @67D8 8D@1@039  (@2398)         ADDL(BW@,1,TE)          ;CK4
A47 @67DA 8E31@@34  (@2399)         SUBL(BW3,4)             ;COUNT 4 REFLN COEFFS
A48 @67DC 91@1@@39  (@24@@)         ADDL(BW@,1,TE)          ;CK5
A49 @67DE 93@1@039  (@24@1)         ADDL(BW@,1,TE)          ;CK6
A4A @67E@ 95@1@039  (@24@2)         ADDL(BW@,1,TE)          ;CK7
A4B @67E2 97@1@039  (@24@3)         ADDL(BW@,1,TE)          ;CK8
A4C @67E4 98310034  (@24@4)         SUBL(BW3,4)             ;COUNT 4 MORE REFLN COEFFS
!                   (@2405) *OUTPUT LOOP
A4D @67E6 9BF1@@39  (@2406) #4      ADDL(BW@,1,TE)          ;CODE(N)
A4E @67E8 9C91003A  (@24@7)         ADDL(BW1,2,TF)          ;D-FRSIZ(N)
A4F @67EA 9EA1@@3A  (@24@8)         ADDL(BW2,2,TF)          ;D(N)
A5@ @67EC A@314DB1  (@24@9)         SUBL(BW3,1),JUMPP(#4)   ;COUNT CODE
           @@@@67C8 (@2410) AAPC$A=#C
A51 @67EE A22@@@3@  (@2411)         CLEAR(RO)
A52 @67F@ A4@@@@2@  (@2412)         NOP(@)
    @67F2           (@2413)         END
    @67F2 @@@@@@@@  (@2414) AAPC$I  DATA 5F'@.@'
     ...
           @@@@3@B@ (@2415) AAPC$Z=#L-AAPC$
```

PAGE 55: [BBN-TENEXD]<MAP>@BN300.NSO.61, 30-Dec-79 16:14:24, Ed: KFIELD
APS3-AAPC (VAPC)

```
                    (02416) ; APU3-DEALU  (DEAL)
                    (02417) ;APU PGM FOR DEAL
                    (02418) ;BINDS TO APS-DEALS
                    (02419) ;DEAL OUT PARAMS FROM INPUT ARRAY TO PITCH, TAP.
                    (02420) ;MULTIPLY NEXT INPUTS (BASEBAND SAMPLES) BY GAIN, AND OUTPUT PRODUCT
                    (02421)         EVEN
    067FC 0000     (02422)         DATA DEALU$SA
    067FD 0019     (02423)         DATA DEALU$SZ
                    (02424) ;
                    (02425) DEALU$  BEGIN APU(DEALU)
                00000002 (02426)         #A=0
  A00 067FE 09F10000 (02427) DEALU$SA MOV(IQA,A1)\NOP          ;PITCH
  A01 06800 32203220 (02428)         NOP4(A1)                 ;MAKE IT FLT
  A02 06802 099C0000 (02429)         MOV(R,OQ)\NOP            ;PUT FLTING PITCH OUT
  A03 06804 08FC3000 (02430)         MOV(IQA,OQ)\NOP          ;TAP
  A04 06806 08EC08EC (02431)         MOV(IQA,M4)              ;GAIN
  A05 06808 08FC0000 (02432)         MOV(IQA,OQ)\NOP          ;K1
  A06 0680A 08FC0000 (02433)         MOV(IQA,OQ)\NOP          ;K2
  A07 0680C 08FC0000 (02434)         MOV(IQA,OQ)\NOP          ;K3
  A08 0680E 08FC0000 (02435)         MOV(IQA,OQ)\NOP          ;K4
  A09 06810 08FC0000 (02436)         MOV(IQA,OQ)\NOP          ;K5
  A0A 06812 08FC0000 (02437)         MOV(IQA,OQ)\NOP          ;K6
  A0B 06814 08FC0000 (02438)         MOV(IQA,OQ)\NOP          ;K7
  A0C 06816 08FC0000 (02439)         MOV(IQA,OQ)\NOP          ;K8
  A0D 06818 08E80000 (02440)         MOV(IQA,M0)\NOP          ;BB(4*K)
  A0E 0681A 020008E8 (02441)         NOP\MOV(IQA,M0)          ;BB(4N+1)
  A0F 0681C 84008400 (02442)         MUL(M0,M4)               ;MULTIPLY BY GAIN
  A10 0681E 08E90000 (02443) #2      MOV(IQA,M1)\NOP          ;BB(4N+2)
  A11 06820 000008E9 (02444)         NOP\MOV(IQA,M1)          ;BB(4N+3)
  A12 06822 849C849C (02445)         MOV(OQ),MUL(M1,M4)       ;BB(4N),BB(4N+1)*GAIN OUT
  A13 06824 08E80000 (02446)         MOV(IQA,M0)\NOP          ;BB(4N+5)
  A14 06826 000079E8 (02447)         NOP\MOV(IQA,M0)          ;BB(4N+6)
  A15 06828 841C841C (02448)         MOV(OQ),MUL(M0,M4)       ;BB(4N+2),4N+3)*GAIN OUT
  A16 0682A 901C001F (02449)         JUMPC(#2,EO)             ;LOOP IF NOT DONE
  A17 0682C 27322232 (02450)         CLEAR(RA)
  A18 06825 00000000 (02451)         NOP
            00000019 (02452)         DEALU$SZ=#A-DEALU$SA
    06830          (02453)         END
```

```
                    (02454) ; APS3-DEAL
                    (02455) *DEAL(Y,A,U,B,V)
                    (02456) * WHERE Y IS BASEBAND SAMPLES *GAIN,A IS RECEIVED PITCH,U IS RECVD K'S,
                    (02457) *B IS RECVD TAP,AND W IS RSOURCE BUFFER,HOME OF ALL INPUT PARAMS
                    (02458) *INPUT STREAM - V(N), WHERE THESE ARE LONG FLOATING NUMBERS
                    (02459) *W CONSISTS OF PITCH,TAP,GAIN,8 K'S,AND 60 BB SAMPLES
                    (02460) *
                    (02461) *OUTPUT STREAM - PITCH,TAP,GAIN, K(1)...K(8),BB*GAIN(1),...BB*GAIN(60)
                    (02462) *
                    (02463)         EVEN
    06830 00006868  (02464)         ADDR DEALS$I
    06832 0000684C  (02465)         ADDR DEALS$ + 2*DEALS$S
    06834 0002      (02466)         DATA 2                      ;SCALAR BLOCK
    06835 003C      (02467)         DATA DEALS$Z
                    (02468)         EVEN
    06836 00005B60  (02469)         ADDR DEALS$A
                    (02470)         EVEN
                    (02471) DEALS$  BEGIN APS(DEALS)
A00 06836 0020C943  (02472)         JSH(DEALS$3,P2)
A01 0683A 2230037C  (02473)         SET(RO)
A02 0683C 04402000  (02474)         LOAD(BR0,C2])               ;SOURCE = V
A03 06835 065000CC  (02475)         LOAD(BR1,M$$)               ;SIZE OF SOURCE-1
A04 06840 00220000  (02476)         SUB(BR0,M$$)                ;SPACING
                    (02477) *INPUT LOOP
A05 06842 0A0AA006  (02478) $1      ADD(BR0,C10],TF)            ;BRING IN NEXT PARAM
A06 06844 0C1005B1  (02479)         SUBL(BP1,1),JUMPP(#1)       ;COUNT IT AND LOOP
A07 06846 05200031  (02480)         CLEAR(RI)
A08 06848 10000020  (02481)         NOP(0)                      ;THAT'S ALL FOR INPUT
A09 0684A 12300032  (02482) DEALS$3 SET(RA)
                    (02483)
A0A 0684C 14C20020  (02484) DEALS$S LOAD(BW0,M$$(1),TF)         ;PUT OUT PITCH JUST LIKE IT COMES IN
A0B 06845 16C20000  (02485)         LOAD(BW0,M$$(1),TF)         ;PUT OUT TAP
A0C 06850 18401000  (02486)         LOAD(BW0,C1])               ;K ARRAY
A0D 06852 1A500002  (02487)         LOAD(BW1,M$$)               ;K APRAY SIZE-1
A0E 06854 1C020002  (02488)         SUB(BW0,M$$)                ;K ARRAY SP MUST BE 2
                    (02489) *OUTPUT LOOP 1
A0F 06856 1E0A9006  (02490) $5      ADD(BWC,C9],TF)             ;K ARRAY SPACING
A10 06858 20110FB1  (02491)         SUBL(BP1,1),JUMPP(#5)       ;COUNT IT AND LOOP
A11 0685A 22400004  (02492)         LOAD(BWC,C2])               ;OUTPUT ARRAY OF BB SAMPLES * GAIN
A12 0685C 24500000  (02493)         LOAD(BW1,M$$)               ;Y SIZE-1
A13 0685E 26020000  (02494)         SUB(BW0,M$$)                ;Y SPACING
                    (02495) *OUTPUT LOOP 2
A14 06860 28A80006  (02496) $6      ADD(BW0,C8],TF)             ;Y SPACING
A15 06862 2A1114B1  (02497)         SUBL(BW1,1),JUMPP(#5)
            00006860 (02498) DEALS$A=PC
A16 06864 2C200030  (02499)         CLEAR(PO)
A17 06866 2E000020  (02500)         NOP(0)
    06868           (02501)         END
    06868 00000000  (02502) DEALS$I DATA    6F'0.0'
     ...
            0000003C (02503) DEALS$Z=#L-DEALS$
```

```
                        (02504) ; APU3-APCIU  (VIAPC)
                        (02505) ;APU PGM FOR APCI
                        (02506) ;BINDS TO APCIS
                        (02507) ;EQN: Y(N)=Y(N-YSIZE)=X(N)+TAP*Y(N-PITCH)
                        (02508) ;CALLED  VIAPC(Y,TAP,X,PITCH)
                        (02509) ;
                        (02510) ;
                        (02511) ; NOTE: THIS ROUTINE MODIFIES ITS APS ROUTINE CODE.
                        (02512) ; PITCH IS INPUT, CONVERTED TO INTEGER, AND WRITTEN TO THE
                        (02513) ; BUS 1 COPY OF THE APS INSTRUCTION TO BE MODIFIED.
                        (02514) ; WHEN THE WRITE HAS COMPLETED, THE BUS1 COPY IS READ
                        (02515) ; AND WRITTEN TO APS PSEUDO MEMORY.
                        (02516) ;
                        (02517) ;
                        (02518)         EVEN
     06874 0007         (02519)         DATA APCIU$SA
     06875 0027         (02520)         DATA APCIU$SZ
                        (02521) ;
                        (02522) APCIU$  BEGIN APU(APCIU)
           00002000     (02523) #A=0
                        (02524) ;
                        (02525) ;
                        (02526) ; CHANGE PITCH TO INTEGER:
                        (02527) ;  MULT BY (2**-15), ALIGN.
                        (02528) ;
                        (02529) APCIU$SA:
A00 06876 08E90000      (02530)         MOV(IQA,M1) \ NOP       ; (2**-15)
A01 06878 08ED0000      (02531)         MOV(IQA,M5) \ NOP       ; FLT PT PITCH
A02 0687A 84A20002      (02532)         MUL(M1,M5) \ NOP
A03 0687C 08B60000      (02533)         MOV(P,A6) \ NOP
A04 0687E 3AC00000      (02534)         ALIGN(A6) \ NOP         ; R <= FXD PITCH
                        (02535) ;
                        (02536) ; SEND FXD PITCH TO BUS1 SO AS TO
                        (02537) ; MODIFY BUS1 COPY OF APS INSTR.
                        (02538) ;
A05 26990 089C0000      (02539)         MOV(R,OQ) \ NOP         ; OQ <= FXD PITCH
                        (02540) ;
                        (02541) ; WAIT TIL BUS1 COPY IS MODIFIED,
                        (02542) ; THEN READ IT AND WRITE TO APS PSEUDO MEM.
                        (02543) ; (ALSO SLR WI SO APS INPUT PGM WILL CONTINUE)
                        (02544) ; (IT WAITING FOR WRITE TO COMPLETE.)
                        (02545) ;
A06 06882 00000000      (02546) VAPCIS#1        NOP
A07 06884 981B0026      (02547)         JUMPC(VAPCIS#1,OQE)     ; LOOP WHILE OQNE
A08 06886 00002000      (02548)         NOP
A09 06888 0F0C0000      (02549)         NOP
A0A 0688A 00000303      (02550)         NOP
A0B 0688C 00000000      (02551)         NOP
A0C 0688E 0W300F00      (02552)         NOP
A0D 06890 20372F37      (02553)         CLEAR(WI)
A0E 06892 00000000      (02554)         NOP
                        (02555) ;
A0F 06894 08FC0000      (02556)         MOV(IQA,OQ) \ NOP       ; READ MODIFIED VERSION
```

```
                    (02557) *                               ; AND WRITE TO APS PSEUDO MEM
                    (02558) *
                    (02559) * WAIT TIL APS COPY WRITTEN
                    (02560) * (CLR WI AGAIN)
                    (02561) *
  A10 06896 00000000 (02562) VAPCI$W2        NOP
  A11 06898 9218801E (02563)          JUMPC(VAPCI$W2,OQE)
  A12 0689A 0000000C (02564)          NOP
  A13 0689C 00000000 (02565)          NOP
  A14 0689E 00000000 (02566)          NOP
  A15 068A0 00020000 (02567)          NOP
  A16 063A2 20000000 (02569)          NOP
  A17 068A4 20372037 (02569)          CLEAR(WI)
  A18 068A6 03000000 (02570)          NOP
                    (02571) *
                    (02572) *
                    (02573) *
  A19 068A8 08E00000 (02574)          MOV(IQA,M0)\NOP ;TAP
  A1A 068AA 0BEC000C (02575)          MOV(IQA,M4)\NOP           ;Y(-PITCH)
  A1B 068AC 84000000 (02576) #1       MUL(M0,M4\NOP             ;TAP*Y(N-PITCH)
  A1C 068AE 08F00000 (02577)          MOV(IQA,A0)\NOP           ;X(N)
  A1D 068B0 08B10081 (02578)          MOV(P,A1)
  A1E 068B2 41004100 (02579)          ADD(A0,A1)                ;X(N)+TAP*Y(N-PITCH)
  A1F 068B4 08F20000 (02580)          MOV(IQA,A2)\NOP           ;DUMMY
  A20 068B6 08EC0000 (02581)          MOV(IQA,M4)\NOP           ;Y(N-PITCH+1)
  A21 068B8 089C0000 (02582)          MOV(R,OQ)\NOP             ;Y(N)
  A22 068BA 089C0000 (02583)          MOV(R,OQ)\NOP             ;Y(N-YSIZE)
  A23 068BC 901C0C1B (02584)          JUMPC(#1,EO)              ;LOOP TIL DONE
  A24 068BE 20322032 (02585)          CLEAR(RA)
  A25 068C0 00000000 (02586)          NOP
  A26 068C2 00000000 (02587)          NOP               ;FOR GOOD LUCK
            00000027 (02588) APCIU$SZ=#A-APCIU$SA
      068C4          (02589)          END
```

```
                    (02590) ; APS-APCIS  INVERSE APC FUNCTION (VIAPC)
                    (02591) *EQN - Y(N)=Y(N-YSIZE)=X(N)+TAP*Y(N-PITCH)
                    (02592) *INPUT STREAM:
                    (02593) *       (2**-15),PITCH,'MOD INSTR',TAP,
                    (02594) *    Y(N-PITCH),X(N)=U(N),DUMMY=U(N);...
                    (02595) *OUTPUT STREAM:
                    (02596) *       FXD PTCH,MODIFIED INSTR,
                    (02597) *Y(1),Y(1-YSIZE),Y(2),Y(2-YSIZE),....
                    (02598) *CALL APCI(Y,TAP,X,PITCH)
                    (02599) *
                    (02600) *
                    (02601) * NOTE: THIS ROUTINE MODIFIES ITS APS ROUTINE CODE.
                    (02602) * PITCH IS INPUT, CONVERTED TO INTEGER, AND WRITTEN TO THE
                    (02603) * BUS 1 COPY OF THE APS INSTRUCTION TO BE MODIFIED.
                    (02604) * WHEN THE WRITE HAS COMPLETED, THE BUS1 COPY IS READ
                    (02605) * AND WRITTEN TO APS PSEUDO MEMORY.
                    (02606) *
                    (02607) *HEADER BLOCK
                    (02608)         EVEN
    068C4 0022693C  (02609)         ADDR APCS$I
    068C6 00036BD4  (02610)         ADDR APCIS$ + 2*APCIS$$
    068C8 0002      (02611)         DATA 2
    068C9 007E      (02612)         DATA APCIS$Z
    068CA 00006934  (02613)         ADDR APCIS$A
                    (02614)         EVEN
                    (02615) APCIS$  BEGIN APS(APCIS)
                    (02616) *
                    (02617) * LEAVE NON-FUNCTIONAL INSTR IN SO DINK WON'T
                    (02618) * HURT ANYTHING.
                    (02619) *
A00 068CC 00400000  (02620)         LOAD(BR0,0)
A01 068CE 02202840  (02621)         JSN(APCIS$3,P2)         ;START OUTPUT
A02 068D0 04300030  (02622)         SET(RO)
                    (02623) *
                    (02624) * GEN (2**-15)
                    (02625) *
A03 068D2 06F203CE  (02626)         LOAD(BR3,SVT$UN1(1),TF)
                    (02627) *
A04 068D4 0B420000  (02628) APCIS$S LOAD(BR0,M$$(1))        ;LOAD TAP
A05 068D6 0A520000  (02629)         LOAD(BR1,M$$(1))        ;LOAD PITCH
                    (02630) *
                    (02631) *
A06 068D8 0C990013  (02632)         MOVB(BR1,BR1,TF)        ;GEN PITCH
                    (02633) *
                    (02634) * WAIT FOR APU PGM TO SIGNAL THAT BUS1 COPY
                    (02635) * OF MODIFIED INSTR HAS BEEN WRITTEN
                    (02636) *
A07 068DA 0E300037  (02637)         SET(W1)
A08 068DC 10000020  (02638)         NOP(0)
A09 068DE 12000020  (02639)         NOP(0)
                    (02640) *
                    (02641) * GEN MODIFIED INSTR ADDR
                    (02642) *
```

```
A0A 068E0 14F268FE (02643)          LOAD(BR3,APCISSM(1),TF)
                   (02644) *
                   (02645) * WAIT ON WRITE AGAIN
                   (02646) *
A0B 06852 16300037 (02647)          SET(WI)
A0C 06854 18000027 (02648)          NOP(0)
A0D 06856 1A000220 (02649)          NOP(0)
                   (02650) *
A0E 063E8 1C890011 (02651)          MOVB(BR0,BP0,TF)        ;GEN TAP
                   (02652) *
                   (02653) *
A0F 0685A 1E000020 (02654)          NOP(0)
A10 0685C 20000020 (02655)          NOP(0)
A11 0685E 22000020 (02656)          NOP(0)
A12 06860 24000020 (02657)          NOP(0)
A13 06862 26000020 (02658)          NOP(0)
A14 06864 28000020 (02659)          NOP(0)
A15 06866 2A000020 (02660)          NOP(0)
A16 06868 2C000020 (02661)          NOP(0)
A17 0686A 2E000020 (02662)          NOP(0)
A18 0686C 30000020 (02663)          NOP(0)
                   (02664) *
                   (02665) *!!!!!NEXT INSTR MODIFIED BY OUTPUT PGM!!!!!!!
                   (02666) *
          00006BFE (02667) APCIS$M = #L
A19 069FE 32400000 (02668)          LOAD(BR0,M$$)   ;LOAD PITCH
                   (02669) *
A1A 06900 34090029 (02670)          ADD8(BR0,BR0)   ;DOUBLE FOR H*ORD ADDR
A1B 06902 36500000 (02671)          LOAD(BP1,[0])   ;Y BASE = Y(0)
A1C 06904 38000000 (02672)          LOAD(BR2,M$$)   ;YBS
A1D 06906 3A120000 (02673)          SUB(BR1,M$$)    ;SUBTR SPACING
A1E 06908 3C190021 (02674)          SUBB(BR1,BR0)   ;Y(0-2-2*PITCH)
                   (02675) *
A1F 0690A 3E401009 (02676)          LOAD(BR0,[1])           ;X(0)=U BASE
A20 0690C 40600020 (02677)          LOAD(BR2,M$$)           ;U SIZE-1
A21 0690E 42020000 (02678)          SUB(BR0,M$$)            ;U SPACING
                   (02679) *INPUT LOOP
A22 06910 449A9006 (02680) #1       ADD(BR1,[8],TF)         ;Y(N-PITCH)
A23 06912 468A9002 (02681)          ADD(BR2,[9],TF)         ;X(N)
A24 06914 48890011 (02682)          MOVB(BR0,BR0,TF)        ;DUMMY
A25 06916 4A2922B1 (02683)          SUBL(BR2,1),JUMPP(#1)   ;COUNT SAMPLE N AND LOOP
A26 06918 4C200231 (02684)          CLEAR(PI)
A27 0691A 4E000220 (02685)          NOP(0)
                   (02686) *OUTPUT PROGRAM
A28 0691C 52300332 (02687) APCIS$3 SET(RA)
                   (02688) *
                   (02689) * GEN BUS1 ADDR OF INPUT PGM INSTR TO
                   (02690) * BE MODIFIED (LEFT H*ORD)
                   (02691) *
A29 0691E 537268FF (02692)          LOAD(BW3,APCIS$M+1(1),TE)
                   (02693) *
                   (02694) * WRITE MODIFIED INSTR TO PSEUDO MEM
                   (02695) *
```

APS-APCIS INVERSE APC FUNCTION (VIAPC)

```
A2A 06920 54F3FFC0 (02696)          LOAD(BW3,APSSLA(1),TF)
                   (02697) *
                   (02698) *
A2B 06922 56400010 (02699)          LOAD(BW0,[0])    ;Y ARRAY
A2C 06924 58500000 (02700)          LOAD(BW1,MS$)    ;Y SIZE-1
A2D 06926 5A020000 (02701)          SUB(BW0,M$$)     ;MINUS SPACING
A2E 06928 5C110039 (02702)          ADDL(BW1,1)      ;SIZE Y
A2F 0692A 5E210010 (02703)          MOVB(BW2,BW0)    ;GET Y BASE -2
A30 0692C 60210022 (02704)          SUBB(BW2,BW1)    ;Y-YSIZE-2
A31 0692E 62210022 (02705)          SUBB(BW2,BW1)    ;Y-2*YSIZE-2
A32 06930 64110031 (02706)          SUBL(BW1,1)                ;YSIZE-1
                   (02707) *  SPACING FOR Y MUST BE 2!!!!!
                   (02708) *OUTPUT LOOP
A33 06932 668A8010 (02709) #2       ADD(BW0,[8],TF)  ;Y(N)
A34 06934 68AAB002 (02710)          ADD(BW2,[8],TF)            ;Y(N-YSIZE)
A35 06936 6A1133B1 (02711)          SUBL(BW1,1),JUMPP(#2)     ;COUNT IT AND LOOP
           0006934 (02712)          APCIS$A=#C                ;CHAIN ANCHOR
A36 06938 6C200300 (02713)          CLEAR(RO)
A37 0693A 6E000020 (02714)          NOP(0)
    0693C          (02715)          END
    0693C 00000000 (02716) APCS$I   DATA 7F'0.0'
     ...
          00000075 (02717) APCIS$Z=#L-APCIS$
```

```
                APU3-PTAP(V,A,U,B,V,C,W)       COMPUTE PITCH AND TAP

                    (02718) * APU3-PTAP(V,A,U,B,V,C,W)       COMPUTE PITCH AND TAP
                    (02719) *                                AND DO PITCH REMOVAL
                    (02720) *
                    (02721) *       V IS "TBPR" (BASEBAND WITH PITCH REMOVED)
                    (02722) *       A IS "TPTC" (PITCH)
                    (02723) *       U IS "TBPCP" (AUTOCORR OF B.B. EXCIT. - PITCH CALC PART)
                    (02724) *       B IS "TQTAP"(QUANTIZED TAP)
                    (02725) *       V IS "TBEQ" (DOWNSAMPLED B.B. EXCITATION)
                    (02726) *       C IS "TCTAP"(LNG FXD CODED TAP - RGHT JUSTIFIED IN LEFT H.ORD)
                    (02727) *       W IS "TPAC" (AUTOCORR. OF B.B. EXCIT. - WHOLE THING)
                    (02728) *
                    (02729) *                DO SMAX(U) TO GET U(MAX) AND MAX(=PITCH-S)
                    (02730) *                ADD 5 TO GET PITCH
                    (02731) *                DO SDIV (U(MAX)/W(0)) TO GET TAP
                    (02732) *                DO QTZT TO GET QUANTIZED AND CODED TAP
                    (02733) *                DO VSMAD(V) TO ACCOMPLISH PITCH REMOVAL
                    (02734) *
                    (02735) *     BINDS TO APS3-PTS$
                    (02736) *
                    (02737) *       INPUT STREAM: U(0),...,U(UBS-1),[FWI],(2**-15),"MODIFIED INSTR",
                    (02738) *              W(0),(2**-15),TAPQTH(0),TAPQVL(0),TAPQTH(1),TAPQVL(1),...,
                    (02739) *              TAPQTH(15),TAPQVL(15),[FWI],
                    (02740) *              V(0-PITCH),V(0),V(1-PITCH),V(1),...,
                    (02741) *              V(VBS-1-PITCH),V(VBS-1),[FI]
                    (02742) *
                    (02743) *       OUTPUT STREAM: A,INTEGER PITCH,"MODIFIED INSTR",
                    (02744) *              B,C,V(0),...,V(VBS-1),[EOI]
                    (02745) *
                    (02746) *       WHERE:
                    (02747) *              "TAPQTH" ARE THE TAP QUANT THRESHOLDS (INTERNAL TO APS PGM
                    (02748) *              "TAPQVL" ARE THE TAP QUANT VALUES (INTERNAL TO APS PGM)
                    (02749) *              "INTEGER PITCH" IS USED BY APS OUTPUT PGM TO MODIFY
                    (02750) *                     APS INPUT PGM, SO IT CAN GENERATE
                    (02751) *                     V(N-PITCH) ADDRESSES.
                    (02752) *              "MODIFIED INSTR" IS APS INSTR TO BE MODIFIED.
                    (02753) *                     IT IS READ FROM BUS1, AND WRITTEN
                    (02754) *                     TO APS PSEUDO MEMORY.
                    (02755) *
                    (02756) *
                    (02757) *
                    (02758) *
                    (02759)       EVEN                        START ON WORD BOUNDARY
     3694A 0002     (02760)       DATA  PTU$SA                START ADDRESS
     3694B 0072     (02761)       DATA  PTU$SZ                SIZE
                    (02762) *
                    (02763) PTU$ BEGIN APU(PTU)             START OF APU MODULE
           00020000 (02764)       BASE                        SET START ADDRESS
                    (02765) *
                    (02766) *
                    (02767) *
                    (02768) *       DO SMAX ON U ARRAY (TERMINATE INPUT ON [FWI])
                    (02769) *
  A0F A694C 00C01682 (02770) PTU$SA NOP \ K(1)                  ---          \ R=1
```

```
A01 0694E 09F72894 (02771)        MOV(IQA,A7)\ MOV(R,A4)       A7=U0          \ A4=1
A02 06950 08152896 (02772)        MOV(ZERO,A6)\ MOV(R,A6)      A6=0           \ A5=1
A03 06952 16A016A0 (02773)        K(2)                         R=2            \ R=2
A04 06954 000039F7 (02774)        NOP \ MOV(IQA,A7)            ---            \ A7=U1
A05 06956 0895C2F5 (02775)        MOV(R,A5) \ MOV(A5),R(A7)      A5=2         \ A5=2, R=U1
A06 06958 91160010 (02776)        JUMPS(PTU$1,FWI)
                   (02777) *
A07 0695A 08F00000 (02778) PTU$P     MOV(IQA,A0) \ NOP           A0=U0          \ ---
A08 0695C 60F40897 (02779)        MOV(A4),MAX(A7,A0) \ MOV(R,A7)  A4=LOC, R=MAX \ A7=MAX
A09 0695E B11F0041 (02780)        CALLS(PTU$4,T2)               ---              \ IF NEW MAX
A0A 06960 000044A0 (02781)        NOP \ ADD(A5,A4)             ---             \ R=LOC
A0B 06962 91160015 (02782)        JUMPS(PTU$2,FWI)               EXIT IF UBS IS ODD
A0C 06964 0897F8F0 (02783)        MOV(R,A7) \ MOV(IQA,A0)      A7=MAX          \ A0=U0
A0D 06966 B11E003E (02784)        CALLS(PTU$3,T1)               IF NEW MAX     \ ---
A0E 06968 44A060F4 (02785)        ADD(A5,A4) \ MOV(A4),MAX(A7,A0)  R=LOC     \ A4=LOC,R=MAX
A0F 0696A 90160007 (02786)        JUMPC(PTU$0,FWI)                 EXIT IF UBS IS EVEN
                   (02787) *
                   (02788) *                                    EXIT IF LOOP BOTTOM
A10 0695C 02E00000 (02789) PTU$1  R(A7) \ NOP                   R=MAX          \ ---
A11 0696E 08910898 (02790)        MOV(R,A1) \ MOV(R,EXO) .     A1=MAX         \ EXO=MAX
A12 06970 B11F0041 (02791)        CALLS(PTU$4,T2)               ---              \ IF NEW MAX
A13 06972 08520000 (02792)        MOV(EXI,A2) \ NOP            A2=MAX          \ ---
A14 06974 10000019 (02793)        JUMP(PTU$5)                   TO COMMON CLEAN UP
                   (02794) *
                   (02795) *                                    IF EXIT LOOP MIDDLE
A15 06976 000002E0 (02796) PTU$2  NOP \ R(A7)                   ---              \ R=MAX
A16 06978 08910898 (02797)        MOV(R,A1) \ MOV(R,EXO)       A1=MAX         \ EXO=MAX
A17 0697A B11E003E (02798)        CALLS(PTU$3,T1)               IF NEW MAX     \ ---
A18 0697C 08520000 (02799)        MOV(EXI,A2)  \ NOP           A2=MAX          \ ---
                   (02800) *
                   (02801) *                                    COMMON CLEAN UP
A19 0697E 622002C2 (02802) PTU$5  MAX(A1,A2) \ R(A6)            R=U-MAX        \ R=LOC(U-MAX)
A1A 06982 089F3000 (02803)        MOV(R,M7) \ NOP              M7 = U(MAX)    \ ---
A1B 06984 911E001F (02804)        JUMPS(PTU$6,T1)               IF FROM RIGHT BOARD
A1C 06986 02C00000 (02805)        R(A6) \ NOP                  R=LOC(U-MAX) \ ---
A1D 06988 08950000 (02806)        MOV(R,A5) \ NOP              A5 = MAX \ ---
A1E 0698A 10000021 (02807)          JUMP(PTU$7)
A1F 0698C 00030898 (02808) PTU$6  NOP \ MOV(R,EXO)               ---            \ EXO = MAX
A20 0698E 08550000 (02809)          MOV(EXI,A5) \ NOP          A5 = MAX \ ---
                   (02810) *
                   (02811) *       ADD 5 TO MAX TO GET PITCH
                   (02812) *
A21 06992 16C00000 (02813) PTU$7    K(4) \ NOP                 R = 4  \ ---
A22 06994 16940000 (02814)          MOV(A4),K(1) \ NOP         A4 = 4, R = 1  \ ---
A23 06996 44B10000 (02815)          MOV(A1),ADD(A5,A4) \ NOP   A1 = 1, R = 4+MAX \ ---
A24 06998 41B50000 (02816)          MOV(A5),ADD(A5,A1) \ NOP   A5 = 4+MAX, R = 5+MAX \ ---
A25 0699A 089C0000 (02817)          MOV(R,OQ) \ NOP            OQ = SA = 5+MAX = PITCH
                   (02818) *
                   (02819) *       GENERATE INTEGER PITCH:  (MULT BY (2**-15), THEN ALIGN)
                   (02820) *       (RESTART APS INPUT PGM BY CLEARING "WI")
                   (02821) *
A26 0699C 20372037 (02822)          CLEAR(WI)
A27 0699E 08E80000 (02823)          MOV(IQA,M0) \ NOP                ; M0 <= (2**-15)
```

```
A29 0699C 083C0003 (02824)          MOV(R,M4) \ NOP                 ; M4 <= PITCH
A29 0699E 94000000 (02825)          MUL(M3,M4) \ NOP                ; P <= PITCH*(2**-15)
A2A 069A0 08B60000 (02826)          MOV(P,A6) \ NOP                 ; A6 <= PITCH*(2**-15)
A2B 069A2 3AC00C00 (02827)          ALIGN(A6) \ NOP                 ; R <= FXD PITCH (RGHT JUST IN L H
                   (02828) *
                   (02829) *        SEND TO BUS1 TO MODIFY BUS1 COPY OF APS INSTR.
                   (02830) *
A2C 069A4 089C0002 (02831)          MOV(R,OQ) \ NOP                 ; OQ <= INTEGER PITCH
                   (02832) *
                   (02833) *        WAIT TIL BUS1 COPY IS MODIFIED, THEN READ
                   (02834) *        IT AND WRITE TO APS PSEUDO MEMORY.
                   (02835) *        (ALSO CLEAR WI SO APS INPUT PGM WILL CONTINUE.)
                   (02836) *        (IT WAS WAITING FOR WRITE TO COMPLETE.)
                   (02837) *
A2D 069A6 00000000 (02838)          NOP
A2E 069A8 00000000 (02839)          NOP
A2F 069AA 00000000 (02840) PTUSW1   NOP
A30 069AC 901B002F (02841)          JUMPC(PTUSW1,OQE)               ; LOOP WHILE OUTPUT QUEUE NOT EMPT
A31 069AE 00000000 (02842)          NOP
A32 069B0 00000000 (02843)          NOP
A33 069B2 00000000 (02844)          NOP
A34 069B4 00000000 (02845)          NOP
A35 069B6 00000000 (02846)          NOP
A36 069B8 20372037 (02847)          CLEAR(WI)
A37 069BA 00000000 (02848)          NOP
                   (02849) *
A38 069BC 09FC0000 (02850)          MOV(IQA,OQ) \ NOP               ; THEN READ MODIFIED APS INSTR
                   (02851) *                                        ; AND WRITE TO APS PSEUDO MEM.
                   (02852) *
                   (02853) *        WAIT TIL APS COPY IS WRITTEN
                   (02854) *
A39 069BE 00000000 (02855) PTUSW2   NOP
A3A 069C0 901B0039 (02856)          JUMPC(PTUSW2,OQE)
                   (02857) *
                   (02858) *
                   (02859) *        SET UP REGS FOR DIVIDE: U(MAX)/U(0)
                   (02860) *        M7 ALREADY = U(MAX)
                   (02861) *        GET W(0) FROM IQ
                   (02862) *
A3B 069C2 08CE0000 (02863)          MOV(IQ,M6) \ NOP            M6 = W(0) \ ---
A3C 069C4 09F00000 (02864)          MOV(IQA,A7) \ NOP           A7 = W(0) \ ---
                   (02865) *
A3D 069C6 10000044 (02866)          JUMP(PTUS8)                 GO DO DIVIDE TO GET TAP
                   (02867) *
                   (02868) *
                   (02869) *                                   SUBROUTINES
A3E 069C8 02800200 (02870) PTUS3    R(A4) \ NOP                  R=LOC          \ ---
A3F 069CA 08960000 (02871)          MOV(P,A6) \ NOP             A6=LOC         \ ---
A40 069CC A000A000 (02872)          RETURN
                   (02873) *
A41 069CE 00000280 (02874) PTUS4    NOP \ R(A4)                  ---            \ R=LOC
A42 069D0 00000896 (02875)          NOP \ MOV(P,A6)             ---            \ A6=LOC
A43 069D2 A000A000 (02876)          RETURN
```

```
:E   66:  [BBN-TENEXD]<MAP>BBN388.MSD.61, 3@-Dec-79 16:14:24, Ed: KFIELD
          APU3-PTAP(V,A,U,B,V,C,W)        COMPUTE PITCH AND TAP

                    (@2877) *
                    (@2878) *
                    (@2879) *       DO SDIV:  DIVIDE B/C
                    (@2880) *       EXPECTS (IN LEFT ADM) B IN M7
                    (@2881) *                             C IN M6
                    (@2882) *                             C IN A@
                    (@2883) *
A44 869D4 16A@@@@@  (@2884) PTU$8   K(2) \ NOP              R=2
A45 869D6 @@96@@@@  (@2885)         MOV(R,A6) \ NOP         A6=2
A46 869D8 2E@@@@@@  (@2886)         RCP(A@) \ NOP           R1=1/C+DEL(=F@)(= 1/C + &)
A47 869DA C@@8@@@@  (@2887)         MOV(R,M@) \ NOP         M@=F@
A48 869DC 84@@@@@@  (@2888)         MUL(M@,M6) \ NOP        P1=F@*C
A49 869DE @@B1@@@@  (@2889)         MOV(P,A1) \ NOP         A1=F@*C
A4A 869E@ 49C@@@@@  (@289@)         SUB(A6,A1) \ NOP        R2=2-F@*C (= 1 - C&)
A4B 869E2 @@8C@@@@  (@2891)         MOV(R,M4) \ NOP         M4=R2
A4C 869E4 84@@@@@@  (@2892)         MUL(M@,M4) \ NOP        P2=F@*R2(=F1)(= 1/C - (&**2)C )
A4D 869E6 @@A8@@@@  (@2893)         MOV(P,M@) \ NOP         M@=F1
A4E 869E8 844@@@@@  (@2894)         MUL(M@,M6) \ NOP        P3=F1*C
A4F 869EA @@B1@@@@  (@2895)         MOV(P,A1) \ NOP         A1=P3
A5@ 869EC 49C@@@@@  (@2896)         SUB(A6,A1) \ NOP        R3=2-F1*C
A51 869EE @@8C@@@@  (@2897)         MOV(R,M4) \ NOP         M4=R3
A52 869F@ 84@@@@@@  (@2898)           MUL(M@,M4) \ NOP      P4=F1*(2-F1*C)(=F2)(= 1/C - (&**4)(C**3)) (=1/C)
A53 869F2 @@A8@@@@  (@2899)         MOV(P,M@) \ NOP         M@=F2
A54 869F4 846@@@@@  (@29@@)           MUL(M@,M7) \ NOP      P5=F2*B (=B/C)
A55 869F6 @@B@@@@@  (@29@1)           MOV(P,A@) \ NOP       A@ = TAP  \ ---
                    (@29@2) *
                    (@29@3) *
                    (@29@4) *       NOW DO QUANTIZE AND CODE OF TAP (TAP IS IN A@)
                    (@29@5) *       USE (2**-15) AS COUNTER INCREMENT
                    (@29@6) *
                    (@29@7) *
A56 869F8 @817@@@@  (@29@8)         MOV(ZERO,A7) \ NOP
A57 869FA @8F6@@@@  (@29@9)         MOV(IQA,A6) \ NOP               ; A6 <= 2**-15 \ ---
A58 869FC 4EE@@@@@  (@291@)         SUB(A7,A6) \ NOP                ; A7 <= -(2**-15) (SO 1ST ADD =@)
A59 869FE @897@@@@  (@2911)         MOV(R,A7) \ NOP                 ; R <= -(2**-15)
                    (@2912) *
A5A 86A@@ C8F1@@@@  (@2913) PTU$9   MOV(IQA,A1) \ NOP               ; A1 <= NEXT THRESH  \ ---
A5B 86AC2 47C@@8F@  (@2914)         ADD(A6,A7) \ MOV(IQA,A@)                ; R<=INCR'D CODE  \ A@<=CURRENT QNT
                    (@2915) *
A5C 86A@4 491702@@  (@2916)         MOV(A7),SUB(A@,A1) \ R(A@)              ; A7<=CODE, R<=TAP-THRESH (T1<=SIGN
A5D 86A@6 @8@@@@@@  (@2917)         MOV(R,NULL) \ NOP               ; MAKE SURE PREV INSTR COMPLETED
A5E 86A@B 91160@63  (@2918)         JUMPS(PTU$11,FWI)               ; JUMP OUT IF NO MORE
A5F 86A@A 911E@@61  (@2919)         JUMPS(PTU$10,T1)                ; T1 SET IF TAP=<THRESH (I.E. DONE)
A6@ 86A@C 1@@@@@5A  (@292@)         JUMP(PTU$9)                     ; LOOP IF TAP .GT. CURR. THRESH.
                    (@2921) *
A61 86A@E @8E@@@@@  (@2922) PTU$1@  MOV(IQA,NULL) \ NOP             ; FLUSH REMAINING INPUT
A62 86A1@ 9@16@@61  (@2923)         JUMPC(PTU$1@,FWI)
                    (@2924) *
A63 86A12 3AE@@89C  (@2925) PTU$11  ALIGN(A7) \ MOV(R,OQ)           ; R<=ALIGNED CODE \ OQ<=QUANT VALUE
A64 86A14 @89C@@@@  (@2926)         MOV(R,OQ) \ NOP                 ; OQ <= ALIGNED CODED TAP
                    (@2927) *
                    (@2928) *
                    (@2929) *
```

```
                    (02930) *       NOW DO PITCH REMOVAL (QUANTIZED TAP IS IN RIGHT ADM'S R REG)
                    (02931) *       (RESTART APS INPUT PROGRAM BY CLEARING 'WI' FLAG)
                    (02932) *
                    (02933) *
  A65 06A16 20372037 (02934)        CLEAR(WI)
  A66 06A18 00000898 (02935)        NOP \ MOV(R,EXO)
  A67 06A1A 08480000 (02936)        MOV(EXI,M0) \ NOP                ; M0 <= QTAP
                    (02937) *
  A68 06A1C 08EC0000 (02938) PTU$12 MOV(IQA,M4) \ NOP                ; M4 <= V(N-PITCH)
  A69 06A1E 84000000 (02939)        MUL(M0,M4) \ NOP                 ; P <= QTAP * V(N-PITCH)
  A6A 06A20 08F10000 (02940)        MOV(IQA,A1) \ NOP                ; A1 <= V(N)
  A6B 06A22 08B50000 (02941)        MOV(P,A5) \ NOP                  ; A5 <= QTAP * V(N-PITCH)
  A6C 06A24 4D200000 (02942)        SUB(A1,A5) \ NOP                 ; R <= V(N) - QTAP*V(N-PITCH)
  A6D 06A26 089C0000 (02943)        MOV(R,OQ) \ NOP                  ; OQ<= V(N) - QTAP*V(N-PITCH)
  A6E 06A28 9C1D0068 (02944)        JUMPC(PTU$12,FI)                 ; DO NEXT N IF MORE
                    (02945) *
                    (02946) *
                    (02947) *
                    (02948) *       DONE
                    (02949) *       DO CLEAN UP
                    (02950) *
  A6F 06A2A 20322032 (02951)        CLEAR(RA)
  A70 06A2C 00000000 (02952)        NOP
  A71 06A2E 1C000000 (02953)        JUMP(0)
                    (02954) *
            00000072 (02955)        PTU$SZ=#A-PTU$SA
      06A30         (02956)        END     PTU$SZ
                    (02957)        EVEN
```

```
                    (02958) ;  APS3-PTAP(Y,A,U,B,V,C,W)
                    (02959) ;
                    (02960) ;
                    (02961) ;       Y IS 'TBPR' (BASEBAND WITH PITCH REMOVED)
                    (02962) ;       A IS 'TPTC' (PITCH)
                    (02963) ;       U IS 'TBPCP' (AUTOCORR OF B.B. EXCIT. - PITCH CALC PART)
                    (02964) ;       B IS 'TQTAP'(QUANTIZED TAP)
                    (02965) ;       V IS 'TBE00' (DOWNSAMPLED B.B. EXCITATION)
                    (02966) ;       C IS 'TCTAP'(LNG FXD CODED TAP - RGHT JUSTIFIED IN LEFT HWORD)
                    (02967) ;       W IS 'TPAC' (AUTOCORR. OF B.B. EXCIT. - WHOLE THING)
                    (02968) ;
                    (02969) ;                 DO SMAX(U) TO GET U(MAX) AND MAX(=PITCH-5)
                    (02970) ;                 ADD 5 TO GET PITCH
                    (02971) ;                 DO SDIV (U(MAX)/W(0)) TO GET TAP
                    (02972) ;                 DO QTZT TO GET QUANTIZED AND CODED TAP
                    (02973) ;                 DO VSMAD(V) TO ACCOMPLISH PITCH REMOVAL
                    (02974) ;
                    (02975) ;
                    (02976) ;       INPUT STREAM: U(0),...,U(UBS-1),[FWI],(2**-15),'MODIFIED INSTR',
                    (02977) ;               W(0),(2**-15),TAPQTH(0),TAPQVL(0),TAPQTH(1),TAPQVL(1),...,
                    (02978) ;               TAPQTH(15),TAPQVL(15),[FWI],
                    (02979) ;               V(0-PITCH),V(0),V(1-PITCH),V(1),...,
                    (02980) ;               V(VBS-1-PITCH),V(VBS-1),[FI]
                    (02981) ;
                    (02982) ;       OUTPUT STREAM: A,INTEGER PITCH,'MODIFIED INSTR',
                    (02983) ;               B,C,Y(0),...,Y(YBS-1),[EO]
                    (02984) ;
                    (02985) ;       WHERE:
                    (02986) ;               'TAPQTH' ARE THE TAP QUANT THRESHOLDS (INTERNAL TO APS PGM
                    (02987) ;               'TAPQVL' ARE THE TAP QUANT VALUES (INTERNAL TO APS PGM)
                    (02988) ;               'INTEGER PITCH' IS USED BY APS OUTPUT PGM TO MODIFY
                    (02989) ;                       APS INPUT PGM, SO IT CAN GENERATE
                    (02990) ;                       V(N-PITCH) ADDRESSES.
                    (02991) ;               'MODIFIED INSTR' IS APS INSTR TO BE MODIFIED.
                    (02992) ;                       IT IS READ FROM BUS1, AND WRITTEN
                    (02993) ;                       TO APS PSEUDO MEMORY.
                    (02994) ;
                    (02995) ;
                    (02996)         EVEN
06A30 00006AAC      (02997)         ADDR    PTS$I                   ; CONSTR INSTR BLK
06A32 00006A8E      (02998)         ADDR    PTS$ + 2*PTS$S          ; SCALAR BLK
06A34 0003          (02999)         DATA    3                       ; NUMBER OF SCALARS
06A35 0084          (03000)         DATA    PTS$Z                   ; MODULE SIZE
06A36 00006AA2      (03001)         ADDR    PTS$A                   ; PTR TO CHAIN ANCHOR
                    (03002)         EVEN
                    (03003) ;
                    (03004) ;
                    (03005) PTS$    BEGIN   APS(PTS)                ; START OF APS MODULE
                    (03006) ;
                    (03007) ;
                    (03008) ;       INPUT PROGRAM
                    (03009) ;       REGISTER USAGE:
                    (03010) ;               BR0: BUFFER 'U', TAPQTH, BUFFER V(N-PITCH) ELEM ADDRS.
```

```
                    (03011) *               BR1: TAPQVL, BUFFER V(N) ELEM ADDRESSES
                    (03012) *               BR2: BUFFER SIZES - 1
                    (03013) *               BR3: SCRATCH
                    (03014) *
                    (03015) *
  A00 06A38 00202A40 (03016)        JSN(PTS$O,P2)                    ; SET OUTPUT PC
  A01 06A3A 02300030 (03017)        SET(RO)                          ; RUN OUTPUT PGM
                    (03018) *
                    (03019) *
                    (03020) *       FIRST GEN U(0)-U(UBS-1) ADDRESSES, THEN [FWI]
                    (03021) *
  A02 06A3C 04401000 (03022)        LOAD(BR0,[1])                    ; LOAD U BASE ADDR
  A03 06A3E 06600000 (03023)        LOAD(BR2,M$$)                    ; LOAD UBS-1
  A04 06A40 08020000 (03024)        SUB(BR0,M$$)                     ; SUBTRACT SPACING
                    (03025) *
  A05 06A42 0A8A9006 (03026) #1     ADD(BR0,[9],TF)                  ; GEN U-ELEM ADDR
  A06 06A44 0C2905B1 (03027)        SUBL(BR2,1),JUMPP(#1)            ; DO UBS ELEMENTS
                    (03028) *
  A07 06A46 0E3000037 (03029)       SET(WI)                          ; SET WI, WAIT TIL APU PGM CLEARS
  A08 06A48 10000020 (03030)        NOP(0)
                    (03031) *
                    (03032) *       GEN (2**-15)
                    (03033) *
  A09 06A4A 12F203CE (03034)        LOAD(BR3,SVT$UN1(1),TF)          ; (2**-15)
                    (03035) *
                    (03036) *       WAIT FOR APU PGM TO SIGNAL THAT BUS1
                    (03037) *       COPY OF MODIFIED INSTR HAS BEEN
                    (03038) *       WRITTEN.
                    (03039) *
  A0A 06A4C 14300037 (03040)        SET(WI)
  A0B 06A4E 16000020 (03041)        NOP(0)
  A0C 06A50 18000020 (03042)        NOP(0)
                    (03043) *
                    (03044) *       GEN "MODIFIED INSTR" ADDRESS.
                    (03045) *
                    (03046) *
  A0D 06A52 1AF26A78 (03047)        LOAD(BR3,PTS$1(1),TF)            ; MODIFIED INSTR (BUS1 ADDR)
                    (03048) *
                    (03049) *       GEN W(0) ADDRESS
                    (03050) *
  A0E 06A54 1CF03E12 (03051)        LOAD(BR3,[3],TF)                 ; W(0) ADDRESS
  A0F 06A56 1E600000 (03052)        LOAD(BR2,M$$)                    ; DUMMY LOAD
  A10 06A58 20600000 (03053)        LOAD(BR2,M$$)                    ; DUMMY LOAD
                    (03054) *
                    (03055) *
                    (03056) *       NOW GEN (2**-15) AND TAPQTH & TAPQVL ADDRESSES, THEN [FWI]
                    (03057) *
  A11 06A5A 22F203CE (03058)        LOAD(BR3,SVT$UN1(1),TF)          ; (2**-15)
                    (03059) *
  A12 06A5C 24426ABC (03060)        LOAD(BR0,TAPQTH(1))              ; LOAD THRESH. TABLE BASE
  A13 06A5E 2660000F (03061)        LOAD(BR2,15)                     ; TABLE SIZE -1
  A14 06A60 28020002 (03062)        SUB(BR0,2)                       ; SUBTR SPACING
                    (03063) *
```

```
A15 06A62 2A526ADC (03064)          LOAD(BR1,TAPQVL(1))              ; LOAD VALUE TABLE BASE
A16 06A64 2C120002 (03065)          SUB(BR1,2)                       ; SUBTR SPACING
                   (03066) *
A17 06A66 2E8A0002 (03067) #2       ADD(BR0,2,TF)                    ; GEN THRESH ADDR
A18 06A68 309A0002 (03068)          ADD(BR1,2,TF)                    ; GEN VALUE ADDR
A19 06A6A 322917B1 (03069)          SUBL(BR2,1),JUMPP(#2)            ; DO 16 ELEMENTS
                   (03070) *
A1A 06A6C 34300037 (03071)          SET(WI)                          ; SET WI, WAIT TIL APU PGM CLRS IT
A1B 06A6E 36000020 (03072)          NOP(0)
A1C 06A70 38000020 (03073)          NOP(0)
                   (03074) *
                   (03075) *
                   (03076) *       NOW GEN V(N-PITCH) AND V(N) ADDRESSES, THEN [FI]
                   (03077) *       (BY NOW, INSTRUCTION WHICH SUBTRACTS PITCH FROM
                   (03078) *       V BUFFER ADDR HAS BEEN MODIFIED BY OUTPUT PGM.)
                   (03079) *
A1D 06A72 3A502C1E (03080)          LOAD(BR1,[2])                    ; LOAD V BASE ADDRESS
A1E 06A74 3C600000 (03081)          LOAD(BR2,M$$)                    ; LOAD VBS-1
A1F 06A76 3E120000 (03082)          SUB(BR1,M$$)                     ; SUBTR SPACING
                   (03083) *
                   (03084) *
                   (03085) *
                   (03086) *
                   (03087) *       !!!!!!!!!!!NEXT INSTR MODIFIED BY OUTPUT PGM!!!!!!!!!!!
           00006A78 (03088) PTS$1 = #L
A20 06A78 4F700000 (03089)          LOAD(BR3,M$$)                    ; LOAD PITCH
                   (03090) *
                   (03091) *
                   (03092) *
A21 06A7A 4239002F (03093)          ADDB(BR3,BR3)                    ; DOUBLE PITCH FOR HWORD ADDRESSIN
A22 06A7C 44090013 (03094)          MOVB(BR0,BR1)                    ; V BASE MINUS SPACING
A23 06A7E 46090027 (03095)          SUBB(BR0,BR3)                    ; V(0-PITCH) MINUS SPACING
                   (03096) *
A24 06A80 488AA00E (03097) #3       ADD(BR0,[10],TF)                 ; GEN V(N-PITCH)
A25 06A82 4A9AA002 (03098)          ADD(BR1,[10],TF)                 ; GEN V(N)
A26 06A84 4C2924B1 (03099)          SUBL(BR2,1),JUMPP(#3)            ; DO VBS ELEMENTS
                   (03100) *
A27 06A86 4E200031 (03101)          CLEAR(RI)                        ; HALT INPUT
A28 06A88 50000020 (03102)          NOP(0)
A29 06A8A 52000020 (03103)          NOP(0)
                   (03104) *
                   (03105) *
                   (03106) *
                   (03107) *       OUTPUT PROGRAM
                   (03108) *       REGISTER USAGE:
                   (03109) *               BW0: A SCALAR ADDR, BUFFER Y ELEM ADDR
                   (03110) *               BW1: SCALAR B ADDR
                   (03111) *               BW2: SCALAR C ADDR, BUFFER Y SIZE-1
                   (03112) *               BW3: SCRATCH
                   (03113) *
                   (03114) *
A2A 06A8C 54300032 (03115) PTS$0    SET(RA)                          ; TURN ON APU
                   (03116) *
```

```
                    (03117) *       FIRST, GET SCALAR A,B,C ADDRESSES, GEN A
                    (03118) *
  A2B 06A8E 56C20000 (03119) PTS$S   LOAD(BW0,M$$(1),TF)             ; GEN SA ADDR
  A2C 06A90 58520000 (03120)         LOAD(BW1,M$$(1))               ; GET SB ADDR
  A2D 06A92 5A620000 (03121)         LOAD(BW2,M$$(1))               ; GET SC ADDR
                    (03122) *
                    (03123) *
                    (03124) *       NOW, GEN BUS1 ADDRESS OF INPUT PGM INSTR TO BE MODIFIED
                    (03125) *       ("INTEGER PITCH" WRITTEN INTO RIGHT HW OF INSTR.)
                    (03126) *
  A2E 06A94 5D726A79 (03127)         LOAD(BW3,PTS$1+1(1),TE)        ; SEND BUS1 ADDR OF R HW OF INSTR
                    (03128) *
                    (03129) *
                    (03130) *       NOW WRITE MODIFIED INSTRUCTION TO APS MEMORY
                    (03131) *       VIA BUS1 PSEUDO MEMORY.
                    (03132) *
  A2F 06A96 5EF3FFC0 (03133)         LOAD(BW3,APS$LA(1),TF)
                    (03134) *
                    (03135) *
                    (03136) *       GEN B,C SCALAR ADDRS, THEN Y(0)-Y(YBS-1), THEN [EO]
                    (03137) *
  A30 06A98 60910012 (03138)         MOVB(BW1,BW1,TF)               ; GEN SB ADDR
  A31 06A9A 63210014 (03139)         MOVB(BW2,BW2,TE)               ; GEN SC ADDR (JUST L HW)
                    (03140) *
  A32 06A9C 64400F1A (03141)         LOAD(BW0,[0])                  ; LOAD Y BASE ADDR
  A33 06A9E 66600000 (03142)         LOAD(BW2,M$$)                  ; LOAD YBS-1
  A34 06AA0 68020000 (03143)         SUB(BW0,M$$)                   ; SUBTR SPACING
                    (03144) *
  A35 06AA2 6A8A8006 (03145) $4      ADD(BW0,[0],TF)                ; GEN Y-ELEM ADDR
  A36 06AA4 6C2135B1 (03146)         SUBL(BW2,1),JUMPP($4)          ; DO YBS ELEMS
                    (03147) *
                    (03148) *
  A37 06AA6 6E200030 (03149)         CLEAR(RO)                      ; HALT OUTPUT
  A38 06AA8 70000020 (03150)         NOP(0)
  A39 06AAA 72000020 (03151)         NOP(0)
                    (03152) *
            00006AA2 (03153) PTS$A=$C                                ; CHAIN ANCHOR
                    (03154) *
      06AAC         (03155)         END     $A-1
                    (03156) *
      06AAC 00000000 (03157) PTS$I   DATA    8F'0.0'
       ...
            00000084 (03158) PTS$Z=$L-PTS$                           ; MODULE SIZE
                    (03159) *
                    (03160) TAPQTH:
      06ABC 0EABF9C0 (03161)         DATA    0.1146233E+00
      06ABE 1CF688C0 (03162)         DATA    0.2262736E+00
      06AC0 2A881CC0 (03163)         DATA    0.3322788E+00
      06AC2 371ACDC0 (03164)         DATA    0.4305055E+00
      06AC4 427EC5C0 (03165)         DATA    0.5194938E+00
      06AC6 4C9AFE40 (03166)         DATA    0.5984800E+00
      06AC8 556AE6C0 (03167)         DATA    0.6673249E+00
      06ACA 5CFA3940 (03168)         DATA    0.7263862E+00
```

```
06ACC 6360@AC@ (@3169)          DATA    @.7763685E+@@
06ACE 68BA2E4@ (@3170)          DATA    @.8181818E+@@
06AD@ 6D295DC@ (@3171)          DATA    @.0520249E+@@
06AD2 7@CE604@ (@3172)          DATA    @.8012981E+@@
06AD4 73C02EC@ (@3173)          DATA    @.9045466E+@@
06AD6 7632DB4@ (@3174)          DATA    @.9234270E+@@
06AD8 7827184@ (@3175)          DATA    @.9386931E+@@
06ADA 79BA25C@ (@3176)          DATA    @.9509933E+@@
              (@3177) TAPQVL:
06ADC 75C325BF (@3178)          DATA    @.5750112E-01
06ADE 15E33D4@ (@3179)          DATA    @.1709973E+@@
06AE@ 23DB4CC@ (@3180)          DATA    @.2801300E+@@
06AE2 3@F4DA4@ (@3181)          DATA    @.3824723E+@@
06AE4 3CF4B6C@ (@3182)          DATA    @.4762181E+@@
06AE6 478692C@ (@3183)          DATA    @.5602592E+@@
06AE8 512C144@ (@3184)          DATA    @.6341577E+@@
06AEA 5959674@ (@3185)          DATA    @.6980409E+@@
06AEC 6@50@04@ (@3186)          DATA    @.7524586E+@@
06AEE 662C0DC@ (@3187)          DATA    @.7982347E+@@
06AF@ 6B0D114@ (@3188)          DATA    @.8363363E+@@
06AF2 6F13264@ (@3189)          DATA    @.8677719E+@@
06AF4 725ED9C@ (@3190)          DATA    @.8935196E+@@
06AF6 750DCBC@ (@3191)          DATA    @.9144832E+@@
06AF8 773A5F4@ (@3192)          DATA    @.9314689E+@@
06AFA 78FB92C@ (@3193)          DATA    @.9451774E+@@
              (@3194) *
```

```
                    (03195) ; APU-ENRG(A,B,C,W,D)   COMPUTE, CODE & QUANTIZE ENERGY (GAIN)
                    (03196) *
                    (03197) *       A IS OUTPUT SCALAR: QUANTIZED GAIN
                    (03198) *       B IS OUTPUT SCALAR: INVERSE OF QUANT. GAIN
                    (03199) *       C IS OUTPUT SCALAR: CODED GAIN
                    (03200) *       W IS INPUT BUFFER:  B.B. WITH PITCH REMOVED
                    (03201) *       D IS INPUT SCALAR:  INVERSE OF DNSMPLED FRAMESIZE
                    (03202) *
                    (03203) *       DO SSMSQ TO GET ENERGY,
                    (03204) *       DO QTZE TO GET CODED & QUANTIZED GAIN.
                    (03205) *
                    (03206) *
                    (03207) *       BINDS TO APS3-ENS$
                    (03208) *
                    (03209) *       INPUT STREAM:   SD,W(0),...,W(NBS-1),[FWI],
                    (03210) *                       (2**-15),ENRQTH(0),ENRQVL(0),ENRQVI(0),...,
                    (03211) *                       ENRQTH(63),ENRQVL(63),ENRQVI(63),[FI]
                    (03212) *
                    (03213) *       OUTPUT STREAM:  SA,SB,SC,[EO]
                    (03214) *
                    (03215) *       WHERE:  "ENRQTH" ARE THE GAIN QUANT THRESHOLDS
                    (03216) *               "ENRQVL" ARE THE GAIN QUANT VALUES
                    (03217) *               "ENRQVI" ARE THE GAIN QUANT INVERSE VALUES
                    (03218) *               (ALL INTERNAL TO APS PGM.)
                    (03219) *
                    (03220) *
                    (03221)         EVEN
    06AFC 0000      (03222)         DATA    ENU$SA          ; START ADDR
    06AFD 0035      (03223)         DATA    ENU$SZ          ; SIZE
                    (03224) *
                    (03225) ENU$    BEGIN   APU(ENU)        ; START OF APU MODULE
          00000000  (03226)         $A = 0
                    (03227) *
                    (03228) *
                    (03229) *       DO SSMSQ ON W ARRAY (TERM ON FWI)
                    (03230) *       PUT RESULT IN A0 (LEFT ADM)
                    (03231) *
                    (03232) *       EQ:  (A0) = D * SUM(W**2)
                    (03233) *
                    (03234) *
A00 06AFE 08170817  (03235) ENU$SA MOV(ZERO,A7)                          A7=0,SUM
A01 06B00 08EF20EF  (03236)        MOV(IQA,M7)                           M7=D
A02 06B02 08090809  (03237)        MOV(ZERO,M1)                          M1=0
A03 06B04 04E004E0  (03238)        MUL(M1,M7)                            P=0
A04 06B06 02E002E0  (03239)        R(A7)                                 R=0
                    (03240) *
A05 06B08 08C884B1  (03241) $1     MOV(IQ,M0) \ MOV(A1),MUL(M1,M5)       W0 IN, P2=SQU3'
A06 06B0A 08EC4737  (03242)        MOV(IQA,M4) \ MOV(A7),ADD(A1,A7)      W0 IN, R2=SUM1'
A07 06B0C 00000000  (03243)        NOP
A08 06B0E 91160017  (03244)        JUMPS(ENU$2,FWI)                        EXIT IF W0 LAST
                    (03245) *
A09 06B10 841108C8  (03246)        MOV(A1),MUL(M0,M4) \ MOV(IQ,M0)       P1=SQU0, W1 IN
A0A 06B12 473708EC  (03247)        MOV(A7),ADD(A1,A7) \ MOV(IQA,M4)      R1=SUM2',W1 IN
```

```
A0B 06B14 00000000 (03248)          NOP
A0C 06B16 91160019 (03249)          JUMPS(ENU$3,FWI)                          EXIT IF W1 LAST
                   (03250) *
A0D 06B18 08C98411 (03251)          MOV(IQ,M1) \ MOV(A1),MUL(M0,M4)           W2 IN, P2=SQU1
A0E 06B1A 08ED4737 (03252)          MOV(IQA,M5) \ MOV(A7),ADD(A1,A7)          W2 IN, R2=SUM3'
A0F 06B1C 00000000 (03253)          NOP
A10 06B1E 9116001B (03254)          JUMPS(ENU$4,FWI)                          EXIT IF W2 LAST
                   (03255) *
A11 06B20 04B108C9 (03256)          MOV(A1),MUL(M1,M5) \ MOV(IQ,M1)           P1=SQU2, W3 IN
A12 06B22 473708ED (03257)          MOV(A7),ADD(A1,A7) \ MOV(IQA,M5)          R1=SUM0, W3 IN
A13 06B24 00000000 (03258)          NOP
A14 06B26 90160005 (03259)          JUMPC($1,FWI)
                   (03260) *
A15 06B28 08084B1 (03261)           MOV(ZERO,M0) \ MOV(A1),MUL(M1,M5)         P2=SQU3
A16 06B2A 080C4737 (03262)          MOV(ZERO,M4) \ MOV(A7),ADD(A1,A7)         R2=SM1
                   (03263) *
A17 06B2C 04110008 (03264) ENU$2    MOV(A1),MUL(M0,M4) \ MOV(ZERO,MC)          P1=SQU0
A18 06B2E 4737080C (03265)          MOV(A7),ADD(A1,A7) \ MOV(ZERO,M4)         R1=SUM2'
                   (03266) *
A19 06B30 08098411 (03267) ENU$3    MOV(ZERO,M1) \ MOV(A1),MUL(M0,M4)          P2=SQU1
A1A 06B32 080D4737 (03268)          MOV(ZERO,M5) \ MOV(A7),ADD(A1,A7)         R2=SUM3
                   (03269) *
A1B 06B34 04B10000 (03270) ENU$4    MOV(A1),MUL(M1,M5) \ NOP                   P1=SQU2
A1C 06B36 47370000 (03271)          MOV(A7),ADD(A1,A7) \ NOP                  R1=SUM0
A1D 06B38 08810881 (03272)          MOV(P,A1)
A1E 06B3A 47374737 (03273)          MOV(A7),ADD(A1,A7)                        R1=SUM2, R2=SUM1
A1F 06B3C 088A088A (03274)          MOV(R,M2)
A20 06B3E 056008560 (03275)         MUL(M2,M7)                                SCALE BY SD
A21 06B40 08B00888 (03276)          MOV(P,A0) \ MOV(P,EXO)                    COMBINE SUMS
A22 06B42 08510000 (03277)          MOV(EXI,A1) \ NOP
A23 06B44 41000000 (03278)          ADD(A0,A1) \ NOP
A24 06B46 08900000 (03279)          MOV(R,A0) \ NOP                           A0 = D*SUM(W**2)
                   (03280) *
                   (03281) *        NOW DO QUANTIZE AND CODE OF ENERGY (IN A0)
                   (03282) *        USE (2**-15) AS COUNTER INCREMENT
                   (03283) *
                   (03284) *        RESTART APS INPUT PGM BY CLEARING 'WI'
                   (03285) *
A25 06B48 20372037 (03286)          CLEAR(WI)
                   (03287) *
                   (03288) *
A26 06B4A 08170000 (03289)          MOV(ZERO,A7)\NOP          ;WILL BE CODE CNTR
A27 06B4C 08F60000 (03290)          MOV(IQA,A6)\NOP                 ;2**-15
A28 06B4E 08F10000 (03291) ENU$5    MOV(IQA,A1)\NOP           ;THRES(N)
A29 06B50 490008F0 (03292)          SUB(A0,A1)\MOV(IQA,A0)          ;E-TH(N)\QG
A2A 06B52 000008F1 (03293)          NOP\MOV(IQA,A1)           ;QGI(N)
A2B 06B54 47C80200 (03294)          ADD(A6,A7)\R(A0)          ;CODE(N)+1\QG(N)
A2C 06B56 911E002F (03295)          JUMPS(ENU$6,T1)           ;JUMP IF E LT TH(N)
A2D 06B58 08970000 (03296)          MOV(R,A7)\NOP             ;CODE(N+1)
A2E 06B5A 901D0028 (03297)          JUMPC(ENU$5,FI)           ;JUMP IF MORE
A2F 06B5C 3AE0023C (03298) ENU$6    ALIGN(A7)\MOV(OQ),R(A1)   ;MAKE CODE INTEGER\QG OUT
A30 06B5E 00000090 (03299)          NOP\MOV(R,OQ)             ;QGI OUT
A31 06B60 089C0000 (03300)          MOV(R,OQ)\NOP             ;CODE G OUT
```

```
          APU-ENRG(A,B,C,W,D)      COMPUTE, CODE & QUANTIZE ENERGY (GAIN)

  A32 06862 20322032 (03301)       CLEAR(RA)
  A33 06864 10000000 (03302)       JUMP(0)
  A34 06866 00000000 (03303)       NOP
                     (03304) *
            00000035 (03305)       ENU$SZ = .A-ENU$SA
      06868          (03306)       END     ENU$SZ
                     (03307)       EVEN
                     (03308) *
                     (03309) *
```

```
                    (03310) * APS3-ENRG(A,B,C,W,D)
                    (03311) *
                    (03312) *       A IS OUTPUT SCALAR: QUANTIZED GAIN
                    (03313) *       B IS OUTPUT SCALAR: INVERSE OF QUANT. GAIN
                    (03314) *       C IS OUTPUT SCALAR: CODED GAIN
                    (03315) *       W IS INPUT BUFFER:  B.B. WITH PITCH REMOVED
                    (03316) *       D IS INPUT SCALAR:  INVERSE OF DNSMPLED FRAMESIZE
                    (03317) *
                    (03318) *
                    (03319) *       INPUT STREAM:   SD,W(0),...,W(WBS-1),[FWI],
                    (03320) *                       (2**-15),ENRQTH(0),ENRQVL(0),ENRQVI(0),...,
                    (03321) *                       ENRQTH(63),ENRQVL(63),ENRQVI(63),[FI]
                    (03322) *
                    (03323) *       OUTPUT STREAM:  SA,SB,SC,[EO]
                    (03324) *
                    (03325) *       WHERE:  'ENRQTH' ARE THE GAIN QUANT THRESHOLDS
                    (03326) *               'ENRQVL' ARE THE GAIN QUANT VALUES
                    (03327) *               'ENRQVI' ARE THE GAIN QUANT INVERSE VALUES
                    (03328) *               (ALL INTERNAL TO APS PGM.)
                    (03329) *
                    (03330)         EVEN
      06B68 00006BBE (03331)         ADDR    ENS$I           ; CONSTR INSTR BLK
      06B6A 00006B70 (03332)         ADDR    ENS$ + 2*ENS$S          ; SCALAR BLK
      06B6C 0004     (03333)         DATA    4               ; NUMBER OF SCALARS
      06B6D 0052     (03334)         DATA    ENS$Z           ; MODULE SIZE
      06B6E 00006B88 (03335)         ADDR    ENS$A           ; PTR TO CHAIN ANCHOR
                     (03336)         EVEN
                     (03337) *
                     (03338) *
                     (03339) ENS$    BEGIN   APS(ENS)        ; START OF APS MODULE
                     (03340) *
                     (03341) *
                     (03342) * INPUT PROGRAM
                     (03343) *  REGISTER USAGE:
                     (03344) *       BR0: BUFFER 'W', ENRQTH ELEM ADDRS
                     (03345) *       BR1: ENRQVL ELEM ADDRS
                     (03346) *       BR2: ENRQVI ELEM ADDRS, W BUFFER SIZE-1
                     (03347) *       BR3: SCALAR D ADDR, SCRATCH
                     (03348) *
                     (03349) *       GET SCALAR ADDRESSES
                     (03350) *
  A00 06B70 0042000B (03351) ENS$S  LOAD(BR0,M$$(1))        ; SA ADDR
  A01 06B72 02520000 (03352)        LOAD(BR1,M$$(1))        ; SB ADDR
  A02 06B74 04620000 (03353)        LOAD(BR2,M$$(1))        ; SC ADDR
  A03 06B76 06F20000 (03354)        LOAD(BR3,M$$(1),TF)     ; GEN SD ADDR
                     (03355) *
  A04 06B78 08010011 (03356)        MOVB(BW0,BR0)           ; PUT SA,SB,SC INTO OUTPUT REGS.
  A05 06B7A 0A110013 (03357)        MOVB(BW1,BR1)
  A06 06B7C 0C210015 (03358)        MOVB(BW2,BR2)
                     (03359) *
  A07 06B7E 0E202040 (03360)        JSN(ENS$O,P2)           ; SET OUTPUT PC
  A08 06B80 10300030 (03361)        SET(RO)                 ; RUN OUTPUT PGM
                     (03362) *
```

```
                    (03363) *       GEN W ADDRESSES
                    (03364) *
  A09 06B82 12403000 (03365)        LOAD(BR0,[3])           ; LOAD W BASE ADDR
  A0A 06B84 14600000 (03366)        LOAD(BR2,M$$)           ; WBS-1
  A0B 06B86 16020000 (03367)        SUB(BR0,M$$)            ; SUBTRACT SPACING
                    (03368) *
  A0C 06B88 18BAB006 (03369) $1     ADD(BR0,[11],TF)        ; GEN W-ELEM ADDR
  A0D 06B8A 1A290CB1 (03370)        SUBL(BR2,1),JUMPP($1)   ; DO WBS ELEMENTS
                    (03371) *
  A0E 06B8C 1C300037 (03372)        SET(WI)
  A0F 06B8E 1E000020 (03373)        NOP(0)
  A10 06B90 20000020 (03374)        NOP(0)
                    (03375) *
                    (03376) *       WAIT FOR APU PGM TO CLR WI, THEN GEN (2**-15),
                    (03377) *       ENRQTH,ENRQVL,ENRQVI ADDRESSES, THEN [FI]
                    (03378) *
  A11 06B92 22F203CE (03379)        LOAD(BR3,SVT$UN1(1),TF)  ; (2**-15)
                    (03380) *
  A12 06B94 244268C2 (03381)        LOAD(BR0,ENRQTH(1))      ; LOAD THRESH TABLE BASE
  A13 06B96 26020002 (03382)        SUB(BR0,2)               ; SUBTR SPACING
  A14 06B98 28526C42 (03383)        LOAD(BR1,ENRQVL(1))      ; LOAD VALUE TABLE BASE
  A15 06B9A 2A120002 (03384)        SUB(BR1,2)               ; SUBTR SPACING
  A16 06B9C 2C626CC2 (03385)        LOAD(BR2,ENRQVI(1))      ; LOAD IVAL TABLE BASE
  A17 06B9E 2E220002 (03386)        SUB(BR2,2)               ; SUBTR SPACING
                    (03387) *
  A18 06BA0 307C003F (03388)        LOAD(BR3,63)             ; TABLE SIZE - 1
                    (03389) *
  A19 06BA2 328A0002 (03390) $2     ADD(BR0,2,TF)            ; GEN THRESH ADDR
  A1A 06BA4 349A0002 (03391)        ADD(BR1,2,TF)            ; GEN VAL ADDR
  A1B 06BA6 36AA0002 (03392)        ADD(BR2,2,TF)            ; GEN IVAL ADDR
  A1C 06BA8 38391981 (03393)        SUBL(BR3,1),JUMPP($2)    ; DO 64 ELEMS
                    (03394) *
  A1D 06BAA 3A200031 (03395)        CLEAR(RI)                ; HALT INPUT
  A1E 06BAC 3C000020 (03396)        NOP(0)
  A1F 06BAE 3E000020 (03397)        NOP(0)
                    (03398) *
                    (03399) *
                    (03400) * OUTPUT PROGRAM
                    (03401) *  REGISTER USAGE:
                    (03402) *       BW0: SA
                    (03403) *       BW1: SB
                    (03404) *       BW2: SC
                    (03405) *       BW3: UNUSED
                    (03406) *
                    (03407) *
  A20 06BB0 40300032 (03408) ENS$O  SET(RA)                  ; TURN ON APU
                    (03409) *
                    (03410) *       GEN SA,SB,SC ADDRS
                    (03411) *
  A21 06BB2 42810010 (03412)        MOVB(BW0,BW0,TF)         ; SA
  A22 06BB4 44910012 (03413)        MOVB(BW1,BW1,TF)         ; SB
  A23 06BB6 46A10014 (03414)        MOVB(BW2,BW2,TF)         ; SC
                    (03415) *
```

```
  A24 06BB8 4B200030 (03416)          CLEAR(RO)                ; HALT OUTPUT
  A25 06BBA 4A000020 (03417)          NOP(0)
  A26 06BBC 4C000020 (03418)          NOP(0)
                     (03419) *
            00006800 (03420) ENS$A = #C                        ; CHAIN ANCHOR
                     (03421) *
      06BBE          (03422)          END     #A-1
                     (03423) *
      06BBE 00000000 (03424) ENS$I    DATA    2F'0.0'
       ...
            00000052 (03425) ENS$Z = #L-ENS$                   ; MODULE SIZE
                     (03426) *
                     (03427) *
                     (03428) ENRQTH:
      06BC2 31382C3B (03429)          DATA    0.3668006E-06
      06BC4 3BBC8CBB (03430)          DATA    0.4450718E-06
      06BC6 487BCEBB (03431)          DATA    0.5400451E-06
      06BC8 57F3693B (03432)          DATA    0.6552047E-06
      06BCA 6AB7F33B (03433)          DATA    0.7951152E-06
      06BCC 0817DBBC (03434)          DATA    0.9647839E-06
      06BCE 09D1F83C (03435)          DATA    0.1170658E-05
      06BD0 0BEA6BBC (03436)          DATA    0.1420463E-05
      06BD2 0E7558BC (03437)          DATA    0.1723574E-05
      06BD4 118B2CBC (03438)          DATA    0.2091366E-05
      06BD6 15498A3C (03439)          DATA    0.2537640E-05
      06BD8 19D4693C (03440)          DATA    0.3079144E-05
      06BDA 1F576D3C (03441)          DATA    0.3736199E-05
      06BDC 260789BC (03442)          DATA    0.4533463E-05
      06BDE 2E24FDBC (03443)          DATA    0.5500853E-05
      06BE0 37FDC0BC (03444)          DATA    0.6674674E-05
      06BE2 43F069BC (03445)          DATA    0.8098974E-05
      06BE4 526FC2BC (03446)          DATA    0.9827205E-05
      06BE6 6407113C (03447)          DATA    0.1192422E-04
      06BE8 795F56BC (03448)          DATA    0.1446872E-04
      06BEA 093459BD (03449)          DATA    0.1755618E-04
      06BEC 0B2B2B3D (03450)          DATA    0.2130247E-04
      06BEE 0D8D493D (03451)          DATA    0.2584819E-04
      06BF0 1071973D (03452)          DATA    0.3136390E-04
      06BF2 13F3DF3D (03453)          DATA    0.3805661E-04
      06BF4 18350S3D (03454)          DATA    0.4617746E-04
      06BF6 1D60623D (03455)          DATA    0.5603122E-04
      06BF8 23A5263D (03456)          DATA    0.6798766E-04
      06BFA 2B405ABD (03457)          DATA    0.8249547E-04
      06BFC 347B12BD (03458)          DATA    0.1000991E-03
      06BFE 3FADF6BD (03459)          DATA    0.1214591E-03
      06C00 4D449EBD (03460)          DATA    0.1473771E-03
      06C02 5DC197BD (03461)          DATA    0.1788258E-03
      06C04 71C342BD (03462)          DATA    0.2169852E-03
      06C06 08A09D3E (03463)          DATA    0.2632874E-03
      06C08 0A77E8BE (03464)          DATA    0.3194700E-03
      06C0A 0CB3C53E (03465)          DATA    0.3876413E-03
      06C0C 0F69A9BE (03466)          DATA    0.4703596E-03
      06C0E 12B39F3E (03467)          DATA    0.5707291E-03
```

```
06C10 16B13F3E (03468)          DATA    0.6925163E-03
06C12 1B88E03E (03469)          DATA    0.84020916E-03
06C14 2169063E (03470)          DATA    0.1019600E-02
06C16 288A268E (03471)          DATA    0.1237172E-02
06C18 3130B9BE (03472)          DATA    0.1501170E-02
06C1A 3BAFDFBE (03473)          DATA    0.1821503E-02
06C1C 486C6F3E (03474)          DATA    0.2210192E-02
06C1E 57E0C0BE (03475)          DATA    0.2681822E-02
06C20 6AA14F3E (03476)          DATA    0.3254093E-02
06C22 0816243F (03477)          DATA    0.3940480E-02
06C24 09CFE2BF (03478)          DATA    0.4791041E-02
06C26 0BE7E4BF (03479)          DATA    0.5813396E-02
06C28 0E7247BF (03480)          DATA    0.7053909E-02
06C2A 118773BF (03481)          DATA    0.8559135E-02
06C2C 1545063F (03482)          DATA    0.1038556E-01
06C2E 19CEEE3F (03483)          DATA    0.1260172E-01
06C30 1F50C73F (03484)          DATA    0.1529079E-01
06C32 25FF773F (03485)          DATA    0.1855367E-01
06C34 2E1B33BF (03486)          DATA    0.2251282E-01
06C36 37F1DEBF (03487)          DATA    0.2731680E-01
06C38 43E1FF3F (03488)          DATA    0.3314590E-01
06C3A 525E453F (03489)          DATA    0.4021887E-01
06C3C 63F1D73F (03490)          DATA    0.4880112E-01
06C3E 794593BF (03491)          DATA    0.5921474E-01
06C40 093265C0 (03492)          DATA    0.7185050E-01
                (03493) ENRQVL:
06C42 12E8A9BE (03494)          DATA    0.5770521E-03
06C44 14D42EBE (03495)          DATA    0.6356456E-03
06C46 16F19BBE (03496)          DATA    0.7001886E-03
06C48 194602BE (03497)          DATA    0.7712853E-03
06C4A 1BD6F83E (03498)          DATA    0.8496010E-03
06C4C 1EAAA33E (03499)          DATA    0.9358689E-03
06C4E 21C7C83E (03500)          DATA    0.1030896E-02
06C50 2535E03E (03501)          DATA    0.1135573E-02
06C52 28FD203E (03502)          DATA    0.1250878E-02
06C54 2D26983E (03503)          DATA    0.1377892E-02
06C56 31BC3EBE (03504)          DATA    0.1517802E-02
06C58 36C910BE (03505)          DATA    0.1671918E-02
06C5A 3C592A3E (03506)          DATA    0.1841684E-02
06C5C 4279DC3E (03507)          DATA    0.2028687E-02
06C5E 4939D63E (03508)          DATA    0.2234678E-02
06C60 50A947BE (03509)          DATA    0.2461586E-02
06C62 58D9FEBE (03510)          DATA    0.2711534E-02
06C64 61DF9B3E (03511)          DATA    0.2986861E-02
06C66 6BCFBC3E (03512)          DATA    0.3290145E-02
06C68 76C231BE (03513)          DATA    0.3624224E-02
06C6A 082D133F (03514)          DATA    0.3992225E-02
06C6C 09019ABF (03515)          DATA    0.4397593E-02
06C6E 09EBB73F (03516)          DATA    0.4844122E-02
06C70 0AED983F (03517)          DATA    0.5335990E-02
06C72 0C09A93F (03518)          DATA    0.5877803E-02
06C74 0D42923F (03519)          DATA    0.6474631E-02
06C76 0E9B413F (03520)          DATA    0.7132061E-02
```

```
**:  (BBN-TENEXD)<MAP>BBN3B0.MSO.61,  30-Dec-79 16:14:24, Ed: KFIELD
      APS3-ENRG(A,B,C,W,D)

06C78 1016EF3F (03521)          DATA    0.7856245E-02
06C7A 11B92B3F (03522)          DATA    0.8653963E-02
06C7C 1385DEBF (03523)          DATA    0.9532681E-02
06C7E 1581593F (03524)          DATA    0.1050062E-01
06C80 17B05C3F (03525)          DATA    0.1156685E-01
06C82 1A1821BF (03526)          DATA    0.1274134E-01
06C84 1CBE6CBF (03527)          DATA    0.1403508E-01
06C86 1FA998BF (03528)          DATA    0.1546020E-01
06C88 22E0A13F (03529)          DATA    0.1703001E-01
06C8A 266B3D3F (03530)          DATA    0.1875923E-01
06C8C 2A51E73F (03531)          DATA    0.2066403E-01
06C8E 2E9DF83F (03532)          DATA    0.2276224E-01
06C90 3359BC3F (03533)          DATA    0.2507350E-01
06C92 3B908BBF (03534)          DATA    0.2761945E-01
06C94 3E4EE43F (03535)          DATA    0.3042391E-01
06C96 44A2883F (03536)          DATA    0.3351313E-01
06C98 4B9AA1BF (03537)          DATA    0.3691603E-01
06C9A 5347E33F (03538)          DATA    0.4066446E-01
06C9C 5BBCB1BF (03539)          DATA    0.4479350E-01
06C9E 650D503F (03540)          DATA    0.4934180E-01
06CA0 6F50103F (03541)          DATA    0.5435193E-01
06CA2 7A9D893F (03542)          DATA    0.5987079E-01
06CA4 8B710CC0 (03543)          DATA    0.6595002E-01
06CA6 894C7B40 (03544)          DATA    0.7264654E-01
06CA8 0A3E31C0 (03545)          DATA    0.8002302E-01
06CAA 0B4B7340 (03546)          DATA    0.8814850E-01
06CAC 0C6DBDC0 (03547)          DATA    0.9709904E-01
06CAE 0DB0D040 (03548)          DATA    0.1069584E+00
06CB0 0F14B0C0 (03549)          DATA    0.1178189E+00
06CB2 1099CB340 (03550)         DATA    0.1297821E+00
06CB4 124CB440 (03551)          DATA    0.1429601E+00
06CB6 14282E40 (03552)          DATA    0.1574762E+00
06CB8 163423C0 (03553)          DATA    0.1734662E+00
06CBA 18754E40 (03554)          DATA    0.1910799E+00
06CBC 1AF112C0 (03555)          DATA    0.2104820E+00
06CBE 1DAD65C0 (03556)          DATA    0.2318542E+00
06CC0 20B0D540 (03557)          DATA    0.2553965E+00
      (03558) ENRQVI:
06CC2 36279143 (03559)          DATA    0.1732946E+04
06CC4 3129A1C3 (03560)          DATA    0.1573204E+04
06CC6 2CA17EC3 (03561)          DATA    0.1428187E+04
06CC8 28844BC3 (03562)          DATA    0.1296537E+04
06CCA 24C82F43 (03563)          DATA    0.1177023E+04
06CCC 21643543 (03564)          DATA    0.1068526E+04
06CCE 1E503CC3 (03565)          DATA    0.9700296E+03
06CD0 1884E743 (03566)          DATA    0.8806128E+03
06CD2 18FB81C3 (03567)          DATA    0.7994383E+03
06CD4 16ADF8C3 (03568)          DATA    0.7257465E+03
06CD6 1496C7C3 (03569)          DATA    0.6588476E+03
06CD8 12B0EC43 (03570)          DATA    0.5981154E+03
06CDA 10F7D9C3 (03571)          DATA    0.5429814E+03
06CDC 0F677043 (03572)          DATA    0.4929297E+03
06CDE 0DFBEF43 (03573)          DATA    0.4474917E+03
```

```
06CE0 0CB1EFC3 (03574)          DATA    0.4062421E+03
06CE2 0B865C43 (03575)          DATA    0.3687950E+03
06CE4 0A7665C3 (03576)          DATA    0.3347996E+03
06CE6 097F8143 (03577)          DATA    0.3039300E+03
06CE8 089F5E43 (03578)          DATA    0.2759211E+03
06CEA 7D3E52C2 (03579)          DATA    0.2504869E+03
06CEC 71B2D442 (03580)          DATA    0.2273971E+03
06CEE 6737C842 (03581)          DATA    0.2064358E+03
06CF0 5DB40842 (03582)          DATA    0.1874066E+03
06CF2 5510D842 (03583)          DATA    0.1701316E+03
06CF4 4D3975C2 (03584)          DATA    0.1544489E+03
06CF6 461B1FC2 (03585)          DATA    0.1402119E+03
06CF8 3FA4C642 (03586)          DATA    0.1272873E+03
06CFA 39C6E942 (03587)          DATA    0.1155540E+03
06CFC 34737EC2 (03588)          DATA    0.1049023E+03
06CFE 2F9DC0C2 (03589)          DATA    0.9523245E+02
06D00 2B3A1BC2 (03590)          DATA    0.8645397E+02
06D02 273E0A42 (03591)          DATA    0.7848469E+02
06D04 23A000C2 (03592)          DATA    0.7125002E+02
06D06 20575342 (03593)          DATA    0.6468223E+02
06D08 1D5C2442 (03594)          DATA    0.5871985E+02
06D0A 1AA74EC2 (03595)          DATA    0.5330709E+02
06D0C 183256C2 (03596)          DATA    0.4839327E+02
06D0E 15F75942 (03597)          DATA    0.4393241E+02
06D10 13F0FDC2 (03598)          DATA    0.3988274E+02
06D12 121A6A42 (03599)          DATA    0.3620637E+02
06D14 106F37C2 (03600)          DATA    0.3286089E+02
06D16 0EEB65C2 (03601)          DATA    0.2983905E+02
06D18 0DBB53C2 (03602)          DATA    0.2708850E+02
06D1A 0C4BB642 (03603)          DATA    0.2459150E+02
06D1C 0B298EC2 (03604)          DATA    0.2232467E+02
06D1E 0A222642 (03605)          DATA    0.2026679E+02
06D20 093305C2 (03606)          DATA    0.1839861E+02
06D22 0859F042 (03607)          DATA    0.1670264E+02
06D24 794DD2C1 (03608)          DATA    0.1516300E+02
06D26 6E1F4B41 (03609)          DATA    0.1376528E+02
06D28 63F8A0C1 (03610)          DATA    0.1249640E+02
06D2A 5AC183C1 (03611)          DATA    0.1134449E+02
06D2C 52630C41 (03612)          DATA    0.1029876E+02
06D2E 4ACBA0C1 (03613)          DATA    0.9349428E+01
06D30 43E69C41 (03614)          DATA    0.8487603E+01
06D32 3DA44A41 (03615)          DATA    0.7705220E+01
06D34 37F5AC41 (03616)          DATA    0.6994957E+01
06D36 32CD23C1 (03617)          DATA    0.6350166E+01
06D38 2E1E5541 (03618)          DATA    0.5764811E+01
06D3A 29DE0841 (03619)          DATA    0.5233414E+01
06D3C 26020CC1 (03620)          DATA    0.4751001E+01
06D3E 22812341 (03621)          DATA    0.4313056E+01
06D40 1F52E7C1 (03622)          DATA    0.3915481E+01
                (03623) *
                (03624) *
```

```
                    (03625) ; APU3-DCOR(Y,U,V)  DIRECT CONVOLUTION
                    (03626) *
                    (03627) *
                    (03628) *  Y(M)=SUM[U(K)*V(K+M)] K=0,1,...R-1,M=0,1,..N-1
                    (03629) *       WHERE R IS SIZEE OF REFERENCE, N SIZE OF OUTPUT
                    (03630) *
                    (03631) *Y(M)=S0,Y(M+1)=S1,Y(M+2)=S2 AND Y(M+3)=S3 COMPUTED TOGETHER
                    (03632) *                M=0,4,...
                    (03633) *REGISTER ASSIGNMENTS:
                    (03634) *M0  U(K)
                    (03635) *M1  U(K+1)
                    (03636) *M2  U(K+2)
                    (03637) *M3  U(K+3)
                    (03638) *M4  V(K+M) OR V(K+M+4)
                    (03639) *M5  V(K+M+1) OR V(K+M+5)
                    (03640) *M6  V(K+M+2) OR V(K+M+6)
                    (03641) *M7  V(K+M+3) OR V(K+M+7)
                    (03642) *
                    (03643) *
                    (03644)         EVEN
     06D42 0000     (03645)         DATA  DCRU$SA
     06D43 00BB     (03646)         DATA  DCRU$SZ
                    (03647) *
                    (03648) DCRU$   BEGIN APU(DCRU)
           00000000 (03649)         #A=0
                    (03650) *
A00 06D44 08EC08EC  (03651) DCRU$SA MOV(IQA,M4)     V(0)
A01 06D46 08110810  (03652)         MOV(ZERO,A1)\MOV(ZERO,A0)       CLEAR S0;S1
A02 06D48 08ED08ED  (03653)         MOV(IQA,M5)     V(1)
A03 06D4A 08130812  (03654)         MOV(ZERO,A3)\MOV(ZERO,A2)       CLEAR S2;S3
A04 06D4C 08EE08EE  (03655)         MOV(IQA,M6)             V(2)
                    (03656) *
A05 06D4E 08E908E8  (03657) DCRUP:  MOV(IQA,M7)     U(0)
A06 06D50 08EF08EF  (03658)         MOV(IQA,M7)     V(M+3)
A07 06D52 00000000  (03659)         NOP             WAIT FOR EO TO BE SET
A08 06D54 00000000  (03660)         NOP
A09 06D56 00000000  (03661)         NOP
A0A 06D58 00000000  (03662)         NOP
A0B 06D5A 00000000  (03663)         NOP
A0C 06D5C 00000000  (03664)         NOP
A0D 06D5E 00000000  (03665)         NOP
A0E 06D60 00000000  (03666)         NOP
A0F 06D62 00000000  (03667)         NOP
A10 06D64 00000000  (03668)         NOP
A11 06D66 00000000  (03669)         NOP
A12 06D68 00000000  (03670)         NOP
A13 06D6A 00000000  (03671)         NOP
A14 06D6C 00000000  (03672)         NOP
A15 06D6E 00000000  (03673)         NOP
A16 06D70 00000000  (03674)         NOP
A17 06D72 00000000  (03675)         NOP
A18 06D74 911C0087  (03676)         JUMPS(DCREND,EO)
A19 06D76 9116003D  (03677)         JUMPS(DCLUP,FWI)        CHECK FOR ONLY ONE PROD NEEDED
```

```
A1A 06D78 840C842C (03678) DCRU0:  MUL(M0,M4)\MUL(M0,M5)   PS0(K);PS1(K)
A1B 06D7A 08E908E9 (03679)         MOV(IQA,M1)             U(K+1)
A1C 06D7C 84508471 (03680)         MOV(A0),MUL(M0,M6)\MOV(A1),MUL(M0,M7)   A0=PS0(K),PS2(K);
                   (03681) *                                               A1=PS1(K),PS3(K)
A1D 06D7E 08EC08EC (03682)         MOV(IQA,M4)                      V(M+K+4)
A1E 06D80 410C4100 (03683)         ADD(A0,A1)\ADD(A0,A1)   S0(K);S1(K)
A1F 06D82 9116004B (03684)         JUMPS(DCLU1,FWI)
                   (03685) *
A20 06D84 84B284D3 (03686) DCRU1:  MOV(A2),MUL(M1,M5)\MOV(A3),MUL(M1,M6)   A2=PS2(K),PS0(K+1);
                   (03687) *                                               A3=PS3(K),PS1(K+1)
A21 06D86 08EA08EA (03688)         MOV(IQA,M2)             U(K+2)
A22 06D88 43514350 (03689)         MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)   A1=S0(K),S2(K);
                   (03690) *                                               A0=S1(K),S3(K)
A23 06D8A 84F08491 (03691)         MOV(A0),MUL(M1,M7)\MOV(A1),MUL(M1,M4)   A0=PS0(K+1),PS2(K+1);
                   (03692) *                                               A1=PS1(K+1),PS3(K+1)
A24 06D8C 41134112 (03693)         MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)   A3=S2(K),S0(K+1);
                   (03694) *                                               A2=S3(K),S1(K+1)
A25 06D8E 08ED08ED (03695)         MOV(IQA,M5)                      V(M+K+5)
A26 06D90 9116005A (03696)         JUMPS(DCLU2,FWI)
                   (03697) *
A27 06D92 85528573 (03698) DCRU2:  MOV(A2),MUL(M2,M6)\MOV(A3),MUL(M2,M7)   A2=PS2(K+1),PS0(K+2);
                   (03699) *                                               A3=PS3(K+1),PS1(K+2)
A28 06D94 08EB08EB (03700)         MOV(IQA,M3)             U(K+3)
A29 06D96 43514350 (03701)         MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)   A1=S0(K+1),S2(K+1);
                   (03702) *                                               A0=S1(K+1),S3(K+1)
A2A 06D98 85108531 (03703)         MOV(A0),MUL(M2,M4)\MOV(A1),MUL(M2,M5)   A0=PS0(K+2),PS2(K+2);
                   (03704) *                                               A1=PS1(K+2),PS3(K+2)
A2B 06D9A 41134112 (03705)         MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)   A3=S2(K+1),S0(K+2);
                   (03706) *                                               A2=S3(K+1),S1(K+2)
A2C 06D9C 08EE08EE (03707)         MOV(IQA,M6)                      V(M+K+6)
A2D 06D9E 91160069 (03708)         JUMPS(DCLU3,FWI)
                   (03709) *
A2E 06DA0 85F28593 (03710) DCRU3:  MOV(A2),MUL(M3,M7)\MOV(A3),MUL(M3,M4)   A2=PS2(K+2),PS0(K+3);
                   (03711) *                                               A3=PS3(K+2),PS1(K+3)
A2F 06DA2 08E808E8 (03712)         MOV(IQA,M0)             U(K+4)
A30 06DA4 43514350 (03713)         MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)   A1=S0(K+2),S2(K+2);
                   (03714) *                                               A0=S1(K+2),S3(K+2)
A31 06DA6 85B085D1 (03715)         MOV(A0),MUL(M3,M5)\MOV(A1),MUL(M3,M6)   A0=PS0(K+3),PS2(K+3);
                   (03716) *                                               A1=PS1(K+3),PS3(K+3)
A32 06DA8 41134112 (03717)         MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)   A3=S2(K+2),S0(K+3);
                   (03718) *                                               A2=S3(K+2),S1(K+3)
A33 06DAA 08EF08EF (03719)         MOV(IQA,M7)                      V(M+K+7)
A34 06DAC 91160078 (03720)         JUMPS(DCLU4,FWI)
                   (03721) *
A35 06DAE 84128433 (03722) DCRU4:  MOV(A2),MUL(M0,M4)\MOV(A3),MUL(M0,M5)   A2=PS2(K+3),PS0(K+4);
                   (03723) *                                               A3=PS3(K+3),PS1(K+4)
A36 06DB0 08E908E9 (03724)         MOV(IQA,M1)             U(K+5)
A37 06DB2 43514350 (03725)         MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)   A1=S0(K+3),S2(K+3);
                   (03726) *                                               A0=S1(K+3),S3(K+3)
A38 06DB4 84508471 (03727)         MOV(A0),MUL(M0,M6)\MOV(A1),MUL(M0,M7)   A0=PS0(K+4),PS2(K+4);
                   (03728) *                                               A1=PS1(K+4),PS3(K+4)
A39 06DB6 41134112 (03729)         MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)   A3=S2(K+3),S0(K+4);
                   (03730) *                                               A2=S3(K+3),S1(K+4)
```

```
A3A 06D80 08EC08EC (03731)          MOV(IQA,M4)                          V(M+K+8)
A3B 06D8A 9116004B (03732)          JUMPS(DCLU1,FWI)
A3C 06D8C 10000020 (03733)          JUMP(DCRU1)
                   (03734) *
                   (03735) *
                   (03736) *
A3D 06D8E 20372037 (03737) DCLU0:   CLEAR(WI)
A3E 06DC0 04000420 (03738)          MUL(M0,M4)\MUL(M0,M5)    PS0(K);PS1(K)
A3F 06DC2 04500471 (03739)          MOV(A0),MUL(M0,M6)\MOV(A1),MUL(M0,M7)    A0=PS0(K),PS2(K);
                   (03740) *                                                 A1=PS1(K),PS3(K)
A40 06DC4 41004100 (03741)          ADD(A0,A1)\ADD(A0,A1)    S0(K);S1(K)
A41 06DC6 009C0810 (03742)          MOV(R,OQ)\MOV(ZERO,A0)   OUT=S0(M);CLEAR S1
A42 06DC8 0811009C (03743)          MOV(ZERO,A1)\MOV(R,OQ)   CLEAR S0;OUT=S1(M)
A43 06DCA 00B209B3 (03744)          MOV(P,A2)\MOV(P,A3)      PS2;PS3
A44 06DCC 43404340 (03745)          ADD(A2,A3)       S2;S3
A45 06DCE 08EC08EC (03746)          MOV(IQA,M4)      V(M+4)
A46 06DD0 08ED08ED (03747)          MOV(IQA,M5)      V(M+5)
A47 06DD2 009C0812 (03748)          MOV(R,OQ)\MOV(ZERO,A2)   OUT=S2(M);CLEAR S3
A48 06DD4 0813009C (03749)          MOV(ZERO,A3)\MOV(R,OQ)   CLEAR S2;OUT=S3(M)
A49 06DD6 08EE08EE (03750)          MOV(IQA,M6)              V(M+6)
A4A 06DD8 10000005 (03751)          JUMP(DCRUP)
                   (03752) *
A4B 06DDA 20372037 (03753) DCLU1:   CLEAR(WI)
A4C 06DDC 04B204D3 (03754)          MOV(A2),MUL(M1,M5)\MOV(A3),MUL(M1,M6)    A2=PS2(K),PS0(K+1);
                   (03755) *                                                 A3=PS3(K),PS1(K+1)
A4D 06DDE 43514350 (03756)          MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)    A1=S0(K),S2(K);
                   (03757) *                                                 A0=S1(K),S3(K)
A4E 06DE0 04F00491 (03758)          MOV(A0),MUL(M1,M7)\MOV(A1),MUL(M1,M4)    A0=PS0(K+1),PS2(K+1);
                   (03759) *                                                 A1=PS1(K+1),PS3(K+1)
A4F 06DE2 41134112 (03760)          MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)    A3=S2(K),S0(K+1);
                   (03761) *                                                 A2=S3(K),S1(K+1)
A50 06DE4 009C0810 (03762)          MOV(R,OQ)\MOV(ZERO,A0)   OUT=S0(M);CLEAR S1
A51 06DE6 0811009C (03763)          MOV(ZERO,A1)\MOV(R,OQ)   CLEAR S0;OUT=S1(M)
A52 06DE8 00B200B3 (03764)          MOV(P,A2)\MOV(P,A3)      PS2;PS3
A53 06DEA 43404340 (03765)          ADD(A2,A3)       S2;S3
A54 06DEC 08EC08EC (03766)          MOV(IQA,M4)      V(M+4)
A55 06DEE 08ED08ED (03767)          MOV(IQA,M5)      V(M+5)
A56 06DF0 009C0812 (03768)          MOV(R,OQ)\MOV(ZERO,A2)   OUT=S2(M);CLEAR S3
A57 06DF2 0813009C (03769)          MOV(ZERO,A3)\MOV(R,OQ)   CLEAR S2;OUT=S3(M)
A58 06DF4 08EE08EE (03770)          MOV(IQA,M6)              V(M+6)
A59 06DF6 10000005 (03771)          JUMP(DCROP)
                   (03772) *
                   (03773) *
A5A 06DF8 20372037 (03774) DCLU2:   CLEAR(WI)
A5B 06DFA 05520573 (03775)          MOV(A2),MUL(M2,M6)\MOV(A3),MUL(M2,M7)    A2=PS2(K+1),PS0(K+2);
                   (03776) *                                                 A3=PS3(K+1),PS1(K+2)
A5C 06DFC 43514350 (03777)          MOV(A1),ADD(A2,A3)\MOV(A0),ADD(A2,A3)    A1=S0(K+1),S2(K+1);
                   (03778) *                                                 A0=S1(K+1),S3(K+1)
A5D 06DFE 05100531 (03779)          MOV(A0),MUL(M2,M4)\MOV(A1),MUL(M2,M5)    A0=PS0(K+2),PS2(K+2);
                   (03780) *                                                 A1=PS1(K+2),PS3(K+2)
A5E 06E00 41134112 (03781)          MOV(A3),ADD(A0,A1)\MOV(A2),ADD(A0,A1)    A3=S2(K+1),S0(K+2);
                   (03782) *                                                 A2=S3(K+1),S1(K+2)
A5F 06E02 009C0810 (03783)          MOV(R,OQ)\MOV(ZERO,A0)   OUT=S0(M);CLEAR S1
```

```
A6@ @6E@4 @811@89C (@3784)          MOV(ZERO,A1)\MOV(R,OQ)  CLEAR S@;OUT=S1(M)
A61 @6E@6 @8B2@8B3 (@3785)          MOV(P,A2)\MOV(P,A3)     PS2;PS3
A62 @6E@8 434@4348 (@3786)          ADD(A2,A3)       S2;S3
A63 @6E@A @8EC@8EC (@3787)          MOV(IQA,M4)      V(M+4)
A64 @6E@C @8ED@8ED (@3788)          MOV(IQA,M5)      V(M+5)
A65 @6E@E @89C@812 (@3789)          MOV(R,OQ)\MOV(ZERO,A2)  OUT=S2(M);CLEAR S3
A66 @6E1@ @813@89C (@3790)          MOV(ZERO,A3)\MOV(R,OQ)  CLEAR S2;OUT=S3(M)
A67 @6E12 @8EE@8EE (@3791)          MOV(IQA,M6)             V(M+6)
A68 @6E14 1@@@@@@5 (@3792)          JUMP(DCRUP)
                   (@3793) *
                   (@3794) *
A69 @6E16 2@372@37 (@3795) DCLU3:   CLEAR(WI)
A6A @6E18 85F28593 (@3796)          MOV(A2),MUL(M3,M7)\MOV(A3),MUL(M3,M4)   A2=PS2(K+2),PS@(K+3);
                   (@3797) *                                                A3=PS3(K+2),PS1(K+3)
A6B @6E1A 43514358 (@3798)          MOV(A1),ADD(A2,A3)\MOV(A@),ADD(A2,A3)   A1=S@(K+2),S2(K+2);
                   (@3799) *                                                A@=S1(K+2),S3(K+2)
A6C @6E1C 858@85D1 (@38@@)          MOV(A@),MUL(M3,M5)\MOV(A1),MUL(M3,M6)   A@=PS@(K+3),PS2(K+3);
                   (@3801) *                                                A1=PS1(K+3),PS3(K+3)
A6D @6E1E 41134112 (@3802)          MOV(A3),ADD(A@,A1)\MOV(A2),ADD(A@,A1)   A3=S2(K+2),S@(K+3);
                   (@3803) *                                                A2=S3(K+2),S1(K+3)
A6E @6E2@ @89C@81@ (@3804)          MOV(R,OQ)\MOV(ZERO,A@)  OUT=S@(M);CLEAR S1
A6F @6E22 @811@89C (@3805)          MOV(ZERO,A1)\MOV(R,OQ)  CLEAR S@;OUT=S1(M)
A7@ @6E24 @8B2@8B3 (@3806)          MOV(P,A2)\MOV(P,A3)     PS2;PS3
A71 @6E26 43484348 (@3807)          ADD(A2,A3)       S2;S3
A72 @6E28 @8EC@8EC (@3808)          MOV(IQA,M4)      V(M+4)
A73 @6E2A @8ED@8ED (@3809)          MOV(IQA,M5)      V(M+5)
A74 @6E2C @89C@812 (@381@)          MOV(R,OQ)\MOV(ZERO,A2)  OUT=S2(M);CLEAR S3
A75 @6E2E @813@89C (@3811)          MOV(ZERO,A3)\MOV(R,OQ)  CLEAR S2;OUT=S3(M)
A76 @6E3@ @8EE@8EE (@3812)          MOV(IQA,M6)             V(M+6)
A77 @6E32 1@@@@@@5 (@3813)          JUMP(DCRUP)
                   (@3814) *
                   (@3815) *
A78 @6E34 2@372@37 (@3816) DCLU4:   CLEAR(WI)
A79 @6E36 84128433 (@3817)          MOV(A2),MUL(M@,M4)\MOV(A3),MUL(M@,M5)   A2=PS2(K+3),PS@(K+4);
                   (@3818) *                                                A3=PS3(K+3),PS1(K+4)
A7A @6E38 43514358 (@3819)          MOV(A1),ADD(A2,A3)\MOV(A@),ADD(A2,A3)   A1=S@(K+3),S2(K+3);
                   (@382@) *                                                A@=S1(K+3),S3(K+3)
A7B @6E3A 845@8471 (@3821)          MOV(A@),MUL(M@,M6)\MOV(A1),MUL(M@,M7)   A@=PS@(K+4),PS2(K+4);
                   (@3822) *                                                A1=PS1(K+4),PS3(K+4)
A7C @6E3C 41134112 (@3823)          MOV(A3),ADD(A@,A1)\MOV(A2),ADD(A@,A1)   A3=S2(K+3),S@(K+4);
                   (@3824) *                                                A2=S3(K+3),S1(K+4)
A7D @6E3E @89C@81@ (@3825)          MOV(R,OQ)\MOV(ZERO,A@)  OUT=S@(M);CLEAR S1
A7E @6E4@ @811@89C (@3826)          MOV(ZERO,A1)\MOV(R,OQ)  CLEAR S@;OUT=S1(M)
A7F @6E42 @8B2@8B3 (@3827)          MOV(P,A2)\MOV(P,A3)     PS2;PS3
A8@ @6E44 434@43%@ (@3828)          ADD(A2,A3)       S2;S3
A81 @6E46 @8EC@8EC (@3829)          MOV(IQA,M4)      V(M+4)
A82 @6E48 @8ED@8ED (@383@)          MOV(IQA,M5)      V(M+5)
A83 @6E4A @89C@812 (@3831)          MOV(R,OQ)\MOV(ZERO,A2)  OUT=S2(M);CLEAR S3
A84 @6E4C @813@89C (@3832)          MOV(ZERO,A3)\MOV(R,OQ)  CLEAR S2;OUT=S3(M)
A85 @6E4E @8EE@8EE (@3833)          MOV(IQA,M6)             V(M+6)
A86 @6E5@ 1@@@@@@5 (@3834)          JUMP(DCRUP)
                   (@3835) *
                   (@3836) *
```

```
                    (03837) *
                    (03838) *
A87 06E52 00000000  (03839) DCREND: NOP
A88 06E54 20322032  (03840)         CLEAR(RA)
A89 06E56 00000000  (03841)         NOP
A8A 06E58 1000000F  (03842)         JUMP(0)
                    (03843) *
                    (03844) *
          0000008B  (03845)         DCRU$SZ=.A-DCRU$SA
    06E5A           (03846)         END
                    (03847)         EVEN
```

```
                    (03848) ; APS3-DCOR(V,U,V)
                    (03849) ;
                    (03850) ;INPUT STREAM:
                    (03851) ;  (V(0),V(1),V(2),U(0),V(3),U(1),V(4),...U(USZ-1),V(USZ+2),WI),
                    (03852) ;   (V(4),V(5),V(6),U(0),V(7),U(1),V(8),...U(USZ-1),V(USZ+6),WI),
                    (03853) ;   (...)..()...(V(VSIZ-USZ-1),...U(0),V(VSZ-USZ-1+3),...
                    (03854) ;    U(USZ-1),V(VSZ-I+2).
                    (03855) ;       WHERE I IS 0,1,2,3 SUCH THAT VSZ-USZ-I=4N
                    (03856) ;
                    (03857) ;OUTPUT STREAM
                    (03858) ; (Y(0),Y(1),Y(2),Y(3)),(...),...(  ,Y(YSZ-1)).
                    (03859)          EVEN
    06E5A 00006EA6  (03860)          ADDR    DCRS$I
    06E5C 00006E62  (03861)          ADDR    DCRS$S  SCALAR BLOCK
    06E5E 0000      (03862)          DATA    0       NO SCALARS
    06E5F 005A      (03863)          DATA    DCRS$Z  SIZE
    06E60 00006E9E  (03864)          ADDR    DCRS$A  ;CHAIN ANCHOR
                    (03865)          EVEN
                    (03866) DCRS$:
                    (03867) DCRS$S   BEGIN APS(DCRS)
A00 06E62 00201A40  (03868)          JSN(DCRS$O,P2)
A01 06E64 02300030  (03869)          SET(RO)
                    (03870) ;
A02 06E66 044F1000  (03871)          LOAD(BR0,[1])   U ST ADDR
A03 06E68 06500000  (03872)          LOAD(BR1,M$$)   U SIZE-1
A04 06E6A 00020000  (03873)          SUB(BR0,M$$)    U ST -SPACING
A05 06E6C 0A602006  (03874)          LOAD(BR2,[2])   V ST ADDR
A06 06E6E 0C700000  (03875)          LOAD(BR3,M$$)   V SIZE-1
A07 06E70 0E220000  (03876)          SUB(BR2,M$$)              V ST ADDR - SPACING
A08 06E72 10390023  (03877)          SUBB(BR3,BR1)             V SIZE - U SIZE =V SIZE-1
                    (03878) ;
A09 06E74 12AAA008  (03879) DCRS1:   ADD(BR2,[10],TF)         V(M)
A0A 06E76 14AAA002  (03880)          ADD(BR2,[1],TF)          V(M+1)
A0B 06E78 16AAA002  (03881)          ADD(BR2,[10],TF)         V(M+2)
A0C 06E7A 18310015  (03882)          MOVB(BW3,BR2)            V(M+2)
                    (03883) ;
A0D 06E7C 1A8A9004  (03884) DCRS2:   ADD(BR0,[9],TF)          U(K)
A0E 06E7E 1CAAA002  (03885)          ADD(BR2,[10],TF)         V(M+3+K)
A0F 06E80 1E190DB1  (03886)          SUBL(BR1,1),JUMPP(DCRS2)         USED ALL V'S?
                    (03887) ;
A10 06E82 20300037  (03888)          SET(WI)
A11 06E84 22000020  (03889)          NOP(0)
A12 06E86 24290016  (03990)          MOVB(BR2,BW3)            V(M+2)=V(M'-2)
A13 06E88 262AA00A  (03891)          ADD(BR2,[10])            V(M'-1);M'=>M
A14 06E8A 28401002  (03892)          LOAD(BR0,[1])            U ST ADDR
A15 06E8C 2A500000  (03993)          LOAD(BR1,M$$)            U SIZE-1
A16 06E8E 2C020000  (03894)          SUB(BR0,M$$)             U ST - SPACING
A17 06E90 2E3909B4  (03895)          SUBL(BR3,4),JUMPP(DCRS1)         JUMPIF NEED MORE INPUT
A18 06E92 30200031  (03896)          CLEAR(RI)
A19 06E94 32000020  (03897)          NOP(0)
                    (03898) ;
                    (03899) ;OUTPUT PROGRAM
A1A 06E96 34300032  (03900) DCRS$O   SET(RA)
```

```
A1B 06E98 3650000E (03901)          LOAD(BW1,[0])            Y ST ADDR
A1C 06E9A 38600000 (03902)          LOAD(BW2,M$$)            Y SIZE-1
A1D 06E9C 3A120000 (03903)          SUB(BW1,M$$)             Y ST ADDR - SPACING
                   (03904) *
                   (03905) *OUTPUT LOOP
A1E 06E9E 3C9A0006 (03906) DCRS3:   ADD(BW1,[8],TF)          Y(N)
A1F 06EA0 3E211EB1 (03907)          SUBL(BW2,1),JUMPP(DCRS3)         MORE OUTPUT?
A20 06EA2 40200030 (03908)          CLEAR(R0)                NO
A21 06EA4 42000020 (03909)          NOP(0)
          00006E9E (03910)          DCRS$A=.C
    06EA6         (03911)          END
    06EA6 00000000 (03912) DCRS$I   DATA  11F'0.0'
     ...
          0000005A (03913)          DCRS$Z=.L-DCRS$
                   (03914) *
                   (03915) *
                   (03916) *
```

```
                    (03917) ; MPMBS$   MODULE TO MOVE BUFFER TO SCALAR
                    (03918) *       MPMBS(Y,SA,NS)
                    (03919) *       JUST LIKE MPTBS, BUT DOESN'T STEP BUFFER ADDR.
                    (03920) *       MOVES 'NS' SAMPLES FROM BUFFER 'Y' TO REAL
                    (03921) *       SCALAR TABLE STARTING WITH SCALAR 'SA'.
                    (03922) *
                    (03923) *       FCB FORMAT (16BITS/HWORD)
                    (03924) *
                    (03925) *       HWORD         LBYTE          RBYTE
                    (03926) *       ----          -----          -----
                    (03927) *        0:         PTR TO NXT FCB & F.L. FLG (LSB)
                    (03928) *        1:            111             Y
                    (03929) *        2:             SA            NS
                    (03930) *        3:             0              0
                    (03931) *        4:             0              0
                    (03932) *        5:             0              0
                    (03933) *
                    (03934) *
                    (03935) *
                    (03936)         EVEN
06EBC 70220FFF      (03937) MPMBS$  MOVWK   R2,R1,MSK$RBYT      EXTRACT BUFFER IDENTIFIER
06EBE 3A21          (03938)         LLS     R2,1               CONVERT TO BCT$ INDEX
06EBF 4074          (03939)         MOVRR   R7,R2
06EC0 8600A1AD6     (03940)         CALL    R0,CK4DB$
06EC2 F0740582      (03941)         MOVMR   R7,BCT$BA(R2)      GET ARRAY BUS
06EC4 9A700006      (03942)         ANDIR   R7,MSK$BMB
                    (03943) *
06EC6 3A73          (03944)         LLS     R7,3               POSITION MEMORY BUS BITS
06EC7 96700001      (03945)         IORIR   R7,$1              STACK BUS ALWAYS EQUALS ONE
                    (03946) *
06EC9 DA040686      (03947)         SMBCL   BIT$RC,BCT$AT(R2)  IF ARRAY TYPE IS COMPLEX
06ECB 96700100      (03948)         IORIR   R7,$100            SET FLAG FOR TRANSFER LOOP
                    (03949) *
06ECD C0540582      (03950)         MOVMRL  R5,BCT$BA(R2)      SET-UP BASE ADDRESS
06ECF 3A62          (03951)         LLS     R6,2               OF ARRAY
06ED0 566A0001      (03952)         IORKR   R6,R5,1
                    (03953) *
06ED2 3E62          (03954)         ROR     R6, 2
06ED3 F0540604      (03955)         MOVMR   R5,BCT$AD(R2)      AND SAMPLE INCREMENT
06ED5 F0320001      (03956)         MOVMR   R3,A$(R1)          CONTRUCT SCALAR ARRAY ADDRESS
06ED7 5F46FF00      (03957)         MOVKR   R4,R3,MSK$LBYT
06ED9 3C47          (03958)         LRS     R4,7
                    (03959)         EVEN
06EDA 9C400382      (03960)         ADDIR   R4,SVT$
06EDC 50360FFF      (03961)         MOVKR   R3,R3,MSK$RBYT
06EDE 2731          (03962)         DECR    R3, 1
                    (03963) *
06EDF 8600183C      (03964)         CALL    R0,GATHER$         GATHER LOGICAL BUFFER INTO SCALAR A
                    (03965) *
06EE1 0E70          (03966)         RETURN
                    (03967)         EVEN
                    (03968) *
```

```
                    (03969) ; DEFINE TOP OF MODULE
                    (03970) *
                    (03971) *
          00006EE2  (03972)         TOE$COR = .L
                    (03973) *
                    (03974) *
     06EE2          (03975)         END
```

DEFINE TOP OF MODULE

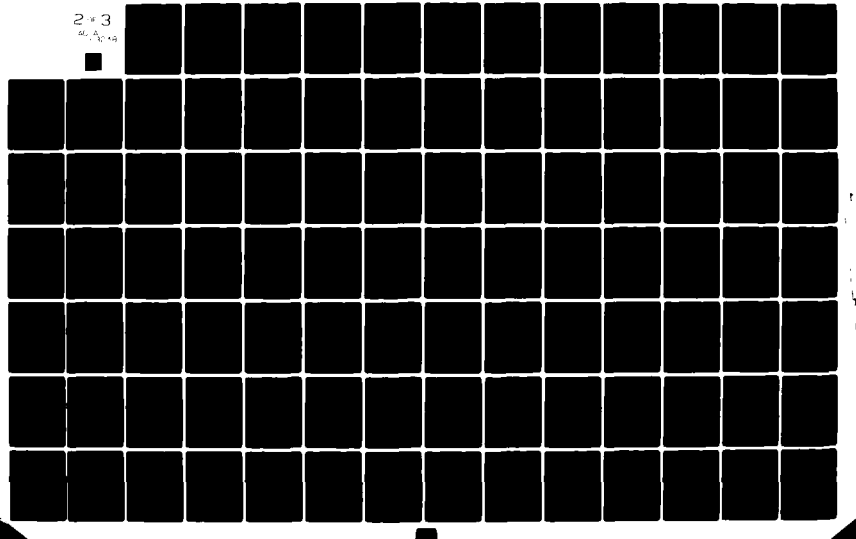
```
AAPC$:      0674C (00131) (02278) (02283) (02415)
AAPC$3:     00032 (02289) (02368)
AAPC$A:     067C8 (02291) (02410)
AAPC$I:     067F2 (02277) (02414)
AAPC$M:     0670E (02314) (02342) (02373)
AAPC$S:     00004 (02278) (02296)
AAPC$Z:     000B0 (02280) (02415)
AFDT$ORG:   008E8 (00011) (00109) (00114) (00119) (00124) (00129) (00134) (00139) (00144) (00149)
                  (00154)
AP$ASSS:    00245 (00012)
AP$BNDR:    00EFA (00013)
AP$BNDR0:   00F63 (00014)
AP$BNDR1:   00F2D (00015)
AP$BSL:     0024D (00016)
AP$CSSC:    00248 (00017)
AP$DONE:    00FB1 (00018) (02027)
AP$DONER:   00FBC (00019)
AP$G0:      0000C (00020)
AP$G1:      0000D (00021)
AP$PBFF:    00010 (00022)
AP$SAID:    0000A (00023)
AP$SSCLR:   0000E (00024)
AP$SSS:     00009 (00025)
APCIS$:     068CC (00141) (02610) (02615) (02717)
APCIS$3:    00028 (02621) (02687)
APCIS$A:    06934 (02613) (02712)
APCIS$M:    068FE (02643) (02667) (02692)
APCIS$S:    00004 (02610) (02628)
APCIS$Z:    0007E (02612) (02717)
APCIU$:     06876 (00140) (02522)
APCIU$SA:   00000 (02519) (02529) (02588)
APCIU$SZ:   00027 (02520) (02588)
APCLP:      00027 (02196) (02250)
APCS$I:     0693C (02609) (02716)
APS$BMO:    00004 (00026)
APS$LA:     1FFC0 (00027) (02032) (02377) (02696) (03133)
APSR:       1FFC0 (00028)
BBTAB:      05D00 (00183)
BCT$AD:     00604 (00030) (03955)
BCT$AT:     00686 (00031) (03947)
BCT$BA:     00582 (00032) (03941) (03950)
BIT$G0:     00000 (00033)
BIT$RC:     00000 (00034) (03947)
CK4DB$:     01AD6 (00036) (03940)
CLP$G0G1:   00792 (00037)
COS$:       03190 (00038)
CSPU$NOS:   021FC (00039) (00112) (00117) (00122) (00132) (00137) (00142) (00147) (00152) (00157)
DCLU0:      0003D (03677) (03737)
DCLU1:      00048 (03694) (03732) (03753)
DCLU2:      0005A (03696) (03774)
DCLU3:      00069 (03708) (03795)
DCLU4:      00078 (03720) (03816)
DCREND:     00087 (03676) (03839)
```

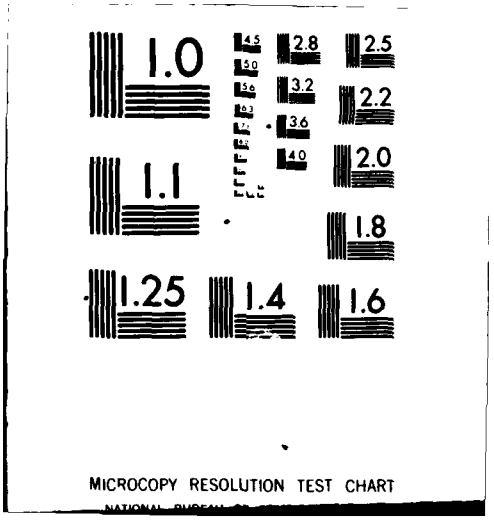
DEFINE TOP OF MODULE

```
DCRS$:      06E62 (00156) (03866) (03913)
DCRS$A:     06E9E (03864) (03910)
DCRS$I:     06EA6 (03860) (03912)
DCRS$O:     0001A (03868) (03900)
DCRS$S:     06E62 (03861) (03867)
DCRS$Z:     0005A (03863) (03913)
DCRS1:      00009 (03879) (03895)
DCRS2:      0000D (03884) (03886)
DCRS3:      0001E (03906) (03907)
DCRU$:      06D44 (00155) (03648)
DCRU$SA:    00000 (03645) (03651) (03845)
DCRU$SZ:    00008 (03646) (03845)
DCRU0:      0001A (03678)
DCRU1:      00029 (03686) (03733)
DCRU2:      00027 (03698)
DCRU3:      0002E (03710)
DCRU4:      00035 (03722)
DCRUP:      00005 (03657) (03751) (03771) (03792) (03813) (03834)
DEALS$:     06838 (00136) (02465) (02471) (02503)
DEALS$3:    00009 (02472) (02482)
DEALS$A:    06860 (02469) (02498)
DEALS$I:    06868 (02464) (02502)
DEALS$S:    0000A (02465) (02484)
DEALS$Z:    0003C (02467) (02503)
DEALU$:     067FE (00135) (02425)
DEALU$SA:   00000 (02422) (02427) (02452)
DEALU$SZ:   00019 (02423) (02452)
DKTAB1:     05D10 (00191)
DKTAB2:     05D50 (00224)
DKTAB3:     05D90 (00257)
DKTAB4:     05DD0 (00290)
DKTAB5:     05DF0 (00307)
DKTAB6:     05E10 (00324)
DKTAB7:     05E30 (00341)
DKTAB8:     05E40 (00350)
DMY$:       00794 (00041) (01601)
DOQ:        00070 (01454) (01456) (01458) (01462) (01464) (01466) (01469) (01471) (01477)
DTQTAB:     05ED0 (00440)
EDTAB:      05E50 (00359)
ENRQTH:     06BC2 (03381) (03428)
ENRQVI:     06CC2 (03385) (03558)
ENRQVL:     06C42 (03383) (03493)
ENS$:       06B70 (00151) (03332) (03339) (03425)
ENS$A:      06B88 (03335) (03420)
ENS$I:      06B8E (03331) (03424)
ENS$O:      00020 (03360) (03408)
ENS$S:      00000 (03332) (03351)
ENS$Z:      00052 (03334) (03425)
ENU$:       06AFE (00150) (03225)
ENU$2:      00017 (03244) (03264)
ENU$3:      00019 (03249) (03267)
ENU$4:      0001B (03254) (03270)
ENU$5:      00028 (03291) (03297)
```

```
ENU$6:      0002F (03295) (03298)
ENU$SA:     00000 (03222) (03235) (03305)
ENU$SZ:     00035 (03223) (03305) (03306)
ERROR$:     01AFA (00043)
F11A$:      03110 (00045)
FDT$:       007E8 (00046) (00162) (00165)
FEOS$:      03040 (00047)
FFT$CBSZ:   0079A (00048)
FLG$CLR:    00000 (00049)
FLG$G0:     00004 (00050)
FLG$G1:     00005 (00051)
FLG$G2:     00006 (00052)
FLG$G3:     00007 (00053)
FLG$RI:     00011 (00054)
FLG$SET:    00020 (00055)
GATHER$:    01B3C (00057) (03964)
H$:         00001 (02059) (02068) (03956)
IFF$1:      0668A (02072) (02076)
IN:         00000 (00548)
INST:       0667A (02030) (02036)
ISVT$:      00502 (00061) (02071) (02076)
K123:       00058 (01653) (01658)
K1LP:       0005A (01655) (01657)
K456:       0005F (01661) (01666)
K4LP:       00061 (01663) (01665)
K78:        00066 (01670) (01675)
K7LP:       00068 (01672) (01674)
KQ$I:       0004E (01524) (01642)
KQ$O:       0006E (01642) (01680)
KQOLP:      00074 (01686) (01688)
KQTAB:      063EC (01650) (01703)
KQUAN:      00055 (01443) (01450)
LCASE1:     0002C (01102) (01130)
LCASE2:     00031 (01106) (01136)
LCASE3:     0001A (01108)
LDONE:      0004F (00616) (00630)
LEFT:       00014 (01099)
LOAD$AP:    01B7E (00063)
LOAD$AP1:   01B8F (00064)
LOOP:       0001D (00577) (00627)
LPQ1:       00073 (01480) (01486)
M$S:        00000 (00069) (00689) (00693) (00706) (00707) (00723) (00724) (00749) (00750) (00755)
                  (00756) (00764) (00765) (00990) (00991) (01010) (01011) (01197) (01198) (01201)
                  (01202) (01205) (01206) (01225) (01226) (01229) (01230) (01525) (01529) (01530)
                  (01534) (01535) (01581) (01582) (01610) (01611) (01630) (01631) (01647) (01648)
                  (01682) (01683) (02296) (02297) (02298) (02299) (02343) (02347) (02348) (02354)
                  (02355) (02381) (02382) (02391) (02392) (02475) (02476) (02484) (02485) (02487)
                  (02488) (02493) (02494) (02629) (02629) (02668) (02672) (02673) (02677) (02678)
                  (02700) (02701) (03023) (03024) (03052) (03053) (03081) (03082) (03089) (03119)
                  (03120) (03121) (03142) (03143) (03351) (03352) (03353) (03354) (03366) (03367)
                  (03872) (03873) (03875) (03876) (03893) (03894) (039F2) (03903)
MPIFF$:     0667E (00163) (02067)
MPMBS$:     06EBC (00166) (03937)
```

AD-A083 238 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL DCA100-79-C-0003
UNCLASSIFIED BBN-4327-VOL-2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART

```
MSK$BMB:    00006 (00066) (03942)
MSK$LBYT:   0FF00 (00067) (03957)
MSK$RBYT:   000FF (00068) (02867) (03937) (03961)
MWDONE:     00001 (01481) (01489) (01496)
MWLAE:      0001F (01572) (01578)
MWLF$A:     063CA (01512) (01693)
MWLF$APS:   062EC (00126) (01509) (01521) (01701)
MWLF$APU:   061D8 (00125) (01298)
MWLF$I:     063DE (01508) (01699)
MWLF$S:     00002 (01509) (01523) (01525)
MWLF$SA:    00000 (01294) (01301) (01501)
MWLF$SZ:    00006 (01295) (01501)
MWLF$Z:     00100 (01511) (01701)
MWLFAE:     00039 (01402) (01448)
MWLFAW:     00053 (01386) (01447)
MWLFD:      0004D (01427) (01441)
MWLFRE:     00017 (01337) (01446)
MWLFRW:     00051 (01316) (01445) (01445)
MWLFSE:     00010 (01318) (01327)
MWLOC:      0004B (01632) (01638)
MWLOD:      0004C (01620) (01640)
MWLOE:      00032 (01603) (01608)
MWLOT:      0002A (01570) (01598)
MWLQSSSM:   00660 (00127) (02025)
MWLSC:      00015 (01546) (01554) (01556)
MWLSE:      00012 (01547) (01551)
MWLT:       00020 (01580)
MWSS$1:     06671 (02025) (02030)
NMWDONE:    0007C (01490) (01494)
OQE:        00010 (00071) (02151) (02166) (02547) (02563) (02841) (02856)
P2120$:     0618C (00121) (01177) (01184) (01251)
P2120$2:    0000F (01194) (01223)
P2120$A:    061C0 (01180) (01244)
P2120$I:    061C8 (01176) (01250)
P2120$S:    00002 (01177) (01197)
P2120$Z:    0004A (01179) (01251)
PRDONE:     00043 (01129) (01149) (01158)
PRTRB$:     060F8 (00120) (01066)
PRTRB$SA:   00000 (01063) (01069) (01162)
PRTRB$SZ:   00046 (01064) (01162) (01163)
PTS$:       06A30 (00146) (02998) (03005) (03158)
PTS$1:      06A70 (03047) (03088) (03127)
PTS$A:      06AA2 (03001) (03153)
PTS$I:      06AAC (02997) (03157)
PTS$O:      0002A (03016) (03115)
PTS$S:      00020 (02990) (03119)
PTS$Z:      00004 (03000) (03158)
PTU$:       0694C (00145) (02763)
PTU$0:      00007 (02770) (02786)
PTU$1:      00010 (02776) (02789)
PTU$10:     00061 (02919) (02922) (02923)
PTU$11:     00063 (02910) (02925)
PTU$12:     00068 (02930) (02944)
```

DEFINE TOP OF MODULE

```
PTU$2:      00015 (02782) (02796)
PTU$3:      0003E (02784) (02798) (02870)
PTU$4:      00041 (02780) (02791) (02874)
PTU$5:      00019 (02793) (02802)
PTU$6:      0001F (02804) (02808)
PTU$7:      00021 (02807) (02813)
PTU$8:      00044 (02866) (02884)
PTU$9:      0005A (02913) (02920)
PTU$SA:     00000 (02760) (02770) (02955)
PTU$SZ:     00072 (02761) (02955) (02956)
PTU$W1:     0002F (02840) (02841)
PTU$W2:     00039 (02855) (02856)
QOUT:       00042 (02215) (02220) (02226) (02229)
RCASE1:     00036 (01117) (01142)
RCASE2:     0003E (01120) (01151)
RCASE3:     00024 (01122)
RIGHT:      0001E (01115) (01134) (01140)
S1011$:     035EA (00073)
SHFTL$R5:   007AE (00074)
SIN$:       03192 (00076)
SINCOS$T:   023FA (00075)
SVT$:       00302 (00077) (01645) (03960)
SVT$UN1:    003CE (00078) (02294) (02626) (03034) (03058) (03379)
SYS$FLGS:   1FFCE (00079) (02033)
TAPQTH:     06ABC (03060) (03160)
TAPQVL:     06ADC (03064) (03177)
TEM$0:      007B4 (00081)
TOE$:       021FE (00082)
TOE$CUR:    06EE2 (00098) (03972)
TOE$PTR:    00288 (00083) (00096)
V1100K$:    060C6 (00116) (00974) (01036)
V1100K$2:   0000A (00979) (01008)
V1100K$A:   060E6 (00970) (01026)
V1100K$I:   060EE (00966) (01033)
V1100K$Z:   00030 (00969) (01036)
V320$:      05FB6 (00111) (00671) (00788)
V3200$6:    0001D (00677) (00744)
V3200$A:    06F12 (00667) (00782)
V3200$I:    06F1A (00663) (00707)
V3200$Z:    00080 (00666) (00788)
VALBUF$:    01C5A (00085)
VAPC$:      06692 (00130) (02126)
VAPC$SA:    00000 (02124) (02133) (02257)
VAPC$SZ:    00059 (02125) (02257)
VAPC$W1:    00006 (02150) (02151)
VAPC$W2:    00010 (02165) (02166)
VAPCI$W1:   00006 (02546) (02547)
VAPCI$W2:   00010 (02562) (02563)
VKTOA$:     06038 (00115) (00841)
VKTOA$SA:   00000 (00838) (00843) (00940)
VKTOA$SZ:   00043 (00839) (00940) (00941)
VLTSY$:     05EF2 (00110) (00545)
VLTSY$SA:   00000 (00542) (00547) (00646)
```

DEFINE TOP OF MODULE

```
  VLTSY$SZ:  0005E (00543) (00646) (00647)
  VSMA2$:    02070 (00086)
  W$:        00002 (00088) (00109) (00114) (00119) (00124) (00129) (00134) (00139) (00144) (00149)
                   (00154) (00162) (00165) (02069)
  X23RD:     00032 (00854) (00855) (00913)
  X4TH:      0002D (00856) (00906)
  X5TH:      0002E (00857) (00905) (00907)
  X6TH:      00027 (00859) (00899)
  X7TH:      00029 (00860) (00898) (00901)
  X8TH:      00023 (00862) (00894)
  XFL$01:    0192C (00090) (02077)
  ZERO:      007BA (00092)

LINES WITH ERRORS:    0  (MAP VERSION 000101.10)   E-    0
```

TABLE OF CONTENTS

```
                (00003) ;SYMBOL DEFINITIONS
                (00004) ;  THE IOS CODE FITS IN THE HOLE AT $4000 - $49FF.  THE CSPU CODE
                (00005) ;     GOES IN THE HOLE AT $3000 - $3FFF.
                (00006) ;
00000001        (00007) H$ = 1          ;HALF-WORD ADDRESS INCREMENT
00000002        (00008) W$ = 2          ;FULL-WORD ADDRESS INCREMENT
                (00009) ;
0000000A        (00010) ACQTHR = 10             ;SYNC-SEARCH ACQUISITION THRESHOLD: NUMBER OF
                (00011) ;                       ; FRAMES OF CONSECUTIVE GOOD SYNC BITS NEEDED
                (00012) ;                       ; TO ACQUIRE SYNC.
00007FFF        (00013) BITS15 = 0"77777"       ;15-BIT MASK
                (00014) ;
                (00015) ;BIT-MASKS SHIFTED 1 PLACE LEFT FOR USE IN THE CORRECT ROUTINE
00000002        (00016) C0 = 2
00000004        (00017) C1 = 4
00000008        (00018) C2 = 8
00000010        (00019) C3 = 16
0000000E        (00020) C210 = C2+C1+C0
00000018        (00021) C32 = C3+C2
0000001C        (00022) C321 = C3+C2+C1
                (00023) ;
                (00024) ;DECODING TABLE ADDRESSES: THESE ARE AT THE BEGINNING OF BBN300.MSO
00005D00        (00025) BBTAB = $5D00           ;TABLE FOR BASEBAND RESIDUAL SAMPLES
00005D10        (00026) DKTAB1 = BBTAB + 8*W$   ;TABLES FOR K1-K8
00005D50        (00027) DKTAB2 = DKTAB1 + 32*W$
00005D90        (00028) DKTAB3 = DKTAB2 + 32*W$
00005DD0        (00029) DKTAB4 = DKTAB3 + 32*W$
00005DF0        (00030) DKTAB5 = DKTAB4 + 16*W$
00005E10        (00031) DKTAB6 = DKTAB5 + 16*W$
00005E30        (00032) DKTAB7 = DKTAB6 + 16*W$
00005E40        (00033) DKTAB8 = DKTAB7 + 8*W$
00005E50        (00034) EDTAB = DKTAB8 + 8*W$   ;TABLE FOR GAIN
00005ED0        (00035) DTQTAB = EDTAB + 64*W$  ;TABLE FOR TAP
                (00036) ;
                (00037) ;SNAP EXEC INTERRUPT ROUTINE ADDRESSES
00004022        (00038) D16$INT1 = $4022        ;MODEM SCROLL INT 1 (TMODEM)
00004030        (00039) D16$INT2 = $4030        ;MODEM SCROLL INT 2 (RMODEM)
000040FA        (00040) D22$INT1 = $40FA        ;ADM INT 1
0000411E        (00041) D23$INT1 = $411E        ;ADAM INT 1
                (00042) ;
00007E8         (00043) FDT$ = $7E8             ;START OF NONARRAY FUNCTION DISPATCH TABLE
                (00044) ;GFLAG SET/CLEAR CONSTANTS
00000020        (00045) SET = 0"40"
0000000C        (00046) CLR = 0
00000004        (00047) G0 = 4
00000005        (00048) G1 = 5
00000006        (00049) G2 = 6
00000007        (00050) G3 = 7
                (00051) ;
                (00052) ;HISTOGRAM TABLES (384. HALFWORDS ON BUS 1 ABOVE RDABB)
0000BC9E        (00053) PHIST = 48286                   ;PITCH
0000BCDE        (00054) THIST = PHIST + 64*H$           ;TAP
0000BCEE        (00055) GHIST = THIST + 16*H$           ;GAIN
```

SYMBOL DEFINITIONS

```
0000BD2E (00056) K1HIST = GHIST + 64*H$          ;K1-K8
0000BD4E (00057) K2HIST = K1HIST + 32*H$
0000BD6E (00058) K3HIST = K2HIST + 32*H$
0000BD8E (00059) K4HIST = K3HIST + 32*H$
0000BD9E (00060) K5HIST = K4HIST + 16*H$
0000BDAE (00061) K6HIST = K5HIST + 16*H$
0000BDBE (00062) K7HIST = K6HIST + 16*H$
0000BDC6 (00063) K8HIST = K7HIST + 8*H$
         (00064) ;
         (00065) ;MODEM-SCROLL INPUT DATA WORD BITS
00000080 (00066) IDW$ONHK = $80          ;LOCAL HANDSET HOOKSWITCH STATE
00000040 (00067) IDW$SI = $40            ;SIGNAL RATE INDICATOR
00000020 (00068) IDW$IC = $20            ;INCOMING CALL
00000010 (00069) IDW$CS = $10            ;CLEAR TO SEND
00000008 (00070) IDW$DM = 8              ;DATA MODE
00000004 (00071) IDW$RR = 4              ;RECEIVER READY
00000002 (00072) IDW$SQ = 2              ;SIGNAL QUALITY
00000001 (00073) IDW$RD = 1              ;RECEIVE DATA
         (00074) ;
00000502 (00075) ISVT$ = $502            ;START OF INTEGER SCALAR TABLE IN SNAP EXEC
00000004 (00076) LOSETHR = 4             ;SYNC-SEARCH LOSE-SYNC THRESHOLD: NUMBER OF
         (00077) ;                       ; SYNC ERRORS WITHOUT 2 CONSECUTIVE GOOD-SYNC
         (00078) ;                       ; FRAMES NEEDED TO LOSE SYNC.
         (00079) ;
         (00080) ;MODEM-SCROLL OUTPUT DATA WORD BITS
00000008 (00081) ODW$TR = 8              ;TERMINAL READY
00000004 (00082) ODW$RS = 4              ;REQUEST TO SEND
00000002 (00083) ODW$SR = 2              ;SIGNALING RATE
00000001 (00084) ODW$SD = 1              ;SEND DATA
         (00085) ;
0000000A (00086) PITCHL = 5*2            ;PITCH DECODER LOWER LIMIT (LEFT SHIFTED 1)
0000004C (00087) PITCHU = 38*2           ;PITCH DECODER UPPER LIMIT (LIKEWISE)
0000FBE8 (00088) RAND$L = $00BE8         ;LOWER AND UPPER LIMITS FOR ERRPTR
00001C77 (00089) RAND$U = $01C77         ; IN RMI$SIM
0001FFCE (00090) SYS$FLGS = $1FFCE       ;MAP SYSTEM FLAGS REGISTER IN PSEUDO-MEMORY
0000000C (00091) TMBITS = ODW$TR+ODW$RS  ;CONTROL BITS USED FOR TRANSMIT MODEM
         (00092) ;
         (00093) ;SYNCSTOP IS A "REGISTER" IN THE AOAM AND AOM USED FOR CSPU-SCROLL
         (00094) ;  COMMUNICATION.
         (00095)         OPADD SYNCSTOP,(16 .LS. 10) + (26 .LS. 5)+4
         (00096) ;
         (00097) ;THE SAF (SET ADDRESS FIELD) INSTRUCTION IS ONE OF THE NEW ONES ADDED
         (00098) ;    TO THE CSPU BY THE "REV. 18" MICROCODE REVISION.
         (00099)         OPADD SAF,(1 .LS. 14) + (29 .LS. 8) + $EF
         (00100)
```

```
                    (00101) ;PATCHES TO NON-ARRAY FUNCTION DISPATCH TABLE
            00008DA (00102) #L = FDT$ + (W$ * 121)
008DA 00003CC4 (00103)          ADDR    PROTECT         ;FCB 121 (SNAP FUNCTION PROTCT)
008DC 00003E08 (00104)          ADDR    CORRECT         ;FCB 122 (CORECT)
008DE 00003F8A (00105)          ADDR    MPGSC           ;FCB 123 (MPGSC)
008E0 0000495C (00106)          ADDR    ADAMINT         ;FCB 124 (IADINT)
008E2 00003B08 (00107)          ADDR    TMODEMINT       ;FCB 125 (ITMINT)
008E4 00003B5A (00108)          ADDR    RMODEMINT       ;FCB 126 (IRMINT)
008E6 00003C80 (00109)          ADDR    AOMINT          ;FCB 127 (IDAINT)
                    (00110) ;
                    (00111) ;
```



```
                    (00112) ;PATCHES TO INTERRUPT SERVICE ROUTINES
                    (00113) ;  THESE PATCHES MAKE SCROLL INTERRUPTS GO TO OUR OWN ROUTINES.
                    (00114) ;  NOTE THAT LOADING THIS FILE WILL MAKE THE NORMAL ADAM/AOM/IOS-2 SNAP
                    (00115) ;    FUNCTIONS INOPERATIVE!
            04022 (00116) #L = D16$INT1
04022 06003B08 (00117) #1:      CALL    R0,TMODEMINT
04024 0F70     (00118)          RET
04025 0800     (00119)          EVEN
04026 00004022 (00120)          JMP     #1
                    (00121) ;
            04030 (00122) #L = D16$INT2
04030 0600 3B5A (00123) #1:     CALL    R0,RMODEMINT
04032 0F70     (00124)          RET
04033 0800     (00125)          EVEN
04034 00004030 (00126)          JMP     #1
                    (00127) ;
            040FA (00128) #L = D22$INT1
040FA 06003C80 (00129) #1:      CALL    R0,AOMINT
040FC 0F70     (00130)          RET
040FD 0800     (00131)          EVEN
040FE 000040FA (00132)          JMP     #1
                    (00133) ;
            0411E (00134) #L = D23$INT1
0411E 0600 495C (00135) #1:     CALL    R0,ADAMINT
04120 0F70     (00136)          RET
04121 0800     (00137)          EVEN
04122 0000411E (00138)          JMP     #1
                    (00139)
```

```
                (00140) ;I/O BUFFER DEFINITIONS
                (00141) ; EXCEPT AS NOTED, ALL BUFFERS ARE ON BUS 1, ARE FIXED-POINT, 16 BITS,
                (00142) ;  AND HAVE SPACING OF 1 HALF-WORD.  THE SOURCE/SINK BUFFERS HAVE
                (00143) ;  ASSIGNED BID NUMBERS, BUT THE OTHERS DON'T.  THEREFORE,  IN THESE
                (00144) ;  I/O ROUTINES, WE USE THE ABSOLUTE ADDRESSES GIVEN BELOW (WHICH ARE
                (00145) ;  TAKEN FROM THE VOCODER INITIALIZATION TYPEOUT).
                (00146) ;   **** THEREFORE, IF THE BUFFER CONFIGURATIONS SHOULD CHANGE, THESE
                (00147) ;   SYMBOLS WILL NEED TO BE MODIFIED. ****
                (00148) ;
     00000084   (00149) SBLNGTH = 180              ;LENGTH OF ALL SPEECH DATA BUFFERS
     0000001E   (00150) SL6=SBLNGTH/6              ;1/6 LENGTH FOR A/D/A COPY ROUTINES
     00000105   (00151) MBLNGTH = 261              ;LENGTH OF ALL MODEM DATA BUFFERS
     0000002B   (00152) ML6 = 258/6                ;1/6 LENGTH FOR TMODEM COPY ROUTINES
                (00153) ;
                (00154) ;NEW BUFFERS:
     0000A75A   (00155) TBTA = 42842               ;TBITS A BUFFER (ALL LENGTHS = MBLNGTH)
     0000A86C   (00156) TBTB = 43104               ;TBITS B BUFFER
     0000A966   (00157) TBTC = 43366               ;FAKE TBITS "SILENCE" DATA
     0000AA6C   (00158) RBTA = 43628               ;RBITS A BUFFER
     0000AB72   (00159) RBTB = 43890               ;RBITS B BUFFER
                (00160) ;
     0000AC78   (00161) TADBA = 44152              ;A/D A BUFFER
     0000AD2C   (00162) TADBB = 44332              ;A/D B BUFFER
     0000ADE0   (00163) TADBC = 44512              ;A/D "ON-HOOK" TONE DATA
     0000AE94   (00164) TSRA =  44692              ;TSOURCE A BUFFER
     0000AF48   (00165) TSRB =  44872              ;TSOURCE B BUFFER
     0000AFFC   (00166) TSNKA = 45052              ;TSINK A BUFFER (LENGTH = 71 FIXED)
     0000B044   (00167) TSNKB = 45124              ;TSINK B BUFFER
     0000B08C   (00168) TSNKC = 45196              ;(NO LONGER USED)
     0000B0D4   (00169) TMDMA = 45268              ;TMODEM A BUFFER
     0000B1DA   (00170) TMDMB = 45530              ;TMODEM B BUFFER
                (00171) ;
     0000B2E0   (00172) RMDMA = 45792              ;RMODEM A BUFFER
     0000B3E6   (00173) RMDMB = 46054              ;RMODEM B BUFFER
     0000B4EC   (00174) RMDMC = 46316              ;RMODEM C BUFFER
     0000B5F2   (00175) RSSPF = 46578              ;SYNC SEARCH PREVIOUS FRAME (LENGTH=MBLNGTH)
     0000B6F8   (00176) RSSSS = 46840              ;SYNC SEARCH SUM-SYNC COUNTS (LENGTH=MBLNGTH)
     0000B7FE   (00177) RSRA =  47102              ;RSOURCE A BUFFER (LENGTH = 71 FLT PT)
     0000B88C   (00178) RSRB =  47244              ;RSOURCE B BUFFER
     0000B91A   (00179) RSNKA = 47386              ;RSINK A BUFFER
     0000B9C8   (00180) RSNKB = 47566              ;RSINK B BUFFER
     0000BA82   (00181) RSNKC = 47746              ;D/A "SILENCE" DATA
     0000BB36   (00182) RDABA = 47926              ;D/A A BUFFER
     0000BBEA   (00183) RDABB = 48106              ;D/A B BUFFER
                (00184) ;
                (00185) ;INITIALIZE ALTERNATING SYNC BITS IN TMODEM BUFFERS
     0000B0D4   (00186) #L = TMDMA
0B0D4 000C      (00187)         DATA    0 + TMBITS
     0000B1DA   (00188) #L = TMDMB
0B1DA 000D      (00189)         DATA    1 + TMBITS
                (00190) ;
                (00191) ;INITIALIZE FINAL (UNUSED) BITS IN TMODEM BUFFERS.  IF THEY WERE LEFT
                (00192) ;  ALONE AND HAPPENED TO ALTERNATE, WE COULDN'T GET SYNC.
```

```
                ;;;;B1D8 (;;193) #L = TMDMA + MBLNGTH - 1
        ;B1D8   ;;0C     (;;194)         DATA    0 + TMBITS
                ;;;;B2DE (;;195) #L = TMDMB + MBLNGTH -1
        ;B2DE   ;;0C     (;;196)         DATA    0 + TMBITS
                         (;;197)
```

```
                (00198) ;INTEGER SCALAR TABLE DEFINITIONS FOR I/O ROUTINES
                (00199) ;
                (00200) ; T/R SOURCE/SINK BUFFER FLAGS.  ALL ARE INITED TO 0 AND USE THE
                (00201) ;   CONVENTION THAT 0 = NON-FULL AND NZ = NON-EMPTY.
                (00202) ;
     00000534   (00203) TSRFA = ISVT$+50                ;TSOURCE FLAGS A & B
     00000535   (00204) TSRFB = ISVT$+51
     00000536   (00205) TBTFA = ISVT$+52                ;TBITS FLAGS A & B
     00000537   (00206) TBTFB = ISVT$+53
     0000053B   (00207) RBTFA = ISVT$+57                ;RBITS FLAGS A & B
     0000053C   (00208) RBTFB = ISVT$+58
     0000053D   (00209) RSNFA = ISVT$+59                ;RSINK FLAGS A & B
     0000053E   (00210) RSNFB = ISVT$+60
     00000534   (00211) #L=TSRFA
00534 0000      (00212)         DATA 4D'0'
 ...
     0000053B   (00213) #L=RBTFA
0053B 0000      (00214)         DATA 4D'0'
 ...
                (00215) ;
                (00216) ;I/O ROUTINE BUFFER/FLAG POINTER OFFSETS (AND INIT VALUES)
                (00217) ;THESE ARE USED BY THE 4 INTERRUPT ROUTINES TO KEEP TRACK OF
                (00218) ;  THE PROPER BUFFERS AND FLAGS.
                (00219) ;
                (00220) ;USED IN ADAMINT:
     00000542   (00221) ADPO =  ISVT$+64                ;A/D OFFSET (0)
     00000543   (00222) TSRPO = ISVT$+65                ;TSOURCE OFFSET (-2)
                (00223) ;USED IN TMODEMINT:
     00000544   (00224) TBTPO = ISVT$+66                ;TBITS OFFSET (0)
     00000545   (00225) TMPO =  ISVT$+67                ;TMODEM OFFSET (0)
                (00226) ;USED IN RMODEMINT:
     00000546   (00227) RMPO =  ISVT$+68                ;RMODEM OFFSET (0)
     00000547   (00228) RBTPO = ISVT$+69                ;RBITS OFFSET (-2)
                (00229) ;USED IN AOMINT:
     00000548   (00230) RSNPO = ISVT$+70                ;RSINK OFFSET (0)
     00000549   (00231) DAPO =  ISVT$+71                ;D/A OFFSET (0)
     00000542   (00232) #L=ADPO
00542 0000      (00233)         DATA    0,-2,0,0,0,-2,0,0
00543 FFFE
00544 0000
00545 0000
00546 0000
00547 FFFE
00548 0000
00549 0000
                (00234) ;
                (00235) ; I/O BUFFER ERROR COUNTERS.  ALL ARE INITED TO 0.
                (00236) ;
     0000054A   (00237) TADFDC = ISVT$+72                ;A/D FRAME DISCARD COUNTER
     0000054B   (00238) TMFFC = ISVT$+73                 ;TMODEM FAKE FRAME COUNTER
     0000054C   (00239) RMFDC = ISVT$+74                 ;RMODEM FRAME DISCARD COUNTER
     0000054D   (00240) RSNNR = ISVT$+75                 ;RSINK DATA NOT READY COUNTER
                (00241) ;
```

```
                (00242) ; MISC. INTEGER MEMORIES (AND INIT VALUES)
                (00243) ;
        00000540 (00244) RUN =    ISVT$+62                  ;VOCODER RUN FLAG
                (00245) ; (63 FREE FOR MPITM USE)
        00000540 (00246) #L=RUN
  0540  0001     (00247)          DATA    1                 ;FOR DEBUGGING, MUST BE "ON"
                (00248) ;
        0000054E (00249) TFRCTR = ISVT$+76                  ;TRANSMITTER FRAME COUNTER (0)
        0000054F (00250) RFRCTR = ISVT$+77                  ;RECEIVER FRAME COUNTER (0)
        00000550 (00251) RLSCTR = ISVT$+78                  ;RCVR LOST-SYNC COUNTER (0)
        00000551 (00252) RONHK = ISVT$+79                   ;LOCAL HANDSET ON-HOOK STATE (0)
        00000552 (00253) RSYNC = ISVT$+80                   ;STATE OF RMODEM SYNC (0)
        00000553 (00254) RBOFO = ISVT$+81                   ;BEGINNING OF FRAME OFFSET: SYNC BIT
                (00255) ; (82 FREE FOR MPITM)               ; POSITION IN RMODEM BUFFER
        0000054A (00256) #L=TADFDC
  054A  0000     (00257)          DATA    9D'0'
   ...
                (00258) ;
                (00259) ; DEBUGGING/DEMONSTRATION AIDS
                (00260) ;
        0000057D (00261) RNOCOR = ISVT$+123                 ;SET TO NZ TO TELL CORRECT NOT TO
                (00262) ; (124 FREE FOR MPITM)              ; CORRECT CHANNEL ERRORS (0)
        0000057F (00263) RERSIM = ISVT$+125                 ;SET TO N>0 TO TELL RMODEMINT TO
                (00264) ; (126 FREE FOR MPITM)              ; SIMULATE N CHANNEL ERRORS PER FRAME(0)
        00000581 (00265) VSTATE = ISVT$+127                 ;VOCODER STATE, IS DISPLAYED BY DAC0 (0)
        0000057D (00266) #L=RNOCOR
  057D  0000     (00267)          DATA    5D'0'
   ...
                (00268)
```

MODEM INTERFACE AND SYSTEM CLOCKS PROGRAMS,

```
                    (00269) "MODEM INTERFACE AND SYSTEM CLOCKS PROGRAMS
                    (00270) ; JJW, 18-SEP-79
                    (00271) ;
                    (00272) ;THERE ARE TWO ENTRY POINTS:
                    (00273) ;  CLKSET: CLOCK RATE SETTING (START ADDRESS = 0)
                    (00274) ;    THIS ENTRY SETS THE DATA AND SAMPLE CLOCKS AND STARTS THEM.
                    (00275) ;    THE 16-BIT CLOCK RATE VALUE IS A LITERAL IN THE INSTRUCTION AT
                    (00276) ;    CLKGO.  (TRIED BINDING AN INTEGER TABLE VALUE HERE, BUT BECAUSE
                    (00277) ;    WE USE NO BIDS, LOAD$ DOES NO BINDING AT ALL!)
                    (00278) ;
                    (00279) ;  RUNMOD: SET CLOCKS AND RUN MODEM TRANSMIT/RECEIVE (START ADDRESS 1)
                    (00280) ;    THIS ENTRY SETS AND STARTS THE CLOCKS AND THEN RUNS THE MODEM
                    (00281) ;    INTERFACE PROGRAM.
                    (00282) ;
                    (00283) ;THESE ENDING ADDRESSES ARE TRUNCATED TO 15 BITS BECAUSE THE LLIT
                    (00284) ;  FIELD OF THE JNE INSTRUCTION IS ONLY THAT BIG.
          000033E4  (00285) RMA$END = (RMDMA+MBLNGTH-1) .AND. BITS15
          000034EA  (00286) RMB$END = (RMDMB+MBLNGTH-1) .AND. BITS15
          000035F0  (00287) RMC$END = (RMDMC+MBLNGTH-1) .AND. BITS15
          000031D8  (00288) TMA$END = (TMDMA+MBLNGTH-1) .AND. BITS15
          000032DE  (00289) TMB$END = (TMDMB+MBLNGTH-1) .AND. BITS15
                    (00290) ;
                    (00291) ;CONSTANT FOR SETTING CLOCK RATES TO MODEM=9600 BPS, SIGNAL
                    (00292) ;  SAMPLE RATE=6621 SAMPLES/SEC.
          0000ECC6  (00293) CLKRATES = $ECC6
                    (00294) ;
                    (00295) ;REGISTER AND FLAG USAGE:
                    (00296) ;  R0 - RCVR BUFFER SWITCH, SET TO MOD$RA, MOD$RB, OR MOD$RC
                    (00297) ;  R1 - RCVR ADDRESS POINTER
                    (00298) ;  R2 - XMTR ADDRESS POINTER
                    (00299) ;  P1 SET SIGNIFIES RCVR DATUM READY
                    (00300) ;  P2 SET SIGNIFIES XMTR READY FOR NEXT DATUM
                    (00301) ;  F1 IS USED AS XMTR BUFFER SWITCH: CLEAR=>TMODEMA, SET=>TMODEMB
                    (00302) ;  F2 IS USED TO REMEMBER WHICH START ADDRESS WAS USED.
                    (00303) ;  INT1 IS USED TO SIGNAL END OF TRANSMITTER BUFFER
                    (00304) ;  INT2 IS USED TO SIGNAL END OF RECEIVER BUFFER
                    (00305) ;
          00004800  (00306) $L = $4800                        ;BOS 1 ADR FOR RTMODEM IOS MODULE
          00000000  (00307) $A = 0
                    (00308)          EVEN
    04800 006E      (00309)          DATA    MOD$SZ,0                ;SCROLL PGM SIZE IN HW, BIDS 1:0
    04801 0000
                    (00310) ;
                    (00311) MOD$BGN: BEGIN  IOS2(RTMODEM)
                    (00312) ;
A00 04802 02300300  (00313) CLKSET: SF2,JUMP(CLKGO)                 ;ENTRY FOR SETTING CLOCKS ONLY
A01 04804 02300380  (00314) RUNMOD: CF2                             ;ENTRY FOR CLOCKS AND MODEM
A02 04806 030@ECC6  (00315) CLKGO:  LOAD(R0,CLKRATES)               ;SET DATA AND SAMPLE CLOCKS
A03 04808 04391800  (00316)         ADDL(R0,0,TP),JIFC(MOD$INIT,F2)
A05 0480C 06306380  (00317)         STOP, CF2
                    (00318) ;
A06 0480E 07000015  (00319) MOD$INIT: LOAD(R0,MOD$RA)               ;INIT RCVR BUFFER SWITCH
A07 04810 0842820F  (00320)         LOAD(R1,RMDMA-1(1),L)           ;INIT RCVR PTR
```

```
A08 04812 09300580 (00321)          CF1                          ;INIT XMTR BUFFER SWITCH
A09 04814 0A82B0D3 (00322)          LOAD(R2,TMDMA-1(1),L)        ;INIT XMTR PTR
A0A 04816 0C340C00 (00323) MOD$LOOP: JIFS(MOD$RCVR,P1)           ;P1=>RCVR HAS DATUM
A0B 04818 00000000
A0D 0481C 0E341C00 (00324)          JIFS(MOD$XMTR,P2)            ;P2=>XMTR READY FOR NEXT DATUM
A0F 04820 0A300000 (00325)          JUMP(MOD$LOOP)
                   (00326) ;
                   (00327) ;  MODEM RECEIVER
                   (00328) ;
A10 04822 11300600 (00329) MOD$RCVR: MW                          ;SET DIRECTION MODEM-TO-MAP
A11 04824 122C001C (00330)          JEQ(MOD$RB,R0,MOD$RB)        ;DISPATCH TO PROPER BUFFER
A13 04828 142C0022 (00331)          JEQ(MOD$RC,R0,MOD$RC)        ; FILLER-UPPER
A15 0482C 165A0001 (00332) MOD$RA:  ADD(R1,1,TM)                 ;INCR RPTR AND READ DATUM
A16 0482E 186833E4 (00333)          JNE(MOD$LOOP,R1,RMA$END)     ;FILLED RMODEMA BUFFER?
A17 04830 00000000
A19 04834 1A00001C (00334)          LOAD(R0,MOD$RB)              ;YES, SET SWITCH TO B
A1A 04836 1B42B3E5 (00335)          LOAD(R1,RMDMB-1(1),L)        ; AND RPTR TO RMODEMB
A1B 04838 0A304000 (00336)          INT2, JUMP(MOD$LOOP)         ; AND INT LINE 2 TO SAY FULL
                   (00337) ;
A1C 0483A 1D5A0001 (00338) MOD$RB:  ADD(R1,1,TM)                 ;SAME FOR RMODEMB
A1D 0483C 1E6834EA (00339)          JNE(MOD$LOOP,R1,RMB$END)
A1F 04840 20000022 (00340)          LOAD(R0,MOD$RC)
A20 04842 2142B4EB (00341)          LOAD(R1,RMDMC-1(1),L)
A21 04844 0A304000 (00342)          INT2, JUMP(MOD$LOOP)
                   (00343) ;
A22 04846 235A0001 (00344) MOD$RC:  ADD(R1,1,TM)                 ;AND FOR RMODEMC
A23 04848 246835F0 (00345)          JNE(MOD$LOOP,R1,RMC$END)
A25 0484C 26000015 (00346)          LOAD(R0,MOD$RA)
A26 0484E 2742B2DF (00347)          LOAD(R1,RMDMA-1(1),L)
A27 04850 0A304000 (00348)          INT2, JUMP(MOD$LOOP)
                   (00349) ;
                   (00350) ;  MODEM TRANSMITTER
                   (00351) ;
A28 04852 2A340A80 (00352) MOD$XMTR: MR, JIFS(MOD$TB,F1)         ;SET DIRECTION MAP-TO-MODEM
A29 04854 00000000
                   (00353) ;                                     ;F1 SET MEANS USE TMODEMB
A2B 04858 2C9A0001 (00354) MOD$TA:  ADD(R2,1,TM)                 ;SEND WORD FROM TMODEMA
A2C 0485A 2EA831D8 (00355)          JNE(MOD$LOOP,R2,TMA$END)     ;EMPTIED TMODEMA?
A2D 0485C 00000000
A2F 04860 3082B1D9 (00356)          LOAD(R2,TMDMB-1(1),L)        ;YES, RESET TPTR, FLIP F1,
A30 04862 0A302100 (00357)          SF1, INT1, JUMP(MOD$LOOP)    ; INT1 TO SAY BUFFER EMPTY
                   (00358) ;
A31 04864 329A0001 (00359) MOD$TB:  ADD(R2,1,TM)                 ;SAME FOR TMODEMB
A32 04866 34A832DE (00360)          JNE(MOD$LOOP,R2,TMB$END)
A33 04868 00000000
A35 0486C 3682B0D3 (00361)          LOAD(R2,TMDMA-1(1),L)
A36 0486E 0A302100 (00362)          CF1, INT1, JUMP(MOD$LOOP)
                   (00363) ;
    04870          (00364)          END
                   (00365) ;
          0000006E (00366) MOD$SZ = .L-MOD$BGN                   ;SIZE OF SCROLL PROGRAM IN HW
    04870 0000     (00367)          DATA    0                   ;DIR=FROM MAP (PROB MEANINGLESS)
    04871 0001     (00368)          DATA    $0001               ;#CHNLS=0 : AQ. MODE = DOUBLE (ALSO)
```

```
04872 0000      (00369)         DATA    0               ;BID3 : BID2
04873 0000      (00370)         DATA    0,0             ;NO CONTROL REGISTERS
04874 0000
04875 FFFF      (00371)         DATA    -1,-1,-1,-1,-1  ;NULL OFFSET, BUFFER CHAIN ANCHORS
04876 FFFF
04877 FFFF
04878 FFFF
04879 FFFF
                (00372)
```

```
                        (00373) "ADAM SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING
                        (00374) ; JJW, 20 SEPT 79
                        (00375) ;
                        (00376) ; WE SAMPLE ON CHANNEL 1, WHICH IS CONNECTED TO THE GTE SPI.
                        (00377) ; THIS PROGRAM IS FUNCTIONALLY IDENTICAL TO ADMSD, BUT BOUND AT
                        (00378) ;   ASSEMBLY TIME.
                        (00379) ;
                00002D2B (00380) ADA$END = (TADBA+SBLNGTH-1) .AND. BITS15
                00002DDF (00381) ADB$END = (TADBB+SBLNGTH-1) .AND. BITS15
                        (00382) ;
                00000002 (00383) ADCNTRL = 2             ;INTERNAL TRIGGER, EXTERNAL SAMPLING CLOCK,
                        (00384) ;                        ; CH1 CONTROLLED BY SR1, SHORT FLT PT
                        (00385) ;
                        (00386) ; REGISTER AND FLAG USAGE:
                        (00387) ;   R0 - CONTROLS MUX
                        (00388) ;   R1 - DATA ADDRESS POINTER
                        (00389) ;   F1 - THE ADAM USES F1 TO TURN SAMPLING ON/OFF
                        (00390) ;   INT1 - SIGNALS END OF BUFFER
                        (00391) ;
                000048A0 (00392) #L = $48A0                      ;BUS 1 ADR FOR ADPROG IOS MODULE
                00000000 (00393) #A = 0
                        (00394)         EVEN
     048A0 002C         (00395)         DATA AD$SZ,0             ;SCROLL PGM SIZE (HW), BIDS 1:0
     048A1 0000
                        (00396) ;
                        (00397) AD$BGN: BEGIN IOS2(ADPROG)
                        (00398) ;
A00 048A2 013006000     (00399)         PR                       ;SET DIRECTION ADAM-TO-MAP
A01 048A4 020000001     (00400)         LOAD(R0,$0001)           ;INIT MUX TO CHNL 1
A02 048A6 031900000     (00401)         ADD(R0,0,TP)
A03 048A8 043000100     (00402)         SF1                      ;START SAMPLING
A04 048AA 0542AC77      (00403) AD$LOOP: LOAD(R1,TADBA-1(1),L)   ;INIT RPTR TO A/D BUFFER A
A05 048AC 061900010     (00404) #1:     ADD(R0,$10,TP)           ;RELEASE S/H
A06 048AE 071000010     (00405)         SUB(R0,$10)
A07 048B0 085A00C1      (00406)         ADD(R1,1,T4)             ;READ SAMPLE, SEND TO BUFFER
A08 048B2 0A682D2B      (00407)         JNE(#1,R1,ADA$END)       ;FILLED BUFFER?
A09 048B4 00000000
A0B 048B8 0C343000      (00408)         JIFS(AD$STOP,SYNCSTOP),INT1  ;YES:  INT1 TO SAY FULL, AND CHECK
                        (00409) ;                                    ;  IF CSPU SAYS TO STOP
A0D 048BC 0542AD2B      (00410)         LOAD(R1,TADBB-1(1),L)    ;INIT RPTR TO A/D BUFFER B
A0E 048BE 0F1900010     (00411) #2:     ADD(R0,$10,TP)
A0F 048C0 101000010     (00412)         SUB(R0,$10)
A10 048C2 115A00C1      (00413)         ADD(R1,1,T4)
A11 048C4 12682DDF      (00414)         JNE(#2,R1,ADB$END)
A13 048C8 143030000     (00415)         JIFC(AD$LOOP,SYNCSTOP), INT1
A15 048CC 163006100     (00416) AD$STOP: CF1, STOP               ;STOP SAMPLING AND HALT
     048CE              (00417)         END
                        (00418) ;
                0000002C (00419) AD$SZ = #L-AD$BGN
     048CE 0002         (00420)         DATA 2                   ;DIR = TO MAP
     048CF 0101         (00421)         DATA $0101               ;#CHNLS=1 : AQ. MODE = DOUBLE
     048D0 0000         (00422)         DATA 0                   ;BID3 : BID2
     048D1 0003         (00423)         DATA 3,0                 ;LOAD 3 CONTROL REGS STARTING WITH 0
```

```
048D2 0000
048D3 0000      (00424)         DATA 0                  ;REG. 0: SAMPLE RATE 1 (FOR EXT CLK)
048D4 0000      (00425)         DATA 0                  ;REG. 1: SAMPLE RATE 2 NOT USED
048D5 0002      (00426)         DATA ADCNTRL            ;REG. 2: ADAM CONTROL BITS
048D6 FFFF      (00427)         DATA -1,-1,-1,-1,-1     ;NULL OFFSET, BUFFER CHAIN ANCHORS
048D7 FFFF
048D8 FFFF
048D9 FFFF
048DA FFFF
                (00428)
```

```
                        (00429) "AOM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT
                        (00430) ; JJW, 28 SEPT 79
                        (00431) ;
                        (00432) ;D/A CHANNEL 0 OUTPUTS AN INTERNALLY-GENERATED RAMP, WHILE CHANNEL 1
                        (00433) ;  (CONNECTED TO THE GTE SPI) OUTPUTS DOUBLE-BUFFERED DATA FROM BUS 1.
                        (00434) ;  THIS PROGRAM IS FUNCTIONALLY SIMILAR TO AOMID, BUT BOUND AT ASSEMBLY
                        (00435) ;  TIME.
                        (00436) ;
                  03BE9 (00437) DAA$END = (RDABA+SBLNGTH-1) .AND. BITS15
                  03C9D (00438) DAB$END = (RDABB+SBLNGTH-1) .AND. BITS15
                        (00439) ;
                  00002 (00440) DACNTRL = 0+2+0          ;SINGLE CHANNEL MODE, EXTERNAL CLOCK,
                        (00441) ;                        ; SHORT FLT PT
                        (00442) ; REGISTER AND FLAG USAGE:
                        (00443) ;   R0 - HOLDS RAMP VALUE FOR CH0 DAC
                        (00444) ;   R1 - DATA ADDRESS POINTER FOR CH1 DAC
                        (00445) ;   F1 - CAN BE USED FOR SCOPE SYNC
                        (00446) ;   INT1 - SIGNALS END OF BUFFER
                        (00447) ;
                  048E0 (00448) #L = $48E0                      ;BUS 1 ADR FOR DAPROG IOS MODULE
                  00000 (00449) #A = 0
                        (00450)         EVEN
048E0 002C              (00451)         DATA DA$SZ,0            ;SCROLL PGM SIZE (HW), BIDS 1:0
048E1 0000
                        (00452) ;
                        (00453) DA$BGN: BEGIN IOS2(DAPROG)
A00 048E2 01300680      (00454)         PW                      ;SET DIRECTION MAP-TO-AOM
A01 048E4 02000000      (00455) DA$LOOP: LOAD(R0,0)             ;INITIALIZE CH0 RAMP
A02 048E6 0342BB35      (00456)         LOAD(R1,RDABA-1(1),L)   ;INIT TPTR TO D/A BUFFER A
A03 048E8 04300100      (00457)         SF1                     ;F1 CAN BE USED FOR SCOPE SYNC
A04 048EA 0519000A      (00458) #1:     ADD(R0,10,TP)           ;OUTPUT RAMP TO CH0 DAC
A05 048EC 065A0001      (00459)         ADD(R1,1,TM)            ;GET SAMPLE FOR CH1 DAC
A06 048EE 08603BE9      (00460)         JNE(#1,R1,DAA$END)      ;EMPTIED BUFFER?
A07 048F0 00000000
A09 048F4 0A343180      (00461)         JIFS(DA$STOP,SYNCSTOP),INT1,CF1  ;YES:  INT1 TO SAY EMPTY, AND
                        (00462) ;                                ; CHECK IF CSPU SAYS TO STOP
A0B 048F8 0C000000      (00463)         LOAD(R0,0)              ;REINIT CH0 RAMP
A0C 048FA 0D42BBE9      (00464)         LOAD(R1,RDABB-1(1),L)   ;INIT TPTR TO D/A BUFFER B
A0D 048FC 0E300100      (00465)         SF1
A0E 048FE 0F19000A      (00466) #2:     ADD(R0,10,TP)
A0F 04900 105A0001      (00467)         ADD(R1,1,TM)
A10 04902 12603C9D      (00468)         JNE(#2,R1,DAB$END)
A11 04904 00000000
A13 04908 14303180      (00469)         JIFC(DA$LOOP,SYNCSTOP), INT1, CF1
A15 0490C 16300000      (00470) DA$STOP: STOP
    0490E               (00471)         END
                        (00472) ;
                  0002C (00473) DA$SZ = #L-DA$BGN
0490E 0000              (00474)         DATA 0                  ;DIR = FROM MAP
0490F 0001              (00475)         DATA $0001              ;#CHNLS=NULL : AQ. MODE = DOUBLE
04910 000F              (00476)         DATA 0                  ;BID3 : BID2
04911 0003              (00477)         DATA 3,0                ;LOAD 3 CONTROL REGS STARTING WITH 0
04912 000F
```

AOM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT

```
04913 0000      (00478)          DATA 0                      ;REG. 0: SAMPLE RATE
04914 0000      (00479)          DATA 0                      ;REG. 1 NOT USED
04915 0002      (00480)          DATA DACNTRL                ;REG. 2: AOM CONTROL BITS
04916 FFFF      (00481)          DATA -1,-1,-1,-1,-1         ;NULL OFFSET, BUFFER CHAIN ANCHORS
04917 FFFF
04918 FFFF
04919 FFFF
0491A FFFF

                (00482)
```

AOM VSTATE-ONLY D/A PROGRAM

```
                        (00483) ;AOM VSTATE-ONLY D/A PROGRAM
                        (00484) ; JJW, 7 OCT 79
                        (00485) ;
                        (00486) ; SPECIAL DIAGNOSTIC PROGRAM FOR DEBUGGING THE NOT-YET-REAL-TIME
                        (00487) ;   VOCODER.  D/A CHANNELS 0 AND 1 OUTPUT THE CONTENTS OF THE VOCODER
                        (00488) ;   STATE WORD.  RUNS ON AOM INTERNAL CLOCK AT A 50 USEC PERIOD ON
                        (00489) ;   EACH CHANNEL
                        (00490) ;
              00004920  (00491) #L = $4920                        ;BUS 1 ADR FOR AVPROG IOS MODULE
              00000000  (00492) #A=0
              00000001  (00493) AVCNTRL = 1+0+0                   ;DUAL CHNL, INT CLK, SHORT FL. PT.
                        (00494)         EVEN
     04920 0010         (00495)         DATA    AV$SZ,0          ;PGM SIZE, BIDS 1:0
     04921 0000
                        (00496) ;
                        (00497) AV$BGN: BEGIN IOS2(AVPROG)
A00  04922 013E0680     (00498)         PW                       ;SET DIRECTION MAP-TO-AOM
A01  04924 02020581     (00499)         LOAD(R0,VSTATE(1),L)     ;VSTATE WORD ADR ON BUS 1
A02  04926 031A0000     (00500) #1:     ADD(R0,0,TM)             ;OUTPUT IT ON CH0
A03  04928 041A0000     (00501)         ADD(R0,0,TM)             ;WE HAVE TO DO CH1 ALSO
A04  0492A 06301000     (00502)         JIFC(#1,SYNCSTOP)        ;LOOP UNLESS CSPU SAYS TO STOP
A05  0492C 00000000
A07  04930 00306000     (00503)         STOP
     04932              (00504)         END
                        (00505) ;
              00000010  (00506) AV$SZ = #L-AV$BGN
     04932 0000         (00507)         DATA    0                ;DIR=FROM MAP
     04933 0001         (00508)         DATA    $0001            ;#CHNLS=NULL ; AQ MODE = DOUBLE
     04934 0000         (00509)         DATA    0                ;BID3 : BID2
     04935 0003         (00510)         DATA    3,0              ;LOAD 3 CONTROL REGS STARTING WITH 0
     04936 0000
     04937 0077         (00511)         DATA    (4800/20)/2-1    ;REG. 0: SAMPLE RATE, 50 USEC
     04938 0000         (00512)         DATA    0                ;REG. 1: NOT USED
     04939 0001         (00513)         DATA    AVCNTRL          ;REG. 2: AOM CONTROL BITS
     0493A FFFF         (00514)         DATA    -1,-1,-1,-1,-1   ;NULL OFFSET, BUFFER CHAIN ANCHORS
     0493B FFFF
     0493C FFFF
     0493D FFFF
     0493E FFFF
                        (00515)
```

```
                        (00516) ;ADAMINT: ADAM INTERRUPT SERVICE ROUTINE
                        (00517) ; JJW, 3-OCT-79
                        (00518) ;
                        (00519) ;ADAMINT COPIES A/D DATA FROM AN A/D BUFFER TO A TSOURCE BUFFER.  IF
                        (00520) ;  THE LOCAL HANDSET IS ON-HOOK, IT COPIES A LOW-AMPLITUDE "ON-HOOK"
                        (00521) ;  TONE FROM TADBC INSTEAD, TO CONVEY THIS FACT TO THE REMOTE LISTENER.
                        (00522) ;  IF THE TSOURCE BUFFER FLAG SHOWS THE BUFFER ISN'T YET AVAILABLE, THE
                        (00523) ;  FRAME OF A/D DATA IS SIMPLY DISCARDED.
                        (00524) ;
            003F4958    (00525) $L = $4958                      ;BUS 1 ADR FOR ADAMINT CSPU MODULE
                        (00526) ;POINTERS TO A/D BUFFERS AND TSOURCE FLAGS AND BUFFER-COPYING ROUTINES.
                        (00527) ;THESE BUFFERS AND FLAGS ARE ACCESSED BY MEANS OF POINTER OFFSETS,
                        (00528) ;  WHICH ARE IN THE INTEGER SCALAR TABLE.
                        (00529) ;       ADPO IS THE A/D BUFFER POINTER OFFSET (-2,0; INIT TO 0 SO THAT
                        (00530) ;          IT GETS SET TO TADBA ON THE FIRST CALL)
                        (00531) ;       TSRPO IS THE TSOURCE BUFFER/FLAG POINTER OFFSET (-2,0;
                        (00532) ;         INIT TO -2 SO THAT IT POINTS TO TSOURCEA ON THE 1ST CALL)
                        (00533) ;
                        (00534)         EVEN
  04950 0000AC76        (00535)         ADDR    TADBA-2         ;A/D BUFFER A (-2 FOR THE BNOVEL)
  04952 0000AD2A        (00536) ADBPTR: ADDR    TADBB-2         ;A/D BUFFER B
                        (00537) ;
  04954 0000498A        (00538)         ADDR    AD$CPYA         ;ROUTINE TO COPY TO TSOURCE A BUFFER
  04956 00004994        (00539) TSRBPTR: ADDR   AD$CPYB         ;ROUTINE TO COPY TO TSOURCE B BUFFER
                        (00540) ;
  04958 00000534        (00541)         ADDR    TSRFA           ;TSOURCE FLAG A
  0495A 00000535        (00542) TSRFPTR: ADDR   TSRFB           ;TSOURCE FLAG B
                        (00543) ;
                        (00544)         EVEN
  0495C F84DFFCE        (00545) ADAMINT: MOVLM  SET+G2,SYS$FLGS  ;SET G2 TO MARK START OF ADAMINT
  04960 F0100542        (00546)         MOVMR   R1,ADPO         ;R1 GETS A/D BUFF PTR OFFSET
  04960 9411FFFE        (00547)         XORIR   R1,-2           ;SWITCH A/D BUFFERS (0,-2,0,...)
  04962 E0100542        (00548)         MOVRM   R1,ADPO
  04964 E2000540        (00549)         CMPMZ   RUN             ;IS THE VOCODER RUNNING?
  04966 80104982        (00550)         JMP     AD$RTN,EQZ      ;NO, SO DO NIL
  04968 F0300543        (00551)         MOVMR   R3,TSRPO        ;R3 GETS TSOURCE OFFSET
  0496A E286495A        (00552)         CMPMZ   @TSRFPTR(R3)    ;IS THE TSOURCE BUFFER EMPTY?
  0496C 81104986        (00553)         JMP     AD$DISC,NEZ     ;NO: DISCARD A/D DATA
  0496E 90A24952        (00554)         MOVIR   R2,@ADBPTR(R1)  ;YES: GET A/D BUFFER-1 PTR
  04970 E2000551        (00555)         CMPMZ   RONHK           ;IS LOCAL HANDSET ON HOOK?
  04972 1A10            (00556)         SKPL    EQZ
  04973 9020ADDE        (00557)         MOVIR   R2,TADBC-2      ;YES: XMIT "ON-HOOK" TONE INSTEAD OF A/D
  04975 0000            (00558)         EVEN
  04976 86864956        (00559)         CALL    R0,@TSRBPTR(R3)  ;COPY FROM A/D BUFFER TO TSOURCE BUFFER
  04978 90100001        (00560)         MOVIR   R1,1
  0497A E096495A        (00561)         MOVRM   R1,@TSRFPTR(R3)  ;SET TSOURCE FLAG TO SHOW FULL
  0497C 9431FFFE        (00562)         XORIR   R3,-2           ;SWITCH TSOURCE BUFFERS (-2,0,-2,...)
  0497E E0300543        (00563)         MOVRM   R3,TSRPO
  04980 E5000542        (00564)         INCM    TFRCTR          ;COUNT FRAME (TO BE) TRANSMITTED
  04982 F80DFFCE        (00565) AD$RTN: MOVLM   CLR+G2,SYS$FLGS  ;CLEAR G2 TO MARK END OF ADAMINT
  04984 0E70            (00566)         RETURN
                        (00567) ;
                        (00568) ; TSOURCE BUFFER ISN'T AVAILABLE, SO WE DISCARD THIS BUFFER OF NEW A/D
```

```
                    (00569) ;   DATA.  JUST COUNT THIS FRAME AS DISCARDED.
04985 0800          (00570)         EVEN
04986 E5000054A     (00571) AD$DISC: INCM   TADFDC
04988 2107          (00572)         HOP     AD$RTN
                    (00573) ;
                    (00574) ;ROUTINES TO COPY FROM THE A/D BUFFER POINTED TO BY (R2)+1
                    (00575) ;  TO A TSOURCE BUFFER.  A SINGLE BMOVE WOULD SUFFICE, BUT THAT WOULD
                    (00576) ;  HOLD OFF INTERRUPTS FOR ABOUT 250 USEC, SO WE BREAK IT UP INTO A FEW,
                    (00577) ;  WHICH IS ALMOST AS FAST, BUT MAKES FOR BETTER INTERRUPT LATENCY.
                    (00578) ;THE REV-19 CSPU MICROCODE REVISION MODIFIED THE BMOVE R,X,MEM SO THAT
                    (00579) ;  (1) R IS LEFT POINTING TO THE LAST SOURCE-WORD MOVED, AND
                    (00580) ;  (2) X IS LEFT WITH ITS ORIGINAL CONTENTS.  THIS LETS US USE
                    (00581) ;  THE BMOVES AS DONE BELOW WITHOUT RESETTING R AND X BEFORE EACH ONE.
04989 0800          (00582)         EVEN
0498A 90400001D     (00583) AD$CPYA: MOVIR  R4,SL6-1
0498C D728AE94      (00584)         BMOVEL  R2,R4,TSRA
0498E D728AED8      (00585)         BMOVEL  R2,R4,TSRA+2*SL6
04990 D728AF0C      (00586)         BMOVEL  R2,R4,TSRA+4*SL6
04992 0E70          (00587)         RETURN
                    (00588)
04993 0800          (00589)         EVEN
04994 90400001D     (00590) AD$CPYB: MOVIR  R4,SL6-1
04996 D728AF48      (00591)         BMOVEL  R2,R4,TSRB
04998 D728AF84      (00592)         BMOVEL  R2,R4,TSRB+2*SL6
0499A D728AFC0      (00593)         BMOVEL  R2,R4,TSRB+4*SL6
0499C 0E70          (00594)         RETURN
                    (00595)
```

```
                    (00596) ;TMODEMINT: TMODEM INTERRUPT SERVICE ROUTINE
                    (00597) ; JJW, 3-OCT-79
                    (00598) ; MODIFIED TO COPY ONLY, NO PROTECT.  JJW, 23-NOV-79
                    (00599) ;
                    (00600) ;TMODEMINT COPIES PROTECTED AND BITSTREAMED DATA FROM A TBITS
                    (00601) ;  BUFFER TO A TMODEM BUFFER.  IF THE TBITS BUFFER FLAG SHOWS THAT
                    (00602) ;  THE BUFFER ISN'T READY YET, A "FAKE" BUFFER OF DATA IS COPIED
                    (00603) ;  FROM TBTC INSTEAD.
                    (00604) ;
           00003B00 (00605) #L = $3B00              ;PUT THE REST OF THE CSPU CODE IN $3B00-$3FFF.
                    (00606) ;
                    (00607) ; POINTER OFFSETS:
                    (00608) ;   TBTPO IS FOR TBITS BUFFERS AND FLAGS (-2,0; INIT TO 0 SO IT
                    (00609) ;     LOOKS AT TBTB THE FIRST TIME)
                    (00610) ;   TMPO IS FOR TMODEM BUFFERS (-2,0; INIT TO 0 SO IT GETS SWITCHED
                    (00611) ;     TO TMDMA THE FIRST TIME)
                    (00612) ;
                    (00613) ;POINTERS TO THE TBITS BUFFERS/FLAGS ARE FOUND IN THE PROTECT MODULE.
                    (00614) ;POINTERS TO TMODEM-COPYING ROUTINES:
                    (00615)         EVEN
03B00 00000536      (00616)         ADDR    TBTFA           ;TBITS BUFFER FLAGS
03B02 00000537      (00617) TBTFPTR: ADDR   TBTFB
03B04 00003B32      (00618)         ADDR    TMI$CPYA        ;ROUTINE TO COPY INTO TMODEMA
03B06 00003B4C      (00619) TMPTR:  ADDR    TMI$CPYB        ;ROUTINE TO COPY INTO TMODEMB
                    (00620) ;
                    (00621)         EVEN
                    (00622) TMODEMINT:
03B08 F84DFFCE      (00623)         MOVLM   SET+G2,SYS$FLGS  ;MARK START OF TMODEMINT
03B0A F02C0545      (00624)         MOVMR   R2,TMPO         ;R2 GETS TMODEM BUFFER PTR OFFSET
03B0C 9421FFFE      (00625)         XORIR   R2,-2           ;SWITCH TO NEXT TMODEM BUFFER
03B0E E020C545      (00626)         MOVRM   R2,TMPO
03B10 E200C540      (00627)         CMPMZ   RUN
03B12 8C103B26      (00628)         JMP     TMI$RTN,EQZ     ;DO NIL IF VOCODER STOPPED
03B14 F030C544      (00629)         MOVMR   R3,TBTPO        ;R3 GETS TBITS PTR OFFSET
03B16 E2863B02      (00630)         CMPMZ   @TBTFPTR(R3)    ;IS THE TBITS BUFFER READY (FULL)?
03B18 8C1C3B2A      (00631)         JMP     TMI$FAKE,EQZ    ;NO, SO WE RUN FAKE DATA TO THE TMODEM
03B1A 90963CC2      (00632)         MOVIR   R1,@TBTBPTR(R3)  ;YES: R1 POINTS TO TBITS BUFFER+1
03B1C 9E100001      (00633)         SUBIR   R1,1            ;MAKE IT POINT TO 1ST WORD
03B1E 86843BC6      (00634)         CALL    R0,@TMPTR(R2)   ;COPY FROM TBITS TO TMODEM BUFFER
03B20 CC863B02      (00635)         MOVZM   @TBTFPTR(R3)    ;CLEAR TBITS FLAG TO SHOW EMPTY
03B22 9431FFFE      (00636)         XORIR   R3,-2           ;SWITCH TBITS BUFFERS (-2,0,-2,...)
03B24 E4C300544     (00637)         MOVRM   R3,TBTPO
03B26 F80DFFCE      (00638) TMI$RTN: MOVLM  CLR+G2,SYS$FLGS  ;MARK END OF TMODEMINT
03B28 0E70          (00639)         RETURN
                    (00640) ;
                    (00641) ;TBITS(X) ISN'T FULL YET, AND WE HAVE TO SEND SOMETHING TO THE
                    (00642) ;  MODEM, SO WE HAVE HANDY A BUFFER (TBTC) OF FAKE TBITS DATA
                    (00643) ;  (CORRESPONDING TO SILENCE), WHICH WE SIMPLY USE INSTEAD.
03B29 0800          (00644)         EVEN
03B2A E5000540      (00645) TMI$FAKE: INCM  TMFFC           ;INCR TMODEM FAKE FRAME COUNT
03B2C 9010A966      (00646)         MOVIR   R1,TBTC
03B2E 86843B06      (00647)         CALL    R0,@TMPTR(R2)
03B30 210B          (00648)         ROP     TMI$RTN         ;RETURN WITHOUT SWITCHING TBITS
```

```
                    (00649) ;                               ; BUFFER POINTERS
                    (00650) ;
                    (00651) ;ROUTINES FOR COPYING FROM THE TBITS BUFFER TO THE TMODEM BUFFER.
                    (00652) ;  R1 COMES POINTING TO THE 1ST WORD OF THE TBITS BUFFER (THE STREAM
                    (00653) ;  STARTS IN THE 2ND WORD, BUT WE WANT THE 3RD WORD-2 FOR THE BMOVEL).
                    (00654) ;WE COPY THE 2ND THRU THE 260TH HALFWORD, OMITTING THE (INITIAL)
                    (00655) ;  SYNC-BIT WORD AND THE (FINAL) UNUSED WORD.
                    (00656) ;THE TBITS BUFFER HAS THESE DATA IN EXACTLY THE SAME POSITIONS (RELATIVE
                    (00657) ;  TO THE START OF THE BUFFER) AS IN THE TMODEM BUFFER.
03B31 0800          (00658)         EVEN
03B32 F042C001      (00659) TMI$CPYA: MOVMR R4,1(R1)        ;MOVE 2ND HALFWORD
03B34 E040B0D5      (00660)         MOVRM   R4,TMDMA+1      ;INTO HALFWORD AFTER SYNC BIT
03B36 9040002A      (00661)         MOVIR   R4,ML6-1
03B38 D718B0D6      (00662)         BMOVEL  R1,R4,TMDMA+2   ;MOVE THE REST IN 3 HUNKS OF 43 FW
03B3A D718B12C      (00663)         BMOVEL  R1,R4,TMDMA+2+2*ML6
03B3C D718B182      (00664)         BMOVEL  R1,R4,TMDMA+2+4*ML6
03B3E 0E70          (00665)         RETURN
                    (00666) ;
03B3F 0800          (00667)         EVEN
03B40 F042C0C1      (00668) TMI$CPYB: MOVMR R4,1(R1)
03B42 E040B1DB      (00669)         MOVRM   R4,TMDMB+1
03B44 9040002A      (00670)         MOVIR   R4,ML6-1
03B46 D718B1DC      (00671)         BMOVEL  R1,R4,TMDMB+2
03B48 D718B232      (00672)         BMOVEL  R1,R4,TMDMB+2+2*ML6
03B4A D718B288      (00673)         BMOVEL  R1,R4,TMDMB+2+4*ML6
03B4C 0E70          (00674)         RETURN
                    (00675)
```

```
                        (00676) ;RMODEMINT: RMODEM INTERRUPT SERVICE MODULE
                        (00677) ; JJW, 25 SEPT 79
                        (00678) ;
                        (00679) ;RMODEMINT CHECKS SYNC ON THE BUFFER JUST INPUT BY THE RMODEM SCROLL
                        (00680) ;  PROGRAM, AND IF SYNC IS GOOD, COPIES THE LATEST "FRAME" OF DATA
                        (00681) ;  FROM THE RMODEM BUFFERS TO A RBITS BUFFER.  A "FRAME" OF DATA IS
                        (00682) ;  MBLNGTH WORDS STARTING WITH A SYNC-BIT WORD; IN GENERAL, IT DOES
                        (00683) ;  NOT START AND END ON RMODEM BUFFER BOUNDARIES.
                        (00684) ;THE STATE OF FRAME SYNC IS GIVEN BY RSYNC:
                        (00685) ;  RSYNC=0 MEANS WE KNOW NOTHING ABOUT SYNC, SO WE MUST CALL SYNCINIT
                        (00686) ;    TO INITIALIZE SYNC-SEARCHING.
                        (00687) ;  RSYNC=-1 MEANS WE ARE STILL IN THE PROCESS ON SCANNING INCOMING
                        (00688) ;    BUFFERS FOR FRAME SYNC (SYNCSRCH).
                        (00689) ;  RSYNC=+1 MEANS WE HAVE SYNC, SO WE CALL SYNCUPDATE TO CHECK THE NEW
                        (00690) ;    FRAME FOR CONTINUED SYNC.  IF SYNCUPDATE SETS RSYNC TO 0, THEN WE
                        (00691) ;    HAVE LOST SYNC AND WE MUST REINITIALIZE WITH SYNCINIT.
                        (00692) ;RMODEMINT ALSO:
                        (00693) ;  COPIES THE ON-HOOK BIT OF THE 1ST DATUM IN EACH RMODEM
                        (00694) ;    BUFFER INTO RONHK FOR USE BY THE ADAMINT MODULE,
                        (00695) ;  SIMULATES CHANNEL ERRORS IF RERSIM IS NONZERO, AND
                        (00696) ;  CHECKS TO BE SURE THAT THE RBITS BUFFER IS AVAILABLE; IF NOT, THEN
                        (00697) ;    THE NEW FRAME OF RMODEM DATA IS SIMPLY DISCARDED.
                        (00698) ;
                        (00699) ;POINTER OFFSETS:
                        (00700) ;  RBTPO IS FOR THE RBITS BUFFERS AND FLAGS (-2,0; INIT TO -2
                        (00701) ;    SO THAT IT POINTS TO RBTBA THE FIRST TIME)
                        (00702) ;  RMPO IS FOR RMODEM BUFFERS (-4,-2,0; INIT TO 0 SO IT GETS SWITCHED
                        (00703) ;    TO RMDMA THE FIRST TIME)
                        (00704) ;
                        (00705) ;POINTERS TO RMODEM BUFFERS AND RBITS FLAGS.  (POINTERS TO
                        (00706) ;  RBITS BUFFERS ARE IN THE CORRECT MODULE.)
                        (00707) ;USE RBTBPTR(RBTPO) TO POINT TO THE CURRENT RBITS BUFFER AND
                        (00708) ;  RBTFPTR(RBTPO) TO POINT TO THE CURRENT FLAG.
03B4D 0800              (00709)         EVEN
03B4E 0000053B          (00710)         ADDR    RBTFA
03B50 0000053C          (00711) RBTFPTR: ADDR   RBTFB
                        (00712) ;
03B52 0000B4EC          (00713)         ADDR    RMDMC   ;USE RMPTR(RMPO) TO POINT TO THE
03B54 0000B2E0          (00714)         ADDR    RMDMA   ; CURRENT RMODEM BUFFER, OR RMPTR-2(RMPO)
03B56 000083E6          (00715)         ADDR    RMDMB   ; TO POINT TO THE PREVIOUS ONE
03B58 000084EC          (00716) RMPTR:  ADDR    RMDMC
                        (00717) ;
                        (00718) ;  NOTE THAT RMODEMINT IS TO BE ENTERED BY CALL R0,RMODEMINT,
                        (00719) ;   SO IT SIMPLY RETURNS.  THERE WILL BE ANOTHER LEVEL THAT
                        (00720) ;   IS DISPATCHED TO BY THE INTERRUPT, SUCH AS:
                        (00721) ;       CALL    R0,RMODEMINT
                        (00722) ;       RET
                        (00723) ;
                        (00724)         EVEN
                        (00725) RMODEMINT:
03B5A F84DFFCE          (00726)         MOVLM   SET+G2,SYS$FLGS  ;MARK START OF RMODEMINT
03B5C F0200546          (00727)         MOVMR   R2,RMPO          ;R2 GETS RMODEM BUFFER PTR OFFSET
03B5E 2622              (00728)         INCR    R2,2             ;STEP TO NEXT RMODEM BUFFER (A,B,C,A,..)
```

```
03B5F 1B20      (00729)          SKPL    LEZ
03B60 9021FFFC  (00730)          MOVIR   R2,-4
03B62 E02C0546  (00731)          MOVRM   R2,RMPO
03B64 E2000540  (00732)          CMPMZ   RUN
03B66 80103BBA  (00733)          JMP     RMI$RTN,EQZ      ;DO NIL IF VOCODER NOT RUNNING
03B68 90943B58  (00734)          MOVIR   R1,@RMPTR(R2)    ;R1 NOW POINTS TO LATEST RMODEM BUFFER
03B6A 7032C080  (00735)          MOVWK   R3,R1,IDW$ONHK   ;GET ON-HOOK BIT FROM 1ST RMODEM DATUM
03B6C E0300551  (00736)          MOVRM   R3,RONHK         ; AND SAVE IT FOR USE BY ADAMINT ROUTINE
03B6E F040057F  (00737)          MOVMR   R4,RERSIM        ;RERSIM=# OF CHANNEL ERRORS PER BUFFER
03B70 1A10      (00738)          SKPL    EQZ              ; TO BE SIMULATED
03B71 86103BC2  (00739)          CALL    R1,RMI$SIM
03B73 E2000552  (00740)          CMPMZ   RSYNC            ;WHAT DO WE KNOW ABOUT SYNC?
03B75 0B00      (00741)          EVEN
03B76 80103BAE  (00742)          JMP     RMI$SINIT,EQZ    ;0 MEANS NIL, SO INIT
03B78 80303BB8  (00743)          JMP     RMI$SRCH,LTZ     ;-1 MEANS WE LACK IT, SO SEARCH
03B7A 861C3C4C  (00744)          CALL    R1,SYNCUPDATE    ;+1 MEANS WE HAVE IT, SO UPDATE
03B7C E2000552  (00745)          CMPMZ   RSYNC            ;DO WE STILL HAVE IT AFTER THE UPDATE?
03B7E 80103BB2  (00746)          JMP     RMI$LOST,EQZ     ;NO, SO REINIT FOR SYNC-SEARCHING
03B80 F0300547  (00747)          MOVMR   R3,RBTPO         ;R3 GETS RBITS POINTER OFFSET
03B82 E2863B50  (00748)          CMPMZ   @RBTFPTR(R3)     ;IS RBITS BUFFER AVAILABLE (EMPTY)?
03B84 81103BBE  (00749)          JMP     RMI$DISC,NEZ     ;NO, SO DISCARD THE FRAME OF DATA
                (00750) ;COPY FRAME OF BITS (INCLUDING SYNC BIT) FROM PREVIOUS AND CURRENT
                (00751) ;  RMODEM BUFFERS TO RBITS BUFFER.  WE COPY (MBLNGTH-RBOFO) BITS
                (00752) ;  FROM (PREV. RMODEM + RBOFO) TO RBITS, AND THEN (RBOFO) BITS FROM
                (00753) ;  (CURRENT RMODEM) TO (RBITS + MBLNGTH-RBOFO).
                (00754) ;NOTE THAT WE JAM ADDRESSES INTO TWO BMOVE INSTRUCTIONS BELOW!
03B86 90943B56  (00755)          MOVIR   R1,@RMPTR-2(R2)  ;R1 = PTR TO PREVIOUS RMODEM BUFFER
03B88 FC100553  (00756)          ADDMR   R1,RBOFO         ;R1 NOW POINTS TO THE SYNC BIT
03B8A 2711      (00757)          DECR    R1,1             ;MAKE R1 BE PTR-1 FOR BMOVE
03B8B 0B00      (00758)          EVEN
03B8C 90C63DFC  (00759)          MOVIR   R4,@RBTBPTR(R3)  ;R4 GETS PTR TO RBITS BUFFER
03B8E EF403B98  (00760)          SAF     R4,RMI$BMO1      ;SET UP 1ST BMOVE WITH 17-BIT ADR
03B90 9050010A  (00761)          MOVIR   R5,MBLNGTH-1     ;R5 GETS NUMBER-1 OF BITS TO BE MOVED
03B92 FE500553  (00762)          SUBMR   R5,RBOFO         ; FROM PREVIOUS RMODEM BUFFER
03B94 9C4A0001  (00763)          ADDIR   R4,1(R5)         ;R4 NOW POINTS TO RBITS FOR 2ND COPY
03B96 EF403BA2  (00764)          SAF     R4,RMI$BMO2      ;SET UP 2ND BMOVE WITH 17-BIT ADR
03B98 D51A0000  (00765) RMI$BMO1: BMOVE  R1,R5,0          ;(ADR GETS SET ABOVE)
03B9A F05C0553  (00766)          MOVMR   R5,RBOFO         ;DO WE NEED ANY BITS FROM 2ND BUFFER?
03B9C 8010 3BA4 (00767)          JMP     RMI$STF,EQZ      ;JUMP IF NO
03B9E 90943B58  (00768)          MOVIR   R1,@RMPTR(R2)    ;R1=PTR TO CURRENT RMODEM BUFFER
03BA0 2711      (00769)          DECR    R1,1             ;PTR-1 FOR BMOVE
03BA1 2751      (00770)          DECR    R5,1             ;NUMBER-1 FOR BMOVE
03BA2 D51A0000  (00771) RMI$BMO2: BMOVE  R1,R5,0          ;(ADR GETS SET ABOVE)
03BA4 E5863B50  (00772) RMI$STF: INCM    @RBTFPTR(R3)     ;SET RBTFLAG TO MARK RBITS BUFFER FULL
03BA6 9431FFFE  (00773)          XORIR   R3,-2            ;SWITCH RBITS BUFFERS (-2,0,-2,...)
03BA8 E0300547  (00774)          MOVRM   R3,RBTPO
03BAA E500054F  (00775)          INCM    RFRCTR           ;COUNT FRAME RECEIVED
03BAC 200D      (00776)          HOP     RMI$RTN
                (00777) ;
03BAD 0B00      (00778)          EVEN
03BAE 86103BE2  (00779) RMI$SINIT: CALL  R1,SYNCINIT      ;INIT THE SYNC-SEARCH
03BB0 2009      (00780)          HOP     RMI$RTN
                (00781) ;
```

```
03BB1 0800      (00782)          EVEN
03BB2 90943B56 (00783) RMI$LOST: MOVIR R1,@RMPTR-2(R2)  ;WE HAVE LOST SYNC:
03BB4 86103BE2 (00784)          CALL    R1,SYNCINIT     ; CALL SYNCINIT ON THE OLDER RMODEM
03BB6 90943B58 (00785)          MOVIR   R1,@RMPTR(R2)   ; BUFFER, THEN SYNCSRCH ON THE NEWER ONE
03BB8 86103C0C (00786) RMI$SRCH: CALL   R1,SYNCSRCH
03BBA F80DFFCE (00787) RMI$RTN: MOVLM  CLR+G2,SYS$FLGS  ;MARK END OF RMODEMINT
03BBC 0E70      (00788)          RETURN
                (00789) ;
                (00790) ;THE RBITS BUFFER ISN'T READY (EMPTY) YET, SO WE HAVE NO PLACE TO
                (00791) ;  PUT THE RMODEM DATA, AND WE HAVE TO DISCARD IT.
03BBD 0B0C      (00792)          EVEN
03BBE E5000C54C (00793) RMI$DISC: INCM  RMFDC           ;COUNT RMODEM FRAME DISCARD
03BC0 2107      (00794)          HOP     RMI$RTN
                (00795) ;
                (00796) ;ROUTINE TO SIMULATE CHANNEL ERRORS.  WE USE THE EXEC CODE AS A "RANDOM"
                (00797) ;  NUMBER GENERATOR.  WE MAINTAIN A POINTER TO THAT PART OF BUS 1.  FOR
                (00798) ;  EACH ERROR, WE PULL IN 2 WORDS, XOR, AND OFF TO 8 BITS, AND USE THAT
                (00799) ;  AS AN OFFSET INTO THE CURRENT RMODEM BUFFER, WHERE WE FLIP THE DATA
                (00800) ;  BIT.  THUS WE CAN ERROR ANY OF THE FIRST 256 BITS OF THE FRAME.
                (00801) ;  DOING IT TWICE PER FRAME GIVES 0.77% ERRORS.
                (00802) ;UPON ENTRY, R1 POINTS TO THE START OF THE RMODEM BUFFER, AND R4 HOLDS
                (00803) ;  THE NUMBER OF ERRORS TO BE MADE.
03BC1 0B00      (00804)          EVEN
03BC2 2741      (00805) RMI$SIM: DECR   R4,1             ;NUMBER-1 OF ERRORS
03BC3 0800      (00806)          EVEN
03BC4 F0503BDE (00807)          MOVMR   R5,ERRPTR        ;PICK UP POINTER INTO EXEC CODE
03BC6 9E700001 (00808)          MOVIR   R7,IDW$RD        ;WILL BE USED TO FLIP DATA BIT
03BC8 FF6A0000 (00809) #1:      MOVMR   R6,0(R5)         ;GET "RANDOM" NUMBER
03BCA F46A0001 (00810)          XORMR   R6,1(R5)         ;MAKE IT MORE SO
03BCC 2663      (00811)          INCR    R6,3             ;AVOID BIAS FOR 0
03BCD 2653      (00812)          INCR    R5,3             ;STEP RANDOM NUMBER PTR
03BCE 9A6000FF (00813)          ANDIR   R6,$FF           ;MAKE THE RANDOM NUMBER 8 BITS
03BD0 4C62      (00814)          ADDRR   R6,R1            ;MAKE PTR INTO RMODEM BUFFER
03BD1 0800      (00815)          EVEN
03BD2 F57C0000 (00816)          XORRM   R7,0(R6)         ;FLIP DATA BIT FOR THE ERROR
03BD4 8C403BC8 (00817)          DJP     R4,#1            ;GO BACK IF NOT ENOUGH ERRORS YET
03BD6 92501C77 (00818)          CMPIR   R5,RAND$U        ;CHECK IF ERRPTR NEEDS RESETTING
03BD8 1B20      (00819)          SKPL    LE
03BD9 90500BE9 (00820)          MOVIR   R5,RAND$L        ;RESET ERRPTR TO LOWER VALUE
03BDB E0503BDE (00821)          MOVRM   R5,ERRPTR
03BDD 0E70      (00822)          RETURN
03BDE 0BE8      (00823) ERRPTR: DATA    RAND$L
                (00824)
```

```
                    (00825) ;  FRAME SYNCHRONIZATION ROUTINES
                    (00826) ;
                    (00827) ; MISC. INTERNAL VARIABLES
03BDF 0000          (00828) ERRSYNC: DATA   0               ;COUNTS SYNC ERRORS
03BE0 0000          (00829) LASTERR: DATA   0               ;=1 IF SYNC ERROR ON PREVIOUS FRAME
03BE1 0000          (00830) OLDSYNC: DATA   0               ;LSB = SYNC BIT ON PREVIOUS FRAME
                    (00831) ;RBOFO: (IST 81)                ;OFFSET FROM START OF RMODEM BUFFER TO
                    (00832) ;                               ; POSITION WHERE SYNC BIT WAS FOUND
                    (00833) ;
                    (00834) ;
                    (00835) ; SYNCINIT: INITIALIZE BUFFERS, ETC. FOR SYNC SEARCHING.  CALL WITH
                    (00836) ;   R1 POINTING TO RMODEM BUFFER.
                    (00837) ; ALSO ZEROS THE GAIN WORD OF BOTH RSOURCE BUFFERS AND ZEROS THE
                    (00838) ;   ENTIRE "LAST" RBITS BUFFER, SINCE IF WE JUST LOST SYNC, THOSE
                    (00839) ;   BUFFERS CONTAIN GARBAGE, AND WE DON'T WANT TO RESUME SYNTHESIZING
                    (00840) ;   WITH THEM.
                    (00841)         EVEN
03BE2 90200104      (00842) SYNCINIT: MOVIR R2,%BLNGTH-1    ;BUFFER OFFSET = LAST WORD IN BUFFER
03BE4 4C14          (00843)         ADDRR   R1,R2           ;R1 NOW POINTS TO END OF RMODEM BUFFER
03BE5 70320001      (00844) SI$LOOP: MOVWK  R3,R1,1         ;PICK UP DATA BIT (BIT 0)
03BE7 E034B5F2      (00845)         MOVRM   R3,RSSPF(R2)    ; AND STASH IT IN RSSPF(I)
03BE9 2711          (00846)         DECR    R1,1
03BEA BC203BE5      (00847)         DJP     R2,SI$LOOP
03BEC E0200552      (00848)         MOVRM   R2,RSYNC        ;SET RSYNC=-1 TO MEAN SEARCH FOR SYNC
03BEE CE00B802      (00849)         MOVZML  RSRA+2*W$       ;ZERO THE 3RD FULLWORD (=GAIN) OF BOTH
03BF0 CE00B890      (00850)         MOVZML  RSRB+2*W$       ; RSOURCE BUFFERS
03BF2 F0300547      (00851)         MOVMR   R3,RBTPO        ;PICK UP PTR OFFSET TO "THIS" RBITS
03BF4 9431FFFE      (00852)         XORIR   R3,-2           ;SWITCH IT TO THE "OTHER" ONE
03BF6 90C63DFC      (00853)         MOVIR   R4,@RBTBPTR(R3)  ;R4 NOW POINTS TO "OTHER" RBITS BUFFER
03BF8 CE000000      (00854)         MOVZML  0(R4)           ;CLEAR THE 1ST FULLWORD
03BFA 2642          (00855)         INCR    R4,W$           ;RBITS+2 FOR BMOVEL DEST. ADR
03BFB EF403C00      (00856)         SAF     R4,SI$BMO       ;SET UP ADR IN BMOVEL BELOW
03BFD 2744          (00857)         DECR    R4,2*W$         ;RBITS-2 FOR BMOVEL SOURCE ADR
03BFE 90200081      (00858)         MOVIR   R2,(MBLNGTH/2)-1  ;SET UP TO CLEAR MBLNGTH-1 HALFWORDS
03C00 D7440000      (00859) SI$BMO: BMOVEL  R4,R2,0         ;(ADR IS SET UP ABOVE)
03C02 9040B6F6      (00860)         MOVIR   R4,RSSSS-W$     ;CLEAR THE RSSSS BUFFER TOO
03C04 CE00B6F8      (00861)         MOVZML  RSSSS
03C06 D744B6FA      (00862)         BMOVEL  R4,R2,RSSSS+W$  ;(R2 IS STILL SET FROM PREVIOUS BMOVEL)
03C08 CC00B7FC      (00863)         MOVZM   RSSSS+MBLNGTH-1  ;CLEAR THE LAST HALFWORD OF RSSSS
03C0A 0E70          (00864)         RETURN
                    (00865) ;
                    (00866) ;
                    (00867) ; SYNCSRCH: SEARCH INCOMING RMODEM BUFFER FOR SYNC, AND IF SYNC
                    (00868) ;   IS FOUND, SET RSYNC TO +1 AND SET RBOFO TO THE FRAME OFFSET
                    (00869) ;   CORRESPONDING TO THE LOGICAL START OF THE FRAME.
                    (00870) ; CALL WITH R1 POINTING TO THE START OF THE RMODEM BUFFER TO BE
                    (00871) ;   SEARCHED.
                    (00872) ; REGISTER USAGE:
                    (00873) ;   R1 - PTR TO THE RMODEM BUFFER BEING SEARCHED
                    (00874) ;   R2 - OFFSET INTO THE RMODEN, RSSPF, RSSSS BUFFERS
                    (00875) ;   R3 - SCRATCH AC
                    (00876) ;   R4 - MAXCNT, THE MAX. OF RSSSS(I)
                    (00877) ;   R5 - NMAX, THE # OF INSTANCES OF MAXCNT IN RSSSS(I)
```

```
                    (00878) ;   R6 - IMAX, THE FRAME OFFSET WHERE MAXCNT WAS FOUND
                    (00879) ; VARIABLES:
                    (00880) ;   RBOFO - BEGINNING OF FRAME OFFSET TO THE SYNC WORD IN THE
                    (00881) ;                 RMODEM BUFFER
                    (00882) ; BUFFERS:
                    (00883) ;   RSSPF - SYNC-SEARCH PREVIOUS FRAME - STORES THE DATA BIT OF THE
                    (00884) ;                 PREVIOUS RMODEM BUFFER
                    (00885) ;   RSSSS - SYNC-SEARCH SUM-OF-SYNC COUNTS - FOR EACH RMODEM WORD,
                    (00886) ;                 STORES THE NUMBER OF CONSECUTIVE FRAMES OF DATA BIT
                    (00887) ;                 ALTERNATION.
03C0B 0800          (00888)         EVEN
03C0C 90200104 (00889) SYNCSRCH: MOVIR R2,NBLNGTH-1     ;INIT BUFFER OFFSET TO LAST IN FRAME
03C0E 4C14          (00890)         ADDRR   R1,R2           ;MAKE R1 POINT THERE TOO
03C0F 0D40          (00891)         CLR     R4              ;MAXCNT <- 0
03C10 70320001 (00892) SS$LOOP: MOVWK  R3,R1,IDW$RD     ;R3 <- NEW DATA BIT (OTHER BITS MASKED)
03C12 F234B5F2 (00893)          CMPMR   R3,RSSPF(R2)    ;DOES THIS BIT ALTERNATE FROM LAST?
03C14 80103C26 (00894)          JMP     SS$NOALT,EQ     ;JUMP IF NO ALTERNATION
03C16 E504B6F8 (00895) SS$ALT:  INCM    RSSSS(R2)       ;YES: COUNT THE BIT ALTERNATION
03C18 F244B6F8 (00896)          CMPMR   R4,RSSSS(R2)    ;IS MAXCNT >= RSSSS(I)?
03C1A 81303C22 (00897)          JMP     NOMAX,GE        ;YES
03C1C F044B6F8 (00898) NEWMAX:  MOVMR   R4,RSSSS(R2)    ;NO: RSSSS(I) BECOMES THE NEW MAXCNT
03C1E 90500001 (00899)          MOVIR   R5,1            ;NMAX <- 1 (UNIQUE SO FAR)
03C20 4064          (00900)         MOVRR   R6,R2           ;AND WE SAVE THE OFFSET IN IMAX
03C21 2006          (00901)         HOP     SS$LPTST
                    (00902) ;
03C22 81103C28 (00904) NOMAX:   JMP     SS$LPTST,NE     ;MAXCNT = RSSSS(I)?
03C24 2651          (00905)         INCR    R5,1            ;YES, SO THE MAX ISN'T UNIQUE
03C25 2002          (00906)         HOP     SS$LPTST
                    (00907) ;
                    (00908)         EVEN
03C26 CC04B6F8 (00909) SS$NOALT: MOVZM RSSSS(R2)        ;CLEAR RSSSS(I)
03C28 E034B5F2 (00910) SS$LPTST: MOVRM R3,RSSPF(R2)     ;SAVE NEW DATA BIT IN RSSPF(I)
03C2A 2711          (00911)         DECR    R1,1            ;DECR RMODEM BUFFER PTR
03C2B 0800          (00912)         EVEN
03C2C 8C203C10 (00913)          DJP     R2,SS$LOOP      ;DECR OFFSET AND TEST IF DONE BUFFER
                    (00914) ; WE HAVE NOW SCANNED THE WHOLE BUFFER.  IS MAXCNT >= THE SYNC
                    (00915) ;   ACQUISITION THRESHOLD, AND IT IT A UNIQUE MAX?  IF SO, WE HAVE
                    (00916) ;   FOUND SYNC.
03C2E 9240000A (00917)          CMPIR   R4,ACQTHR       ;MAXCNT >= ACQTHR?
03C30 80303C48 (00918)          JMP     SS$ND,LT        ;ND
03C32 92500001 (00919)          CMPIR   R5,1            ;YES: IS NMAX=1?
03C34 81103C48 (00920)          JMP     SS$NO,NE        ;NO
03C36 E0500552 (00921)          MOVRM   R5,RSYNC        ;YES: WE HAVE SYNC. SET RSYNC TO +1
03C38 E0600553 (00922)          MOVRM   R6,RBOFO        ;SAVE OFFSET IN RBOFO
03C3A 90300004 (00923)          MOVIR   R3,LOSETHR
03C3C E03F3BDF (00924)          MOVRM   R3,ERRSYNC      ;INIT ERRSYNC TO COUNT ERRORS
03C3E CC003BE0 (00925)          MOVZM   LASTERR         ; FOR SYNCUPDATE NEXT FRAME
03C40 9C1C0001 (00926)          ADDIR   R1,1(R6)        ;R1 POINTS TO WORD HOLDING SYNC
03C42 70320001 (00927)          MOVWK   R3,R1,IDW$RD    ;PICK UP SYNC BIT AND SAVE IT IN
03C44 E03C3BE1 (00928)          MOVRM   R3,OLDSYNC      ; OLDSYNC FOR USE IN SYNCUPDATE ON NEXT
03C46 0E70          (00929)         RETURN                  ; FRAME
                    (00930) ;
```

FRAME SYNCHRONIZATION ROUTINES

```
03C47 0800      (00931)          EVEN
03C48 E0200552 (00932) SS$NO:   MOVRM   R2,RSYNC        ;NO SYNC YET: SET TO -1 (=SEARCH)
03C4A 0E70      (00933)          RETURN                  ; FOR GOOD MEASURE
                (00934) ;
                (00935) ;
                (00936) ; SYNCUPDATE:  CHECK RMODEM BUFFER TO SEE IF THE DATA BIT AT THE
                (00937) ;   EXPECTED SYNC POSITION (START OF BUFFER+RBOFO) HAS THE EXPECTED
                (00938) ;   VALUE.  IF THERE ARE LOSETHR ERRORS WITHOUT 2 CONSECUTIVE CORRECT
                (00939) ;   COMPARISONS, CLEAR RSYNC TO SIGNIFY LOSS OF FRAME SYNCHRONIZATION.
                (00940) ; ENTER WITH R1 POINTING TO THE LATEST RMODEM BUFFER.  ALSO, THE
                (00941) ;   WORD RBOFO HOLDS THE OFFSET IN THE FRAME THAT POINTS TO THE
                (00942) ;   SYNC WORD TO BE VERIFIED.
                (00943) ; VARIABLES:
                (00944) ;   RBOFO - BEGINNING OF FRAME OFFSET
                (00945) ;   OLDSYNC - SYNC BIT FROM PREVIOUS FRAME
                (00946) ;   LASTERR - NZ IF SYNC ERROR ON PREVIOUS FRAME
                (00947) ;   ERRSYNC - COUNTS SYNC ERRORS (FROM LOSETHR DOWN TO 0)
03C4B 0800      (00948)          EVEN
03C4C FC100553 (00949) SYNCUPDATE: ADDMR R1,RBOFO       ;R1 NOW POINTS TO NEW SYNC WORD
03C4E 6022      (00950)          MOVWR   R2,R1          ;PICK UP THE NEW SYNC WORD
03C4F 0800      (00951)          EVEN
03C50 F4203BE1 (00952)          XORMR   R2,OLDSYNC      ;XOR WITH LAST SYNC BIT
03C52 2C04      (00953)          SRBS    0,R2           ;SKIP IF DATA BIT SET (GOOD SYNC)
03C53 200C      (00954)          HOP     SU$ERR
03C54 E2003BE0 (00955)          CMPMZ   LASTERR         ;WAS THERE AN ERROR LAST FRAME (NEZ)?
03C56 81103C5C (00956)          JMP     SU$ELF,NEZ      ;YES
03C58 90300004 (00957)          MOVIR   R3,LOSETHR      ;NO:  THAT'S (AT LEAST) 2 CONSECUTIVE
03C5A E0303BDF (00958)          MOVRM   R3,ERRSYNC      ; CORRECT FRAMES, SO RESET ERROR COUNT
03C5C CC003BE0 (00959) SU$ELF:  MOVZM   LASTERR         ;REMEMBER NONERROR FOR NEXT FRAME
03C5E 2009      (00960)          HOP     SU$VALID
                (00961) ;
03C5F 0800      (00962)          EVEN
03C60 90200001 (00963) SU$ERR:  MOVIR   R2,1            ;COME HERE ON SYNC BIT ERROR
03C62 E0203BE0 (00964)          MOVRM   R2,LASTERR      ;REMEMBER ERROR FOR NEXT FRAME
03C64 E400 3BDF (00965)          DECM    ERRSYNC         ;HAVE WE COUNTED LOSETHR ERRORS?
03C66 81203C6E (00966)          JMP     SU$LOSE,LEZ     ;YES, THEREFORE WE'VE LOST SYNC
03C68 90100001 (00967) SU$VALID: MOVIR  R1,IDW$RD       ;COMPLEMENT OLDSYNC IN PREPARATION
03C6A F5103BE1 (00968)          XORRM   R1,OLDSYNC      ; FOR USE NEXT FRAME
03C6C 0E70      (00969)          RETURN
                (00970) ;
03C6D 0800      (00971)          EVEN
03C6E CC000552 (00972) SU$LOSE: MOVZM   RSYNC           ;ZERO RSYNC TO SAY WE LOST SYNC
03C70 E5000550 (00973)          INCM    RLSCTR          ;COUNT LOSS OF SYNC
03C72 0E70      (00974)          RETURN
                (00975)
```

AOMINT: AOM INTERRUPT SERVICE ROUTINE

```
                        (00976) ;AOMINT: AOM INTERRUPT SERVICE ROUTINE
                        (00977) ; JJW, 3 OCT 79
                        (00978) ;
                        (00979) ;AOMINT COPIES SYNTHESIS WAVEFORM DATA FROM AN RSINK BUFFER TO A D/A
                        (00980) ;  BUFFER.  IF THE RSINK BUFFER ISN'T READY YET, A FRAME OF SILENCE
                        (00981) ;  IS COPIED FROM RSNKC INSTEAD.
                        (00982) ;POINTER OFFSETS:
                        (00983) ;  RSNPO IS FOR RSINK BUFFERS AND FLAGS (-2,0; INIT TO 0 SO IT POINTS
                        (00984) ;    TO RSNKB THE FIRST TIME).  USE RSIBPTR(RSNPO) TO POINT TO THE
                        (00985) ;    RSINK BUFFER AND RSIFPTR(RSNPO) FOR THE FLAG.
                        (00986) ;  DAPO IS FOR D/A BUFFERS (-2,0; INIT TO 0 SO IT GETS SWITCHED TO
                        (00987) ;    DA$CPYA THE FIRST TIME).  USE DABPTR(DAPO) TO POINT TO THE
                        (00988) ;    D/A BUFFER COPYING ROUTINE.
                        (00989) ;
03C73 0800      (00990)         EVEN
03C74 00003CA8  (00991)         ADDR    DA$CPYA         ;ROUTINE TO COPY TO D/A BUFFER A
03C76 00003CB2  (00992) DABPTR: ADDR    DA$CPYB         ;ROUTINE TO COPY TO D/A BUFFER B
                        (00993) ;
03C78 0000B918  (00994)         ADDR    RSNKA-2         ;RSINK A BUFFER (-2 FOR THE BMOVEL)
03C7A 0000B9CC  (00995) RSIBPTR: ADDR   RSNKB-2         ;RSINK B BUFFER
03C7C 0000053D  (00996)         ADDR    RSNFA           ;RSINK A FLAG (0=EMPTY, 1=FULL)
03C7E 0000053E  (00997) RSIFPTR: ADDR   RSNFB           ;RSINK B FLAG
                        (00998) ;
                        (00999)         EVEN
03C80 F84DFFCE  (01000) AOMINT: MOVLM   SET+G2,SYS$FLGS  ;MARK START OF AOMINT
03C82 F0100549  (01001)         MOVMR   R1,DAPO         ;R1 GETS D/A BUFFER PTR OFFSET
03C84 9411FFFE  (01002)         XORIR   R1,-2           ;SWITCH TO OTHER BUFFER (0,-2,0,...)
03C86 E0100549  (01003)         MOVRM   R1,DAPO
03C88 E200C540  (01004)         CMPMZ   RUN
03C8A 80103C9C  (01005)         JMP     AOM$RTN,EQZ     ;DO NIL IF VOCODER NOT RUNNING
03C8C F0300548  (01006)         MOVMR   R3,RSNPO        ;R3 GETS RSINK PTR OFFSET
03C8E 90A63C7A  (01007)         MOVIR   R2,@RSIBPTR(R3)  ;R2 POINTS TO RSINK BUFFER-1
03C90 E2863C7E  (01008)         CMPMZ   @RSIFPTR(R3)    ;IS THE RSINK BUFFER READY (FULL)?
03C92 80103CA0  (01009)         JMP     AOM$NRDY,EQZ    ;JUMP IF NOT READY
03C94 86823C76  (01010)         CALL    R0,@DABPTR(R1)  ;COPY FROM RSINK TO D/A BUFFER
03C96 CC863C7E  (01011)         MOVZM   @RSIFPTR(R3)    ;CLR RSINK FLAG TO SHOW BUFFER EMPTY
03C98 9431FFFE  (01012)         XORIR   R3,-2           ;SWITCH RSINK BUFFERS (-2,0,-2,...)
03C9A E0300548  (01013)         MOVRM   R3,RSNPO
03C9C F80DFFCE  (01014) AOM$RTN: MOVLM  CLR+G2,SYS$FLGS  ;MARK END OF AOMINT
03C9E 0E70      (01015)         RETURN
                        (01016) ;
03C9F 0800      (01017)         EVEN
03CA0 E5000540  (01018) AOM$NRDY: INCM  RSNNR           ;COUNT RSINK BUFFER NOT READY
03CA2 9020BA80  (01019)         MOVIR   R2,RSNKC-2      ;COPY SILENCE FROM RSNKC TO D/A BUFFER
03CA4 86823C76  (01020)         CALL    R0,@DABPTR(R1)
03CA6 2108      (01021)         HOP     AOM$RTN         ;RETURN WITHOUT SWITCHING RSINK BUFFS
                        (01022) ;
03CA7 0800      (01023)         EVEN
03CA8 90400010  (01024) DA$CPYA: MOVIR  R4,SL6-1
03CAA D7288B36  (01025)         BMOVEL  R2,R4,RDABA
03CAC D7288B72  (01026)         BMOVEL  R2,R4,RDABA+2*SL6
03CAE D7288BAE  (01027)         BMOVEL  R2,R4,RDABA+4*SL6
03CB0 0E70      (01028)         RETURN
```

```
                    (01029) ;
03CB1 0800          (01030)          EVEN
03CB2 90400810 (01031) DA$CPYB: MOVIR   R4,SL6-1
03CB4 D728BBEA (01032)          BMOVEL  R2,R4,RDABB
03CB6 D728BC26 (01033)          BMOVEL  R2,R4,RDABB+2*SL6
03CB8 D728BC62 (01034)          BMOVEL  R2,R4,RDABB+4*SL6
03CBA 0E70          (01035)          RETURN
                    (01036)
```

```
(01037) TPROTECT(AB) -- PROTECT AND BITSTREAM THE CODED TRANSMITTER FRAME.
(01038) ;
(01039) ;PROTECT TAKES QUANTIZED AND CODED ANALYSIS PARAMETERS AND RESIDUAL
(01040) ;  SAMPLES FROM A TSINK BUFFER, AND FORMS A BITSTREAM, INCLUDING
(01041) ;  ERROR-PROTECTING CODES FOR SOME BITS OF CERTAIN PARAMETERS, IN
(01042) ;  A TBITS BUFFER.  PROTECT NO LONGER RUNS AT INTERRUPT LEVEL, BUT IS
(01043) ;  INVOKED VIA FCB.
(01044) ;PROTECT ALSO ACCUMULATES HISTOGRAM DATA FOR P, T, G, K1-K8.
(01045) ;
(01046) ;THIS ROUTINE IS ENTERED WITH R1 POINTING TO THE FCB#.  1(R1)
(01047) ;  IS THE BUFFER/FLAG/ROUTINE POINTER OFFSET, -2 FOR "A" OR 0 FOR "B".
(01048) ;
(01049) ;FORMAT OF INPUT (TSINK) BUFFER IS:
(01050) ;  PITCH(5/6)
(01051) ;  TAP(4/4)
(01052) ;  GAIN(5/6)
(01053) ;  K1(5/5)
(01054) ;  K2(5/5)
(01055) ;  K3(5/5)
(01056) ;  K4(3/4)
(01057) ;  K5(3/4)
(01058) ;  K6(3/4)
(01059) ;  K7(1/3)
(01060) ;  K8(1/3)
(01061) ;  BB1 - BB60(0/3)
(01062) ;WHERE (N/M) MEANS N HIGH-ORDER BITS PROTECTED OUT OF M TOTAL FOR PARAM.
(01063) ;
(01064) ;
(01065) ;FORMAT OF OUTPUT (TBITS) BUFFER IS:
(01066) ;  POSITION FOR THE SYNC BIT (IN THE TMODEM BUFFER)
(01067) ;  HIGH 4 BITS PITCH (7/4)
(01068) ;  4 BITS TAP(7/4)
(01069) ;  HIGH 4 BITS GAIN (7/4)
(01070) ;  LOWEST PITCH BIT (1)
(01071) ;  LOWEST GAIN BIT (1)
(01072) ;  HIGH 4 BITS K1 (7/4)
(01073) ;  HIGH 4 BITS K2 (7/4)
(01074) ;  LOW PROT BIT PITCH, LOW PROT BIT GAIN, LOW K1, LOW K2 (7/4)
(01075) ;  HIGH 4 K3 (7/4)
(01076) ;  HIGH 3 BITS K4, LOW BIT K3 (7/4)
(01077) ;  3 HIGH BITS K5, HIGH BIT K6 (7/4)
(01078) ;  LOWEST K4, LOWEST K5 (2)
(01079) ;  LOWEST K6, LOW 2 K7 (3)
(01080) ;  LOW 2 PROT K6, HIGH K7, HIGH K8 (7/4)
(01081) ;  LOW 2 K8 (2)
(01082) ;  BB1 - BB60(3)
(01083) ;... FOR A TOTAL OF 260 BITS, WHICH LEAVES THE LAST BIT POSITION UNUSED.
(01084) ;THERE'S ONE EXTRA BIT POSITION IN THE FRAME.
(01085) ;
(01086) ;REGISTER USAGE:
(01087) ;  R1 = PTR-1 TO INPUT (TSINK) BUFFER
(01088) ;  R2 = PTR TO OUTPUT (TBITS) BUFFER
(01089) ;  R3 = COUNTER IN PRBBLP
```

```
                    (01090) ;  R4-R6 ARE USED FOR HOLDING PIECES OF PARAMETERS.  R6 IS
                    (01091) ;    ALSO USED IN OUT7, BUT THE RETURN RESTORES IT.
                    (01092) ;  R7 = HOLDS DATUM FOR INPUT TO OUT7, OUT3, OUT2 ROUTINES.
                    (01093) ;
                    (01094) ;POINTERS TO BUFFERS AND FLAGS.  SOME OF THESE ARE ALSO USED BY
                    (01095) ;  TMODEMINT.
03CBB 0800          (01096)          EVEN
03CBC 0000AFFB      (01097)          ADDR    TSNKA-1          ;TSINK BUFFERS (-1 FOR POPXI'S)
03CBE 0000B043      (01098) TSNBPTR: ADDR    TSNKB-1
                    (01099) ;
03CC0 0000A75B      (01100)          ADDR    TBTA+1           ;TBITS BUFFERS (BITSTREAM STARTS AT +1
03CC2 0000AB61      (01101) TBTBPTR: ADDR    TBTB+1           ; TO ALLOW FOR SYNC BIT IN TMDM BUFFS)
                    (01102) ;
03CC4 F84BFFCE      (01103) PROTECT: MOVLM   SET+G1,SYSSFLGS
03CC6 F8320001      (01104)          MOVMR   R3,1(R1)         ;PICK UP POINTER OFFSET
03CC8 90963CBE      (01105)          MOVIR   R1,@TSNBPTR(R3)  ;R1 GETS PTR-1 TO TSINK BUFFER
03CCA 90A63CC2      (01106)          MOVIR   R2,@TBTBPTR(R3)  ;R2 GETS OUTPUT PTR (TO TBT + 1)
                    (01107) ;
03CCC 301E          (01108)          POPXI R1,R7       ;PITCH(5/6)
03CCD 0800          (01109)          EVEN
03CCE E50EBC9E      (01110)          INCM PHIST(R7)  ;ADD INTO HISTOGRAM DATA
03CD0 504E0003      (01111)          MOVKR R4,R7,3     ;LOW 2 BITS TO R4
03CD2 3872          (01112)          ARS R7,2          ;HIGH 4 BITS ONLY
03CD3 0800          (01113)          EVEN
03CD4 866003D70     (01114)          CALL R6,OUT7      ;PUT 7 BITS OUT
                    (01115) ;
03CD6 301E          (01116)          POPXI R1,R7       ;TAP(4/4)
03CD7 0800          (01117)          EVEN
03CD8 E50EBCDE      (01118)          INCM THIST(R7)
03CDA 866003D70     (01119)          CALL R6,OUT7      ;PUT 7 TAP BITS OUT
                    (01120) ;
03CDC 301E          (01121)          POPXI R1,R7       ;GAIN(5/6)
03CDD 0800          (01122)          EVEN
03CDE E50EBCEE      (01123)          INCM GHIST(R7)
03CE0 505E0003      (01124)          MOVKR R5,R7,3     ;LOW 2 BITS TO R5
03CE2 3872          (01125)          ARS R7,2          ;HIGH 4 BITS ONLY
03CE3 0800          (01126)          EVEN
03CE4 866003D70     (01127)          CALL R6,OUT7      ;HIGH GAIN OUT
                    (01128) ;
03CE6 50780001      (01129)          MOVKR R7,R4,1     ;LOWEST PITCH BIT
03CE8 9670000C      (01130)          IORIR R7,TMBITS   ;ADD MODEM BITS
03CEA 342E          (01131)          PUSHXI R2,R7     ;PUT OUT 1 BIT
03CEB 0800          (01132)          EVEN
                    (01133) ;
03CEC 507A0001      (01134)          MOVKR R7,R5,1     ;GET LOWEST GAIN BIT
03CEE 9670000C      (01135)          IORIR R7,TMBITS
03CF0 342E          (01136)          PUSHXI R2,R7     ;PUT OUT 1 BIT
03CF1 0800          (01137)          EVEN
                    (01138) ;
03CF2 50680002      (01139)          MOVKR R6,R4,2     ;LOWEST PROT PITCH BIT TO R6
03CF4 4C6C          (01140)          ADDRR R6,R6      ;SHIFT LEFT (FASTER THAN LLS) TO MAKE ROOM
03CF5 0800          (01141)          EVEN
03CF6 566A0002      (01142)          IORKR R6,R5,2     ;OR IN LOWEST PROT GAIN BIT
```

```
                PROTECT(AB) -- PROTECT AND BITSTREAM THE CODED TRANSMITTER FRAME.

03CF8 301E      (01143)         POPXI R1,R7       ;K1(5/5)
03CF9 0800      (01144)         EVEN
03CFA E50EBD2E  (01145)         INCM K1HIST(R7)
03CFC 566E0001  (01146)         IORKR R6,R7,1     ;OR IN LOWEST K1 BIT
03CFE 4C6C      (01147)         ADDRR R6,R6       ;MAKE ROOM
03CFF 3871      (01148)         ARS R7,1          ;HIGH 4 BITS K1 ONLY
03D00 86603D70  (01149)         CALL R6,OUT7      ;PUT OUT PROTECTED HIGH 4 BITS K1
                (01150) ;
03D02 301E      (01151)         POPXI R1,R7       ;K2(5/5)
03D03 0800      (01152)         EVEN
03D04 E50EBD4E  (01153)         INCM K2HIST(R7)
03D06 566E0001  (01154)         IORKR R6,R7,1     ;OR IN LOWEST BIT K2
03D08 3871      (01155)         ARS R7,1          ;HIGH 4 BITS K2 ONLY
03D09 0800      (01156)         EVEN
03D0A 86603D70  (01157)         CALL R6,OUT7      ;OUTPUT PROTECTED HIGH 4 BITS K2
                (01158) ;
03D0C 407C      (01159)         MOVRR R7,R6       ;PICK UP BITS STASHED IN R6
03D0D 0800      (01160)         EVEN
03D0E 86603D70  (01161)         CALL R6,OUT7      ;OUTPUT LOWEST BITS PROTECTED OF P,G,K1,K2
                (01162) ;
03D10 301E      (01163)         POPXI R1,R7       ;K3(5/5)
03D11 0800      (01164)         EVEN
03D12 E50E8D6E  (01165)         INCM K3HIST(R7)
03D14 504E0001  (01166)         MOVKR R4,R7,1     ;SAVE LOWEST BIT OF K3 IN R4
03D16 3871      (01167)         ARS R7,1          ;HIGH 4 BITS ONLY
03D17 0800      (01168)         EVEN
03D18 86603D70  (01169)         CALL R6,OUT7      ;OUTPUT HIGH 4 BITS K3 PROTECTED
                (01170) ;
03D1A 301E      (01171)         POPXI R1,R7       ;K4(3/4)
03D1B 0800      (01172)         EVEN
03D1C E50EBD8E  (01173)         INCM K4HIST(R7)
03D1E 505E0001  (01174)         MOVKR R5,R7,1     ;LOW BIT K4 TO R5
03D20 9A70000E  (01175)         ANDIR R7,0^16     ;CLEAR SLOT FOR LO K3
03D22 4678      (01176)         IORRR R7,R4       ;OR IN LOW BIT K3 TO HIGH 3 BITS K4
03D23 0800      (01177)         EVEN
03D24 86603D70  (01178)         CALL R6,OUT7      ;OUTPUT PROTECTED 3 HI K4, LOW K3
                (01179) ;
03D26 301E      (01180)         POPXI R1,R7       ;K5(3/4)
03D27 4C5A      (01181)         ADDRR R5,R5       ;LOW K4, SHIFT TO MAKE ROOM
03D28 E50EBD9E  (01182)         INCM K5HIST(R7)
03D2A 565E0001  (01183)         IORKR R5,R7,1     ;OR IN LOWEST K5 BIT
03D2C 3018      (01184)         POPXI R1,R4       ;K6(3/4)
03D2D 0800      (01185)         EVEN
03D2E E50EBDAE  (01186)         INCM K6HIST(R7)
03D30 50680007  (01187)         MOVKR R6,R4,7     ;LOW 3 BITS K6
03D32 3843      (01188)         ARS R4,3          ;HIGH BIT K6
03D33 0800      (01189)         EVEN
03D34 9A70000E  (01190)         ANDIR R7,0^16     ;SAVE 3 HIGH BITS K5
03D36 4678      (01191)         IORRR R7,R4       ;OR IN 1 HIGH BIT K6
03D37 0800      (01192)         EVEN
03D38 86603D70  (01193)         CALL R6,OUT7      ;OUTPUT 3 HIGH K5, 1 HIGH K6, PROTECTED
                (01194) ;
03D3A 407A      (01195)         MOVRR R7,R5       ;LOWEST K4, LOWEST K5
```

```
                PROTECT(AB) -- PROTECT AND BITSTREAM THE CODED TRANSMITTER FRAME.

03D3B 0000      (01196)         EVEN
03D3C 86603D88  (01197)         CALL R6,OUT2      ;OUTPUT LOWEST K4, LOWEST K5, UNPROTECTED
                (01198) ;
03D3E 504C0006  (01199)         MOVKR R4,R6,6     ;LOWEST 2 PROT K4
03D40 3016      (01200)         POPXI R1,R3       ;K7(1/3)
03D41 0000      (01201)         EVEN
03D42 E50EBD8E  (01202)         INCM K7HIST(R7)
03D44 50760003  (01203)         MOVKR R7,R3,3     ;SAVE LOW 2 BITS K7
03D46 3832      (01204)         ARS R3,2          ;HIGH BIT K7 ONLY
03D47 4646      (01205)         IORRR R4,R3       ;OR IN HIGH K7 TO 2 LOW (PROT) K6
03D48 3A62      (01206)         LLS R6,2          ;SHIFT LOWEST K6 BIT
03D49 0000      (01207)         EVEN
03D4A 567C0004  (01208)         IORKR R7,R6,4     ;OR LOWEST K6 IN WITH LOW 2 K7
03D4C 86603D7C  (01209)         CALL R6,OUT3      ;OUTPUT LOW K6, LOW 2 K7
                (01210) ;
03D4E 4C48      (01211)         ADDRR R4,R4       ;LOW 2 PROT K6, HI K7
03D4F 301E      (01212)         POPXI R1,R7       ;K8(1/3)
03D50 E50EBDC6  (01213)         INCM K8HIST(R7)
03D52 505E0003  (01214)         MOVKR R5,R7,3     ;LOW 2 K8
03D54 3872      (01215)         ARS R7,2          ;HIGH K8 ONLY
03D55 4678      (01216)         IORRR R7,R4       ;LOW 2 PROT K6, HIGH K7, HIGH K8
03D56 86603D70  (01217)         CALL R6,OUT7      ;OUTPUT THEM
                (01218) ;
03D58 407A      (01219)         MOVRR R7,R5       ;LOW 2 K8
03D59 0000      (01220)         EVEN
03D5A 86603D88  (01221)         CALL R6, OUT2     ;OUTPUT LOW 2 K8 UNPROTECTED
                (01222) ;
03D5C 90300038  (01223)         MOVIR   R3,60-1           ;NUMBER OF BASEBAND SAMPLES-1
03D5E 301E      (01224) PRBBLP: POPXI   R1,R7             ;GET NEXT BB SAMPLE
03D5F 4C7E      (01225)         ADDRR   R7,R7             ;(THIS IS THE SAME CODE AS OUT3)
03D60 2623      (01226)         INCR    R2,3
03D61 0000      (01227)         EVEN
03D62 90FE3DB2  (01228)         MOVIR   R7,@PTR3(R7)
03D64 C875FFFD  (01229)         POPMI   R7,-3(R2)
03D66 C875FFFE  (01230)         POPMI   R7,-2(R2)
03D68 C875FFFF  (01231)         POPMI   R7,-1(R2)
03D6A 8C303D5E  (01232)         DJP     R3,PRBBLP         ;DONE?
03D6C F800BFFCE (01233)         MOVLM   CLR+G1,SYS$FLGS
03D6E 0E70      (01234)         RETURN
                (01235) ;
                (01236) ;OUT7 TAKES 4-BIT DATUM IN R7 AND OUTPUTS 7-BIT HAMMING CODE
                (01237) ;NOTE THAT WE MODIFY THE INSTRUCTION AT BMO7 BY JAMMING THE
                (01238) ;  17-BIT OUTPUT POINTER INTO ITS ADDRESS FIELD.
03D6F 0000      (01239)         EVEN
03D70 EF203D78  (01240) OUT7:   SAF     R2,BMO7           ;STORE OUTPUT PTR IN BMOVE INSTR.
03D72 4C7E      (01241)         ADDRR   R7,R7             ;CONVERT R7 TO FULLWORD OFFSET
03D73 2627      (01242)         INCR    R2,7              ;UPDATE OUTPUT PTR IN R2
03D74 90FE3D92  (01243)         MOVIR   R7,@HPTR(R7)      ;R7 GETS PTR TO START-1
03D76 90060006  (01244)         MOVIR   R6,7-1            ;WILL TRANSFER 7 WORDS
03D78 D57C0000  (01245) BMO7:   BMOVE   R7,R6,0           ;MOVE 7 WORDS FROM (R7)+1 TO OUTPUT
03D7A 0E70      (01246)         RETURN
                (01247) ;
                (01248) ;OUT3 OUTPUTS 3 BITS GIVEN IN R7
```

PROTECT(AB) -- PROTECT AND BITSTREAM THE CODED TRANSMITTER FRAME.

```
03D7B 0800      (01249)          EVEN
03D7C 4C7E      (01250) OUT3:    ADDRR   R7,R7           ;CONVERT R7 TO FULLWORD OFFSET
03D7D 2623      (01251)          INCR    R2,3            ;UPDATE OUTPUT POINTER
03D7E 90FE3DB2 (01252)          MOVIR   R7,@PTR3(R7)    ;R7 GETS PTR-1
03D80 C875FFFD (01253)          POPMI   R7,-3(R2)       ;MOVE 3 WORDS
03D82 C875FFFE (01254)          POPMI   R7,-2(R2)
03D84 C875FFFF (01255)          POPMI   R7,-1(R2)
03D96 0E70      (01256)          RETURN
                (01257) ;
                (01258) ;OUT2 OUTPUTS 2 BITS GIVEN IN R7
03D87 0800      (01259)          EVEN
03D88 4C7E      (01260) OUT2:    ADDRR   R7,R7           ;CONVERT R7 TO FULLWORD OFFSET
03D89 2622      (01261)          INCR    R2,2            ;UPDATE OUTPUT POINTER
03D8A 90FE3DC2 (01262)          MOVIR   R7,@PTR2(R7)    ;R7 GETS PTR-1
03D8C C875FFFE (01263)          POPMI   R7,-2(R2)
03D8E C875FFFF (01264)          POPMI   R7,-1(R2)
03D90 0E70      (01265)          RETURN
                (01266)
                (01267) ;TABLE OF HAMMING CODE SEQUENCES.  INDEX WITH 2*(4-BIT VALUE)
                (01268) ;  TO BE PROTECTED; TABLE ENTRY IS PTR-1 TO THE 7-BIT HAMMING
                (01269) ;  CODEWORD.
03D91 0800      (01270)          EVEN
03D92 00003DF1 (01271) HPTR:    ADDR H000-1
03D94 0F003DC9 (01272)          ADDR H151-1
03D96 000F3DE3 (01273)          ADDR H052-1
03D98 00003DE8 (01274)          ADDR H103-1
03D9A 000F3DD2 (01275)          ADDR H114-1
03D9C 000F3DCB (01276)          ADDR H045-1
03D9E 00003DD1 (01277)          ADDR H146-1
03DA0 000F3DEA (01278)          ADDR H017-1
03DA2 00003DEE (01279)          ADDR H160-1
03DA4 000F3DD3 (01280)          ADDR H031-1
03DA6 00003DDF (01281)          ADDR H132-1
03DA8 000F3DD4 (01282)          ADDR H063-1
03DAA 000F3DEC (01283)          ADDR H074-1
03DAC 00003DE2 (01284)          ADDR H125-1
03DAE 000F3DCD (01285)          ADDR H026-1
03DB0 00003DD9 (01286)          ADDR H177-1
                (01287)
                (01288) ;TABLE OF PTR-1 TO 3-BIT SEQUENCES
03DB2 00003DF2 (01289) PTR3:    ADDR P0-1
03DB4 00003DD7 (01290)          ADDR P1-1
03DB6 00003DCE (01291)          ADDR P2-1
03DB8 00003DE0 (01292)          ADDR P3-1
03DBA 00003DF0 (01293)          ADDR P4-1
03DBC 00003DCA (01294)          ADDR P5-1
03DBE 00003DD5 (01295)          ADDR P6-1
03DC0 00003DDA (01296)          ADDR P7-1
                (01297)
                (01298) ;TABLE OF PTR-1 TO 2-BIT SEQUENCES
03DC2 00003DF3 (01299) PTR2:    ADDR Q0-1
03DC4 00003DD8 (01300)          ADDR Q1-1
03DC6 00003DCF (01301)          ADDR Q2-1
```

PROTECT(AB) -- PROTECT AND BITSTREAM THE CODED TRANSMITTER FRAME.

```
03DC8 000030E1 (01302)          ADDR Q3-1
               (01303)
               (01304) ;TABLE OF CODE SEQUENCES.
03DCA 000D     (01305) H151:    DATA 1 + TMBITS
03DCB 000D     (01306) P5:      DATA 1 + TMBITS
03DCC 000C     (01307) H045:    DATA 0 + TMBITS
03DCD 000D     (01308)          DATA 1 + TMBITS
03DCE 000C     (01309) H026:    DATA 0 + TMBITS
03DCF 000C     (01310) P2:      DATA 0 + TMBITS
03DD0 000D     (01311) Q2:      DATA 1 + TMBITS
03DD1 000C     (01312)          DATA 0 + TMBITS
03DD2 000D     (01313) H146:    DATA 1 + TMBITS
03DD3 000D     (01314) H114:    DATA 1 + TMBITS
03DD4 000C     (01315) H031:    DATA 0 + TMBITS
03DD5 000C     (01316) H063:    DATA 0 + TMBITS
03DD6 000D     (01317) P6:      DATA 1 + TMBITS
03DD7 000D     (01318)          DATA 1 + TMBITS
03DD8 000C     (01319) P1:      DATA 0 + TMBITS
03DD9 000C     (01320) Q1:      DATA 0 + TMBITS
03DDA 000D     (01321) H177:    DATA 1 + TMBITS
03DDB 000D     (01322) P7:      DATA 1 + TMBITS
03DDC 000D     (01323)          DATA 1 + TMBITS
03DDD 000D     (01324)          DATA 1 + TMBITS
03DDE 000D     (01325)          DATA 1 + TMBITS
03DDF 000D     (01326)          DATA 1 + TMBITS
03DE0 000D     (01327) H132:    DATA 1 + TMBITS
03DE1 000C     (01328) P3:      DATA 0 + TMBITS
03DE2 000D     (01329) Q3:      DATA 1 + TMBITS
03DE3 000D     (01330) H125:    DATA 1 + TMBITS
03DE4 000C     (01331) H052:    DATA 0 + TMBITS
03DE5 000D     (01332)          DATA 1 + TMBITS
03DE6 000C     (01333)          DATA 0 + TMBITS
03DE7 000D     (01334)          DATA 1 + TMBITS
03DE8 000C     (01335)          DATA 0 + TMBITS
03DE9 000D     (01336) H103:    DATA 1 + TMBITS
03DEA 000C     (01337)          DATA 0 + TMBITS
03DEB 000C     (01338) H017:    DATA 0 + TMBITS
03DEC 000C     (01339)          DATA 0 + TMBITS
03DED 000C     (01340) H074:    DATA 0 + TMBITS
03DEE 000D     (01341)          DATA 1 + TMBITS
03DEF 000D     (01342) H160:    DATA 1 + TMBITS
03DF0 000D     (01343)          DATA 1 + TMBITS
03DF1 000D     (01344) P4:      DATA 1 + TMBITS
03DF2 000C     (01345) H000:    DATA 0 + TMBITS
03DF3 000C     (01346) P0:      DATA 0 + TMBITS
03DF4 000C     (01347) Q0:      DATA 0 + TMBITS
03DF5 000C     (01348)          DATA 0 + TMBITS
03DF6 000C     (01349)          DATA 0 + TMBITS
03DF7 000C     (01350)          DATA 0 + TMBITS
03DF8 000C     (01351)          DATA 0 + TMBITS
               (01352)
```

```
                    (01353) ;CORRECT(A8): UNBITSTREAM, ERROR-CORRECT, AND DECODE
                    (01354) ;
                    (01355) ;CORRECT TAKES A FRAME OF BITSTREAM (RBITS BUFFER) FROM THE
                    (01356) ;  RECEIVER MODEM AND FORMS IT INTO PARAMETERS AND RESIDUAL SAMPLES,
                    (01357) ;  DOING ERROR-CORRECTING DECODING FOR THE PROTECTED DATA BITS.  EACH
                    (01358) ;  UNBITSTREAMED AND CORRECTED VALUE IS THEN LOOKED UP IN THE
                    (01359) ;  APPROPRIATE DECODING TABLE, SO THAT THE OUTPUT (TSOURCE) BUFFER IS
                    (01360) ;  FULL-WORD FLOATING POINT VALUES READY FOR THE SYNTHESIS PROCESS.
                    (01361) ;CORRECT NO LONGER RUNS AT INTERRUPT LEVEL, BUT IS INVOKED VIA FCB.
                    (01362) ;
                    (01363) ;THIS ROUTINE IS ENTERED (VIA FCB) WITH R1 POINTING TO THE
                    (01364) ;  FCB#.  1(R1) IS THE BUFFER/FLAG POINTER OFFSET, EQUAL TO -2 FOR "A"
                    (01365) ;  OR 0 FOR "B".
                    (01366) ;REGISTER USAGE:
                    (01367) ;  R1 = PTR-1 TO THE INPUT (RBITS) BUFFER (USED BY POPXI'S)
                    (01368) ;  R2 = PTR TO THE OUTPUT (RSOURCE) BUFFER (USED IN PUSHMIL'S)
                    (01369) ;  R6 - SCRATCH IN PARAM DECODING AND IN GET74, ETC.
                    (01370) ;  R7 - GET74, ETC. RETURN VALUES IN HERE
                    (01371) ;POINTERS TO BUFFERS.  THE RBITS POINTERS ARE ALSO USED BY RMODEMINT.
03DF9 0000          (01372)         EVEN
03DFA 0000AA6C      (01373)         ADDR    RBTA            ;RBITS BUFFERS
03DFC 0000AB72      (01374) RBTBPTR: ADDR   RBTB
                    (01375) ;
03DFE 0000B7FE      (01376)         ADDR    RSRA            ;RSOURCE BUFFERS
03E00 0000B88C      (01377) RSRBPTR: ADDR   RSRB
                    (01378) ;
                    (01379) ;TEMPS FOR PARTIAL, UNBITSTREAMED, CODED PARAM VALUES
03E02 0000          (01380) PITCH:  DATA    0               ;ALSO K3, K4, K7
03E03 0000          (01381) TAP:    DATA    0               ;ALSO K5, K8
03E04 0000          (01382) GAIN:   DATA    0               ;ALSO K6
03E05 0000          (01383) K1:     DATA    0
03E06 0000          (01384) K2:     DATA    0
      00003E02      (01385) K3 = PITCH
      00003E02      (01386) K4 = PITCH
      00003E03      (01387) K5 = TAP
      00003E04      (01388) K6 = GAIN
      00003E02      (01389) K7 = PITCH
      00003E03      (01390) K8 = TAP
                    (01391) ;
03E07 0000          (01392)         EVEN
03E08 F048FFCE      (01393) CORRECT: MOVLM  SET+G1,SYS$FLGS
03E0A F0220001      (01394)         MOVMR   R2,1(R1)        ;PICK UP POINTER OFFSET
03E0C 9A943DFC      (01395)         MOVIR   R1,@RBTBPTR(R2)  ;R1 GETS PTR TO RBITS BUFFER (POINTS TO
                    (01396) ;                                ; SYNC BIT, SO IS PTR-1 TO THE DATA)
03E0E 9AA43E00      (01397)         MOVIR   R2,@RSRBPTR(R2)  ;R2 GETS PTR TO RSOURCE BUFFER
                    (01398) ;
03E10 86003ED2      (01399)         CALL    R0,GET74        ;GET AND DECODE HI 4 OF PITCH
03E12 3A72          (01400)         LLS     R7,2            ;SHIFT BITS TO FINAL RESTING PLACE
03E13 0000          (01401)         EVEN
03E14 E0703E02      (01402)         MOVRM   R7,PITCH        ;AND SAVE THEM
                    (01403) ;
03E16 86003ED2      (01404)         CALL    R0,GET74        ;G&D TAP (4 BITS)
03E18 E0703E03      (01405)         MOVRM   R7,TAP          ;SAVE IT
```

```
                    (01406) ;
03E1A 86003ED2 (01407)          CALL    R0,GET74        ;G&D HI 4 OF GAIN
03E1C 3A72     (01408)          LLS     R7,2            ;SHIFT
03E1D 0800     (01409)          EVEN
03E1E E0703E04 (01410)          MOVRM   R7,GAIN         ;AND SAVE THEM
                    (01411) ;
03E20 86003F08 (01412)          CALL    R0,GET1         ;GET LOWEST PITCH BIT
03E22 F7703E02 (01413)          IORRM   R7,PITCH        ;AND OR IT INTO PITCH
                    (01414) ;
03E24 86003F08 (01415)          CALL    R0,GET1         ;GET LOWEST GAIN BIT
03E26 F7703E04 (01416)          IORRM   R7,GAIN         ;LIKEWISE
                    (01417) ;
03E28 86003ED2 (01418)          CALL    R0,GET74        ;G&D HI 4 OF K1
03E2A 4C7E     (01419)          ADDRR   R7,R7           ;SHIFT TO POSITION (FASTER THAN LLS)
03E2B 0800     (01420)          EVEN
03E2C E0703E05 (01421)          MOVRM   R7,K1           ;AND STORE
                    (01422) ;
03E2E 86003ED2 (01423)          CALL    R0,GET74        ;G&D HI 4 OF K2
03E30 4C7E     (01424)          ADDRR   R7,R7           ;LIKEWISE
03E31 0800     (01425)          EVEN
03E32 E0703E06 (01426)          MOVRM   R7,K2
                    (01427) ;
03E34 86003ED2 (01428)          CALL    R0,GET74        ;G&D B1 PITCH, B1 GAIN, B0 K1, B0 K2
03E36 506E0010 (01429)          MOVKR   R6,R7,C3        ;EXTRACT B1 PITCH
03E38 3862     (01430)          ARS     R6,2            ;SHIFT IT INTO POSITION
03E39 0800     (01431)          EVEN
03E3A F6603E02 (01432)          IORMR   R6,PITCH        ;R6 NOW HAS THE CODED PITCH (LSHFTD 1)
03E3C 9260000A (01433)          CMPIR   R6,PITCHL       ;MAKE SURE PITCH VALUE IS WITHIN BOUNDS
03E3E 1B30     (01434)          SKPL    GE
03E3F 9060000A (01435)          MOVIR   R6,PITCHL       ;VALUE WAS TOO SMALL
03E41 9260004C (01436)          CMPIR   R6,PITCHU
03E43 1B20     (01437)          SKPL    LE
03E44 9060004C (01438)          MOVIR   R6,PITCHU       ;VALUE WAS TOO LARGE
                    (01439)     EVEN
03E46 340E     (01440)          PUSHXI  R0,R7           ;TEMP. SAVE R7 ON THE STACK
03E47 3A66     (01441)          LLS     R6,7-1          ;MAKE INTO UNNORM FLTING NUMBER
03E48 90700042 (01442)          MOVIR   R7,$42          ;RIGHT EXP FOR FLT INTEGER
03E4A 84640000 (01443)          MOVRML  R6,0(R2)        ;PUT OUT PITCH AS UNNORM FLT PT INTGR
03E4C 2622     (01444)          INCR    R2,2            ;STEP OUTPUT PTR TO TAP
03E4D 0800     (01445)          EVEN
03E4E F6603E03 (01446)          MOVMR   R6,TAP          ;GET CODED TAP
03E50 C62C5ED0 (01447)          PUSHMIL R2,DTQTAB(R6)   ;DECODE AND STORE TAP
03E52 310E     (01448)          POPXD   R0,R7           ;RESTORE R7 FOR THE OTHER 3 BITS
03E53 0800     (01449)          EVEN
03E54 506E0008 (01450)          MOVKR   R6,R7,C2        ;EXTRACT B1 GAIN
03E56 3861     (01451)          ARS     R6,1            ;SHIFT TO RIGHT SPOT
03E57 0800     (01452)          EVEN
03E58 F6603E04 (01453)          IORMR   R6,GAIN         ;OR IN THE REST OF THE GAIN BITS
03E5A C62C5E50 (01454)          PUSHMIL R2,EQTAB(R6)    ;DECODE AND STORE GAIN
03E5C 506E0004 (01455)          MOVKR   R6,R7,C1        ;EXTRACT B0 K1
03E5E 3861     (01456)          ARS     R6,1
03E5F 0800     (01457)          EVEN
03E60 F6603E05 (01458)          IORMR   R6,K1
```

```
03E62 C62C5D10 (01459)          PUSHMIL R2,DKTAB1(R6)
03E64 9A700002 (01460)          ANDIR   R7,C0           ;EXTRACT B0 K2
03E66 F6703E06 (01461)          IORMR   R7,K2
03E68 C62E5D50 (01462)          PUSHMIL R2,DKTAB2(R7)
                (01463) ;
03E6A 86003ED2 (01464)          CALL    R0,GET74        ;G&D HI 4 OF K3
03E6C 4C7E     (01465)          ADDRR   R7,R7           ;POSITION THEM
03E6D 0000     (01466)          EVEN
03E6E E0703E02 (01467)          MOVRM   R7,K3           ;AND STORE THEM
                (01468) ;
03E70 86003ED2 (01469)          CALL    R0,GET74        ;G&D HI 3 OF K4, LO BIT OF K3
03E72 506E0002 (01470)          MOVKR   R6,R7,C0        ;EXTRACT LO BIT OF K3
03E74 F6603E02 (01471)          IORMR   R6,K3           ;R6 NOW HAS CODED K3
03E76 C62C5D90 (01472)          PUSHMIL R2,DKTAB3(R6)   ;DECODE AND STORE K3
03E78 506E001C (01473)          MOVKR   R6,R7,C321      ;MASK HI 3 OF K4
03E7A E0603E02 (01474)          MOVRM   R6,K4           ;AND SAVE THEM
                (01475) ;
03E7C 86003ED2 (01476)          CALL    R0,GET74        ;G&D HI 3 OF K5, HI BIT OF K6
03E7E 506E001C (01477)          MOVKR   R6,R7,C321      ;EXTRACT HI 3 OF K5
03E80 E0603E03 (01478)          MOVRM   R6,K5           ;AND SAVE THEM
03E82 9A700002 (01479)          ANDIR   R7,C0           ;EXTRACT HI BIT OF K6
03E84 3A73     (01480)          LLS     R7,3            ;POSITION IT CORRECTLY
03E85 0000     (01481)          EVEN
03E86 E0703E04 (01482)          MOVRM   R7,K6           ;AND SAVE IT
                (01483) ;
03E88 86003F08 (01484)          CALL    R0,GET1         ;GET LO BIT OF K4
03E8A F6703E02 (01485)          IORMR   R7,K4
03E8C C62E5DD0 (01486)          PUSHMIL R2,DKTAB4(R7)   ;DECODE AND STORE K4
                (01487) ;
03E8E 86003F08 (01488)          CALL    R0,GET1         ;GET LO K5
03E90 F6703E03 (01489)          IORMR   R7,K5
03E92 C62E5DF0 (01490)          PUSHMIL R2,DKTAB5(R7)   ;LIKEWISE
                (01491) ;
03E94 86003F08 (01492)          CALL    R0,GET1         ;GET LO K6
03E96 F7703E04 (01493)          IORRM   R7,K6           ;OR IT INTO THE HI K6 AND SAVE
                (01494) ;
03E98 86003EFE (01495)          CALL    R0,GET2         ;GET LO 2 BITS OF K7
03E9A E0703E02 (01496)          MOVRM   R7,K7           ;AND SAVE THEM
                (01497) ;
03E9C 86003ED2 (01498)          CALL    R0,GET74        ;G&D B2,B1 OF K6, B2 OF K7, B2 OF K8
03E9E 506E0018 (01499)          MOVKR   R6,R7,C32       ;EXTRACT B2,B1 OF K6
03EA0 3861     (01500)          ARS     R6,1            ;SHIFT TO CORRECT POSITION
03EA1 0000     (01501)          EVEN
03EA2 F6603E04 (01502)          IORMR   R6,K6           ;OR IN B3,B0 OF K6
03EA4 C62C5E10 (01503)          PUSHMIL R2,DKTAB6(R6)   ;DECODE AND STORE K6
03EA6 4C7E     (01504)          ADDRR   R7,R7           ;LSHIFT TO LINE UP B2 OF K7
03EA7 0000     (01505)          EVEN
03EA8 506E0008 (01506)          MOVKR   R6,R7,C2        ;EXTRACT IT
03EAA F6603E02 (01507)          IORMR   R6,K7           ;OR IN THE REST
03EAC C62C5E30 (01508)          PUSHMIL R2,DKTAB7(R6)   ;DECODE AND STORE K7
03EAE 4C7E     (01509)          ADDRR   R7,R7           ;SHIFT B2 OF K8 TO CORRECT POSITION
03EAF 0000     (01510)          EVEN
03EB0 9A700008 (01511)          ANDIR   R7,C2           ;MASK OFF OTHERS
```

```
03EB2 E0703E03 (01512)          MOVRM   R7,K8              ;AND SAVE IT
               (01513) ;
03EB4 86003EFE (01514)          CALL    R0,GET2           ;GET LO 2 OF K8
03EB6 F6703E03 (01515)          IORMR   R7,K8
03EB8 C62E5E40 (01516)          PUSHMIL R2,DKTAB8(R7)     ;DECODE AND STORE K8
               (01517) ;
03EBA 9C50003B (01518)          MOVIR   R5,60-1           ;NUMBER-1 OF BASEBAND SAMPLES
03EBC 301C     (01519) #1:      POPXI   R1,R6             ;GET BB DATA BIT FROM STREAM
03EBD 0800     (01520)          EVEN
03EBE 507C0001 (01521)          MOVKR   R7,R6,1           ;(THIS CODE IS SAME AS GET3)
03EC0 4C7E     (01522)          ADDRR   R7,R7             ;LEFT SHIFT 1 TO MAKE ROOM FOR NEXT
03EC1 301C     (01523)          POPXI   R1,R6
03EC2 567C0001 (01524)          IORKR   R7,R6,1
03EC4 4C7E     (01525)          ADDRR   R7,R7
03EC5 301C     (01526)          POPXI   R1,R6
03EC6 567C0001 (01527)          IORKR   R7,R6,1
03EC8 4C7E     (01528)          ADDRR   R7,R7             ;CONVERT TO FULLWORD INDEX
03EC9 0800     (01529)          EVEN
03ECA C62E5D00 (01530)          PUSHMIL R2,BBTAB(R7)      ;DECODE AND STORE BB SAMPLE
03ECC BC503EBC (01531)          DJP     R5,#1             ;DONE YET?
03ECE F808FFCE (01532)          MOVLM   CLR+G1,SYS$FLGS
03ED0 0E70     (01533)          RETURN                    ;YES!
               (01534) ;
               (01535) ;THESE UNBITSTREAMING ROUTINES RETURN THEIR VALUES IN R7 LEFT
               (01536) ;  SHIFTED ONE PLACE, SO THAT THEY CAN ACT AS FULL-WORD OFFSETS INTO
               (01537) ;  THE DECODING TABLES.
               (01538) ;
               (01539) ;PULL IN 7 SUCCESSIVE WORDS FROM THE BITSTREAM AND FORM THEIR DATA
               (01540) ;  BITS INTO A 7-BIT VALUE.  THIS VALUE IS USED AS AN INDEX INTO
               (01541) ;  CORTAB TO YIELD AN ERROR-CORRECTED 4-BIT VALUE.  IF RNOCOR IS NZ,
               (01542) ;  OMIT THE ERROR-CORRECTION AND RETURN ONLY THE 4 INFORMATION BITS.
               (01543) ;R1 = PTR-1 TO BITSTREAM
               (01544) ;R6 = SCRATCH
               (01545) ;R7 = HOLDS RETURNED VALUE
               (01546) ;THIS CAN BE CALLED WITH CALL R0,GET74.
03ED1 0800     (01547)          EVEN
03ED2 301C     (01548) GET74:   POPXI   R1,R6             ;GET MODEM WORD INTO R6
03ED3 0800     (01549)          EVEN
03ED4 507C0001 (01550)          MOVKR   R7,R6,1           ;EXTRACT DATA BIT TO R7
03ED6 4C7E     (01551)          ADDRR   R7,R7             ;SHIFT LEFT 1 FASTER THAN LLS
03ED7 301C     (01552)          POPXI   R1,R6
03ED8 567C0001 (01553)          IORKR   R7,R6,1
03EDA 4C7E     (01554)          ADDRR   R7,R7
03EDB 301C     (01555)          POPXI   R1,R6
03EDC 567C0001 (01556)          IORKR   R7,R6,1
03EDE 4C7E     (01557)          ADDRR   R7,R7
03EDF 301C     (01558)          POPXI   R1,R6
03EE0 567C0001 (01559)          IORKR   R7,R6,1
03EE2 4C7E     (01560)          ADDRR   R7,R7
03EE3 301C     (01561)          POPXI   R1,R6
03EE4 567C0001 (01562)          IORKR   R7,R6,1
03EE6 4C7E     (01563)          ADDRR   R7,R7
03EE7 301C     (01564)          POPXI   R1,R6
```

```
03EE8 567C0001 (01565)          IORKR   R7,R6,1
03EEA 4C7E     (01566)          ADDRR   R7,R7
03EEB 301C     (01567)          POPXI   R1,R6
03EEC 567C0001 (01568)          IORKR   R7,R6,1
03EEE E200057D (01569)          CMPMZ   RNOCOR          ;DO WE ERROR-CORRECT?
03EF0 81103EF6 (01570)          JMP     NOCORR,NEZ      ;JUMP IF NO
03EF2 F07E3F0A (01571) ERCORR:  MOVMR   R7,CORTAB(R7)   ;LOOK UP CORRECTED 4-BIT VALUE
03EF4 0E70     (01572)          RETURN
               (01573) ;
03EF5 0800     (01574)          EVEN
03EF6 506E0010 (01575) NOCORR:  MOVKR   R6,R7,C3        ;NO ERROR CORRECTION: GET B3 INFO BIT
03EF8 4C7E     (01576)          ADDRR   R7,R7           ;SHIFT THE OTHERS
03EF9 0800     (01577)          EVEN
03EFA 566E000E (01578)          IORKR   R6,R7,C210      ;EXTRACT AND ADD B2,B1,B0 INFO BITS
03EFC 407C     (01579)          MOVRR   R7,R6
03EFD 0E70     (01580)          RETURN
               (01581) ;
               (01582) ;GET1 AND GET2 UNBITSTREAM 1 AND 2 BITS, ALSO RETURNING RESULT
               (01583) ;  LEFT-SHIFTED ONE BIT.
               (01584)          EVEN
03EFE 301C     (01585) GET2:    POPXI   R1,R6
03EFF 0800     (01586)          EVEN
03F00 507C0001 (01587)          MOVKR   R7,R6,1
03F02 4C7E     (01588)          ADDRR   R7,R7
03F03 301C     (01589) GT1:     POPXI   R1,R6
03F04 567C0001 (01590)          IORKR   R7,R6,1
03F06 4C7E     (01591)          ADDRR   R7,R7
03F07 0E70     (01592)          RETURN
               (01593) ;
               (01594)          EVEN
03F08 0D70     (01595) GET1:    CLR     R7
03F09 2107     (01596)          HOP     GT1
               (01597) ;
               (01598) ;HAMMING 7/4 CODE CORRECTION TABLE.  INDEX TABLE WITH 7-BIT RECEIVED
               (01599) ;  CODEWORD; TABLE VALUE IS THE CORRECTED 4-BIT VALUE,
               (01600) ;  LEFT SHIFTED 1 BIT (TO BE USED AS A FULL-WORD OFFSET IN THE DECODING
               (01601) ;  TABLES).
               (01602) CORTAB:
03F0A 0000     (01603)          DATA    0^00^^2     ;RCVD 000
03F0B 0000     (01604)          DATA    0^00^^2     ;RCVD 001
03F0C 0000     (01605)          DATA    0^00^^2     ;RCVD 002
03F0D 0006     (01606)          DATA    0^03^^2     ;RCVD 003
03F0E 0000     (01607)          DATA    0^00^^2     ;RCVD 004
03F0F 000A     (01608)          DATA    0^05^^2     ;RCVD 005
03F10 001C     (01609)          DATA    0^16^^2     ;RCVD 006
03F11 000E     (01610)          DATA    0^07^^2     ;RCVD 007
03F12 0000     (01611)          DATA    0^00^^2     ;RCVD 010
03F13 0012     (01612)          DATA    0^11^^2     ;RCVD 011
03F14 0004     (01613)          DATA    0^02^^2     ;RCVD 012
03F15 000E     (01614)          DATA    0^07^^2     ;RCVD 013
03F16 0008     (01615)          DATA    0^04^^2     ;RCVD 014
03F17 000E     (01616)          DATA    0^07^^2     ;RCVD 015
03F18 000E     (01617)          DATA    0^07^^2     ;RCVD 016
```

```
03F19 000E     (01618)          DATA  0'07**2     ;RCVD 017
03F1A 0000     (01619)          DATA  0'00**2     ;RCVD 020
03F1B 0012     (01620)          DATA  0'11**2     ;RCVD 021
03F1C 001C     (01621)          DATA  0'16**2     ;RCVD 022
03F1D 0016     (01622)          DATA  0'13**2     ;RCVD 023
03F1E 001C     (01623)          DATA  0'16**2     ;RCVD 024
03F1F 001A     (01624)          DATA  0'15**2     ;RCVD 025
03F20 001C     (01625)          DATA  0'16**2     ;RCVD 026
03F21 001C     (01626)          DATA  0'16**2     ;RCVD 027
03F22 0012     (01627)          DATA  0'11**2     ;RCVD 030
03F23 0012     (01628)          DATA  0'11**2     ;RCVD 031
03F24 0014     (01629)          DATA  0'12**2     ;RCVD 032
03F25 0012     (01630)          DATA  0'11**2     ;RCVD 033
03F26 0018     (01631)          DATA  0'14**2     ;RCVD 034
03F27 0012     (01632)          DATA  0'11**2     ;RCVD 035
03F28 001C     (01633)          DATA  0'16**2     ;RCVD 036
03F29 000E     (01634)          DATA  0'07**2     ;RCVD 037
03F2A 0000     (01635)          DATA  0'00**2     ;RCVD 040
03F2B 000A     (01636)          DATA  0'05**2     ;RCVD 041
03F2C 0004     (01637)          DATA  0'02**2     ;RCVD 042
03F2D 0016     (01638)          DATA  0'13**2     ;RCVD 043
03F2E 000A     (01639)          DATA  0'05**2     ;RCVD 044
03F2F 000A     (01640)          DATA  0'05**2     ;RCVD 045
03F30 000C     (01641)          DATA  0'06**2     ;RCVD 046
03F31 000A     (01642)          DATA  0'05**2     ;RCVD 047
03F32 0004     (01643)          DATA  0'02**2     ;RCVD 050
03F33 0002     (01644)          DATA  0'01**2     ;RCVD 051
03F34 0004     (01645)          DATA  0'02**2     ;RCVD 052
03F35 0004     (01646)          DATA  0'02**2     ;RCVD 053
03F36 0018     (01647)          DATA  0'14**2     ;RCVD 054
03F37 000A     (01648)          DATA  0'05**2     ;RCVD 055
03F38 0004     (01649)          DATA  0'02**2     ;RCVD 056
03F39 000E     (01650)          DATA  0'07**2     ;RCVD 057
03F3A 0010     (01651)          DATA  0'10**2     ;RCVD 060
03F3B 0016     (01652)          DATA  0'13**2     ;RCVD 061
03F3C 0016     (01653)          DATA  0'13**2     ;RCVD 062
03F3D 0016     (01654)          DATA  0'13**2     ;RCVD 063
03F3E 0018     (01655)          DATA  0'14**2     ;RCVD 064
03F3F 000A     (01656)          DATA  0'05**2     ;RCVD 065
03F40 001C     (01657)          DATA  0'16**2     ;RCVD 066
03F41 0016     (01658)          DATA  0'13**2     ;RCVD 067
03F42 0018     (01659)          DATA  0'14**2     ;RCVD 070
03F43 0012     (01660)          DATA  0'11**2     ;RCVD 071
03F44 0004     (01661)          DATA  0'02**2     ;RCVD 072
03F45 0016     (01662)          DATA  0'13**2     ;RCVD 073
03F46 0018     (01663)          DATA  0'14**2     ;RCVD 074
03F47 0018     (01664)          DATA  0'14**2     ;RCVD 075
03F48 0018     (01665)          DATA  0'14**2     ;RCVD 076
03F49 001E     (01666)          DATA  0'17**2     ;RCVD 077
03F4A 0006     (01667)          DATA  0'00**2     ;RCVD 100
03F4B 0006     (01668)          DATA  0'03**2     ;RCVD 101
03F4C 0006     (01669)          DATA  0'03**2     ;RCVD 102
03F4D 0006     (01670)          DATA  0'03**2     ;RCVD 103
```

```
03F4E 0008    (01671)      DATA  0^04^*2    ;RCVD 104
03F4F 001A    (01672)      DATA  0^15^*2    ;RCVD 105
03F50 000C    (01673)      DATA  0^06^*2    ;RCVD 106
03F51 0006    (01674)      DATA  0^03^*2    ;RCVD 107
03F52 0008    (01675)      DATA  0^04^*2    ;RCVD 110
03F53 0002    (01676)      DATA  0^01^*2    ;RCVD 111
03F54 0014    (01677)      DATA  0^12^*2    ;RCVD 112
03F55 0006    (01678)      DATA  0^03^*2    ;RCVD 113
03F56 0008    (01679)      DATA  0^04^*2    ;RCVD 114
03F57 0008    (01680)      DATA  0^04^*2    ;RCVD 115
03F58 0008    (01681)      DATA  0^04^*2    ;RCVD 116
03F59 000E    (01682)      DATA  0^07^*2    ;RCVD 117
03F5A 0010    (01683)      DATA  0^10^*2    ;RCVD 120
03F5B 001A    (01684)      DATA  0^15^*2    ;RCVD 121
03F5C 0014    (01685)      DATA  0^12^*2    ;RCVD 122
03F5D 0006    (01686)      DATA  0^03^*2    ;RCVD 123
03F5E 001A    (01687)      DATA  0^15^*2    ;RCVD 124
03F5F 001A    (01688)      DATA  0^15^*2    ;RCVD 125
03F60 001C    (01689)      DATA  0^16^*2    ;RCVD 126
03F61 001A    (01690)      DATA  0^15^*2    ;RCVD 127
03F62 0014    (01691)      DATA  0^12^*2    ;RCVD 130
03F63 0012    (01692)      DATA  0^11^*2    ;RCVD 131
03F64 0014    (01693)      DATA  0^12^*2    ;RCVD 132
03F65 0014    (01694)      DATA  0^12^*2    ;RCVD 133
03F66 0008    (01695)      DATA  0^04^*2    ;RCVD 134
03F67 001A    (01696)      DATA  0^15^*2    ;RCVD 135
03F68 0014    (01697)      DATA  0^12^*2    ;RCVD 136
03F69 001E    (01698)      DATA  0^17^*2    ;RCVD 137
03F6A 0010    (01699)      DATA  0^10^*2    ;RCVD 140
03F6B 0002    (01700)      DATA  0^01^*2    ;RCVD 141
03F6C 000C    (01701)      DATA  0^06^*2    ;RCVD 142
03F6D 0006    (01702)      DATA  0^03^*2    ;RCVD 143
03F6E 000C    (01703)      DATA  0^06^*2    ;RCVD 144
03F6F 000A    (01704)      DATA  0^05^*2    ;RCVD 145
03F70 000C    (01705)      DATA  0^06^*2    ;RCVD 146
03F71 000C    (01706)      DATA  0^06^*2    ;RCVD 147
03F72 0002    (01707)      DATA  0^01^*2    ;RCVD 150
03F73 0002    (01708)      DATA  0^01^*2    ;RCVD 151
03F74 0004    (01709)      DATA  0^02^*2    ;RCVD 152
03F75 0002    (01710)      DATA  0^01^*2    ;RCVD 153
03F76 0008    (01711)      DATA  0^04^*2    ;RCVD 154
03F77 0002    (01712)      DATA  0^01^*2    ;RCVD 155
03F78 000C    (01713)      DATA  0^06^*2    ;RCVD 156
03F79 001E    (01714)      DATA  0^17^*2    ;RCVD 157
03F7A 0010    (01715)      DATA  0^10^*2    ;RCVD 160
03F7B 0010    (01716)      DATA  0^10^*2    ;RCVD 161
03F7C 0010    (01717)      DATA  0^10^*2    ;RCVD 162
03F7D 0016    (01718)      DATA  0^13^*2    ;RCVD 163
03F7E 0010    (01719)      DATA  0^10^*2    ;RCVD 164
03F7F 001A    (01720)      DATA  0^15^*2    ;RCVD 165
03F80 000C    (01721)      DATA  0^06^*2    ;RCVD 166
03F81 001E    (01722)      DATA  0^17^*2    ;RCVD 167
03F82 0010    (01723)      DATA  0^10^*2    ;RCVD 170
```

```
03F83 0002      (01724)         DATA  0^01^*2    ;RCVD 171
03F84 0014      (01725)         DATA  0^12^*2    ;RCVD 172
03F85 001E      (01726)         DATA  0^17^*2    ;RCVD 173
03F86 0018      (01727)         DATA  0^14^*2    ;RCVD 174
03F87 001E      (01728)         DATA  0^17^*2    ;RCVD 175
03F88 001E      (01729)         DATA  0^17^*2    ;RCVD 176
03F89 001E      (01730)         DATA  0^17^*2    ;RCVD 177
                (01731)
```

```
                    (01732) ;MPGSC -- G-FLAG SET/CLEAR
                    (01733) ;JJW, 18 OCT 79
                    (01734) ;
                    (01735) ;FCB # 123
                    (01736) ;  G-FLAG NUMBER IN 1(R1)
                    (01737) ;  0/1 = CLEAR/SET IN 2(R1)
                    (01738)          EVEN
03F8A F0220001      (01739) MPGSC:   MOVMR   R2,1(R1)          ;GET G-FLAG NUMBER
03F8C 2624          (01740)          INCR    R2,4              ;ADD OFFSET FOR SYS$FLGS
03F8D F0320002      (01741)          MOVMR   R3,2(R1)          ;GET CLR/SET BIT
03F8F 3A35          (01742)          LLS     R3,5              ;SHIFT TO PROPER POSITION
03F90 56260020      (01743)          IORKR   R2,R3,$20         ;AND OFF AND PUT IN R2
03F92 E021FFCE      (01744)          MOVRM   R2,SYS$FLGS       ;DO THE MOVE THAT SETS/CLRS IT
03F94 0E70          (01745)          RETURN
                    (01746) ;
            00000069 (01747) SPARE = $3FFE - .L                ;ROOM TO SPARE IN THIS "HOLE"
                    (01748) ;
                    (01749) ;
                    (01750) ;"BREAKPOINT" FOR DEBUGGING.  TO HALT CSPU EXECUTION AND SAVE
                    (01751) ;  REGISTERS ON THE STACK, PATCH A "CALL R1,BKPT" INTO THE PROGRAM
                    (01752) ;  WITH MPLOOK:
                    (01753) ;        LOC
                    (01754) ;        $8610,
                    (01755) ;        $3FFE
                    (01756) ;
03F95 80003F95      (01757) BKPT:    JMP     BKPT
                    (01758) ;
03F97               (01759)          END                       ;END OF FILE
```

```
ACQTHR:     0000A (00010) (00917)
AD$BGN:     0048A2 (00397) (00419)
AD$CPYA:    0498A (00538) (00583)
AD$CPYB:    04994 (00539) (00590)
AD$DISC:    04986 (00553) (00571)
AD$LOOP:    00004 (00403) (00415)
AD$RTN:     04982 (00550) (00565) (00572)
AD$STOP:    00015 (00408) (00416)
AD$SZ:      0002C (00395) (00419)
ADA$END:    02D2B (00380) (00407)
ADAMINT:    0495C (02106) (00135) (00545)
ADB$END:    02DDF (00381) (00414)
ADBPTR:     04952 (00536) (00554)
ADCNTRL:    00002 (00383) (00426)
ADPO:       00542 (00221) (00232) (00546) (00548)
AOM$NRDY:   03CA0 (01009) (01018)
AOM$RTN:    03C9C (01005) (01014) (01021)
AOMINT:     03C80 (00109) (00129) (01000)
AV$BGN:     04922 (03497) (00506)
AV$SZ:      00010 (00495) (00506)
AVCNTRL:    00001 (00493) (00513)
BBTAB:      05D00 (00025) (00026) (01530)
BITS15:     07FFF (00013) (00295) (00286) (00287) (00288) (00289) (00380) (00381) (00437) (00438)
BKPT:       03F95 (01757) (01757)
BMO7:       03D78 (01240) (01245)
C0:         00002 (00016) (00020) (01460) (01470) (01479)
C1:         00004 (00017) (00020) (00022) (01455)
C2:         00008 (00018) (00020) (00021) (00022) (01450) (01506) (01511)
C210:       0000E (00020) (01578)
C3:         00010 (00019) (00021) (00022) (01429) (01575)
C32:        00018 (00021) (01499)
C321:       0001C (00022) (01473) (01477)
CLKGO:      00002 (00313) (00315)
CLKRATES:   0ECC6 (00293) (00315)
CLKSET:     00000 (00313)
CLR:        00008 (03046) (00565) (00638) (00787) (01014) (01233) (01532)
CORRECT:    03E08 (00104) (01393)
CORTAB:     03F0A (01571) (01602)
D16$INT1:   04022 (02038) (00116)
D16$INT2:   04030 (00039) (00122)
D22$INT1:   040FA (00040) (00128)
D23$INT1:   0411E (00041) (00134)
DA$BGN:     048E2 (00453) (00473)
DA$CPYA:    03CA8 (00991) (01024)
DA$CPYB:    03CB2 (00992) (01031)
DA$LOOP:    00001 (00455) (00469)
DA$STOP:    00015 (00461) (00470)
DA$SZ:      0002C (00451) (00473)
DAA$END:    03BE9 (00437) (00460)
DAB$END:    03C9D (00438) (00468)
DABPTR:     03C76 (00992) (01010) (01020)
DACNTRL:    00002 (00440) (00480)
DAPO:       00549 (00231) (01001) (01003)
```

```
DKTAB1:    05D10 (00026) (00027) (01459)
DKTAB2:    05D50 (00027) (00028) (01462)
DKTAB3:    05D90 (00028) (00029) (01472)
DKTAB4:    05DD0 (00029) (00030) (01486)
DKTAB5:    05DF0 (00030) (00031) (01490)
DKTAB6:    05E10 (00031) (00032) (01503)
DKTAB7:    05E30 (00032) (00033) (01508)
DKTAB8:    05E40 (00033) (00034) (01516)
DTQTAB:    05ED0 (00035) (01447)
EDTAB:     05E50 (00034) (00035) (01454)
ERCORR:    03EF2 (01571)
ERRPTR:    03BDE (00007) (00821) (00823)
ERRSYNC:   03BDF (00828) (00924) (00958) (00965)
FDT$:      007E8 (00043) (00102)
G0:        00004 (00047)
G1:        00005 (00048) (01103) (01233) (01393) (01532)
G2:        00006 (00049) (00545) (00565) (00623) (00638) (00726) (00787) (01000) (01014)
G3:        00007 (00050)
GAIN:      03E04 (01382) (01398) (01410) (01416) (01453)
GET1:      03F08 (01412) (01415) (01484) (01488) (01492) (01595)
GET2:      03EFE (01495) (01514) (01585)
GET74:     03ED2 (01399) (01404) (01407) (01418) (01423) (01428) (01464) (01469) (01476) (01498)
                 (01548)
GHIST:     0BCEE (00055) (00056) (01123)
GT1:       03F03 (01589) (01596)
H$:        00001 (00007) (00054) (00055) (00056) (00057) (00058) (00059) (00060) (00061) (00062)
                 (00063)
H000:      03DF2 (01271) (01345)
H017:      03DEB (01278) (01338)
H026:      03DCE (01285) (01309)
H031:      03DD4 (01280) (01315)
H045:      03DCC (01276) (01307)
H052:      03DE4 (01273) (01331)
H063:      03DD5 (01282) (01316)
H074:      03DED (01283) (01340)
H103:      03DE9 (01274) (01336)
H114:      03DD3 (01275) (01314)
H125:      03DE3 (01284) (01330)
H132:      03DE0 (01281) (01327)
H146:      03DD2 (01277) (01313)
H151:      03DCA (01272) (01305)
H160:      03DEF (01279) (01342)
H177:      03DDA (01286) (01321)
HPTR:      03D92 (01243) (01271)
IDW$CS:    00010 (00069)
IDW$DM:    00008 (00070)
IDW$IC:    00020 (00068)
IDW$ONHK:  00080 (00066) (00735)
IDW$RD:    00001 (00073) (00808) (00892) (00927) (00967)
IDW$RR:    00004 (00071)
IDW$SI:    00040 (00067)
IDW$SQ:    00002 (00072)
ISVT$:     00582 (00075) (00203) (00204) (00205) (00206) (00207) (00208) (00209) (00210) (00221)
```

```
                    (00222) (00224) (00225) (00227) (00228) (00230) (00231) (00237) (00238) (00239)
                    (00240) (00244) (00249) (00250) (00251) (00252) (00253) (00254) (00261) (00263)
                    (00265)
  K1:          03E05 (01383) (01421) (01458)
  K1HIST:      0BD2E (00056) (00057) (01145)
  K2:          03E06 (01394) (01426) (01461)
  K2HIST:      0BD4E (00057) (00058) (01153)
  K3:          03E02 (01385) (01467) (01471)
  K3HIST:      0BD6E (00058) (00059) (01165)
  K4:          03E02 (01386) (01474) (01485)
  K4HIST:      0BD8E (00059) (00060) (01173)
  K5:          03E03 (01387) (01478) (01489)
  K5HIST:      0BD9E (00060) (00061) (01182)
  K6:          03E04 (01388) (01482) (01493) (01502)
  K6HIST:      0BDAE (00061) (00062) (01186)
  K7:          03E02 (01389) (01496) (01507)
  K7HIST:      0BDBE (00062) (00063) (01202)
  K8:          03E03 (01390) (01512) (01515)
  K8HIST:      0BDC6 (00063) (01213)
  LASTERR:     03BE0 (00829) (00925) (00955) (00959) (00964)
  LOSETHR:     00004 (00076) (00923) (00957)
  MBLNGTH:     00105 (00151) (00193) (00195) (00285) (00286) (00287) (00288) (00289) (00761) (00842)
                    (00858) (00863) (00889)
  ML6:         00028 (00152) (00661) (00663) (00664) (00670) (00672) (00673)
  MOD$BGN:     04802 (00311) (00366)
  MOD$INIT:    00006 (00316) (00319)
  MOD$LOOP:    0000A (00323) (00325) (00333) (00336) (00339) (00342) (00345) (00348) (00355) (00357)
                    (00360) (00362)
  MOD$RA:      00015 (00319) (00332) (00346)
  MOD$RB:      0001C (00330) (00330) (00334) (00338)
  MOD$RC:      00022 (00331) (00331) (00340) (00344)
  MOD$RCVR:    00010 (00323) (00329)
  MOD$SZ:      0006E (00309) (00366)
  MOD$TA:      00028 (00354)
  MOD$TB:      00031 (00352) (00359)
  MOD$XMTR:    00028 (00324) (00352)
  MPGSC:       03F8A (00105) (01739)
  NEWMAX:      03C1C (00898)
  NOCORR:      03EF6 (01570) (01575)
  NOMAX:       03C22 (00897) (00904)
  ODW$RS:      00F04 (00092) (00091)
  ODW$SD:      00001 (00084)
  ODW$SR:      00002 (00083)
  ODW$TR:      00008 (00081) (00091)
  OLDSYNC:     03BE1 (00830) (00928) (00952) (00968)
  OUT2:        03D88 (01197) (01221) (01260)
  OUT3:        03D7C (01209) (01250)
  OUT7:        03D70 (01114) (01119) (01127) (01149) (01157) (01161) (01169) (01178) (01193) (01217)
                    (01240)
  P0:          03DF3 (01289) (01346)
  P1:          03DD8 (01290) (01319)
  P2:          03DCF (01291) (01310)
  P3:          03DE1 (01292) (01328)
```

```
P4:          03DF1 (01293) (01344)
P5:          03DCB (01294) (01306)
P6:          03DD6 (01295) (01317)
P7:          03DDB (01296) (01322)
PHIST:       0BC9E (00053) (00054) (01110)
PITCH:       03E02 (01380) (01385) (01386) (01389) (01402) (01413) (01432)
PITCHL:      00000A (00086) (01433) (01435)
PITCHU:      0004C (00087) (01436) (01438)
PRBBLP:      03D5E (01224) (01232)
PROTECT:     03CC4 (00103) (01103)
PTR2:        03DC2 (01262) (01299)
PTR3:        03DB2 (01228) (01252) (01289)
Q0:          03DF4 (01299) (01347)
Q1:          03DD9 (01300) (01320)
Q2:          03DD0 (01301) (01311)
Q3:          03DE2 (01302) (01329)
RAND$L:      00BE8 (00088) (00820) (00823)
RAND$U:      01C77 (00089) (00818)
RBUFO:       00553 (00254) (00756) (00762) (00766) (00922) (00949)
RBTA:        00A6C (00158) (01373)
RBTB:        00B72 (00159) (01374)
RBTBPTR:     03DFC (00759) (00853) (01374) (01395)
RBTFA:       00538 (00207) (00213) (00710)
RBTFB:       0053C (00208) (00711)
RBTFPTR:     03B50 (00711) (00748) (00772)
RBTPO:       00547 (00228) (00747) (00774) (00851)
RDABA:       00B36 (00182) (00437) (00456) (01025) (01026) (01027)
RDABB:       00B3A (00183) (00438) (00464) (01032) (01033) (01034)
RERSIM:      0057F (00263) (00737)
RFRCTR:      00549 (00250) (00775)
RLSCTR:      00550 (00251) (00973)
RMA$END:     033E4 (00285) (00333)
RMB$END:     034EA (00286) (00339)
RMC$END:     035F0 (00287) (00345)
RMDMA:       0B2E0 (00172) (00285) (00320) (00347) (00714)
RMDMB:       0B3E6 (00173) (00286) (00335) (00715)
RMDMC:       0B4EC (00174) (00287) (00341) (00713) (00716)
RMFDC:       00540 (00239) (00793)
RMI$BMO1:    03B98 (00760) (00765)
RMI$BMO2:    03BA2 (00764) (00771)
RMI$DISC:    03BBE (00749) (00793)
RMI$LOST:    03BB2 (00746) (00783)
RMI$RTN:     03BBA (00733) (00776) (00780) (00787) (00794)
RMI$SIM:     03BC2 (00739) (00805)
RMI$SINI:    03BAE (00742) (00779)
RMI$SRCH:    03BB8 (00743) (00786)
RMI$STF:     03BA4 (00767) (00772)
RMODEMIN:    03B5A (00108) (00123) (00725)
RMPO:        00546 (00227) (00727) (00731)
RMPTR:       03B58 (00716) (00734) (00755) (00768) (00783) (00785)
RNOCOR:      0057D (00261) (00266) (01569)
RONHK:       00551 (00252) (00555) (00736)
RSIBPTR:     03C7A (00995) (01007)
```

```
RSIFPTR:   03C7E (00997) (01008) (01011)
RSNFA:     00530 (00209) (00996)
RSNFB:     00535 (00210) (00997)
RSNKA:     0B91A (00179) (00994)
RSNKB:     0B9CE (00180) (00995)
RSNKC:     0BA82 (00181) (01019)
RSNNR:     00540 (00240) (01010)
RSNPO:     00548 (00230) (01006) (01013)
RSRA:      0B7FE (00177) (00849) (01376)
RSRB:      0B88C (00178) (00850) (01377)
RSRBPTR:   03E00 (01377) (01397)
RSSPF:     0B5F2 (00175) (00845) (00893) (00910)
RSSSS:     0B6F0 (00176) (00860) (00861) (00862) (00863) (00895) (00896) (00898) (00909)
RSYNC:     00552 (00253) (00740) (00745) (00948) (00921) (00932) (00972)
RUN:       00540 (00244) (00246) (00549) (00627) (00732) (01004)
RUNMOD:    00001 (00314)
SBLNGTH:   000B4 (00149) (00150) (00380) (00381) (00437) (00438)
SET:       00020 (00045) (00545) (00623) (00726) (01000) (01103) (01393)
SI$BMO:    03C00 (00856) (00859)
SI$LOOP:   03BE5 (00844) (00847)
SL6:       0001E (00150) (00583) (00585) (00586) (00590) (00592) (00593) (01024) (01026) (01027)
                 (01031) (01033) (01034)
SPARE:     00069 (01747)
SS$ALT:    03C16 (00895)
SS$LOOP:   03C10 (00892) (00913)
SS$LPTST:  03C20 (00901) (00904) (00906) (00910)
SS$NO:     03C4B (00918) (00920) (00932)
SS$NOALT:  03C26 (00894) (00909)
SU$ELF:    03C5C (00956) (00959)
SU$ERR:    03C60 (00954) (00963)
SU$LOSE:   03C6E (00966) (00972)
SU$VALID:  03C68 (00960) (00967)
SYNCINIT:  03BE2 (00779) (00784) (00842)
SYNCSRCH:  03C0C (00786) (00889)
SYNCUPDA:  03C4C (00744) (00949)
SYSSFLGS:  1FFCE (00090) (00545) (00565) (00623) (00638) (00726) (00787) (01000) (01014) (01103)
                 (01233) (01393) (01532) (01744)
TAD9A:     0AC78 (00161) (00380) (00483) (00535)
TADBB:     0AD2C (00162) (00381) (00410) (00536)
TADBC:     0ADE0 (00163) (00557)
TADFDC:    00540 (00237) (00256) (00571)
TAP:       03E03 (01381) (01387) (01390) (01405) (01446)
TBTA:      0A75A (00155) (01100)
TBTB:      0AB60 (00156) (01101)
TBTBPTR:   03CC2 (00632) (01101) (01106)
TBTC:      0A966 (00157) (00646)
TBTFA:     00536 (00205) (00616)
TBTFB:     00537 (00206) (00617)
TBTFPTR:   03B02 (00617) (00630) (00635)
TBTPO:     00544 (00224) (00629) (00637)
TFRCTR:    00540 (00249) (00564)
THIST:     0BCDE (00054) (00055) (01118)
TMA$END:   031D0 (00288) (00355)
```

```
  TMB$END:    032DE (00289) (00360)
  TMBITS:     0000C (00091) (00187) (00189) (00194) (00196) (01130) (01135) (01305) (01306) (01307)
                    (01308) (01309) (01310) (01311) (01312) (01313) (01314) (01315) (01316) (01317)
                    (01318) (01319) (01320) (01321) (01322) (01323) (01324) (01325) (01326) (01327)
                    (01328) (01329) (01330) (01331) (01332) (01333) (01334) (01335) (01336) (01337)
                    (01338) (01339) (01340) (01341) (01342) (01343) (01344) (01345) (01346) (01347)
                    (01348) (01349) (01350) (01351)
  TMDMA:      0B0D4 (00169) (00186) (00193) (00288) (00322) (00361) (00660) (00662) (00663) (00664)
  TMDMB:      0B1DA (00170) (00188) (00195) (00289) (00356) (00669) (00671) (00672) (00673)
  TMFFC:      00548 (00238) (00645)
  TMI$CPYA:   03B32 (00618) (00659)
  TMI$CPYB:   03B40 (00619) (00660)
  TMI$FAKE:   03B2A (00631) (00645)
  TMI$RTN:    03B26 (00628) (00638) (00648)
  TMODEMIN:   03B00 (00107) (00117) (00622)
  TMPO:       00545 (00225) (00624) (00626)
  TMPTR:      03B06 (00619) (00634) (00647)
  TSNBPTR:    03CBE (01098) (01105)
  TSNKA:      0AFFC (00166) (01097)
  TSNKB:      0B004 (00167) (01098)
  TSNKC:      0B08C (00168)
  TSRA:       0AE94 (00164) (00584) (00585) (00586)
  TSRB:       0AF40 (00165) (00591) (00592) (00593)
  TSRBPTR:    04956 (00539) (00559)
  TSRFA:      00534 (00203) (00211) (00541)
  TSRFB:      00535 (00204) (00542)
  TSRFPTR:    0495A (00542) (00552) (00561)
  TSRPO:      00543 (00222) (00551) (00563)
  TSTATE:     00501 (00265) (00499)
  W$:         00002 (00008) (00026) (00027) (00028) (00029) (00030) (00031) (00032) (00033) (00034)
                    (00035) (00102) (00849) (00850) (00855) (00857) (00860) (00862)

LINES WITH ERRORS:     0  (MAP VERSION 000101.10)   E-    0
```

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


```
                    (00001) ;[BBN-TENEXD]<MAP>BBNPAT.MSO.4,  8-Jan-80 18:55:38, Ed: KFIELD
PAGE  2:  [BBN-TENEXD]<MAP>BBNPAT.MSO.4,  8-Jan-80 18:55:38, Ed: KFIELD
          BBN SPEECH CODER - PATCHES TO MAP-300 SNAPII EXEC

                    (00002) "BBN SPEECH CODER - PATCHES TO MAP-300 SNAPII EXEC
                    (00003) *
                    (00004) * THIS PATCH FILE ASSUMES MAP HAS BEEN LOADED WITH
                    (00005) * SNAPII EXEC, RELEASE 3.5.
                    (00006) *
                    (00007) * THIS PATCH ASSUMES REV. 18 (OR LATER) MICROCODE.
                    (00008) *
                    (00009) * PATCH PREBINDING ROUTINES TO WORK WITH PREBINDING
                    (00010) * BUFFERS ABOVE $7FFF.
                    (00011) * (NOTE: THIS PATCH USES PATCH SPACE FROM $1C98 - $1CA3.)
                    (00012) *
                    (00013) * DEFINE "SAF" (SET ADDRESS FIELD) AND "LAF" (LOAD
                    (00014) * ADDRESS FIELD) INSTRUCTIONS TO ASSEMBLER.
                    (00015) * THESE INSTRUCTIONS ARE NEW CSPU INSTRUCTIONS
                    (00016) * AVAILABLE WITH "REV 18" CSPU MICROCODE.
                    (00017) * THEY SET/LOAD THE LOW ORDER 17 BITS OF THE
                    (00018) * REFERENCED FULL-WORD FROM/INTO THE INDICATED
                    (00019) * REGISTER.
                    (00020) *
                    (00021)   OPADD   SAF,(1 .LS. 14) + (29 .LS. 8) + $EF
                    (00022)   OPADD   LAF,(1 .LS. 14) + (29 .LS. 8) + $EE
                    (00023) *
                    (00024) *
          00000582  (00025)   BCT$BA = $0582
          0000024A  (00026)   AP$CSL = $024A
          00000E38  (00027)   SCD$04R= $0E38
                    (00028) *
                    (00029) *
          00000E60  (00030)   #L = $0E60
     00E6F EE3A0582 (00031)   LAF     R3,BCT$BA(R5)
                    (00032) *
          00000F6C  (00033)   #L = $0F6C
     00F6C EE440582 (00034)   LAF     R4,BCT$BA(R2)
                    (00035) *
          00000F8C  (00036)   #L = $0F8C
     00F8C EF540582 (00037)   SAF     R5,BCT$BA(R2)
                    (00038) *
          00000FB1  (00039)   #L = $0FB1
     00FB1 00001C98 (00040)   JMP     PATCH1
                    (00041) RET1:
                    (00042) *
          00000E58  (00043)   #L = $0E58
     00E58 00001CA0 (00044)   JMP     PATCH2
                    (00045) *
PAGE  3:  [BBN-TENEXD]<MAP>BBNPAT.MSO.4,  8-Jan-80 18:55:38, Ed: KFIELD
          BBN SPEECH CODER - PATCHES TO MAP-300 SNAPII EXEC

          00001C98 (00046)   #L = $1C98
```

```
                        (00047) PATCH1:
      01C98 C06000024A  (00048)   MOVMRL   R6,APSCSL
      01C9A 3A62        (00049)   LLS      R6,2
      01C9B 3A72        (00050)   LLS      R7,2
      01C9C 3C62        (00051)   LRS      R6,2
      01C9D 3C72        (00052)   LRS      R7,2
      01C9E 80000FB3    (00053)   JMP      RET1
                        (00054) *
            00001CA0    (00055)   #L = $1CA0
                        (00056) PATCH2:
      01CA0 3A32        (00057)   LLS      R3,2
      01CA1 3C32        (00058)   LRS      R3,2
      01CA2 80000E3B    (00059)   JMP      SCD$04H
                        (00060) *
                        (00061) *
      01CA4             (00062)   END
PAGE    4:  [BBN-TENEXD]<MAP>BBNPAT.MSO.4,  8-Jan-80 18:55:38, Ed: KFIELD
            BBN SPEECH CODER - PATCHES TO MAP-300 SNAPII EXEC

  APSCSL:    0024A (00026) (00048)
  BCT$BA:    00582 (00025) (00031) (00034) (00037)
  PATCH1:    01C98 (00048) (00047)
  PATCH2:    01CA0 (00044) (00056)
  RET1:      00FB3 (20041) (00053)
  SCD$04H:   00E3B (00027) (00059)


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

```
CC BBN-TENEXD]<MAP>BBNHSP.R35.26, 17-Dec-79 19:18:57, Ed: KFIELD
C
C
C
        INTEGER FUNCTION DCOR(Y,U,V)
C
C       USAGE:  IER = DCOR(Y,U,V)
C
C       DO CORRELATION ON VECTOR V WITH KERNAL U
C        RESULT IN Y.
C
        INTEGER Y,U,V,FCBGN
C
        DCOR = FCBGN(191,Y,0,U,0,V,0,0,0,0)
C
        RETURN
        END
C
C
C
C
        INTEGER FUNCTION PTAP(Y,A,U,B,V,C,W)
C
C USAGE IER = PTAP(Y,A,U,B,V,C,W)
C
C COMPUTE PITCH AND TAP AND DO PITCH REMOVAL
C
C Y IS OUTPUT BUFFER: B.B. WITH PITCH REMOVED
C A IS OUTPUT SCALAR: PITCH
C U IS INPUT BUFFER: B.B. AUTOCORRELATION (PITCH CALC. PART)
C B IS OUTPUT SCALAR: QUANTIZED TAP
C V IS INPUT BUFFER: DOWNSAMPLED B.B. EXCITATION
C C IS OUTPUT SCALAR: CODED TAP (INTEGER - RIGHT JUST. IN L HWORD)
C W IS INPUT BUFFER: B.B. AUTOCORR. (WHOLE THING)
C
C
        INTEGER FCBGN,Y,A,U,B,V,C,W
        PTAP = FCBGN(212,Y,A,U,B,V,C,W,0,14)
        RETURN
        END
C
C
C
C
        INTEGER FUNCTION ENRG(A,B,C,W,D)
C
C USAGE IER = ENRG(A,B,C,W,D)
C
C COMPUTE, CODE & QUANTIZE ENERGY (GAIN)
C
C A IS OUTPUT SCALAR: QUANTIZED GAIN
```



```
C B IS OUTPUT SCALAR: INVERSE OF QUANTIZED GAIN
C C IS OUTPUT SCALAR: CODED GAIN
C W IS INPUT BUFFER:  B.B. WITH PITCH REMOVED
C D IS INPUT SCALAR:  INVERSE OF DOWNSAMPLED FRAMESIZE
C
C
        INTEGER FCBGN,A,B,C,W,D
        ENRG = FCBGN(199,1,A,1,B,1,C,W,D,14)
        RETURN
        END
C
C
C
        INTEGER FUNCTION VLTSV(Y,U,V,W)
C
C USAGE  IER = VLTSV(Y,U,V,W)
C
C  NOTE: USE CALLING SEQUENCE #17 (Y,A,U,V,W), WITH A=0
C
        INTEGER FCBGN,Y,U,V,W
        VLTSV = FCBGN(132,Y,0,U,0,V,0,W,0,17)
        RETURN
        END
C
C
C
        INTEGER FUNCTION VKTOA(Y,U)
C
C USAGE  IER = VKTOA(Y,U)
C
C  U IS ARRAY OF K'S
C  Y IS ARRAY OF A'S
C
        INTEGER FCBGN,Y,U
        VKTOA = FCBGN(133,Y,0,U,0,0,0,0,0,11)
        RETURN
        END
C
C
C
        INTEGER FUNCTION PRTRB(Y,A,U,B,V)
C
C USAGE  IER = PRTRB(Y,A,U,B,V)
C
C  U IS DOWNSAMPLED INPUT VECTOR
C  V IS VECTOR OF RANDOM NUMBERS
C  Y IS PERTURBED UPSAMPLED OUTPUT VECTOR
C  SA,SB ARE EXTERNAL CONSTANTS
C
        INTEGER FCBGN,Y,A,U,B,V
        PRTRB = FCBGN(134,Y,A,U,B,V,0,0,0,4)
        RETURN
        END
```

```
C
C
C
      INTEGER FUNCTION MWLF(Y,A,U,V)
C
C  USAGE  IER = MWLF(Y,A,U,V)
C
C
C  Y IS ARRAY OF FIXED K'S
C  U IS ARRAY OF (CODED K,QUANTIZED K)
C  V IS INPUT ARRA OF AUTOCORRELATION COEFFS
C
      INTEGER FCBGN,Y,A,U,V
      MWLF=FCBGN(135,Y,A,U,0,V,0,0,0,9)
      RETURN
      END
C
C
C
      INTEGER FUNCTION VAPC(Y,A,U,B,V,C,D)
C
C  USAGE  IER = VAPC(Y,A,U,B,V,C,D)
C  Y IS ARRAY OF CODED PARAMETERS
C  U IS ARRAY OF QUANTIZE ERRORS
C  V IS INPUT
C  A IS PITCH (REAL SCALAR)
C  B IS TAP, FOLLOWED BY INVERSE OF GAIN (REAL SCALARS)
C  C IS CODED TAP, CODED GAIN, & CODED REFL. COEFFS
C     (INTEGERS IN LEFT HWARD OF REAL SCALARS)
C  D IS B.B. QUANT INFO (REAL SCALARS)
C     (3 QUANT THRESHOLDS, 4 QUANT VALUES)
C
C
      INTEGER FCBGN,Y,A,U,B,V,C,D
      VAPC=FCBGN(150,Y,A,U,B,V,C,0,0,16)
      RETURN
      END
C
C
C
      INTEGER FUNCTION DEAL(Y,A,U,B,V)
C
C  USAGE IER = DEAL(Y,A,U,B,V)
C
C Y IS BASE BAND OUTPUT
C U IS REFLN COEF OUTPUT
C V IS INPUT PARAM BUFFER
C A IS PITCH
C B IS TAP
```



```
C
      INTEGER FCBGN,Y,A,U,B,V
      DEAL=FCBGN(190,Y,A,U,B,V,0,0,0,4)
      RETURN
      END
C
C
C
      INTEGER FUNCTION VIAPC(Y,A,U,B)
C
C  USAGE  IER = VIAPC(Y,A,U,B)
C
C Y IS OUTPUT BASEBAND SAMPLES
C A IS TAP
C U IS INPUT
C B IS PITCH
C
      INTEGER FCBGN,Y,A,U,B
      VIAPC=FCBGN(196,Y,A,U,B,0,0,0,0,2)
      RETURN
      END
C
C
C
      INTEGER FUNCTION MPIFF(ISA,ISB,FLID)
C
C SUPPORT MODULE FOR THE INTEGER "IF A.NE.0 .AND. B.EQ.0"
C  CONDITIONAL FCB.
C
C CALLING SEQUENCE:
C  IER = MPIFF(ISA,ISB,FLID)
C  WHERE:
C   ISA = INTEGER SCALAR ID (0 .LE. ISA .LE. 127)
C   ISB = INTEGER SCALAR ID (0 .LE. ISB .LE. 127)
C   FLID = FUNCTION LIST ID (0 .LE. FLID .LE. 63)
C
      INTEGER FCBRG,FCBSZ,HWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LEVEL
C
      MPIFF = 0
C
      DO 10 I=1,11
   10 FCBRG(I) = 0
C
C FCB #105
C
      FCBRG(2) = 105
C
      IF ((FLID .LT. 0) .OR. (FLID .GT. 63)) MPIFF = -3
      IF ((ISB .LT. 0) .OR. (ISB .GT. 127)) MPIFF = -2
      IF ((ISA .LT. 0) .OR. (ISA .GT. 127)) MPIFF = -1
```

```
C
      IF (MPIFF .NE. 0) CALL MPERR(MPIFF)
      IF (MPIFF .NE. 0) RETURN
C
      FCBRG(3) = ISA
      FCBRG(5) = ISR
      FCBRG(7) = FLID
C
      MPIFF = RUNMP(FCBRG(1),FCBSZ(1,5))
C
      RETURN
      END
C
C
C
      INTEGER FUNCTION MPXBM(FCBNO,Y,A,U,V)
C
C SUPPORT MODULE FOR THE EXECUTE BOUND VERSION OF MWLF FCB
C
C CALLING SEQUENCE:
C
C       MAP = MPXBM(FCBNO,Y,A,U,V)
C
C WHERE:
C
C       FCBNO = FCB NUMBER OF CREATED BOUND MWLF FUNCTION
C       Y,A,U,V = PARAMETERS TO MWLF
C
      INTEGER FCBRG,FCBSZ,HWS,FCB,HRI,RUNMP,FCBNO,Y,A,U,V
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LEVEL
C
      MPXBM = 0
      DO 10 I=1,11
10    FCBRG(I) = 0
C
      IF ((V .LT. 1) .OR. (V .GT. 63)) MPXBM=-5
      IF ((U .LT. 1) .OR. (U .GT. 63)) MPXBM=-4
      IF ((A .LT. 0) .OR. (A .GT. 255)) MPXBM=-3
      IF ((Y .LT. 1) .OR. (Y .GT. 63)) MPXBM=-2
      IF ((FCBNO .LT. 0) .OR. (FCBNO .GT. 255)) MPXBM=-1
      IF (MPXBM .NE. 0) CALL MPERR(MPXBM)
      IF (MPXBM .NE. 0) RETURN
C
      FCBRG(2) = FCBNO
      FCBRG(3) = 2
      FCBRG(4) = Y
      FCBRG(5) = A
      FCBRG(6) = U
      FCBRG(8) = V
C
      MPXBM = RUNMP(FCBRG(1),FCBSZ(1,5))
      RETURN
      END
```



```
C
C
C
C HOST SUPPORT FOR 4 FCB'S THAT CALL THE SPEECH CODER SCROLL
C   INTERRUPT ROUTINES AS SUBROUTINES.  THEY PASS NO ARGUMENTS TO
C   THE MAP.
C
      INTEGER FUNCTION IADINT(I)
C
C FCB # 124
      IADINT=INTFN(124,0)
      RETURN
      END
C
      INTEGER FUNCTION ITMINT(I)
C
C FCB # 125
      ITMINT=INTFN(125,0)
      RETURN
      END
C
      INTEGER FUNCTION IRMINT(I)
C
C FCB # 126
      IRMINT=INTFN(126,0)
      RETURN
      END
C
      INTEGER FUNCTION IDAINT(I)
C
C FCB # 127
      IDAINT=INTFN(127,0)
      RETURN
      END
C
      INTEGER FUNCTION INTFN(IFCBN,IARG)
C CALLER FOR IADINT, ITMINT, IRMINT, IDAINT
C  ALSO CALLER FOR PROPAR, PROBB, CORPAR, CORBB
      INTEGER FCBRG,FCBSZ,HWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LE
      INTFN=0
      DO 10 I=1,11
10      FCBRG(I)=0
      FCBRG(2)=IFCBN
C
      IF(IARG.NE.0 .AND. IARG.NE.-2) INTFN=-1
      IF(INTFN.NE.0) CALL MPERR(INTFN)
      IF(INTFN.NE.0) RETURN
C
      FCBRG(4)=IARG
      INTFN=RUNMP(FCBRG(1),FCBSZ(1,2))
      RETURN
      END
```

```
C
C
C
      INTEGER FUNCTION MPGSC(GFLAG,SETCLR)
C
C SUPPORT MODULE FOR MPGSC -- GFLAG SET/CLEAR
C
C CALLING SEQUENCE:
C  IER = MPGSC(GFLAG,SETCLR)
C  WHERE:
C   GFLAG = G-FLAG NUMBER (0,1,2,3)
C   SETCLR = 0 TO CLEAR IT, 1 TO SET IT
C
      INTEGER GFLAG,SETCLR
      INTEGER FCBRG,FCBSZ,HWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LEVEL
C
      MPGSC = 0
C
      DO 10 I=1,11
10    FCBRG(I) = 0
C
C FCB #123
C
      FCBRG(2) = 123
C
      IF ((SETCLR .LT. 0) .OR. (SETCLR .GT. 1)) MPGSC = -2
      IF ((GFLAG .LT. 0) .OR. (GFLAG .GT. 3)) MPGSC = -1
C
      IF (MPGSC .NE. 0) CALL MPERR(MPGSC)
      IF (MPGSC .NE. 0) RETURN
C
      FCBRG(5) = GFLAG
      FCBRG(7) = SETCLR
C
      MPGSC = RUNMP(FCBRG(1),FCBSZ(1,5))
C
      RETURN
      END
C
C
C
C  HOST SUPPORT FOR PROTCT, CORECT
C  CALLING SEQUENCE (E.G., PROTCT):
C       IVALUE=PROTCT(IAB)
C  WHERE IAB = -2 MEANS PROTECT TSNKA, 0 MEANS PROTECT TSNKB
C
C
      INTEGER FUNCTION PROTCT(IAB)
C
```



```
C FCB #121
      PROPAR=INTFN(121,IAB)
      RETURN
      END
C
C
      INTEGER FUNCTION CORECT(IAB)
C
C FCB #122
      CORECT=INTFN(122,IAB)
      RETURN
      END
C
C
C
   INTEGER FUNCTION MPMBS( BID, SID, NSC )
C
C
C   SUPPORT MODULE FOR THE MOVE BUFFER INTO SCALAR TABLE FCB
C     SAME AS MPTBS EXCEPT NO CHANGE TO BUFFER ADDRESS
C
C
C    CALLING SEQUENCE:
C       MAP = MPMBS( BID, SID, NSC )
C    WHERE:
C       BID = MAP LOGICAL BUFFER IDENTIFIER ( 1.LE. BID .LE.63 )
C       SID = SCALAR IDENTIFIER (0.LE. SID .LE.191 )
C       NSC = NUMBER OF SCALARS TO TRANSFER
C
C       MPMBS =  0 => NO ERROR, ARGUMENTS INSIDE PROPER BOUNDS
C             = -1 => INDICATES 1-ST ARGUMENT (BID) IS OUTSIDE OF LEGAL
C                                    ** BOUNDS
C             = -2 => INDICATES 2-ND ARGUMENT (SID) IS OUTSIDE OF LEGAL
C                                    ** BOUNDS
C             = -3 => INDICATES 3-RD ARGUMENT (NSC) IS OUTSIDE OF LEGAL
C                                    ** BOUNDS
C         NOTE: ANY NON-ZERO VALUE IMPLIES MAP EXECUTION HAS BEEN ABORTE
C                                    **D
C
C
C
      INTEGER FCBRG, FCBSZ, HWS,FCB,HRI,BID, SID, RUNMP
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LEVEL
C
C
C
      MPMBS=0
      DO 10 I = 1, 11
```

```
10    FCBRG(1) = 0
      FCBRG(2) = 111
      IF ( (NSC.LE.0) .OR. (SID+NSC .GT. 192) )  MPMBS = -3
      IF( SID.LT.0 .OR. SID.GT.191 )  MPMBS = -2
      IF( BID.LE.0 .OR. BID.GT.63 )  MPMBS = -1
      IF(MPMBS.NE.0)CALL MPERR(MPMBS)
      IF(MPMBS.NE.0)  RETURN
      FCBRG(3) = BID
      FCBRG(4) = SID
      FCBRG(5) = NSC
      MPMBS = RUNMP(FCBRG(1), FCBSZ(1,1))
      RETURN
      END
```

```
C [BBN-TENEXD]<KFIELD>DCA96A.FOR.34, 11-Dec-79 17:32:34, Ed: KFIELD
      SUBROUTINE DCA96C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                        DCA96A
C       DCA96 ADDRESSES - BUFFER AND SCALAR
C       CONFIGURATION AND PRINTOUT OF BUFFER ADDRESSES.
C       (CALL THIS SUBR "DCA96C" SO STANDARD MAINLINE CAN
C       BE USED.)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      LOGICAL*1 NAME(10)
C
C       MAP-RELATED VARIABLES
C
      REAL HWSZ,FWSZ,DWSZ
      INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
      INTEGER PRCSR,AP
C
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
C
      INTEGER TADRA,TADRB,TADRC,TSRA, TSRB, THAMW
      INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
      INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
      INTEGER TSRL, TINF0,TINF1,TINF, TLPFB,TBE0
      INTEGER TBE1, TBE,  TBEZ, TWSRZ
      INTEGER TPAC, TBPCP
      INTEGER TBPR, TBROL,TBROR,TSNKA,TSNKB,TSNKC
      INTEGER TBTA, TBTB, TBTC
      INTEGER TMDMA,TMDMB
      INTEGER RBTA, RBTB
      INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
      INTEGER RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1
      INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
      INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
      INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
      INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C       MAP SCALARS
C
      INTEGER TDCN, TFSZI,TKTHR,TPTC
      INTEGER TE,   TQTAP,TQG,  TQGI, TBT1
      INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
      INTEGER TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4
      INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
      INTEGER TOFSI
      INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
      INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
      INTEGER RPTC
```



```
C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RIPTC
      INTEGER RUN
      INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO
      INTEGER RBTPO,RSNPO,DAPO, TADFDC
      INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
      INTEGER RFRCTR,     RLSCTR
      INTEGER RONHK,RSYNC,RBOFO,RNOCOR
      INTEGER RERSIM,     VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IDS
C
      COMMON /BSIDS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINF0,TINF1,TINF, TLPFB,TBE0,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TBPR, TBROL,TBROR,TSNKA,TSNKB,TSNKC,
     ;TBTA, TBTB, TBTC,
     ;TMDMA,TMDMB,
     ;RBTA, RBTB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1,
     ;RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCN, TFSZI,TKTHR,TPTC,
     ;TE,   TQTAP,TQG,  TQGI, TBT1,
     ;TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRF8,T5,
     ;TOFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     ;RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONHK,RSYNC,RBOFO,RNOCOR,
     ;RERSIM,     VSTATE
```

```
C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     ;ATADBC,STADBC,
     ;ATBTC, STBTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBRDL,STBRDL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC


C
C
C
        DATA PRCSR/1/,AP/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
C
C
C
C       OPEN MAP, INHIBIT BUFFER CHECKING
C
        CALL MPOPN(0)
        CALL MPIBC(0)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP BUFFERS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP BUFFERS.
C       ALL MAP BUFFER ID'S (BID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER BUFFERS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER BUFFERS.
C       AN "A" FOLLOWED BY A BUFFER NAME SPECIFIES
C       THE HOST VARIABLE CONTAINING THE ADDRESS OF
C       THAT BUFFER.  SIMILARLY, AN "S" FOLLOWED BY A
C       BUFFER NAME SPECIFIES THE HOST VARIABLE
C       CONTAINING THE SIZE OF THAT BUFFER.
C
C
```



```
C
C
C       DEFINE MAP BUFFER ID'S
C
        IBUS1 = 64
        IBUS2 = 128
        IBUS3 = 192
C
C
C A/D INPUT FROM ADAM
        TADRA = 0
C
C A/D INPUT FROM ADAM
        TADBB = 0
C
C A/D "TONE" DATA
        TADBC = 0
C
C TSOURCE A BUFFER
        TSRA = 1
C
C TSOURCE B BUFFER
        TSRB = 2
C
C HAMMING WINDOW COEFFS.
        THAMW = 3
C
C WINDOWED TSRA OR TSRB
        TWSR = 4
C
C TWSR WITH 8 ZEROS ON RIGHT
        TWSRZ = 6
C
C AUTOCORRELATION VALUES
        TACV = 7
C
C OUTPUT FROM MWLF, INCLUDING REFLECTION COEFFS AND ERRORS
        TRFER = 8
C
C OUTPUT FROM MWLF: CODED & Q'TIZED RF COEFFS (TCRF & TQRF)
        TCQRF = 9
C
C CODED REFLECTION COEFFS
        TCRF = 10
C
C Q'TIZED REFLECTION COEFFS
        TQRF = 11
C
C LINEAR PREDICTION COEFFS
        TLPC = 7
C
```

```
C REVERSE OF TLPC
      TLPCR = 5
C
C CURRENT TSOURCEX DATA IN LONG REAL FORMAT
      TSR = 13
C
C TSR WITH 8 MEMORY SAMPLES IN FRONT
      TSRM = 14
C
C FIRST 8 SAMPLES OF TSRM
      TSRF = 15
C
C LAST 8 SAMPLES OF TSRM
      TSRL = 16
C
C INVERSE FILTERED SAMPLES (CURRENT FRAME)
      TINF0 = 17
C
C TINF0 PLUS 37 PREVIOUS FRAME ELEMENTS IN FRONT
      TINF1 = 18
C
C 257 INVERSE FILTERED SAMPLES CENTERED ON PREV FRAME
      TINF = 19
C
C LPF COEFFS TO GET BASEBAND EXCITATION
      TLPFB = 20
C
C DOWNSAMPLED BASEBAND (BB) EXCITATION
      TBE0 = 21
C
C PREVIOUS FRAME'S TBE0
      TBE1 = 22
C
C LAST HALF OF TBE1, ALL OF TBE0
      TBE  = 23
C
C TBE PLUS 38 ZEROS
      TBEZ = 24
C
C PITCH AUTOCORRELATION BUFFER
      TPAC = 28
C
C BB EXCITATION PITCH CALC PART (TPAC(5) - TPAC(38))
      TBPCP = 29
C
C BB EXCITATION WITH PITCH REMOVED
      TBPR = 40
C
C BB RESIDUAL DIFFERENCE BUFFER (LEFT HALF)
      TBRDL = 0
C
```



```
C BB RESIDUAL DIFFERENCE BUFFER (RIGHT HALF)
      TBRDR = 33
C
C TSINK A BUFFER
      TSNKA = 35
C
C TSINK B BUFFER
      TSNKB = 36
C
C TSINK "SILENCE" BUFFER (NO LONGER USED)
      TSNKC = 0
C
C TBITS A BUFFER
      TBTA = 0
C
C TBITS B BUFFER
      TBTB = 0
C
C TBITS C BUFFER
      TBTC = 0
C
C TMODEM A BUFFER
      TMDMA = 0
C
C TMODEM B BUFFER
      TMDMB = 0
C
C RMODEM A BUFFER
      RMDMA = 0
C
C RMODEM B BUFFER
      RMDMB = 0
C
C RMODEM C BUFFER
      RMDMC = 0
C
C RBITS A BUFFER
      RBTA = 0
C
C RBITS B BUFFER
      RBTB = 0
C
C SYNC SEARCH PREVIOUS FRAME
      RSSPF = 0
C
C SYNC SEARCH SUM SYNC
      RSSSS = 0
C
C RSOURCE A BUFFER
      RSRA = 37
C
```

```
C RSOURCE B BUFFER
      RSRB = 38
C
C CURRENT FRAME REFLECTION COEFFS
      RRF0 = 39
C
C BB RESIDUAL
      RBR  = 40
C
C BB EXCITATION
      RBE  = 41
C
C HPF'D BB EXCITATION
      RHBE0 = 43
C
C CURRENT FRAME HPF'D BB EXCITATION, PLUS 12 PREV FRAME ELEMENTS
      RHBE1 = 44
C
C 84 HPF'D BB EXCITATION SAMPLES CENTERED ON PREV FRAME
      RHBE  = 45
C
C LPF COEFFS (WITH GAIN OF 3) TO GET UPSAMPLED BB EXCITATION
      RLPU = 0
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH 3RD
      RLPU1 = 46
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH SECOND
      RLPU2 = 47
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH 1ST
      RLPU3 = 48
C
C UPSAMPLED, LPF'D BB EXCITATION
      RUBE = 49
C
C EVERY 3RD ELEMENT OF RUBE STARTING WITH FIRST
      RUBE1 = 50
C
C EVERY 3RD ELEMENT OF RUBE STARTING WITH SECOND
      RUBE2 = 51
C
C EVERY 3RD ELEMENT OF RUBE STARTING WITH THIRD
      RUBE3 = 52
C
C RANDOM NUMBER ARRAY
      RRND = 53
C
C CURRENT FRAME OF UPSAMPLED AND PERTURBED BB SAMPLES
      RUPB0 = 54
C
```



```
C RUPB0 PLUS 37 PREV FRAME ELEMENTS IN FRONT
      RUPB1 = 55
C
C 254 UPSAMPLED AND PERTURBED BB SAMPLES CENTERED ON PREV FRAME
      RUPB = 56
C
C HPF COEFFS (WITH GAIN OF 3) TO GET UPSAMPLED, PERTURBED SAMPLES
      RHPU = 57
C
C HPF'D, UPSAMPLED, PERTURBED SAMPLES
      RHUP = 58
C
C FILTERED EXCITATION SAMPLES
      RFES = 58
C
C PREV FRAME'S RRF0 (BID #39)
      RRF1 = 59
C
C SYNTHESIS FILTER MEMORY
      RSFM = 60
C
C SYNTHESIZED SPEECH
      RSNKA = 61
C
C SYNTHESIZED SPEECH
      RSNKB = 62
C
C D/A 'SILENCE' BUFFER
      RSNKC = 0
C
C D/A OUTPUT TO AOM
      RDABA = 0
C
C D/A OUTPUT TO AOM
      RDABB = 0
C
C
C
C
C     DEFINE BUFFER SIZES (NUMBER OF SAMPLES PER BUFFER)
C
      STADBA = 180.
      STADBB = 180.
      STADBC = 180.
      STSRA  = 180.
      STSRB  = 180.
      STHAMW = 180.
      STWSR  = 180.
      STWSRZ = 180.
      STACV  = 9.
      STRFER = 16.
```

```
      STCORF = 8.
      STCRF  = 8.
      STORF  = 8.
      STLPC  = 9.
      STLPCR = 9.
      STSR   = 180.
      STSRM  = 180.
      STSRF  = 8.
      STSRL  = 8.
      STINF0 = 180.
      STINF1 = 217.
      STINF  = 254.
      STLPFR = 75.
      STBE0  = 60.
      STBE1  = 60.
      STBE   = 90.
      STBEZ  = 90.+30.
      STPAC  = 30. + 1.
      STBPCP = (30. - 5.) + 1.
      STBPR  = 60.
      STRRDL = 60.
      STRRDR = 60.
      STSNKA = 71.
      STSNKB = 71.
      STSNKC = 71.
      STBTA  = 261.
      STBTB  = 261.
      STBTC  = 261.
      STMDMA = 261.
      STMDMB = 261.
C
      SRMDMA = 261.
      SRMDMB = 261.
      SRMDMC = 261.
      SRBTA  = 261.
      SRBTB  = 261.
      SRSSPF = 261.
      SRSSSS = 261.
      SRSRA  = 71.
      SRSRB  = 71.
      SRRF0  = 8.
      SRBR   = 60.
      SRBE   = 60.
      SRHBE0 = 60.
      SRHBE1 = 72.
      SRHBE  = 84.
      SRLPU  = 75.
      SRLPU1 = SRLPU/3.
      SRLPU2 = SRLPU/3.
      SRLPU3 = SRLPU/3.
      SRUBE  = 180.
```



```
      SRUBE1 = SRUBE/3.
      SRUBE2 = SRUBE/3.
      SRUBE3 = SRUBE/3.
      SRRND  = 60.
      SRUPB0 = 180.
      SRUPB1 = 217.
      SRUPB  = 254.
      SRHPU  = 75.
      SRHUP  = 180.
      SRFES  = 180.
      SRRF1  = 8.
      SRSFM  = 8.
      SRSNKA = 180.
      SRSNKB = 180.
      SRSNKC = 180.
      SRDABA = 180.
      SRDABB = 180.
C
C
C
C     DEFINE BUFFER ADDRESSES
C
C     BUS1 BUFFERS:
C
C     PUT BUS1 BUFFERS AT TOP END OF BUS1 MEMORY:
C     SET BUFFER BASE ON BUS1 TO HW SIZE OF BUS1
C     ($C000 = 49152. = 48K HWORDS) MINUS APPROX HW SIZE
C     OF BUS1 BUFFERS.
C
      BASE1  = 49152. - 6310.
C NEW 'BITS' BUFFERS
      ATBTA  = BASE1
      ATBTB  = EVEN(ATBTA  + HWSZ*STBTA)
      ATBTC  = EVEN(ATBTB  + HWSZ*STBTB)
      ARBTA  = EVEN(ATBTC  + HWSZ*STBTC)
      ARBTB  = EVEN(ARBTA  + HWSZ*SRBTA)
C
      ATADBA = EVEN(ARBTB  + HWSZ*SRBTB)
      ATADBB = EVEN(ATADBA + HWSZ*STADBA)
      ATADBC = EVEN(ATADBB + HWSZ*STADBB)
      ATSRA  = EVEN(ATADBC + HWSZ*STADBC)
      ATSRB  = EVEN(ATSRA  + HWSZ*STSRA)
      ATSNKA = EVEN(ATSRB  + HWSZ*STSRB)
      ATSNKB = EVEN(ATSNKA + HWSZ*STSNKA)
      ATSNKC = EVEN(ATSNKB + HWSZ*STSNKB)
      ATMDMA = EVEN(ATSNKC + HWSZ*STSNKC)
      ATMDMB = EVEN(ATMDMA + HWSZ*STMDMA)
```

```
C
      ARMDMA = EVEN(ATMDMB + HWSZ*ST4DMB)
      ARMDMB = EVEN(ARMDMA + HWSZ*SR4DMA)
      ARMDMC = EVEN(ARMDMB + HWSZ*SRMDMB)
      ARSSPF = EVEN(ARMDMC + HWSZ*SR4D4C)
      ARSSSS = EVEN(ARSSPF + HWSZ*SRSSPF)
      ARSRA  = EVEN(ARSSSS + HWSZ*SRSSSS)
      ARSRB  = ARSRA  + FWSZ*SRSRA
      ARS4KA = ARSRB  + FWSZ*SRSRB
      ARSNKB = EVEN(ARSNKA + HWSZ*SRSNKA)
      ARSNKC = EVEN(ARSNKB + HWSZ*SRSNKB)
      ARDABA = EVEN(ARSNKC + HWSZ*SRSNKC)
      ARDABB = EVEN(ARDABA + HWSZ*SRDABA)
      TOP1   = EVEN(ARDABB + HWSZ*SRDABB)
C
C     BUS2 BUFFERS:
C
      BASE2  = 0.
      ATBRDL = BASE2
      ATBRDR = ATBRDL + FWSZ*STBRDL
      ATHAMW = ATBRDR + FWSZ*STBRDR
C
      ARRF0  = ATHAMW + FWSZ*STHAMW
      ARBE   = ARRF0  + FWSZ*SRRF0
      ARBR   = ARBE   + FWSZ*SRBE
      ARRND  = ARBR   + FWSZ*SRBR
      ARHUP  = ARRND  + FWSZ*SRRND
      ARFES  = ARHUP
      ARRF1  = ARHUP  + FWSZ*SRHUP
      ARSFM  = ARRF1  + FWSZ*SRRF1
      TOP2   = ARSFM  + FWSZ*SRSFM
C
      ATBPR  = ARBR
C
C     BUS3 BUFFERS
C
      BASE3  = 0.
      ATINF  = BASE3
      ATINF0 = ATINF  + FWSZ*STINF1
      ATINF1 = ATINF  + FWSZ*STINF0
      ATWSR  = ATINF0 + FWSZ*STINF0
      ATWSRZ = ATWSR
      ATRFER = ATWSRZ + FWSZ*STWSRZ
      ATCRF  = ATRFER + FWSZ*STRFER
      ATCRF  = ATCRF
      ATRF   = ATCRF  + HWSZ
      ATSRM  = ATCRF  + FWSZ*STCRF
      ATSRF  = ATSRM
      ATSR   = ATSRF  + FWSZ*STSRF
      ATSRL  = ATSRM  + FWSZ*STSR
      ATACV  = ATSRM  + FWSZ*STSRM
```



```
      ATLPC  = ATACV
      ATLPCR = ATLPC
      ATLPFB = ATACV  + FWSZ*STACV
      ATBE1  = ATLPFB + FWSZ*STLPFB
      ATBE   = ATBE1  + FWSZ*STBE1*.5
      ATBE0  = ATBE1  + FWSZ*STBE1
      ATBE2  = ATBE
      ATPAC  = ATBE2  + FWSZ*STBE2
      ATBPCP = ATPAC  + FWSZ*5.
C
      ARHBE  = ATPAC  + FWSZ*STPAC
      ARHBE0 = ARHBE  + FWSZ*SRHBE1
      ARHBE1 = ARHBE  + FWSZ*SRHBE0
      ARLPU  = ARHBE0 + FWSZ*SRHBE0
      ARLPU3 = ARLPU
      ARLPU2 = ARLPU3 + FWSZ
      ARLPU1 = ARLPU2 + FWSZ
      ARHPU  = ARLPU  + FWSZ*SRLPU
      ARUPB  = ARHPU  + FWSZ*SRHPU
      ARUPB0 = ARUPB  + FWSZ*SRUPB1
      ARUPB1 = ARUPB  + FWSZ*SRUPB0
      ARUBE  = ARUPB0 + FWSZ*SRUPB0
      ARUBE1 = ARUBE
      ARUBE2 = ARUBE1 + FWSZ
      ARUBE3 = ARUBE2 + FWSZ
      TOP3   = ARUBE  + FWSZ*SRUBE
C
C
C
C
C
C     CONFIGURE BUFFERS
C
C     BUS1 BUFFERS:
C
      CALL MPCLB(IBUS1+TSRA, ATSRA, STSRA, RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TSRB, ATSRB, STSRB, RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TSNKA,ATSNKA,STSNKA,FKD,  CNTG,LNG)
      CALL MPCLB(IBUS1+TSNKB,ATSNKB,STSNKB,FKD,  CNTG,LNG)
C
      CALL MPCLB(IBUS1+RSRA, ARSRA, SRSRA, RL,   CNTG,LNG)
      CALL MPCLB(IBUS1+RSRB, ARSRB, SRSPB, RL,   CNTG,LNG)
      CALL MPCLB(IBUS1+RSNKA,ARSNKA,SRSNKA,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+RSNKB,ARSNKB,SRSNKB,RL,   CNTG,SHRT)
C
C     BUS2 BUFFERS:
C
      CALL MPCLB(IBUS2+TBRDR,ATBRDR,STBRDR,RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+THAMW,ATHAMW,STHAMW,RL,   CNTG,LNG)
```

```
C
      CALL MPCLB(IBUS2+RRF0, ARRF0, SRRF0, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RBR,  ARBR,  SRBR,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RBE,  ARBE,  SRBE,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RRND, ARRND, SRRND, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RHUP, ARHUP, SRHUP, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RRF1, ARRF1, SRRF1, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RSFM, ARSFM, SRSFM, RL,   CNTG,LNG)
C
C     BUS3 BUFFERS:
C
      CALL MPCLB(IBUS3+TINF0,ATINF0,STINF0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TINF1,ATINF1,STINF1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TINF, ATINF, STINF, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TWSR, ATWSR, STWSR, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TWSRZ,ATWSRZ,STWSRZ,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TRFER,ATRFER,STRFER,RL,   CNTG,LNG)
C
C     DO DUMMY MPCLB OF TCQRF, SPECIFYING TSCQRF AND FXD,LNG,
C     SO TCRF AND TQRF CAN BE DEFINED USING MPEVN AND MPODD.
C     THEN DO REAL MPCLB OF TCQRF, SPECIFYING STCQRF AND RL,LNG.
C
      CALL MPCLB(IBUS3+TCQRF,ATCQRF,STCQRF*2.,FXD,  CNTG,LNG)
      CALL MPEVN(TCRF,TCQRF)
      CALL MPODD(TQRF,TCQRF)
      CALL MPCLB(IBUS3+TCQRF,ATCQRF,STCQRF,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSR,  ATSR,  STSR,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRM, ATSRM, STSRM, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRF, ATSRF, STSRF, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRL, ATSRL, STSRL, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TACV, ATACV, STACV, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TLPC, ATLPC, STLPC, RL,   CNTG,LNG)
      CALL MPCRB(TLPCR,TLPC)
      CALL MPCLB(IBUS3+TLPFR,ATLPFR,STLPFR,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE0, ATBE0, STBE0, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE1, ATBE1, STBE1, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE,  ATBE,  STBE,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBEZ, ATBEZ, STBEZ, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TPAC, ATPAC, STPAC, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBPCP,ATBPCP,STBPCP,RL,   CNTG,LNG)
C
      CALL MPCLB(IBUS3+RHBE0,ARHBE0,SRHBE0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RHBE1,ARHBE1,SRHBE1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RHBE, ARHBE, SRHBE, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RLPU1,ARLPU1,SRLPU1,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RLPU2,ARLPU2,SRLPU2,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RLPU3,ARLPU3,SRLPU3,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RHPU, ARHPU, SRHPU, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUBE, ARUBE, SRUBE, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUBE1,ARUBE1,SRUBE1,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUBE2,ARUBE2,SRUBE2,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUBE3,ARUBE3,SRUBE3,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUPB0,ARUPB0,SRUPB0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUPB1,ARUPB1,SRUPB1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUPB, ARUPB, SRUPB, RL,   CNTG,LNG)
```



```
C
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP (REAL) SCALARS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP (REAL) SCALARS.
C       ALL MAP (REAL) SCALAR ID'S (SID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER SCALARS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER SCALARS.
C
C
C       DEFINE MAP (REAL) SCALAR ID'S
C
C
C NEGATIVE D.C. VALUE OF CURRENT FRAME
      TDCN  = 50
C
C NEGATIVE INVERSE OF FRAMESIZE
      TFSZI = 51
C
C THRESHOLD FOR MWLF
      TKTHR = 52
C
C PITCH (BETWEEN 5 AND 38, INCLUSIVE)
      TPTC  = 55
C
C ENERGY OF PITCH-REMOVED BASEBAND EXCITATION
      TE    = 57
C
C QUANTIZED TTAP  (TQTAP,TQG,TQGI MUST BE CONSECUTIVE)
      TQTAP = 58
C
C QUANTIZED GAIN (GAIN=SQRT(TE))  (TQTAP,TQG,TQGI MUST BE CONSEC.)
      TQG   = 59
C
C INVERSE OF QUANTIZED GAIN  (TQTAP,TQG,TQGI MUST BE CONSECUTIVE)
      TQGI  = 60
C
C B.B. QUANTIZATION THRESHOLD 1  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBT1  = 61
C
C B.B. QUANT. THRESH. 2  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBT2  = 62
C
C B.B. QUANT. THRESH. 3  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBT3  = 63
C
```

```
C B.B. QUANT. VALUE 0  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBQ0  = 64
C
C B.B. QUANT. VALUE 1  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBQ1  = 65
C
C B.B. QUANT. VALUE 2  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBQ2  = 66
C
C B.B. QUANT. VALUE 3  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
      TBQ3  = 67
C
C DECODED TAP
      RTAP  = 68
C
C COEFF 1 FOR BUTTERWORTH FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
      RBWC1 = 70
C
C COEFF 2 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
      RBWC2 = 71
C
C COEFF 3 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
      RBWC3 = 72
C
C COEFF 4 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
      RBWC4 = 73
C
C MEMORY 1 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
      RBWM1 = 74
C
C MEMORY 2 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
      RBWM2 = 75
C
C MEMORY 3 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
      RBWM3 = 76
C
C MEMORY 4 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
      RBWM4 = 77
C
C FIRST CONSTANT FOR PERTURBATION
      RPC1  = 78
C
C SECOND CONSTANT FOR PERTURBATION
      RPC2  = 79
C
C CODED TAP (INTEGER: 1ST HWORD OF REAL SCALAR)  (TCTAP,TCG,TCRF1-8 CONS
                                        **EC)
      TCTAP = 80
C
C CODED GAIN (INTEGER) (TCTAP,TCG,TCRF1-8 MUST BE CONSEC)
      TCG   = 81
```



```
C
C PREV FRAME CODED REFL COEF(1) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF1 = 82
C
C PREV FRAME CODED REFL COEF(2) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF2 = 83
C
C PREV FRAME CODED REFL COEF(3) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF3 = 84
C
C PREV FRAME CODED REFL COEF(4) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF4 = 85
C
C PREV FRAME CODED REFL COEF(5) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF5 = 86
C
C PREV FRAME CODED REFL COEF(6) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF6 = 87
C
C PREV FRAME CODED REFL COEF(7) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF7 = 88
C
C PREV FRAME CODED REFL COEF(8) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF8 = 89
C
C DECODED PITCH (INTEGER)
      RPTC  = 91
C
C CONSTANT FIVE
      T5    = 92
C
C INVERSE OF DOWNSAMPLED FRAME SIZE
      TDFSI = 94
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP INTEGER SCALARS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP INTEGER SCALARS.
C       ALL MAP INTEGER SCALAR ID'S (ISID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER SCALARS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER SCALARS.
C
C
C
C       DEFINE MAP INTEGER SCALAR ID'S
C
C       (FOR BUFFER STATUS FLAGS:  0=EMPTY, 1=FULL)
```

```
C
C TSOURCE FLAG A
      TSRFA = 50
C
C TSOURCE FLAG B
      TSRFB = 51
C
C TBITS FLAG A
      TBTFA = 52
C
C TBITS FLAG B
      TBTFB = 53
C
C INTEGER PITCH
      TIPTC = 54
C
C RBITS FLAG A
      RBTFA = 57
C
C RBITS FLAG B
      RBTFB = 58
C
C RSINK FLAG A
      RSNFA = 59
C
C RSINK FLAG B
      RSNFB = 60
C
C DECODED (INTEGER) PITCH
      RIPTC = 61
C
C SYSTEM RUN FLAG (0 => STOP)
      RUN   = 62
C
C FREE FOR MPITM
      I     = 63
C
C A/D POINTER OFFSET
      ADPO  = 64
C
C TSOURCE POINTER OFFSET
      TSRPO = 65
C
C TBITS POINTER OFFSET
      TBTPO = 66
C
C TMODEM POINTER OFFSET
      TMPO  = 67
C
C RMODEM POINTER OFFSET
      RMPO  = 68
```



```
C
C RBITS POINTER OFFSET
      RBTPO = 69
C
C RSINK POINTER OFFSET
      RSNPO = 70
C
C D/A POINTER OFFSET
      DAPO  = 71
C
C A/D FRAME DISCARD COUNTER
      TADFDC= 72
C
C TMODEM FAKE FRAME COUNTER
      TMFFC = 73
C
C RMODEM FRAME DISCARD COUNTER
      RMFDC = 74
C
C RSINK DATA-NOT-READY COUNTER
      RSNNR = 75
C
C TRANSMITTER FRAME COUNTER
      TFRCTR= 76
C
C RECEIVER FRAME COUNTER
      RFRCTR= 77
C
C RECEIVER LOST-SYNC COUNTER
      RLSCTR= 78
C
C LOCAL HANDSET ON-HOOK STATE
      RONHK = 79
C
C STATE OF RMODEM SYNC
      RSYNC = 80
C
C BEGINNING-OF-FRAME OFFSET (SYNC BIT POSITION)
      RBOFO = 81
C
C FREE FOR MPITM
      I     = 82
C
C NO-ERROR-CORRECTION SWITCH (NO CORRECTIONS IF NZ)
      RNOCOR= 123
C
C FREE FOR MPITM
      I     = 124
C
C CHANNEL ERROR SIMULATOR FLAG (NON-ZERO => # SIM ERRORS PER FRAME)
      RERSIM= 125
```

```
C
C FREE FOR MPITM
      I     = 126
C
C VOCODER STATE (FOR DISPLAY)
      VSTATE= 127
C
C
C
      CALL SCOPY(' BASE1  = ',NAME)
      TYPE 111,NAME,BASE1
      CALL SCOPY(' ATBTA  = ',NAME)
      TYPE 111,NAME,ATBTA
      CALL SCOPY(' ATBTB  = ',NAME)
      TYPE 111,NAME,ATBTB
      CALL SCOPY(' ATBTC  = ',NAME)
      TYPE 111,NAME,ATBTC
      CALL SCOPY(' ARBTA  = ',NAME)
      TYPE 111,NAME,ARBTA
      CALL SCOPY(' ARBTB  = ',NAME)
      TYPE 111,NAME,ARBTB
      CALL SCOPY(' ATADBA = ',NAME)
      TYPE 111,NAME,ATADBA
      CALL SCOPY(' ATADBB = ',NAME)
      TYPE 111,NAME,ATADBB
      CALL SCOPY(' ATADBC = ',NAME)
      TYPE 111,NAME,ATADBC
      CALL SCOPY(' ATSRA  = ',NAME)
      TYPE 111,NAME,ATSRA
      CALL SCOPY(' ATSRB  = ',NAME)
      TYPE 111,NAME,ATSRB
      CALL SCOPY(' ATSNKA = ',NAME)
      TYPE 111,NAME,ATSNKA
      CALL SCOPY(' ATSNKB = ',NAME)
      TYPE 111,NAME,ATSNKB
      CALL SCOPY(' ATSNKC = ',NAME)
      TYPE 111,NAME,ATSNKC
      CALL SCOPY(' ATMDMA = ',NAME)
      TYPE 111,NAME,ATMDMA
      CALL SCOPY(' ATMDMB = ',NAME)
      TYPE 111,NAME,ATMDMB
      CALL SCOPY(' ARMDMA = ',NAME)
      TYPE 111,NAME,ARMDMA
      CALL SCOPY(' ARMDMB = ',NAME)
      TYPE 111,NAME,ARMDMB
      CALL SCOPY(' ARMDMC = ',NAME)
      TYPE 111,NAME,ARMDMC
      CALL SCOPY(' ARSSPF = ',NAME)
      TYPE 111,NAME,ARSSPF
      CALL SCOPY(' ARSSSS = ',NAME)
      TYPE 111,NAME,ARSSSS
```



```
      CALL SCOPY(' ARSRA  = ',NAME)
      TYPE 111,NAME,ARSRA
      CALL SCOPY(' ARSRB  = ',NAME)
      TYPE 111,NAME,ARSRB
      CALL SCOPY(' ARSNKA = ',NAME)
      TYPE 111,NAME,ARSNKA
      CALL SCOPY(' ARSNKB = ',NAME)
      TYPE 111,NAME,ARSNKB
      CALL SCOPY(' ARSNKC = ',NAME)
      TYPE 111,NAME,ARSNKC
      CALL SCOPY(' ARDABA = ',NAME)
      TYPE 111,NAME,ARDABA
      CALL SCOPY(' ARDABB = ',NAME)
      TYPE 111,NAME,ARDABB
      CALL SCOPY(' TOP1   = ',NAME)
      TYPE 111,NAME,TOP1
C
      CALL SCOPY(' BASE2  = ',NAME)
      TYPE 111,NAME,BASE2
      CALL SCOPY(' ATBRDL = ',NAME)
      TYPE 111,NAME,ATBRDL
      CALL SCOPY(' ATBRDR = ',NAME)
      TYPE 111,NAME,ATBRDR
      CALL SCOPY(' ATHAMW = ',NAME)
      TYPE 111,NAME,ATHAMW
      CALL SCOPY(' ARRF0  = ',NAME)
      TYPE 111,NAME,ARRF0
      CALL SCOPY(' ARBE   = ',NAME)
      TYPE 111,NAME,ARBE
      CALL SCOPY(' ARBR   = ',NAME)
      TYPE 111,NAME,ARBR
      CALL SCOPY(' ARRND  = ',NAME)
      TYPE 111,NAME,ARRND
      CALL SCOPY(' ARHUP  = ',NAME)
      TYPE 111,NAME,ARHUP
      CALL SCOPY(' ARFES  = ',NAME)
      TYPE 111,NAME,ARFES
      CALL SCOPY(' ARRF1  = ',NAME)
      TYPE 111,NAME,ARRF1
      CALL SCOPY(' ARSFM  = ',NAME)
      TYPE 111,NAME,ARSFM
      CALL SCOPY(' TOP2   = ',NAME)
      TYPE 111,NAME,TOP2
C
      CALL SCOPY(' ATBPR  = ',NAME)
      TYPE 111,NAME,ATBPR
```

```
C
      CALL SCOPY(' BASE3  = ',NAME)
      TYPE 111,NAME,BASE3
      CALL SCOPY(' ATINF  = ',NAME)
      TYPE 111,NAME,ATINF
      CALL SCOPY(' ATINF0 = ',NAME)
      TYPE 111,NAME,ATINF0
      CALL SCOPY(' ATINF1 = ',NAME)
      TYPE 111,NAME,ATINF1
      CALL SCOPY(' ATWSR  = ',NAME)
      TYPE 111,NAME,ATWSR
      CALL SCOPY(' ATWSRZ = ',NAME)
      TYPE 111,NAME,ATWSRZ
      CALL SCOPY(' ATRFER = ',NAME)
      TYPE 111,NAME,ATRFER
      CALL SCOPY(' ATCQRF = ',NAME)
      TYPE 111,NAME,ATCQRF
      CALL SCOPY(' ATCRF  = ',NAME)
      TYPE 111,NAME,ATCRF
      CALL SCOPY(' ATQRF  = ',NAME)
      TYPE 111,NAME,ATQRF
      CALL SCOPY(' ATSRM  = ',NAME)
      TYPE 111,NAME,ATSRM
      CALL SCOPY(' ATSRF  = ',NAME)
      TYPE 111,NAME,ATSRF
      CALL SCOPY(' ATSR   = ',NAME)
      TYPE 111,NAME,ATSR
      CALL SCOPY(' ATSRL  = ',NAME)
      TYPE 111,NAME,ATSRL
      CALL SCOPY(' ATACV  = ',NAME)
      TYPE 111,NAME,ATACV
      CALL SCOPY(' ATLPC  = ',NAME)
      TYPE 111,NAME,ATLPC
      CALL SCOPY(' ATLPCR = ',NAME)
      TYPE 111,NAME,ATLPCR
      CALL SCOPY(' ATLPFB = ',NAME)
      TYPE 111,NAME,ATLPFB
      CALL SCOPY(' ATBE   = ',NAME)
      TYPE 111,NAME,ATBE
      CALL SCOPY(' ATBEZ  = ',NAME)
      TYPE 111,NAME,ATBEZ
      CALL SCOPY(' ATBE1  = ',NAME)
      TYPE 111,NAME,ATBE1
      CALL SCOPY(' ATBE0  = ',NAME)
      TYPE 111,NAME,ATBE0
      CALL SCOPY(' ATPAC  = ',NAME)
      TYPE 111,NAME,ATPAC
      CALL SCOPY(' ATBPCP = ',NAME)
      TYPE 111,NAME,ATBPCP
      CALL SCOPY(' ARHBE  = ',NAME)
      TYPE 111,NAME,ARHBE
```



```
      CALL SCOPY(' ARHBE0 = ',NAME)
      TYPE 111,NAME,ARHBE0
      CALL SCOPY(' ARHBE1 = ',NAME)
      TYPE 111,NAME,ARHBE1
      CALL SCOPY(' ARLPU  = ',NAME)
      TYPE 111,NAME,ARLPU
      CALL SCOPY(' ARLPU1 = ',NAME)
      TYPE 111,NAME,ARLPU1
      CALL SCOPY(' ARLPU2 = ',NAME)
      TYPE 111,NAME,ARLPU2
      CALL SCOPY(' ARLPU3 = ',NAME)
      TYPE 111,NAME,ARLPU3
      CALL SCOPY(' ARHPU  = ',NAME)
      TYPE 111,NAME,ARHPU
      CALL SCOPY(' ARUBE  = ',NAME)
      TYPE 111,NAME,ARUBE
      CALL SCOPY(' ARUBE1 = ',NAME)
      TYPE 111,NAME,ARUBE1
      CALL SCOPY(' ARUBE2 = ',NAME)
      TYPE 111,NAME,ARUBE2
      CALL SCOPY(' ARUBE3 = ',NAME)
      TYPE 111,NAME,ARUBE3
      CALL SCOPY(' ARUPB  = ',NAME)
      TYPE 111,NAME,ARUPB
      CALL SCOPY(' ARUPB0 = ',NAME)
      TYPE 111,NAME,ARUPB0
      CALL SCOPY(' ARUPB1 = ',NAME)
      TYPE 111,NAME,ARUPB1
      CALL SCOPY(' TOP3   = ',NAME)
      TYPE 111,NAME,TOP3
C
  111 FORMAT(10A1,F7.0)
C
C
      RETURN
      END
C
C
C
      REAL FUNCTION EVEN(X)
C
C     USAGE:  Y = EVEN(X)
C
C     X IS REAL INPUT
C     Y IS: X   IF X IS EVEN
C           X+1 IF X IS ODD
C
C
      EVEN = X
      IF (AMOD(X,2.) .NE. 0.) EVEN = X+1.
C
      RETURN
      END
```

```
C  <DCA96>DCA96C.FOR.1   Sat 29-Dec-79 7:34PM          Page 1

C [BBN-TENEXD]<KFIELD>DCA96C.FOR.35, 11-Dec-79 17:29:49, Ed: KFIELD
      SUBROUTINE DCA96C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                        DCA96C
C       DCA96 CONFIGURATION - BUFFER AND SCALAR
C       CONFIGURATION.
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       MAP-RELATED VARIABLES
C
      REAL MWSZ,FWSZ,DWSZ
      INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
      INTEGER PRCSR,AP
C
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
C
      INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, THAMW
      INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
      INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
      INTEGER TSRL, TINF0,TINF1,TINF, TLPF0,TRE0
      INTEGER TBE1, TBE,  TBEZ, TWSRZ
      INTEGER TPAC, TBPCP
      INTEGER TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC
      INTEGER TBTA, TBTB, TBTC
      INTEGER TMDMA,TMDMB
      INTEGER RBTA, RBTB
      INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
      INTEGER RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1
      INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
      INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
      INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
      INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C       MAP SCALARS
C
      INTEGER TDCN, TFSZI,TKTHR,TPTC
      INTEGER TE,   TQTAP,TQG,  TQGI, TBT1
      INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
      INTEGER TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4
      INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
      INTEGER TDFSI
      INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
      INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
      INTEGER RPTC
```

```
C  <DCA96>DCA96C.FOR.1   Sat 29-Dec-79 7:34PM          Page 1:1

C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RIPTC
      INTEGER RUN
      INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO
      INTEGER RBTPO,RSNPO,DAPO, TADFDC
      INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
      INTEGER RFRCTR,     RLSCTR
      INTEGER RONHK,RSYNC,RBOFO,RNOCOR
      INTEGER RERSIM,     VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IDS
C
      COMMON /BSIDS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINF0,TINF1,TINF, TLPF0,TRE0,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC,
     ;TBTA, TBTB, TBTC,
     ;TMDMA,TMDMB,
     ;RBTA, RBTB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1,
     ;RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCN, TFSZI,TKTHR,TPTC,
     ;TE,   TQTAP,TQG,  TQGI, TBT1,
     ;TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRF8,T5,
     ;TDFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     ;RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONHK,RSYNC,RBOFO,RNOCOR,
     ;RERSIM,     VSTATE
```

```
C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     ;ATADBC,STADBC,
     ;ATBTC, STBTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBRDL,STBRDL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC


C
C
C
        DATA PRCSR/1/,AP/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
C
C
C
C       OPEN MAP, INHIBIT BUFFER CHECKING
C       SET SNAP ERROR REPORT LEVEL:
C               0 => NO REPORT
C               1 => REPORT CODE; CONTINUE
C               2 => REPORT CODE; HALT
C               3 => REPORT MSG.; CONTINUE
C               4 => REPORT MSG.; HALT
C               5 => REPORT CODE; STATUS; HALT
C               6 => REPORT MSG.; STATUS; HALT
C
        IERPRT = 3
        CALL MPOPN(IERPRT)
        CALL MPIBC(0)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP BUFFERS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP BUFFERS.
C       ALL MAP BUFFER ID'S (BID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER BUFFERS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER BUFFERS.
```



```
C       AN "A" FOLLOWED BY A BUFFER NAME SPECIFIES
C       THE HOST VARIABLE CONTAINING THE ADDRESS OF
C       THAT BUFFER.  SIMILARLY, AN "S" FOLLOWED BY A
C       BUFFER NAME SPECIFIES THE HOST VARIABLE
C       CONTAINING THE SIZE OF THAT BUFFER.
C
C
C
C
C       DEFINE MAP BUFFER ID'S
C
        IBUS1 = 64
        IBUS2 = 128
        IBUS3 = 192
C
C
C A/D INPUT FROM ADAM
        TADBA = 12
C
C A/D INPUT FROM ADAM
        TADBB = 31
C
C A/D 'TONE' DATA
        TADBC = 0
C
C TSOURCE A BUFFER
        TSRA = 1
C
C TSOURCE B BUFFER
        TSRB = 2
C
C HAMMING WINDOW COEFFS.
        THAMW = 3
C
C WINDOWED TSRA OR TSRB
        TWSR = 4
C
C TWSR WITH 8 ZEROS ON RIGHT
        TWSRZ = 6
C
C AUTOCORRELATION VALUES
        TACV = 7
C
C OUTPUT FROM MWLF, INCLUDING REFLECTION COEFFS AND ERRORS
        TRFER = 8
C
C OUTPUT FROM MWLF: CODED & Q'TIZED RF COEFFS (TCRF & TQRF)
        TCQRF = 9
C
C CODED REFLECTION COEFFS
        TCRF = 10
```

```
C
C Q'TIZED REFLECTION COEFFS
      TQRF = 11
C
C LINEAR PREDICTION COEFFS
      TLPC = 7
C
C REVERSE OF TLPC
      TLPCR = 5
C
C (NOTE: TADBA = 12)
C
C CURRENT TSOURCEX DATA IN LONG REAL FORMAT
      TSR = 13
C
C TSR WITH 8 MEMORY SAMPLES IN FRONT
      TSRM = 14
C
C FIRST 8 SAMPLES OF TSRM
      TSRF = 15
C
C LAST 8 SAMPLES OF TSRM
      TSRL = 16
C
C INVERSE FILTERED SAMPLES (CURRENT FRAME)
      TINF0 = 17
C
C TINF0 PLUS 37 PREVIOUS FRAME ELEMENTS IN FRONT
      TINF1 = 18
C
C 254 INVERSE FILTERED SAMPLES CENTERED ON PREV FRAME
      TINF = 19
C
C LPF COEFFS TO GET BASEBAND EXCITATION
      TLPFB = 20
C
C DOWNSAMPLED BASEBAND (BB) EXCITATION
      TBE0 = 21
C
C PREVIOUS FRAME'S TBE0
      TBE1 = 22
C
C LAST HALF OF TBE1, ALL OF TBE0
      TBE  = 23
C
C TBE PLUS 30 ZEROS
      TBEZ = 24
C
C (NOTE: RMDMC = 25; RDABA = 26; RDABB = 27)
C
C PITCH AUTOCORRELATION BUFFER
      TPAC = 28
```



```
C
C BB EXCITATION PITCH CALC PART (TPAC(5) - TPAC(30))
      TBPCP = 29
C
C BB EXCITATION WITH PITCH REMOVED
      TBPR = 40
C
C BB RESIDUAL DIFFERENCE BUFFER (LEFT HALF)
      TBRDL = 0
C
C
C (NOTE: RMDMA = 30; TADBB = 31; RMDMB = 32)
C BB RESIDUAL DIFFERENCE BUFFER (RIGHT HALF)
      TBRDR = 33
C
C (NOTE: TMDMB = 34)
C
C TSINK A BUFFER
      TSNKA = 35
C
C TSINK B BUFFER
      TSNKB = 36
C
C TSINK 'SILENCE' BUFFER (NO LONGER USED)
      TSNKC = 0
C
C TBITS A BUFFER
      TBTA = 0
C
C TBITS B BUFFER
      TBTB = 0
C
C TBITS C BUFFER
      TBTC = 0
C
C TMODEM A BUFFER
      TMDMA = 42
C
C TMODEM B BUFFER
      TMDMB = 34
C
C RMODEM A BUFFER
      RMDMA = 30
C
C RMODEM B BUFFER
      RMDMB = 32
C
C RMODEM C BUFFER
      RMDMC = 25
C
C RBITS A BUFFER
      RBTA = 0
```

```
C
C RBITS B BUFFER
      RBTB = 0
C
C SYNC SEARCH PREVIOUS FRAME
      RSSPF = 0
C
C SYNC SEARCH SUM SYNC
      RSSSS = 0
C
C RSOURCE A BUFFER
      RSRA = 37
C
C RSOURCE B BUFFER
      RSRB = 38
C
C CURRENT FRAME REFLECTION COEFFS
      RRF0 = 39
C
C BB RESIDUAL
      RBR  = 40
C
C BB EXCITATION
      RRE  = 41
C
C (NOTE: TMDMA = 42)
C
C HPF'D BB EXCITATION
      RHBE0 = 43
C
C CURRENT FRAME HPF'D BB EXCITATION, PLUS 12 PREV FRAME ELEMENTS
      RHBE1 = 44
C
C 84 HPF'D BB EXCITATION SAMPLES CENTERED ON PREV FRAME
      RHBE  = 45
C
C LPF COEFFS (WITH GAIN OF 3) TO GET UPSAMPLED BB EXCITATION
      RLPU = 0
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH 3RD
      RLPU1 = 46
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH SECOND
      RLPU2 = 47
C
C EVERY 3RD ELEMENT OF RLPU STARTING WITH 1ST
      RLPU3 = 48
C
C UPSAMPLED, LPF'D BB EXCITATION
      RUBE = 49
C
```



```
C EVERY 3RD ELEMENT OF RUBE STARTING WITH FIRST
      RUBE1 = 50
C
C EVERY 3RD ELEMENT OF RUBE STARTING WITH SECOND
      RUBE2 = 51
C
C EVERY 3RD ELEMENT OF RUBE STARTING WITH THIRD
      RUBE3 = 52
C
C RANDOM NUMBER ARRAY
      RRND = 53
C
C CURRENT FRAME OF UPSAMPLED AND PERTURBED BB SAMPLES
      RUPB0 = 54
C
C RUPB0 PLUS 37 PREV FRAME ELEMENTS IN FRONT
      RUPB1 = 55
C
C 254 UPSAMPLED AND PERTUBED BB SAMPLES CENTERED ON PREV FRAME
      RUPB = 56
C
C HPF COEFFS (WITH GAIN OF 3) TO GET UPSAMPLED, PERTURBED SAMPLES
      RHPU = 57
C
C HPF'D, UPSAMPLED, PERTURBED SAMPLES
      RHUP = 58
C
C FILTERED EXCITATION SAMPLES
      RFES = 58
C
C PREV FRAME'S RRF0 (BID #39)
      RRF1 = 59
C
C SYNTHESIS FILTER MEMORY
      RSFM = 60
C
C SYNTHESIZED SPEECH
      RSNKA = 61
C
C SYNTHESIZED SPEECH
      RSNKB = 62
C
C D/A 'SILENCE' BUFFER
      RSNKC = 0
C
C D/A OUTPUT TO AOM
      RDABA = 26
C
C D/A OUTPUT TO AOM
      RDABB = 27
C
```

```
C
C
C
C       DEFINE BUFFER SIZES (NUMBER OF SAMPLES PER BUFFER)
C
        STADBA = 100.
        STADBB = 100.
        STADBC = 100.
        STSRA  = 100.
        STSRB  = 100.
        STHAMW = 100.
        STWSR  = 100.
        STWSRZ = 100.
        STACV  = 9.
        STRFER = 16.
        STCQRF = 8.
        STCPF  = 8.
        STQRF  = 9.
        STLPC  = 9.
        STLPCR = 9.
        STSR   = 100.
        STSRM  = 199.
        STSRF  = 8.
        STSRL  = 8.
        STINF0 = 100.
        STINF1 = 217.
        STINF  = 254.
        STLPF3 = 75.
        STRE0  = 60.
        STRE1  = 60.
        STBF   = 90.
        STBFZ  = 90.+30.
        STP4C  = 30. + 1.
        STAPCP = (39. - 5.) + 1.
        STBPR  = 60.
        STBRDL = 60.
        STBRDR = 60.
        STSNKA = 71.
        STSNKB = 71.
        STSNKC = 71.
        STBTA  = 261.
        STBTB  = 261.
        STBTC  = 261.
        STMDMA = 261.
        STMDMB = 261.
```



```
C
        SRMDMA = 261.
        SRMDMB = 261.
        SRMDMC = 261.
        SRBTA  = 261.
        SRBTB  = 261.
        SRSSPF = 261.
        SRSSSS = 261.
        SRSRA  = 71.
        SRSRB  = 71.
        SRRF0  = 8.
        SRBR   = 60.
        SRBF   = 60.
        SRHBE0 = 60.
        SRHBE1 = 72.
        SRHBE  = 94.
        SRLPU  = 75.
        SRLPU1 = SRLPU/3.
        SRLPU2 = SRLPU/3.
        SRLPU3 = SRLPU/3.
        SRUBE  = 100.
        SRUBE1 = SRUBE/3.
        SRUBE2 = SRUBE/3.
        SRUBE3 = SRUBE/3.
        SRRND  = 60.
        SRUPB0 = 100.
        SRUPB1 = 217.
        SRUPB  = 254.
        SRHPU  = 75.
        SRHUP  = 100.
        SRFES  = 100.
        SRRF1  = 8.
        SRSFM  = 8.
        SRSNKA = 100.
        SRSNKB = 100.
        SRSNKC = 100.
        SRDABA = 100.
        SRDABB = 100.
C
C
C
C
C       DEFINE BUFFER ADDRESSES
C
C       BUS1 BUFFERS:
C
C       PUT BUS1 BUFFERS AT TOP END OF BUS1 MEMORY:
C       SET BUFFER BASE ON BUS1 TO HW SIZE OF BUS1
C       ($C800 = 49152. = 48K HWORDS) MINUS APPROX HW SIZE
C       OF BUS1 BUFFERS.
C
        BASE1  = 49152. - 6310.
```

```
C MEM "BITS" BUFFERS
      ATBTA  = BASE1
      ATBTB  = EVEN(ATBTA  + HWSZ*STBTA)
      ATBTC  = EVEN(ATBTB  + HWSZ*STBTB)
      ARBTA  = EVEN(ATBTC  + HWSZ*STBTC)
      ARBTB  = EVEN(ARBTA  + HWSZ*SRBTA)
C
      ATADBA = EVEN(ARBTB  + HWSZ*SRBTB)
      ATADBB = EVEN(ATADBA + HWSZ*STADBA)
      ATADBC = EVEN(ATADBB + HWSZ*STADBB)
      ATSRA  = EVEN(ATADBC + HWSZ*STADBC)
      ATSRB  = EVEN(ATSRA  + HWSZ*STSRA)
      ATSNKA = EVEN(ATSRB  + HWSZ*STSRB)
      ATSNKB = EVEN(ATSNKA + HWSZ*STSNKA)
      ATSNKC = EVEN(ATSNKB + HWSZ*STSNKB)
      ATMDMA = EVEN(ATSNKC + HWSZ*STSNKC)
      ATMDMB = EVEN(ATMDMA + HWSZ*STMDMA)
C
      ARMDMA = EVEN(ATMDMB + HWSZ*STMDMB)
      ARMDMB = EVEN(ARMDMA + HWSZ*SRMDMA)
      ARMDMC = EVEN(ARMDMB + HWSZ*SRMDMB)
      ARSSPF = EVEN(ARMDMC + HWSZ*SRMDMC)
      ARSSSS = EVEN(ARSSPF + HWSZ*SRSSPF)
      ARSRA  = EVEN(ARSSSS + HWSZ*SRSSSS)
      ARSRB  = ARSRA  + FWSZ*SRSRA
      ARSNKA = ARSRB  + FWSZ*SRSRB
      ARSNKB = EVEN(ARSNKA + HWSZ*SRSNKA)
      ARSNKC = EVEN(ARSNKB + HWSZ*SRSNKB)
      ARDABA = EVEN(ARSNKC + HWSZ*SRSNKC)
      ARDABB = EVEN(ARDABA + HWSZ*SRDABA)
      TOP1   = EVEN(ARDABB + HWSZ*SRDABB)
C
C     BUS2 BUFFERS:
C
      BASE2  = 0.
      ATBRDL = BASE2
      ATBRDR = ATBRDL + FWSZ*STBRDL
      ATHAMW = ATBRDR + FWSZ*STBRDR
C
      ARRF0  = ATHAMW + FWSZ*STHAMW
      ARBE   = ARRF0  + FWSZ*SRRF0
      ARBR   = ARBE   + FWSZ*SRBE
      ARRND  = ARBR   + FWSZ*SRBR
      ARHUP  = ARRND  + FWSZ*SRRND
      ARFES  = ARHUP
      ARRF1  = ARHUP  + FWSZ*SRHUP
      ARSFM  = ARRF1  + FWSZ*SRRF1
      TOP2   = ARSFM  + FWSZ*SRSFM
C
      ATBPR  = ARBR
C
```



```
C     BUS3 BUFFERS
C
      BASE3  = 0.
      ATINF  = BASE3
      ATINF0 = ATINF  + FWSZ*STINF1
      ATINF1 = ATINF  + FWSZ*STINF0
      ATWSR  = ATINF0 + FWSZ*STINF0
      ATWSRZ = ATWSR
      ATRFER = ATWSRZ + FWSZ*STWSRZ
      ATCQRF = ATRFER + FWSZ*STRFER
      ATCRF  = ATCQRF
      ATQRF  = ATCQRF + HWSZ
      ATSRM  = ATCQRF + FWSZ*STCQRF
      ATSRF  = ATSRM
      ATSR   = ATSRF  + FWSZ*STSRF
      ATSRL  = ATSRM  + FWSZ*STSR
      ATACV  = ATSRM  + FWSZ*STSRM
      ATLPC  = ATACV
      ATLPCR = ATLPC
      ATLPFB = ATACV  + FWSZ*STACV
      ATBE1  = ATLPFB + FWSZ*STLPFB
      ATBE   = ATBE1  + FWSZ*STBE1*.5
      ATBE0  = ATBE1  + FWSZ*STBE1
      ATBEZ  = ATBE
      ATPAC  = ATBEZ  + FWSZ*STBEZ
      ATBPCP = ATPAC  + FWSZ*5.
C
      ARHBE  = ATPAC  + FWSZ*STPAC
      ARHBE0 = ARHBE  + FWSZ*SRHBE1
      ARHBE1 = ARHBE  + FWSZ*SRHBE0
      ARLPU  = ARHBE0 + FWSZ*SRHBE0
      ARLPU3 = ARLPU
      ARLPU2 = ARLPU3 + FWSZ
      ARLPU1 = ARLPU2 + FWSZ
      ARHPU  = ARLPU  + FWSZ*SRLPU
      ARUPB  = ARHPU  + FWSZ*SRHPU
      ARUPB0 = ARUPB  + FWSZ*SRUPB1
      ARUPB1 = ARUPB  + FWSZ*SRUPB0
      ARUBE  = ARUPB0 + FWSZ*SRUPB0
      ARUBE1 = ARUBE
      ARUBE2 = ARUBE1 + FWSZ
      ARUBE3 = ARUBE2 + FWSZ
      TOP3   = ARUBE  + FWSZ*SRUBE
C
C
C
C
C
C     CONFIGURE BUFFERS
C
C     BUS1 BUFFERS:
```

```
C
      CALL MPCLB(IBUS1+TADBA,ATADBA,STADBA,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TADBB,ATADBB,STADBB,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TSRA, ATSRA, STSRA, RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TSRB, ATSRB, STSRB, RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+TSNKA,ATSNKA,STSNKA,FXD,  CNTG,LNG)
      CALL MPCLB(IBUS1+TSNKB,ATSNKB,STSNKB,FXD,  CNTG,LNG)
      CALL MPCLB(IBUS1+TMDMA,ATMDMA,STMDMA,FXD,   CNTG,LNG)
      CALL MPCLB(IBUS1+TMDMB,ATMDMB,STMDMB,FXD,   CNTG,LNG)
C
      CALL MPCLB(IBUS1+RMDMA,ARMDMA,SRMDMA,FXD,   CNTG,LNG)
      CALL MPCLB(IBUS1+RMDMB,ARMDMB,SRMDMB,FXD,   CNTG,LNG)
      CALL MPCLB(IBUS1+RMDMC,ARMDMC,SRMDMC,FXD,   CNTG,LNG)
      CALL MPCLB(IBUS1+RSRA, ARSRA, SRSRA, RL,   CNTG,LNG)
      CALL MPCLB(IBUS1+RSRB, ARSRB, SRSRB, RL,   CNTG,LNG)
      CALL MPCLB(IBUS1+RSNKA,ARSNKA,SRSNKA,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+RSNKB,ARSNKB,SRSNKB,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+RDABA,ARDABA,SRDABA,RL,   CNTG,SHRT)
      CALL MPCLB(IBUS1+RDABB,ARDABB,SRDABB,RL,   CNTG,SHRT)
C
C     BUS2 BUFFERS:
C
      CALL MPCLB(IBUS2+TBRDR,ATBRDR,STBRDR,RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+THAMW,ATHAMW,STHAMW,RL,   CNTG,LNG)
C
      CALL MPCLB(IBUS2+RRF0, ARRF0, SRRF0, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RBR,  ARBR,  SRBR,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RBE,  ARBE,  SRBE,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RRND, ARRND, SRRND, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RHUP, ARHUP, SRHUP, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RRF1, ARRF1, SRRF1, RL,   CNTG,LNG)
      CALL MPCLB(IBUS2+RSFM, ARSFM, SRSFM, RL,   CNTG,LNG)
C
C     BUS3 BUFFERS:
C
      CALL MPCLB(IBUS3+TINF0,ATINF0,STINF0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TINF1,ATINF1,STINF1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TINF, ATINF, STINF, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TWSR, ATWSR, STWSR, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TWSRZ,ATWSRZ,STWSRZ,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TRFER,ATRFER,STRFER,RL,   CNTG,LNG)
C
C     DO DUMMY MPCLB OF TCQRF, SPECIFYING STCQRF*2 AND FXD,LNG,
C     SO TCRF AND TQRF CAN BE DEFINED USING MPEVN AND MPODD.
C     THEN DO REAL MPCLB OF TCQRF, SPECIFYING STCQRF AND RL,LNG.
```



```
C
      CALL MPCLB(IBUS3+TCQRF,ATCQRF,STCQRF*2.,FXD,  CNTG,LNG)
      CALL MPEVN(TCRF,TCQRF)
      CALL MPODD(TQRF,TCQRF)
      CALL MPCLB(IBUS3+TCQRF,ATCQRF,STCQRF,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSR,  ATSR,  STSR,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRM, ATSRM, STSRM, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRF, ATSRF, STSRF, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TSRL, ATSRL, STSRL, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TACV, ATACV, STACV, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TLPC, ATLPC, STLPC, RL,   CNTG,LNG)
      CALL MPCRB(TLPCR,TLPC)
      CALL MPCLB(IBUS3+TLPFB,ATLPFB,STLPFB,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE0, ATBE0, STBE0, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE1, ATBE1, STBE1, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBE,  ATBE,  STBE,  RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBEZ, ATBEZ, STBEZ, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TPAC, ATPAC, STPAC, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+TBPCP,ATBPCP,STBPCP,RL,   CNTG,LNG)
C
      CALL MPCLB(IBUS3+RHBE0,ARHBE0,SRHBE0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RHBE1,ARHBE1,SRHBE1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RHBE, ARHBE, SRHBE, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RLPU1,ARLPU1,SRLPU1,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RLPU2,ARLPU2,SRLPU2,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RLPU3,ARLPU3,SRLPU3,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RHPU, ARHPU, SRHPU, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUBE, ARUBE, SRUBE, RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUBE1,ARUBE1,SRUBE1,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUBE2,ARUBE2,SRUBE2,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUBE3,ARUBE3,SRUBE3,RL,   3,   LNG)
      CALL MPCLB(IBUS3+RUPB0,ARUPB0,SRUPB0,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUPB1,ARUPB1,SRUPB1,RL,   CNTG,LNG)
      CALL MPCLB(IBUS3+RUPB, ARUPB, SRUPB, RL,   CNTG,LNG)
C
C
C
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP (REAL) SCALARS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP (REAL) SCALARS.
C       ALL MAP (REAL) SCALAR ID'S (SID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER SCALARS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER SCALARS.
C
C
```

```
C       DEFINE MAP (REAL) SCALAR ID'S
C
C
C NEGATIVE D.C. VALUE OF CURRENT FRAME
        TDCN  = 50
C
C NEGATIVE INVERSE OF FRAMESIZE
        TFSZI = 51
C
C THRESHOLD FOR MWLF
        TKTHR = 52
C
C PITCH (BETWEEN 5 AND 30, INCLUSIVE)
        TPTC  = 55
C
C ENERGY OF PITCH-REMOVED BASEBAND EXCITATION
        TE    = 57
C
C QUANTIZED TTAP  (TQTAP,TQG,TQGI MUST BE CONSECUTIVE)
        TQTAP = 58
C
C QUANTIZED GAIN (GAIN=SQRT(TE))  (TQTAP,TQG,TQGI MUST BE CONSEC.)
        TQG   = 59
C
C INVERSE OF QUANTIZED GAIN  (TQTAP,TQG,TQGI MUST BE CONSECUTIVE)
        TQGI  = 60
C
C B.B. QUANTIZATION THRESHOLD 1  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBT1  = 61
C
C B.B. QUANT. THRESH. 2  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBT2  = 62
C
C B.B. QUANT. THRESH. 3  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBT3  = 63
C
C B.B. QUANT. VALUE 0  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBQ0  = 64
C
C B.B. QUANT. VALUE 1  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBQ1  = 65
C
C B.B. QUANT. VALUE 2  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBQ2  = 66
C
C B.B. QUANT. VALUE 3  (TBT1,2,3,TBQ0,1,2,3 MUST BE CONSEC.)
        TBQ3  = 67
C
C DECODED TAP
        RTAP  = 68
C
```



```
C COEFF 1 FOR BUTTERWORTH FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
        RBWC1 = 70
C
C COEFF 2 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
        RBWC2 = 71
C
C COEFF 3 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
        RBWC3 = 72
C
C COEFF 4 FOR B.W. FILTER  (RBWC1,2,3,4 MUST BE CONSEC.)
        RBWC4 = 73
C
C MEMORY 1 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
        RBWM1 = 74
C
C MEMORY 2 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
        RBWM2 = 75
C
C MEMORY 3 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
        RBWM3 = 76
C
C MEMORY 4 FOR B.W. FILTER  (RBWM1,2,3,4 MUST BE CONSEC.)
        RBWM4 = 77
C
C FIRST CONSTANT FOR PERTURBATION
        RPC1  = 78
C
C SECOND CONSTANT FOR PERTURBATION
        RPC2  = 79
C
C CODED TAP (INTEGER: 1ST HWORD OF REAL SCALAR)  (TCTAP,TCG,TCRF1-8 CONS
                                                  **EC)
        TCTAP = 80
C
C CODED GAIN (INTEGER) (TCTAP,TCG,TCRF1-8 MUST BE CONSEC)
        TCG   = 81
C
C PREV FRAME CODED REFL COEF(1) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
        TCRF1 = 82
C
C PREV FRAME CODED REFL COEF(2) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
        TCRF2 = 83
C
C PREV FRAME CODED REFL COEF(3) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
        TCRF3 = 84
C
C PREV FRAME CODED REFL COEF(4) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
        TCRF4 = 85
C
C PREV FRAME CODED REFL COEF(5) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
        TCRF5 = 86
```

```
C
C PREV FRAME CODED REFL COEF(6) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF6 = 87
C
C PREV FRAME CODED REFL COEF(7) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF7 = 88
C
C PREV FRAME CODED REFL COEF(8) (INTEGER) (TCTAP,TCG,TCRF1-8 CONSEC)
      TCRF8 = 89
C
C DECODED PITCH (INTEGER)
      RPTC  = 91
C
C CONSTANT FIVE
      T5    = 92
C
C INVERSE OF DOWNSAMPLED FRAME SIZE
      TDFSI = 94
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CONFIGURE MAP INTEGER SCALARS.
C
C       THIS MODULE CONFIGURES THE SYSTEM MAP INTEGER SCALARS.
C       ALL MAP INTEGER SCALAR ID'S (ISID'S) ARE
C       SYMBOLICALLY NAMED, WITH NAMES STARTING WITH
C       "T" FOR TRANSMITTER SCALARS, AND NAMES STARTING
C       WITH "R" FOR RECEIVER SCALARS.
C
C
C
C       DEFINE MAP INTEGER SCALAR ID'S
C
C       (FOR BUFFER STATUS FLAGS:  0=EMPTY, 1=FULL)
C
C TSOURCE FLAG A
      TSRFA = 50
C
C TSOURCE FLAG B
      TSRFB = 51
C
C TBITS FLAG A
      TBTFA = 52
C
C TBITS FLAG B
      TBTFB = 53
C
C INTEGER PITCH
      TIPTC = 54
```



```
C
C RBITS FLAG A
      RBTFA = 57
C
C RBITS FLAG B
      RBTFB = 58
C
C RSINK FLAG A
      RSNFA = 59
C
C RSINK FLAG B
      RSNFB = 60
C
C DECODED (INTEGER) PITCH
      RIPTC = 61
C
C SYSTEM RUN FLAG (0 => STOP)
      RUN   = 62
C
C FREE FOR MPITM
      I     = 63
C
C A/D POINTER OFFSET
      ADPO  = 64
C
C TSOURCE POINTER OFFSET
      TSRPO = 65
C
C TBITS POINTER OFFSET
      TBTPO = 66
C
C TMODEM POINTER OFFSET
      TMPO  = 67
C
C RMODEM POINTER OFFSET
      RMPO  = 68
C
C RBITS POINTER OFFSET
      RBTPO = 69
C
C RSINK POINTER OFFSET
      RSNPO = 70
C
C D/A POINTER OFFSET
      DAPO  = 71
C
C A/D FRAME DISCARD COUNTER
      TADFDC= 72
C
C TMODEM FAKE FRAME COUNTER
      TMFFC = 73
```

```
C
C RMODEM FRAME DISCARD COUNTER
      RMFDC = 74
C
C RSINK DATA-NOT-READY COUNTER
      RSNNR = 75
C
C TRANSMITTER FRAME COUNTER
      TFRCTR= 76
C
C RECEIVER FRAME COUNTER
      RFRCTR= 77
C
C RECEIVER LOST-SYNC COUNTER
      RLSCTR= 78
C
C LOCAL HANDSET ON-HOOK STATE
      RONHK = 79
C
C STATE OF RMODEM SYNC
      RSYNC = 80
C
C BEGINNING-OF-FRAME OFFSET (SYNC BIT POSITION)
      RBOFO = 81
C
C FREE FOR MPITM
      I     = 82
C
C NO-ERROR-CORRECTION SWITCH (NO CORRECTIONS IF NZ)
      RNOCOR= 123
C
C FREE FOR MPITM
      I     = 124
C
C CHANNEL ERROR SIMULATOR FLAG (NON-ZERO => # SIM ERRORS PER FRAME)
      RERSIM= 125
C
C FREE FOR MPITM
      I     = 126
C
C VOCODER STATE (FOR DISPLAY)
      VSTATE= 127
C
C
C
      RETURN
      END
C
C
C
      REAL FUNCTION EVEN(X)
```



```
C
C     USAGE:  Y = EVEN(X)
C
C     X IS REAL INPUT
C     Y IS: X    IF X IS EVEN
C           X+1 IF X IS ODD
C
      EVEN = X
      IF (AMOD(X,2.) .NE. 0.) EVEN = X+1.
C
      RETURN
      END
```

```
C  <DCA96>DCA96D.FOR.1   Sat 29-Dec-79 7:37PM          Page 1

C [88R-TENEX01<KFIELD>DCA96D.FOR.1, 12-Dec-79 19:21:52, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DCA96D - DUMMY ROUTINES FOR UNSUPPORTED
C                FILE-TO-FILE AND TIMING MODES OF
C                OPERATION.
C
C       INCLUDES DUMMY ROUTINES FOR:
C                IDELT(I,J)
C                IRDCLK(I)
C                SETCLK(I)
C                SCOPY(IARRAY,JARRAY)
C                CNFIRM(IARRAY)
C                KOPE(IARRAY,J,K)
C                KSIN(IARRAY,JARRAY,K,L)
C                KOPN(IARRAY,J,K,LOG)
C                KSOUT(IARRAY,JARRAY,K,L)
C                KCLOSE(IARRAY)
C
C KDF 12/79
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        INTEGER FUNCTION IDELT(I,J)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION IRDCLK(I)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        SUBROUTINE SETCLK(I)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        SUBROUTINE SCOPY(IARRAY,JARRAY)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
```

```
C  <DCA96>DCA96D.FOR.1   Sat 29-Dec-79 7:37PM          Page 1:1

C
C
C
        LOGICAL FUNCTION CNFIRM(IARRAY)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KOPE(IARRAY,J,K)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KSIN(IARRAY,JARRAY,K,L)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KOPN(IARRAY,J,K,LOG)
        DIMENSION IARRAY(1)
        LOGICAL LOG
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KSOUT(IARRAY,JARRAY,K,L)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KCLOSE(IARRAY)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
```

```
C [BBN-TENEXD]<KFIELD>DCA96E.FOR.74, 31-Dec-79 14:04:55, Ed: KFIELD
      SUBROUTINE DCA96E
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       DCA96E
C               DCA96 EXECUTION
C
C       EXECUTION MODE ENTERED BY USER:
C       1) REAL TIME SYSTEM (DCART)
C       2) FILE TO FILE SYSTEM (DCAFF)
C       3) TIMING SYSTEM (DCATM)
C       4) SCOPE TIMING SYSTEM (DCASP)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       HOST ARRAYS FOR FILE TO FILE TEST SYSTEM
C
        INTEGER XTSRA(264),XRSNKA(264)
        INTEGER HDRBLK(256),SKPHDR(20)
        INTEGER ISIG(100),IST(82)
        REAL SIG(100)
        EQUIVALENCE (HDRBLK(1),SKPHDR(1),SIG(1),IST(1))
        LOGICAL CNFIRM,LTSNK,LRIST
C
C       MAP FUNCTION LIST NAMES
C
        INTEGER DCARTS,DCALP,DCAFFT,DCATIM,DRSTRT
        INTEGER ANLZA,ANLZB,SYNZA,SYNZB
C
C       OTHER MAP-RELATED VARIABLES
C
        INTEGER WMOVN
        REAL HWSZ,FWSZ,DWSZ
        INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
        INTEGER PRCSR,AP,ME
C
        LOGICAL*1 CHRC,CHRE,CHRL,CHRR,CHRS,CHRT,CHRQ,CHRQM,ICMD
        REAL MDMPRG,ADMPRG,AOMPRG
        INTEGER MDMSCL,ADAM,AOM,IOS2
C
C
C
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
```



```
C
        INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, THAMW
        INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
        INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
        INTEGER TSRL, TINF0,TINF1,TINF, TLPFB,TBE0
        INTEGER TBE1, TBE,  TBEZ, TWSRZ
        INTEGER TPAC, TBPCP
        INTEGER TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC
        INTEGER TBTA, TBTB, TBTC
        INTEGER TMDMA,TMDMB
        INTEGER RBTA, RBTB
        INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
        INTEGER RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1
        INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
        INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
        INTEGER RUPB, RHPU, RHUP, RFSS, RRF1, RSPM
        INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C       MAP SCALARS
C
        INTEGER TDCM, TFSZI,TKTMR,TPTC
        INTEGER TE,   TQTAP,TQC,  TQCI, TBT1
        INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
        INTEGER TCTAP,TCC,  TCRF1,TCRF2,TCRF3,TCRF4
        INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
        INTEGER TDFSI
        INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
        INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
        INTEGER RPTC
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
        INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RIPTC
        INTEGER RUN
        INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO
        INTEGER RBTPO,RSNPO,DAPO, TADFOC
        INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
        INTEGER RFRCTR,      RLSCTR
        INTEGER RONHR,RSYNC,RBOFO,RNOCOR
        INTEGER RERSIM,      VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IOS
C
        COMMON /BSIOS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINF0,TINF1,TINF, TLPFB,TBE0,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC,
```

```
     ;TBTA, TBTB, TBTC,
     ;TMDMA,TMDMB,
     ;BATA, BATB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSPB, RRFF, RAR,  RBE,  RHBE0,RHBE1,
     ;RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RAND, RUPB0,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCM, TFSZI,TKTMR,TPTC,
     ;TE,   TQTAP,TQG,  TQGI, TBT1,
     ;TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRF8,T5,
     ;TDFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     ;RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADPDC,     TMFPC,RMPDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONHK,RSYNC,RBOFO,RNOCOR,
     ;RERSIN,     VSTATE
C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     ;ATADBC,STADBC,
     ;ATBTC, STBTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBRDL,STBRDL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC
C
C       COMMON BLOCK - FUNCTION LIST NAMES
C
        COMMON /FLS/
     ;DCARTS,DCALP,DCAFFT,DCATIM,DRSTRT,
     ;ANLZA,ANLZB,SYNZA,SYNZB
C
```



```
C
C
        DATA PROSR/1/,AP/1/,NE/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
C
        DATA CHRC,CHRE,CHRL,CHRR,CHRS,CHRT/"C","E","L","R","S","T"/
        DATA CHRQ,CHRQM/"Q","?"/
C  SCROLL IDENTIFIERS
        DATA MDMSCL,ADAM,AOM,IOS2/16,23,22,2/
C  BUS 1 ADDRESSES OF SCROLL PROGRAMS
        DATA MDMPRG,ADMPRG,AOMPRG/18432.,18592.,18656./
C
C
C
        IBUS1 =  64
        IBUS2 = 128
        IBUS3 = 192
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C!!!!!!!! FORCE MODE TO INDICATE "REAL TIME SYSTEM" !!!!!!!!
C (OTHER MODES NOT FULLY SUPPORTED IN FINAL VOCODER SYSTEM.)
C
        MODE = 1
        IF (MODE .EQ. 1) GOTO 100
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       GET EXECUTION MODE FROM USER...
C
   9    TYPE 10
  10    FORMAT(" EXECUTION MODES:"/
     ;"   1 => REAL TIME SYSTEM"/
     ;"   2 => FILE TO FILE SYSTEM"/
     ;"   3 => TIMING SYSTEM"/
     ;"   4 => SCOPE TIMING SYSTEM"/
     ;" ENTER EXECUTION MODE: "$)
        ACCEPT 11,MODE
  11    FORMAT(I3)
C
        IF ((MODE .LT. 1) .OR. (MODE .GT. 4)) GOTO 9
C
C       EXECUTE CODE FOR INDICATED MODE (DO MODE CODE)
C
        IF (MODE .EQ. 1) GOTO 100
        IF (MODE .EQ. 2) GOTO 200
        IF (MODE .EQ. 3) GOTO 300
        IF (MODE .EQ. 4) GOTO 400
```

```
C
C
C       ALL MODES RETURN HERE
C
 900    CALL MPCLS(0)
C
        RETURN
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE DCA96 REAL TIME SYSTEM
C
 100    CONTINUE
C
C  CONFIGURE A BUFFER OVER EACH SCROLL PROGRAM AND LOAD IT
C
        CALL MPCLB(IBUS1+63,MDMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(MDMSCL,IOS2,63)
        CALL MPCLB(IBUS1+63,ADMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(ADAM,IOS2,63)
        CALL MPCLB(IBUS1+63,AOMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(AOM,IOS2,63)
C
C
C
C  START THE REAL TIME SYSTEM
C
        CALL MPXFL(DCARTS)
C
        TYPE 108
108     FORMAT(' VOCODER IS IN OPERATION.'/)
110     TYPE 111
111     FORMAT(' COMMANDS ARE:'/
        1 '    S:   SUSPEND TASK (LEAVING VOCODER RUNNING)'/
        1 '    Q:   QUIT (HALT VOCODER)'/
        2 '    E N: SIMULATE N ERRORS PER FRAME'/
        3 '    C N: ERROR-CORRECT IF N=0'/
        4 '    L:   CAUSE RCVR TO LOSE SYNC'/
        5 '    T:   TYPE OUT VOCODER STATE')
112     TYPE 113
113     FORMAT(/' ENTER COMMAND: '$)
        ACCEPT 114,ICMD,N
114     FORMAT(A1,I)
        IF(ICMD.EQ.CHRQM) GO TO 110
        IF(ICMD.EQ.CHRQ) GO TO 120
        IF(ICMD.EQ.CHRE) GO TO 130
        IF(ICMD.EQ.CHRC) GO TO 140
        IF(ICMD.EQ.CHRL) GO TO 150
        IF(ICMD.EQ.CHRT) GO TO 160
        IF(ICMD.EQ.CHRS) GO TO 170
115     TYPE 116
116     FORMAT(' BAD COMMAND!')
        GO TO 110
```

```
C  "Q" COMMAND -- QUIT (HALT VOCODER)
120     CALL MPITM(RUN,0,0)
        CALL MPSAA(ADAM)
        CALL MPSAA(AOM)
C  (THERE'S NO WAY TO STOP THE MODEM SCROLL!)
C
C       RETURN TO COMMON CODE
        GO TO 900
C
C "E" COMMAND -- SET ERROR SIMULATION
130     IF(N.LT.0 .OR. N.GT.10) GO TO 115
        CALL MPITM(RERSIN,N,0)
        GO TO 112
C
C  "C" COMMAND -- CONTROL ERROR CORRECTION
140     IF(N.NE.0) N=1
        CALL MPITM(RNOCOR,N,0)
        GO TO 112
C
C  "L" COMMAND -- CAUSE THE RCVR TO LOSE SYNC BY JAMMING INTO RBOFO
150     CALL MPITM(RSYNC,1,50)
        GO TO 112
C
C "T" COMMAND -- STOP VOCODER, READ INTEGER SCALARS, THEN RESTART,
C    THEN TYPE OUT THEIR VALUES
160     CALL MPITM(RUN,0,0)
        CALL MPIRS(TSPFA,IST(TSRFA),32,0)
        CALL MPXFL(DRSTRT)
        CALL TIST(IST)
        GO TO 112

C  "S" COMMAND -- SUSPEND THIS TASK, BUT LET MAP CONTINUE.
C  (UPON RESUMPTION OF TASK, IT WILL CONTINUE OUT OF THE PAUSE)
170     PAUSE '(VOCODER CONTINUING)'
        GO TO 110
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE FILE TO FILE SYSTEM
C
 200    CONTINUE
C
C
C
C       DEFINE INPUT AND OUTPUT FILES
C
 203    TYPE 210
 210    FORMAT(' READ INPUT SIGNAL FROM FILE: '$)
        NBLKS = KOPE(XTSRA,0,0)
        IF(NBLKS.LE.0) GO TO 203
C
        CALL SCOPY('SKIP HEADER BLOCK (256 WORDS)? ',SKPHDR)
        IF (.NOT. CNFIRM(SKPHDR)) GOTO 215
        ISTAT = KSIN(XTSRA,HDRBLK,256,NWXFRD)
C
 215    TYPE 220
 220    FORMAT(' WRITE OUTPUT SIGNAL TO FILE: '$)
        NBLKS = KOPN(XRSNKA,0,0,.TRUE.)
        IF(NBLKS.LE.0) GO TO 215
C
C       CONFIGURE DUMMY MAP BUFFER (RL,LNG) FOR MPWDB'S
C       !!!!!NOTE: THIS BUFFER RESIDES ON BUS 2 AT 15000-15359
C            (I.E. THE TOP OF BUS 2 MEMORY).!!!!!
C
        IDUM = 63
        CALL MPCLB(IBUS2+IDUM,15000.,100.,RL,CNTG,LNG)
C
        IEOF = -1
        NFRAME=0
        NCLIPS=0
        CALL MPIST(RUN,1)
C  ASK ABOUT FRAME TYPEOUTS
        LTSNK=.FALSE.
        LRIST=.FALSE.
        TYPE 226
226     FORMAT(' FRAME TYPEOUTS? (T, R, OR NIL): '$)
        ACCEPT 227,ICMD
227     FORMAT(A1)
        IF(ICMD.EQ.CHRR) LRIST=.TRUE.
        IF(ICMD.NE.CHRT) GO TO 230
        LTSNK=.TRUE.
        TYPE 228
228     FORMAT(' FILE FOR TSINK DATA: '$)
        CALL ASSIGN(2,XTSRA,-1)
        WRITE(2,229)
229     FORMAT(/'  FR#   P  T  G K1 K2 K3 K4 K5 K6 K7 K8 BASEBAND'/)
C
```

```
C  TOP OF LOOP
C
 230    ISTAT = KSIN(XTSRA,ISIG,180,NWXFRD)
        IF (ISTAT .EQ. IEOF) GOTO 290
        DO 235 I=1,180
 235    SIG(I) = FLOAT(ISIG(I)) / 32768.
C
C  SET UP FOR PROPER A/D AND D/A BUFFERS
        IADBUF=TADBA
        IDABUF=RDABA
        ITSNK=TSNKA
        IF(MOD(NFRAME,2).EQ.0) GO TO 240
        IADBUF=TADBB
        IDABUF=RDABB
        ITSNK=TSNKB
C
C       XFER SIG TO MAP
C
240     CALL MPWDB(IDUM,SIG,4,1,SIG(180))
        CALL VMOVN(IADBUF,IDUM)
C
C
C       EXECUTE 1 FRAME OF SYSTEM
C
        CALL MPXFL(DCAFFT)
C
C
C  NOW MOVE BITS FROM TMODEM BUFFER TO RMODEM BUFFER, AS A CHANNEL
C    DOES.
        GO TO(241,242,243,244,245,246) 1+MOD(NFRAME,6)

241     CALL VMOV(RMDMB,TMDMA)
        GO TO 248

242     CALL VMOV(RMDMC,TMDMB)
        GO TO 248

243     CALL VMOV(RMDMA,TMDMA)
        GO TO 248

244     CALL VMOV(RMDMB,TMDMB)
        GO TO 248

245     CALL VMOV(RMDMC,TMDMA)
        GO TO 248
```



```
246     CALL VMOV(RMDMA,TMDMB)
        GO TO 248
C
C       READ OUTPUT (RSNKA) AND WRITE TO FILE
C
C       RECONFIGURE DUMMY BUFFER 'IDUM'
C
248     CALL MPCLB(IBUS2+IDUM,15000.,180.,RL,CNTG,LNG)
C
        CALL VMOVN(IDUM,IDABUF)
        CALL MPRDB(IDUM,SIG,4,1,SIG(180))
C
        DO 250 I=1,180
          XSIG = SIG(I) * 32768.
          IF (ABS(XSIG) .LE. 32767.) GO TO 250
          NCLIPS=NCLIPS+1
          XSIG=SIGN(32767.,XSIG)
 250      ISIG(I) = IFIX(XSIG)
C
C       WRITE OUT INTEGER OUTPUT
C
        ISTAT = KSOUT(XRSNKA,ISIG,180,NWXFRD)
C
C       LOOP.
C
C  IF REQUESTED, DO FRAME TYPEOUT
        IF(.NOT.LTSNK) GO TO 260
        CALL MPRDB(ITSNK,ISIG,2,0,ISIG(71))
        WRITE(2,255) NFRAME,(ISIG(I), I=1,71)
255     FORMAT(I5,1X,11I3,I2,29I1,/,40X,30I1)
260     IF(LRIST) CALL RIST
        NFRAME=NFRAME+1
        GOTO 230
C
C       HERE ON EOF.
C
 290    CALL RIST
        TYPE 295,NFRAME,NCLIPS
295     FORMAT(I7,' FRAMES,',I5,' OUTPUT SAMPLES WERE CLIPPED.')
        ISTAT = KCLOSE(XTSRA)
        ISTAT = KCLOSE(XRSNKA)
        IF(LTSNK) CALL CLOSE(2)
C
C
C       RETURN TO COMMON CODE
C
        GOTO 900
^L
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE TIMING SYSTEM
C
 300    CONTINUE
C
C       GET TIMING PARAMETERS.
C
        TYPE 310
 310    FORMAT(' ENTER ITERATION VALUE (N/6): '$)
        ACCEPT 311,N
 311    FORMAT(I3)
C
C
C  PUT SOMETHING REASONABLE IN TADBA, TADBB
C
        CALL VCOST(TADBA,17.)
        CALL VCOST(TADBB,17.)
C
C
C
C
C       SET UP TIMING (100 USEC TICKS)
C       READ CLOCK
C
        CALL SETCLK(1)
        IT1 = IRDCLK(0)
C
C       EXECUTE DCAFFT '6N' TIMES
C
        CALL MPIST(RUN,1)
        CALL MPFOR(1,1,N,DCATIM)
C
C       WAIT ON MAP, READ CLOCK
C
        CALL MPRST(1,HONE,1,1)
        IT2 = IRDCLK(0)
        TIME = FLOAT(IDELT(IT1,IT2))
C
C       CONVERT TIME, PRINT IT
C
        TIMSMS = TIME * .1
        TYPE 320,6*N,TIMSMS
 320    FORMAT(I6,' ITERATION(S) TOOK',F10.4,' MSECS.')
C
        CALL RIST
C
C       RETURN TO COMMON CODE
C
        GOTO 900
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE SCOPE TIMING SYSTEM
C
 400    CONTINUE
C
C
C  PUT SOMETHING REASONABLE IN TADBA,TADBB
C
        CALL VCOST(TADBA,17.)
        CALL VCOST(TADBB,17.)
C
C
C       START UP INFINITE MPIWL ON DCATIM
C
        CALL MPIST(RUN,1)
        CALL MPIWL(RUN,NE,0,DCATIM)
C
C       WAIT FOR USER TO SAY QUIT
C
        TYPE 410
 410    FORMAT(' TYPE ANY CHARACTER TO HALT MAP: '$)
        ACCEPT 411,ICHAR
 411    FORMAT(A1)
C
C       STOP MPIWL
C
        CALL MPITM(RUN,0,0)
        CALL RIST
C
C       RETURN TO COMMON CODE
C
        GOTO 900
C
C
C
C
C
C
C
C
C
C
C
C
C
        END
^L
```

```
C
      SUBROUTINE RIST
C  READ IST AND TYPE OUT CONTENTS
      DIMENSION IST(82)
C
      CALL MPIRS(1,IST(1),82,0)
      CALL TIST(IST)
C
      RETURN
      END
C
      SUBROUTINE TIST(IST)
C     TYPE OUT SELECTED VALUES FROM IST ARRAY
C
C
C [ABN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C     MAP BUFFER NAMES
C
      INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, THAMW
      INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
      INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
      INTEGER TSRL, TINF0,TINF1,TINF, TLPF0,TBE0
      INTEGER TBE1, TBE,  TBEZ, TWSRZ
      INTEGER TPAC, T0PCP
      INTEGER T0PR, T0ROL,T0ROR,TSNKA,TSNKB,TSNKC
      INTEGER T0TA, T0TB, T0TC
      INTEGER TMDMA,TMDMB
      INTEGER R0TA, R0TB
      INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
      INTEGER RSRB, RRF0, RRR,  RBE,  RHBE0,RHBE1
      INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
      INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
      INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
      INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C     MAP SCALARS
C
      INTEGER TDCN, TFSZI,TKTHR,TPTC
      INTEGER TE,   TQTAP,TQC,  TQCI, TBT1
      INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
      INTEGER TCTAP,TCC,  TCRF1,TCRF2,TCRF3,TCRF4
      INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
      INTEGER TDFSI
      INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
      INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
      INTEGER RPTC
```



```
C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RIPTC
      INTEGER RUN
      INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO
      INTEGER RBTPO,RSNPO,DAPO, TADFDC
      INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
      INTEGER RFRCTR,     RLSCTR
      INTEGER RONNK,RSYNC,R0OFO,RNOCOR
      INTEGER RERSIM,     VSTATE
C
C     COMMON BLOCK - BUFFER AND SCALAR IOS
C
      COMMON /BSIOS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINF0,TINF1,TINF, TLPF0,TBE0,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, T0PCP,
     ;T0PR, T0RDL,T0RDR,TSNKA,TSNKB,TSNKC,
     ;T0TA, T0TB, T0TC,
     ;TMDMA,TMDMB,
     ;R0TA, R0TB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSRB, RRF0, RRR,  RBE,  RHBE0,RHBE1,
     ;RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCN, TFSZI,TKTHR,TPTC,
     ;TE,   TQTAP,TQC,  TQCI, TBT1,
     ;TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCC,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRF8,T5,
     ;TDFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     ;RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONNK,RSYNC,R0OFO,RNOCOR,
     ;RERSIM,     VSTATE
```

```
C  <DCA96>DCA96E.FOR.1   Mon 31-Dec-79 2:07PM         Page 6:2

C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /RAS/
     ;ATADBC,STADRC,
     ;ATRTC, STRTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBRDL,STBRDL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCAC04.FOR CCCCCCCC
C
        DIMENSION IST(1)
C
        TYPE 10
        TYPE 20,IST(TSRFA),IST(ADPO), IST(TFRCTR),IST(RSYNC)
        TYPE 30,IST(TSRFB),IST(TSRPO),IST(RFRCTR),IST(RBOFO)
        TYPE 40,IST(TBTFA),IST(TBTPO),            IST(RLSCTR)
        TYPE 50,IST(TBTFB),IST(TMPO), IST(TADFDC)
        TYPE 60,IST(RBTFA),IST(RMPO), IST(TMFFC), IST(RONHK)
        TYPE 70,IST(RBTFB),IST(RBTPO),IST(RMFDC)
        TYPE 80,IST(RSNFA),IST(RSNPO),IST(RSNNR)
        TYPE 90,IST(RSNFB),IST(DAPO)
C
 10     FORMAT(' VOCODER STATE VARIABLES:',/)
 20     FORMAT(' TSRFA:',I2,'   ADPO: ',I3,'    TFRCTR:',I6,'     RSYNC:
     ;I6)
                              ** ',
 30     FORMAT(' TSRFB:',I2,'   TSRPO:',I3,'    RFRCTR:',I6,'     RBOFO:
     ;I6)
                              ** ',
 40     FORMAT(' TBTFA:',I2,'   TBTPO:',I3,'           ',6X,'     RLSCTR
     ;I6)
                              **:',
 50     FORMAT(' TBTFB:',I2,'   TMPO: ',I3,'    TADFDC:',I6)
 60     FORMAT(' RBTFA:',I2,'   RMPO: ',I3,'    TMFFC: ',I6,'     RONHK:
     ;I6)
                              ** ',
 70     FORMAT(' RBTFB:',I2,'   RBTPO:',I3,'    RMFDC: ',I6)
 80     FORMAT(' RSNFA:',I2,'   RSNPO:',I3,'    RSNNR: ',I6)
 90     FORMAT(' RSNFB:',I2,'   DAPO: ',I3)
C
        RETURN
        END
```

```
C  <DCA96>DCA96F.FOR.1   Sat 29-Dec-79 7:36PM          Page 1

C [BBN-TENEXD]<KFIELD>DCA96F.FOR.22, 17-Dec-79 19:23:35, Ed: KFIELD
      SUBROUTINE DCA96F




CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                        DCA96F
C       DCA96 FUNCTION PREBINDING AND FUNCTION LIST
C               DEFINITIONS.
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       MAP FUNCTION NAMES
C
      INTEGER SSUM, VMUL, VMOV, DCVM
      INTEGER VKTOA,VLTSV
      INTEGER SADD
      INTEGER VSMA2,DFL22,PRTRB
      INTEGER VMOVM,VFLT
      INTEGER DEAL, DECOD,VAPC, VIAPC
      INTEGER ENRG, PTAP
C
C
C       MAP FUNCTION LIST NAMES
C
      INTEGER DCARTS,DC4LP,DCAFFT,DCATIM,ORSTRT
      INTEGER AMLZA,AMLZB,SYNZA,SYNZB
C
C
C       OTHER MAP-RELATED VARIABLES
C
      REAL HWSZ,FWSZ,DWSZ
      INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
      INTEGER PRCSR,AP,ME
      INTEGER A,B,IG3,ISET,ICLR
      INTEGER MDMSCL,ADAM,AOM,IOS2,MDMSA,ADAMSA,AOMSA
C
C
C       MAP PRE-BINDING BUFFER
C
      INTEGER PRBUF
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
```

```
C  <DCA96>DCA96F.FOR.1   Sat 29-Dec-79 7:36PM          Page 1:1

C
      INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, THAMW
      INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
      INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
      INTEGER TSRL, TINF0,TINF1,TIMF, TLPFB,TBE0
      INTEGER TBE1, TBE,  TBEZ, TWSRZ
      INTEGER TPAC, TBPCP
      INTEGER TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC
      INTEGER TBTA, TBTB, TBTC
      INTEGER TMDMA,TMDMB
      INTEGER RBTA, RBTB
      INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
      INTEGER RSRB, RRF0, RBR,  RBE,  RHBE0,RHBE1
      INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
      INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
      INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
      INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDARB
C
C
C       MAP SCALARS
C
      INTEGER TDCN, TFSZI,TKTHR,TPTC
      INTEGER TE,   TQTAP,TQG,  TQGI, TBT1
      INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
      INTEGER TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4
      INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
      INTEGER TDFSI
      INTEGER RTAP, RRWC1,RBWC2,RBWC3,RBWC4
      INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
      INTEGER RPTC
C
      INTEGER TSRFA,TSRFB,TBIFA,TBIFB,TIPTC
      INTEGER RBIFA,RBIFB,RSNFA,RSNFB,RIPTC
      INTEGER RUN
      INTEGER ADPO, TSRPO,TBIPO,TMPO, RMPO
      INTEGER RBIPO,RSNPO,DAPO, TADFDC
      INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
      INTEGER RFRCTR,     RLSCTR
      INTEGER RONHK,RSYNC,RBOFO,RNUCOR
      INTEGER RERSIN,     VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IDS
C
      COMMON /BSIDS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINF0,TINF1,TIMF, TLPFB,TBE0,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC,
```

```
     $TBTA, TRTR, TBTC,
     $TMDMA,TMDMB,
     $RRT', RRTR,
     $RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     $RSR8, RRF8, RRR,  RBE,  RHBE8,RHBE1,
     $RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     $RUBE1,RUBE2,RUBE3,RRND, RUPB8,RUPB1,
     $RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     $RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     $TDCN, TFSZI,TRTHR,TPTC,
     $TE,   TQTAP,TQG,  TQGI, TBT1,
     $TBT2, TRT3, TBQ8, TBQ1, TBQ2, TBQ3,
     $TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     $TCRF5,TCRF6,TCRF7,TCRF8,T5,
     $TDFSI,
     $RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     $RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     $RPTC,
     $TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     $RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     $RUN,
     $ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     $RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     $RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     $RONHK,RSYNC,RBOFO,RNOCOR,
     $RERSTM,     #STATE

C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     $ATADBC,STADBC,
     $ATBTC, STBTC,
     $ARLPU, SRLPU,
     $ARSNKC,SRSNKC,
     $ATBRDL,STBRDL,
     $ATMDMA,STMDMA,
     $ATMDMB,STMDMB,
     $ARDABA,SRDABA,
     $ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC
C
C       COMMON BLOCK
C       FUNCTION LIST NAMES
C
        COMMON /FLS/
     $DCARTS,DCALP,DCAFFT,DCATIM,DRSTRT,
     $ANLZA,ANLZB,SYNZA,SYNZB
```



```
C
C
C
C
C
        DATA PRCSR/1/,AP/1/,NE/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA RWSZ/1./,FWSZ/2./,DWSZ/4./
C  I/O SCROLL IDENTIFIERS
        DATA A,B/-2,0/,IG3,ISET,ICLR/3,1,0/
C  SCROLL PROGRAM STARTING ADDRESSES
        DATA MDMSCL,ADAM,AOM,IOS2/16,23,22,2/
        DATA MDMSA,ADAMSA,AOMSA/1,0,0/
C
C
        IBUS1 =  64
        IBUS2 = 128
^L
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DEFINE FUNCTION LIST NAMES
C
C
        ANLZA =  1
        ANLZB =  2
        SYNZA =  3
        SYNZB =  4
C       REAL TIME SYSTEM STARTUP:
        DCARTS = 10
C       REAL TIME SYSTEM LOOP:
        DCALP = 11
C       FILE-TO-FILE SYSTEM:
        DCAFFT = 12
C       TIMING SYSTEM
        DCATIM = 13
C       RESTARTING RT SYSTEM
        DRSTRT = 14
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DEFINE FUNCTION LISTS
C
C       THE FOLLOWING ARRAY AND NON-ARRAY FUNCTIONS ARE
C       ARN-WRITTEN:
C       MPIFF(IA,IB,FLNAME): IF ISA .NE. 0  .AND. ISB = 0 ,
C                            EXECUTE FLNAME.
C       MWLF(Y,A,U,V): MWLD WITH FIXED POINT OUTPUT
C                      (SEE DETAILED DOCUMENTATION)
C       VKTOA(V,U): CONVERT REFLECTION COEFFS TO LINEAR PREDICTOR COEFFS
C                      (SEE DETAILED DOCUMENTATION)
C       PRTRB(Y,A,U,B,V): Y=PERTURBED AND UPSAMPLED OUTPUT
C                         SA=PRTRB CONSTANT #1
C                         U=INPUT ARRAY
C                         SB=PRTRB CONSTANT #2
C                         V=RANDOM NUMBER ARRAY (INPUT)
C                         (SEE DETAILED DOCUMENTATION)
C       VLTSY(Y,U,V,W): VECTOR LATTICE SYNTHESIS FILTER
C                       Y = FILTERED OUTPUT SAMPLES
C                       U = FILTER MEMORY
C                       V = FILTER COEFFICIENTS
C                       W = INPUT SAMPLES
C                       (SEE DETAILED DOCUMENTATION)
C       DEAL(Y,A,U,B,V): SEE DETAILED DOCUMENTATION
C       DECOD(V,U): SEE DETAILED DOCUMENTATION
C       VAPC(Y,A,U,B,V,C,D): SEE DETAILED DOCUMENTATION
C       VIAPC(Y,A,U,B): SEE DETAILED DOCUMENTATION
C       MPXBM(FCBNO,Y,A,U,V): EXECUTE BOUND VERSION OF MWLF.
C                     SAME AS MPXBF, BUT PUTS PARAMS TO MWLF
```



```
C                     (Y,Q,U,V) IN FCB ENTRY AS THEY'D BE
C                     IN UN-PRE-BOUND MWLF FCB.
C       PTAP(Y,A,U,B,V,C,W): COMPUTE PITCH & TAP AND
C                            DO PITCH REMOVAL.
C               Y = OUTPUT: B.B. WITH PTICH REMOVED
C               A = OUTPUT: PITCH
C               U = INPUT:  B.B. AUTOCORRELATION (PITCH CALC. PART)
C               B = OUTPUT: QUANTIZED TAP
C               V = INPUT:  DOWNSAMPLED B.B. EXCITATION
C               C = OUTPUT: CODED TAP (INTEGER - R JUST. IN L RWORD)
C               W = INPUT:  B.B. AUTOCORR. (WHOLE THING)
C       ENRG(A,B,C,W,D): COMPUTE,QUANTIZE & CODE ENERGY(GAIN)
C               A = OUTPUT: QUANTIZED GAIN
C               B = OUTPUT: INVERSE OF QUANT. GAIN
C               C = OUTPUT: CODED GAIN
C               W = INPUT:  B.B. WITH PITCH REMOVED
C               D = INPUT:  INVERSE OF DNSMPLD FRAMESIZE
C       DCOR(Y,U,V): DISCRETE CORRELATION
C               Y = OUTPUT
C               U = INPUT: KERNAL
C               V = INPUT
C       PROTCT(AB): PROTECT & BITSTREAM XMTR FRAME
C       CORECT(AB): UNBITSTREAM AND CORRECT RCVR FRAME
C               AB = -2 FOR "A", 0 FOR "B" BUFFERS
C       MPGSC(G,SC): SET/CLEAR G-FLAG
C               G = G-FLAG NUMBER
C               SC = 0 FOR CLEAR, 1 FOR SET
C       MPMBS(Y,SA,NS): MOVE BUFFER TO SCALAR
C               SAME AS MPTBS, BUT DOESN'T ADVANCE 'Y'
C               BUFFER ADDRESS.
C       IADINT, ITMINT, IRMINT, IDAINT: RUN AN INTERRUPT ROUTINE
C
C       ALL OTHER FUNCTIONS USED BELOW ARE PART OF THE
C       CSPI SNAP-II EXECUTIVE.
C
C
C       DO PRE-BINDING OF ARRAY FUNCTIONS
C
C       DEFINE PRE-BINDING BUFFER AT $8000 = 32768. ON BUS1
C                        WITH SIZE   $2000 =  8192.
C
        PBBUF =    63
       APBBUF = 32768.
       SPBBUF =  8192.
C
       CALL MPCLB(IBUS1+PBBUF,APBBUF,SPBBUF,FXD,CNTG,LNG)
C
C       PUT PRE-BOUND FUNCTIONS IN FCBS #214 - 252
```

```
C
      CALL MPCBF(PBRUF,214)
C
C     #214
            CALL SSUM(TDCN,TSRA,TFSZI)
C     #215
            CALL VMUL(TWSR,1,TSRA,TDCN,THAMW,0)
C     #216
            CALL VMOV(TSRF,TSRL)
C     #217
            CALL VMOV(TSR,TSRA)
C     #218
            CALL SSUM(TDCN,TSRB,TFSZI)
C     #219
            CALL VMUL(TWSR,1,TSRB,TDCN,THAMW,0)
C     #220
            CALL VMOV(TSR,TSRB)
C     #221
            CALL DCOR(TACV,TWSR,TWSRZ)
C     #222
            CALL VWLF(TRFER,TKTHR,TCQRF,TACV)
C     #223
            CALL VKTOA(TLPC,TQRF)
C     #224
            CALL VMOV(TINF,TINF1)
C     #225
            CALL DCOR(TINF0,TLPCR,TSRM)
C     #226
            CALL VMOV(TBE1,TBE0)
C     #227
            CALL DCVM(TBE0,3,TINF,TLPFB)
C     #228
            CALL DCOR(TPAC,TBE,TBEZ)
C     #229  (DUMMY)
            CALL VMOV(RMDMB,TMDMA)
C     #230
            CALL PTAP(TBPR,TPTC,TPPCP,TQTAP,TBE0,TCTAP,TPAC)
C     #231  (DUMMY)
            CALL VMOV(RMDMC,TMDMB)
C     #232  (DUMMY)
            CALL VMOV(RMDMA,TMDMA)
C     #233  (DUMMY)
            CALL VMOV(RMDMB,TMDMB)
C     #234  (DUMMY)
            CALL VMOV(RMDMC,TMDMA)
C     #235
            CALL ENRG(TQG,TQGI,TCG,TBPR,TDFSI)
C     #236
            CALL VAPC(TSNKA,TPTC,TBRDR,TQTAP,TBPR,TCTAP,TBT1)
C     #237
            CALL VAPC(TSNKB,TPTC,TBRDR,TQTAP,TBPR,TCTAP,TBT1)
```



```
C     #238
            CALL DEAL(RBR,RPTC,RRF0,RTAP,RSRA)
C     #239
            CALL VIAPC(RBR,RTAP,RBR,RPTC)
C     #240
            CALL DEAL(RBR,RPTC,RRF0,RTAP,RSRB)
C     #241
            CALL VMOV(RHBE,RHBE1)
C     #242
            CALL DFL22(RHBE0,RBWC1,RBE,RBWM1)
C     #243
            CALL DCOR(RUBE1,RLPU1,RHBE)
C     #244
            CALL DCOR(RUBE2,RLPU2,RHBE)
C     #245
            CALL DCOR(RUBE3,RLPU3,RHBE)
C     #246
            CALL VMOV(RUPB,RUPB1)
C     #247
            CALL PRTRB(RUPB0,RPC1,RHBE0,RPC2,RRND)
C     #248
            CALL DCOR(RHUP,RHPU,RUPB)
C     #249
            CALL VSMA2(RFES,1,RUBE,1,RHUP,9)
C     #250
            CALL VLTSY(RSNKA,RSFM,RRF1,RFES)
C     #251
            CALL VMOV(RRF1,RRF0)
C     #252
            CALL VLTSY(RSNKB,RSFM,RRF1,RFES)
C     #253
            CALL VMOV(RMDMA,TMDMB)
C
      CALL MPTBF(0)
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DCARTS: F.L. FOR REAL TIME SYSTEM STARTUP:
C
      CALL MPBFL(DCARTS)
            CALL MPRNS(MDMSCL,IOS2,MDMSA)
            CALL MPRAA(ADAM,ADAMSA,AOM,AOMSA)
            CALL MPIST(RUN,1)
            CALL MPIWL(RUN,NE,0,DCALP)
      CALL MPEFL(DCARTS)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
C
C       DCALP: F.L. FOR REAL TIME SYSTEM LOOP:
C
        CALL MPBFL(DCALP)
                CALL MPIFF(TSRFA,TBTFB,ANLZA)
                CALL MPIFF(RBTFB,RSNFA,SYNZA)
                CALL MPIFF(TSRFB,TBTFA,ANLZB)
                CALL MPIFF(RBTFA,RSNFB,SYNZB)
        CALL MPEFL(DCALP)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DRSTRT: F.L. FOR RESTARTING REAL TIME SYSTEM:
C
        CALL MPBFL(DRSTRT)
                CALL MPIST(RUN,1)
                CALL MPIWL(RUN,NE,0,DCALP)
        CALL MPEFL(DRSTRT)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DCAFFT: F.L. FOR FILE-TO-FILE SYSTEM AND FOR TIMING:
C
        CALL MPBFL(DCAFFT)
                CALL MPWT(PRCSR,AP)
                CALL IADINT
                CALL IRMINT
                CALL MPIFF(TSRFA,TBTFB,ANLZA)
                CALL MPIFF(RBTFB,RSNFA,SYNZA)
                CALL MPIFF(TSRFB,TBTFA,ANLZB)
                CALL MPIFF(RBTFA,RSNFB,SYNZB)
                CALL MPWT(PRCSR,AP)
                CALL IDAINT
                CALL ITMINT
        CALL MPEFL(DCAFFT)
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       DCATIM: F.L. FOR TIMING AND SCOPE TIMING SYSTEMS
C
        CALL MPBFL(DCATIM)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(229)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(231)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(232)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(233)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(234)
                CALL MPXFL(DCAFFT)
                CALL MPXBF(253)
        CALL MPEFL(DCATIM)
^L
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C       ANLZA: F.L. FOR ANALYSIS USING "A" BUFFERS:
C
        CALL MPRFL(ANLZA)
C
C FOR SCOPE SYNC...
                CALL MPGSC(IG3,ISET)
C LPC...
C
C
C               USE TSRA; REMOVE DC; MULT BY HAMMING WINDOW:
C
C-              CALL SSUM(TDCN,TSRA,TFSZI)
                CALL MPXBF(214)
C-              CALL VMUL(TWSR,1,TSRA,TDCN,THAMW,0)
                CALL MPXBF(215)
C-              CALL VMOV(TSRF,TSRL)
                CALL MPXBF(216)
C-              CALL VMOV(TSR,TSRA)
                CALL MPXBF(217)
C
C
C               AUTOCORRELATE TWSR:
C
C-              CALL DCOR(TACV,TWSR,TWSRZ)
                CALL MPXBF(221)
C
C               FINISHED WITH TSRA
                CALL MPIST(TSRFA,0)
C
C               DELAY REFLECTION COEFFICIENTS:
C               PUT LAST FRAME CODED&QUANTIZED REFL COEFFS (TCQRF)
C               INTO MEMORY (SCALARS TCRF1-TCRF8)
C
                CALL MPMBS(TCQRF,TCRF1,8)
C
C               COMPUTE REFLECTION COEFFS, CODE & QUANTIZE THEM:
C
C-              CALL MWLF(      TRFER,TKTHR,TCQRF,TACV)
                CALL MPXBM(222,TRFER,TKTHR,TCQRF,TACV)
C
C               COMPUTE LP COEFFS (TLPC):
C
C-              CALL VRTOA(TLPC,TQRF)
                CALL MPXBF(223)
C
C               COMPUTE INVERSE FILTERED SAMPLES AFTER
C               MOVING LAST FRAME MEMORY INTO PLACE:
```



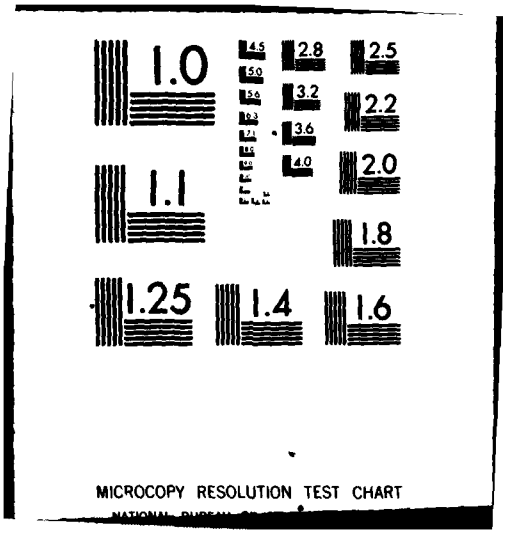
```
C
C-              CALL VMOV(TINF,TINF1)
                CALL MPXBF(224)
C-              CALL DCOR(TINF0,TLPCR,TSRM)
                CALL MPXBF(225)
C
C BBEXT...
C
C
C               SAVE LAST FRAME DNSMPLD B.B. EXCITATION;
C               COMPUTE CURRENT FRAME:
C
C-              CALL VMOV(TBE1,TBE0)
                CALL MPXBF(226)
C-              CALL DCVM(TBE0,3,TINF,TLPF0)
                CALL MPXBF(227)
C
C               ERROR-PROTECT PREVIOUS XMTR FRAME (MOSTLY UNDER DCVM)
                CALL PROTCT(8)
                CALL MPIST(TBTF0,1)
C
C PEDET...
C
C
C               DO PITCH AND TAP DETERMINATION & PITCH REMOVAL
C
C-              CALL DCOR(TPAC,TBE,TBEZ)
                CALL MPXBF(228)
C
C-              CALL PTAP(TBPR,TPTC,TBPCP,TQTAP,TBE0,TCTAP,TPAC)
                CALL MPXBF(230)
C
C               DO ENERGY CALCULATION:
C
C-              CALL ENRG(TQG,TQGI,TCG,TBPR,TDFSI)
                CALL MPXBF(235)
C
C APC...
C
C-              CALL VAPC(TSNKA,TPTC,TBRDR,TQTAP,TBPR,TCTAP,TBTI)
                CALL MPXBF(236)
C
C
        CALL MPEFL(ANLZA)
^L
```

```
C  <DCA96>DCA96F.FOR.1   Sat 29-Dec-79 7:36PM          Page 5

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       ANLZB: F.L. FOR ANALYSIS USING "B" BUFFERS:
C
        CALL MPBFL(ANLZB)
C
C FOR SCOPE SYNC...
                CALL MPGSC(IG3,ISET)
C
C LPC...
C
C
C               USE TSRB; REMOVE DC; MULT BY HAMMING WINDOW:
C
C-              CALL SSUM(TDCN,TSRB,TFSZI)
                CALL MPXBF(219)
C-              CALL VMUL(TWSR,1,TSRB,TDCN,THAMW,0)
                CALL MPXBF(219)
C-              CALL VMOV(TSRF,TSRL)
                CALL MPXBF(216)
C-              CALL VMOV(TSR,TSRB)
                CALL MPXBF(220)
C
C               AUTOCORRELATE TWSR:
C
C-              CALL DCOR(TACV,TWSR,TWSRZ)
                CALL MPXBF(221)
C
C               FINISHED WITH TSRB
                CALL MPIST(TSRFB,0)
C
C               DELAY REFLECTION COEFFICIENTS:
C               PUT LAST FRAME CODED&QUANTIZED REFL COEFFS (TCQRF)
C               INTO MEMORY (SCALARS TCRF1-TCRF8)
C
                CALL MPWBS(TCQRF,TCRF1,8)
C
C               COMPUTE REFLECTION COEFFS, CODE & QUANTIZE THEM:
C
C-              CALL MWLF(      TRFER,TKTHR,TCQRF,TACV)
                CALL MPXBM(222,TRFER,TKTHR,TCQRF,TACV)
C
C               COMPUTE LP COEFFS (TLPC):
C
C-              CALL VKTOA(TLPC,TQRF)
                CALL MPXBF(223)
C
C               COMPUTE INVERSE FILTERED SAMPLES AFTER
C               MOVING LAST FRAME MEMORY INTO PLACE:
C
C-              CALL VMOV(TINF,TINF1)
                CALL MPXBF(224)
```

```
C  <DCA96>DCA96F.FOR.1   Sat 29-Dec-79 7:36PM          Page 5:1

C-              CALL DCOR(TINF0,TLPCR,TSRM)
                CALL MPXBF(225)
C
C BBEXT...
C
C
C               SAVE LAST FRAME UNSMPLD B.B. EXCITATION;
C               COMPUTE CURRENT FRAME:
C
C-              CALL VMOV(TBE1,TBE0)
                CALL MPXBF(226)
C-              CALL DCVM(TBE0,3,TINF,TLPFB)
                CALL MPXBF(227)
C
C               ERROR-PROTECT PREVIOUS XMTR FRAME (MOSTLY UNDER DCVM)
                CALL PROTCT(A)
                CALL MPIST(TRTFA,1)
C
C PEDET...
C
C
C               DO PITCH AND TAP DETERMINATION & PITCH REMOVAL
C
C-              CALL DCOR(TPAC,TBE,TBEZ)
                CALL MPXBF(228)
C
C-              CALL PTAP(TBPR,TPTC,TBPCP,TQTAP,TBE0,TCTAP,TPAC)
                CALL MPXBF(230)
C
C               DO ENERGY CALCULATION:
C
C-              CALL ENRG(TQC,TQGI,TCG,TBPR,TDFSI)
                CALL MPXBF(235)
C
C APC...
C
C-              CALL VAPC(TSNKB,TPTC,TBRDR,TQTAP,TBPR,TCTAP,TBT1)
                CALL MPXBF(237)
C
C
        CALL MPEFL(ANLZB)
-L
```

AD-A083 238 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL DCA100-79-C-0003
UNCLASSIFIED BBN-4327-VOL-2 NL

END
DATE FILMED
6-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SYNZA: F.L. FOR SYNTHESIS USING "A" BUFFERS
C
        CALL MPBFL(SYNZA)
C
C FOR SCOPE SYNC...
                CALL MPGSC(IG3,ICLR)
C
C APCI...
C
C-              CALL DEAL(RBR,RPTC,RRF0,RTAP,RSRA)
                CALL MPXBF(238)
C-              CALL VIAPC(RBR,RTAP,RRR,RPTC)
                CALL MPXBF(239)
C
C HFR...
C
C
C               SAVE PREV FRAME RESULT,
C               HIPASS FILTER B.B. EXCITATION SAMPLES:
C
C-              CALL VMOV(RHBE,RHBE1)
                CALL MPXBF(241)
C-              CALL DFL22(RHBE0,RAWC1,RBE,RBWM1)
                CALL MPXBF(242)
C
C               UPSAMPLE (3:1) AND LOWPASS FILTER:
C
C-              CALL DCOR(RUBE1,RLPU1,RHBE)
                CALL MPXBF(243)
C-              CALL DCOR(RUBE2,RLPU2,RHBE)
                CALL MPXBF(244)
C-              CALL DCOR(RUBE3,RLPU3,RHBE)
                CALL MPXBF(245)
C
C               SAVE PREV FRAME RESULT,
C               PERTURB AND UPSAMPLE,  HIPASS FILTER:
C
C-              CALL VMOV(RUPB,RUPB1)
                CALL MPXBF(246)
C-              CALL PRTRR(RUPB0,RPC1,RHBE0,RPC2,RRND)
                CALL MPXBF(247)
C-              CALL DCOR(RHUP,RHPU,RUPB)
                CALL MPXBF(248)
C
C               CORRECT NEXT RCVR FRAME (UNDER DCOR)
                CALL CORECT(0)
                CALL MPIST(RATFB,0)
C
C               SUM HPF AND LPF OUTPUTS:
```



```
C
C-              CALL VSMA2(RFES,1,RUBE,1,RHUP,0)
                CALL MPXBF(249)
C
C SWFL...
C
C-              CALL VLTSV(RSWKA,RSFM,RRF1,RFES)
                CALL MPXBF(250)
C-              CALL VMOV(RRF1,RRF0)
                CALL MPXBF(251)
C
C               SET FLAG TO SAY THAT RSWKA IS FULL
                CALL MPIST(RSWFA,1)
C
C
      CALL MPEFL(SYNZA)
^L
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SYNZB: F.L. FOR SYNTHESIS USING "B" BUFFERS
C
        CALL MPBFL(SYNZB)
C
C FOR SCOPE SYNC...
                CALL MPGSC(IG3,ICLR)
C
C APCI...
C
C-              CALL DEAL(RBR,RPTC,RRF0,RTAP,RSBR)
                CALL MPXBF(240)
C-              CALL VIAPC(RBR,RTAP,RRR,RPTC)
                CALL MPXBF(239)
C
C HPR...
C
C
C               SAVE PREV FRAME RESULT,
C               HIPASS FILTER B.B. EXCITATION SAMPLES:
C
C-              CALL VMOV(RHBE,RHBE1)
                CALL MPXBF(241)
C-              CALL DFL22(RHBE0,RBWC1,RBE,RBWM1)
                CALL MPXBF(242)
C
C               UPSAMPLE (3:1) AND LOWPASS FILTER:
C
C-              CALL DCOR(RUBE1,RLPU1,RHBE)
                CALL MPXBF(243)
C-              CALL DCOR(RUBE2,RLPU2,RHBE)
                CALL MPXBF(244)
C-              CALL DCOR(RUBE3,RLPU3,RHBE)
                CALL MPXBF(245)
C
C               SAVE PREV FRAME RESULT,
C               PERTURB AND UPSAMPLE,  HIPASS FILTER:
C
C-              CALL VMOV(RUPB,RUPB1)
                CALL MPXBF(246)
C-              CALL PRTRB(RUPB0,RPC1,RHBE3,RPC2,RRND)
                CALL MPXBF(247)
C-              CALL DCOR(RHUP,RHPU,RUPB)
                CALL MPXBF(248)
C
C               CORRECT NEXT RCVR FRAME (UNDER DCOR)
                CALL CORECT(A)
                CALL MPIST(RATFA,0)
C
C               SUM HPF AND LPF OUTPUTS:
```



```
C
C-              CALL VSMA2(RFES,1,RUBE,1,RHUP,0)
                CALL MPXBF(249)
C
C SNFL...
C
C-              CALL VLTSY(RSNKB,RSFM,RRF1,RFES)
                CALL MPXBF(252)
C-              CALL VMOV(RRF1,RRF0)
                CALL MPXBF(251)
C
                CALL MPIST(RSNFB,1)
C
C
        CALL MPEFL(SYNZB)
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C
        RETURN
        END
```

```
C [BBN-TENEXD]<KFIELD>DCA96I.FOR.28, 31-Dec-79 14:46:59, Ed: KFIELD
      SUBROUTINE DCA96I
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                        DCA96I
C       DCA96 INITIALIZATION - BUFFER AND SCALAR
C       INITIALIZATION.
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       HOST ARRAYS FOR INITIALIZATION
C
      REAL HWORK(270)
      INTEGER IWORK(270)
C
C       MAP-RELATED VARIABLES
C
      REAL HWSZ,FWSZ,DWSZ
      INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
      INTEGER PRCSR,AP
C
C
C       MAP FUNCTION NAMES
C
      INTEGER VCLR
C
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
C
      INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, THAMW
      INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
      INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
      INTEGER TSRL, TINFB,TINF1,TINF, TLPFB,TREB
      INTEGER TBE1, TBE,  TBEZ, TWSRZ
      INTEGER TPAC, TBPCP
      INTEGER TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC
      INTEGER TBTA, TBTB, TBTC
      INTEGER TMDMA,TMDMB
      INTEGER RBTA, RBTB
      INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
      INTEGER RSRB, RRFB, RBR,  RBE,  RHBEB,RHBE1
      INTEGER RHBE, RLPU, RLPB1,RLPB2,RLPB3,RUBE
      INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPBB,RUPB1
      INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
      INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
```



```
C       MAP SCALARS
C
      INTEGER TDCN, TFSZI,TKTHR,TPTC
      INTEGER TE,   TQTAP,TQG,  TQGI, TBT1
      INTEGER TBT2, TBT3, TBQB, TBQ1, TBQ2, TBQ3
      INTEGER TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4
      INTEGER TCRF5,TCRF6,TCRF7,TCRFB,T5
      INTEGER TDFSI
      INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
      INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
      INTEGER RPTC
C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RIPTC
      INTEGER RUN
      INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO
      INTEGER RBTPO,RSNPO,DAPO, TADFDC
      INTEGER TMFFC,RMFDC,RSNNR,TFRCTR
      INTEGER RFRCTR,     RLSCTR
      INTEGER RONNK,RSYNC,RBOFO,RNOCOR
      INTEGER RERSIN,     VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IDS
C
      COMMON /BSIDS/
     ;TADBA,TADBB,TADBC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINFB,TINF1,TINF, TLPFB,TREB,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TBPR, TBRDL,TBRDR,TSNKA,TSNKB,TSNKC,
     ;TBTA, TBTB, TBTC,
     ;TMDMA,TMDMB,
     ;RBTA, RBTB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSRB, RRFB, RBR,  RBE,  RHBEB,RHBE1,
     ;RHBE, RLPU, RLPB1,RLPB2,RLPB3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RRND, RUPBB,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCN, TFSZI,TKTHR,TPTC,
     ;TE,   TQTAP,TQG,  TQGI, TBT1,
     ;TBT2, TBT3, TBQB, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRFB,T5,
     ;TDFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
```

```
     ;RRTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONNK,RSYNC,RROFO,RNOCOR,
     ;RERSIM,     VSTATE

C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     ;ATADBC,STADBC,
     ;ATBTC, STBTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBRDL,STBRDL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC
C
C
C
        DATA PRCSR/1/,AP/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
C
        IBUS1 =  64
        IBUS2 = 128
        IBUS3 = 192
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INITIALIZE MAP BUFFERS AND SCALARS.
C
C       THIS MODULE INITIALIZES ALL MAP BUFFERS AND SCALARS (REAL &
C       INTEGER) WHICH REQUIRE INITIALIZATION.
C       AN "H" FOLLOWED BY A SCALAR NAME SPECIFIES THE
C       HOST VARIABLE CONTAINING THE HOST COPY OF THE
C       INITIAL CONDITIONS OF THAT SCALAR.
C       HOST ARRAYS 'HWORK'(REAL) AND 'IWORK'(INTEGER) ARE
C       USED TO CONTAIN THE HOST COPIES OF THE INITIAL
C       CONDITIONS OF ALL BUFFERS.
C
```



```
C
C       INITIALIZATION CONSTANTS:
C
        IUPFSZ = 180
        IDNFSZ = 60
        ILPF   = 0
        IHPF   = 1
C
C       !!!USE BUFFER 'RUBE' AS TEMP BUFFER FOR INITIALIZATION
C       !!!OF 'TADBC', SINCE MPWDB CAN'T GO DIRECTLY TO A
C       !!!RL,SHRT BUFFER. ('RUBE' IS RL,CNTG,LNG WITH SIZE
C       !!!OF 180.  'RUBE' DOES NOT REQUIRE INITIALIZATION.)
C
        IDUM = RUBE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INITIALIZE BUFFERS.
C
C
C INITIALIZE 'TADBC' (A/D 'TONE' BUFFER) TO SQUARE WAVE
C       AT APPROX 225 HZ.  (EACH BUFFER = APPROX 27.2 MSEC.  =>
C       APPROX 37 BUFFERS/SEC.;  6 CYCLES/BUFFER  =>  APPROX 225 HZ.)
C
C       USE BID 63 AS TEMP BID
C       (TADBC IS NOT A SNAP BUFFER)
C
        TADBC = 63
        CALL MPCLB(IBUS1+TADBC,ATADBC,STADBC,RL,CNTG,SHRT)
C
        AMP = .055
        NCYCLE = 6
        CALL SQWAV(HWORK,IUPFSZ,NCYCLE,AMP)
        CALL MPWDB(IDUM,HWORK,4,1,HWORK(IUPFSZ))
        CALL VMOVW(TADBC,IDUM)
C
C
C INITIALIZE 'THAMW'  (HAMMING WINDOW COEFFS)
C
        CALL HAMMNG(HWORK,IUPFSZ)
        CALL MPWDB(THAMW,HWORK,4,1,HWORK(IUPFSZ))
C
C
C INITIALIZE 'TWSRZ' TO ZEROS
C
        CALL VCLR(TWSRZ)
C
C
C INITIALIZE 'TSRM' TO ZEROS
C
        CALL VCLR(TSRM)
```

```
C
C
C INITIALIZE "TCQRF" TO ZEROS
C
      CALL VCLR(TCQRF)
C
C
C INITIALIZE "TINF1" TO ZEROS
C
      CALL VCLR(TINF1)
C
C
C INITIALIZE "TLPFB" TO LPF COEFFS
C       (75 COEFFS WITH NO PADDING)
C
      NZRPD = 0
      NCOEFS = 75
      CALL LPFWPF(HWORK,NZRPD,NCOEFS,ILPF)
      CALL MPWDB(TLPFB,HWORK,4,1,HWORK(NCOEFS))
C
C
C INITIALIZE "TBEZ" TO ZEROS
C
      CALL VCLR(TBEZ)
C
C
C INITIALIZE "TBRDL" TO ZEROS
C
C       USE BID 63 AS TEMP BID
C       (TBRDL IS NOT A SNAP BUFFER)
C
      TBRDL = 63
      CALL MPCLB(IBUS2+TBRDL,ATBRDL,STBRDL,RL,CNTG,LNG)
C
      CALL VCLR(TBRDL)
C
C
C INITIALIZE "TBRDR" TO ZEROS
C
      CALL VCLR(TBRDR)
C
C
C INITIALIZE "TSNKB" TO ZEROS
C
      CALL VCLR(TSNKB)
C
C
C INIT "TBTC" TO THE BITSTREAM EQUIV OF "SILENCE".
C       (ASSUME CODED ENERGY = 0 WILL TAKE CARE OF IT,
C       SO INITIALIZE TO $000C = 12)
C
```



```
C       USE BID 63 AS TEMP BID
C       (TBTC IS NOT A SNAP BUFFER)
C
      TBTC = 63
      CALL MPCLB(IBUS1+TBTC,ATBTC,STBTC,FXD,CNTG,LNG)
C
      DO 5 I=1,IFIX(STBTC)
5     IWORK(I) = 12
C
      CALL MPWDB(TBTC,IWORK,2,1,IWORK(IFIX(STBTC)))
C
C
C INITIALIZE "TMDMA" TO $000C'S (=12)
C       SYNC BIT (= LOW BIT OF 1ST HWORD) = 0
C
C
      DO 7 I=1,IFIX(STMDMA)
7     IWORK(I) = 12
C
      CALL MPWDB(TMDMA,IWORK,2,1,IWORK(IFIX(STMDMA)))
C
C
C INITIALIZE "TMDMB" TO $000C'S (=12)
C EXCEPT 1ST HWORD <= $000D (=13)
C       SYNC BIT (= LOW BIT OF 1ST HWORD) = 1
C
C
      DO 8 I=1,IFIX(STMDMB)
8     IWORK(I) = 12
      IWORK(1) = IWORK(1) + 1
C
      CALL MPWDB(TMDMB,IWORK,2,1,IWORK(IFIX(STMDMB)))
C
C
C INITIALIZE "RSRB" TO ZEROS
C
      CALL VCLR(RSRB)
C
C
C INITIALIZE "RBE" TO ZEROS
C
      CALL VCLR(RBE)
C
C
C INITIALIZE "RHBE1" TO ZEROS
C
      CALL VCLR(RHBE1)
C
C
C INITIALIZE "RLPD" TO LPF COEFFS * 3
C       (75 COEFFS WITH NO PADDING)
```

```
C
C       USE BID 63 AS TEMP BID
C       (RLPU IS NOT A SNAP BUFFER)
C
        RLPU = 63
        CALL MPCLB(IBUS3+RLPU,ARLPU,SRLPU,RL,CNTG,LNG)
C
        NZRPD = 0
        NCOEFS = 75
        CALL LPFMPF(MWORK,NZRPD,NCOEFS,ILPF)
        DO 10 I=1,NCOEFS
   10   MWORK(I) = MWORK(I) * 3.
        CALL MPWDB(RLPU,MWORK,4,1,MWORK(NCOEFS))
C
C
C INITIALIZE "RRND" TO GAUSSIAN RANDOM VALUES
C
        CALL RANDOM(PWORK,IDNFSZ)
        CALL MPWDB(RRND,MWORK,4,1,MWORK(IDNFSZ))
C
C
C INITIALIZE "RUPR1" TO ZEROS
C
        CALL VCLR(RUPR1)
C
C
C INITIALIZE "RMPU" TO MPF COEFFS * 3
C       (75 COEFFS WITH NO PADDING)
C
        NZRPD = 0
        NCOEFS = 75
        CALL LPFMPF(MWORK,NZRPD,NCOEFS,IMPF)
        DO 20 I=1,NCOEFS
   20   MWORK(I) = MWORK(I) * 3.
        CALL MPWDB(RMPU,MWORK,4,1,MWORK(NCOEFS))
C
C
C INITIALIZE "RRF1" TO ZEROS
C
        CALL VCLR(RRF1)
C
C
C INITIALIZE "RSFM" TO ZEROS
C
        CALL VCLR(RSFM)
C
C
C INITIALIZE "RSNKC" TO "SILENCE"
C
C       USE BID 63 AS TEMP BID
C       (RSNKC IS NOT A SNAP BUFFER)
```



```
C
        RSNKC = 63
        CALL MPCLB(IBUS1+RSNKC,ARSNKC,SRSNKC,RL,CNTG,SHRT)
C
        CALL VCLR(RSNKC)
C
C
C INITIALIZE "RDABA" TO ZEROS
C
C
        CALL VCLR(RDABA)
C
C
C INITIALIZE "RDABB" TO ZEROS
C
C
        CALL VCLR(RDABB)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INITIALIZE (REAL) SCALARS.
C
C
C
C
C
C INITIALIZE "TFSZI"  (NEGATIVE INVERSE OF FRAMESIZE)
C
        HTFSZI = -1./180.
        CALL MPWST(TFSZI,HTFSZI,1,1)
C
C
C INITIALIZE "TKTHR"  (MWLF THRESHOLD)
C
        HTKTHR = .9995
        CALL MPWST(TKTHR,HTKTHR,1,1)
C
C
C INITIALIZE "TBT1"-"TBT3"  (B.B. QUANT. THRESHOLDS)
C
        HTBT1  = 0.531
        HTBT2  = 1.249
        HTBT3  = 2.371
        CALL MPWST(TBT1,HTBT1,1,1)
        CALL MPWST(TBT2,HTBT2,1,1)
        CALL MPWST(TBT3,HTBT3,1,1)
C
C
C INITIALIZE "TBQ0"-"TBQ3"  (B.B. QUANT. VALUES)
```

```
C
      HTBQ0  = 0.233
      HTBQ1  = 0.839
      HTBQ2  = 1.666
      HTBQ3  = 3.075
      CALL MPWST(TBQ0,HTBQ0,1,1)
      CALL MPWST(TBQ1,HTBQ1,1,1)
      CALL MPWST(TBQ2,HTBQ2,1,1)
      CALL MPWST(TBQ3,HTBQ3,1,1)
C
C
C INITIALIZE 'RBWC1'-'RBWC4'  (BUTTERWORTH FILTER COEFFS)
C
      HRBWC1 = -2.
      HRBWC2 =  1.
      HRBWC3 = -1.7085
      HRBWC4 =   .7459
      CALL MPWST(RBWC1,HRBWC1,1,1)
      CALL MPWST(RBWC2,HRBWC2,1,1)
      CALL MPWST(RBWC3,HRBWC3,1,1)
      CALL MPWST(RBWC4,HRBWC4,1,1)
C
C
C INITIALIZE 'RBWM1'-'RBWM4' TO ZEROS (BUTTERWORTH FILTER MEMORY)
C       (SCALAR ## IS PRESET BY EXEC TO EQUAL 0)
C
      CALL MPTSS(RBWM1,0)
      CALL MPTSS(RBWM2,0)
      CALL MPTSS(RBWM3,0)
      CALL MPTSS(RBWM4,0)
C
C
C INITIALIZE 'RPC1'  (1ST PERTURBATION CONSTANT)
C
      HRPC1 = .7
      CALL MPWST(RPC1,HRPC1,1,1)
C
C
C INITIALIZE 'RPC2'  (2ND PERTURBATION CONSTANT)
C
      HRPC2 = -(HRPC1/35.) * (2.**10)
      CALL MPWST(RPC2,HRPC2,1,1)
C
C
C INITIALIZE 'T5' TO FIVE
C
      HT5 = 5.
      CALL MPWST(T5,HT5,1,1)
C
C
C INITIALIZE 'TDFSI' (INVERSE OF DOWNSAMPLED FRAME SIZE)
```



```
C
      HTDFSI = 1./60.
      CALL MPWST(TDFSI,HTDFSI,1,1)
C
C
C
C
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INITIALIZE INTEGER SCALARS.
C
C
C INITIALIZE 'TSRFA' TO ZERO
C
      CALL MPIST(TSRFA,0)
C
C
C INITIALIZE 'TSRFB' TO ZERO
C
      CALL MPIST(TSRFB,0)
C
C
C INITIALIZE 'TBTFA' TO ZERO
C
      CALL MPIST(TBTFA,0)
C
C
C INITIALIZE 'TBTFB' TO ZERO
C
      CALL MPIST(TBTFB,0)
C
C
C INITIALIZE 'RBTFA' TO ZERO
C
      CALL MPIST(RBTFA,0)
C
C
C INITIALIZE 'RBTFB' TO ZERO
C
      CALL MPIST(RBTFB,0)
C
C
C INITIALIZE 'RSNFA' TO ZERO
C
      CALL MPIST(RSNFA,0)
C
C
C INITIALIZE 'RSNFB' TO ZERO
```

```
C
      CALL MPIST(RSNFR,0)
C
C
C INITIALIZE 'RUN' TO 0 (WILL SET TO 1 WHEN STARTING VOCODER)
C
      CALL MPIST(RUN,0)
C
C
C INITIALIZE 'ADPO' TO 0
C
      CALL MPIST(ADPO,0)
C
C
C INITIALIZE 'TSRPO' TO -2
C
      CALL MPIST(TSRPO,-2)
C
C
C INITIALIZE 'TBTPO' TO 0
C
      CALL MPIST(TBTPO,0)
C
C
C INITIALIZE 'TMPO' TO 0
C
      CALL MPIST(TMPO,0)
C
C
C INITIALIZE 'RMPO' TO 0
C
      CALL MPIST(RMPO,0)
C
C
C INITIALIZE 'RBTPO' TO -2
C
      CALL MPIST(RBTPO,-2)
C
C
C INITIALIZE 'RSNPO' TO 0
C
      CALL MPIST(RSNPO,0)
C
C
C INITIALIZE 'DAPO' TO 0
C
      CALL MPIST(DAPO,0)
C
C
C INITIALIZE 'TADFDC' TO 0
C
      CALL MPIST(TADFDC,0)
```



```
C
C
C INITIALIZE 'TMFFC' TO 0
C
      CALL MPIST(TMFFC,0)
C
C
C INITIALIZE 'RMFDC' TO 0
C
      CALL MPIST(RMFDC,0)
C
C
C INITIALIZE 'RSNNR' TO 0
C
      CALL MPIST(RSNNR,0)
C
C
C INITIALIZE 'TFRCTR' TO 0
C
      CALL MPIST(TFRCTR,0)
C
C
C INITIALIZE 'RFRCTR' TO 0
C
      CALL MPIST(RFRCTR,0)
C
C
C INITIALIZE 'RLSCTR' TO 0
C
      CALL MPIST(RLSCTR,0)
C
C
C INITIALIZE 'RONHK' TO 0
C
      CALL MPIST(RONHK,0)
C
C
C INITIALIZE 'RSYNC' TO 0
C
      CALL MPIST(RSYNC,0)
C
C
C INITIALIZE 'RNOCOR' TO 0
C
      CALL MPIST(RNOCOR,0)
C
C
C INITIALIZE 'RERSIM' TO 0
C
      CALL MPIST(RERSIM,0)
C
```

```
C
C INITIALIZE "VSTATE" TO 0
C
      CALL MPIST(VSTATE,0)
C
C
C
C
C
      RETURN
      END
```

```
C [BBN-TENEXD]<KFIELD>DCA964.FOR.16,  5-Dec-79 22:39:28, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                       DCA9.6
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS POUT SEGMENT CONTAINS CALLS TO SUBROUTINES
C       DCA96C (CONFIGURATION - BUFFER AND SCALAR
C               CONFIGURATION)
C       DCA96I (INITIALIZATION - BUFFER AND SCALAR
C               INITIALIZATION)
C       DCA96F (FUNCTION PREBINDING AND FUNCTION LIST
C               DEFINITIONS)
C       DCA96E (EXECUTION).
C       COMMUNICATION BETWEEN THESE THREE SUBROUTINES
C       IS VIA MAP-300 MEMORY, WHICH CONTAINS
C       SYSTEM BUFFERS, SCALARS AND FUNCTION LISTS,
C       AND VIA COMMON BLOCKS "BSIDS" (BUFFER AND
C       SCALAR ID NAME DEFINITIONS), "BAS" (BUFFER
C       ADDRESSES AND SIZES NEEDED BY DCA96I),
C       "FLS" (FUNCTION LIST NAME DEFINITIONS).
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C       USER INTERACTION VARIABLE(S)
C
        INTEGER Q
C
C       MAP FUNCTION LIST NAMES
C
        INTEGER DCARTS,DCALP,DCAFFT,DCATIM,ORSTRT
        INTEGER ANLZA,ANLZB,SYNZA,SYNZB
C
C
C [BBN-TENEXD]<KFIELD>DCACOM.FOR.8,  5-Dec-79 14:42:27, Ed: WOLF
C       MAP BUFFER NAMES
C
        INTEGER TADRA,TADBB,TADRC,TSRA, TSRB, THAMW
        INTEGER TWSR, TLPCR,TACV, TRFER,TCQRF
        INTEGER TCRF, TQRF, TLPC, TSR,  TSRM, TSRF
        INTEGER TSRL, TINFB,TINF1,TINF, TLPFB,TBEB
        INTEGER TBE1, TBE,  TBEZ, TWSRZ
        INTEGER TPAC, TBPCP
        INTEGER TAPP, TARDL,TBRDR,TSNKA,TSNKB,TSNKC
        INTEGER TBTA, TBTB, TBTC
        INTEGER TMDMA,TMDMB
```



```
        INTEGER RBTA, RBTB
        INTEGER RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA
        INTEGER RSRB, RRFB, RBR,  RBE,  RHBEB,RHBE1
        INTEGER RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE
        INTEGER RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1
        INTEGER RUPB, RHPU, RHUP, RFES, RRF1, RSFM
        INTEGER RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C       MAP SCALARS
C
        INTEGER TDCN, TFSZI,TKTHR,TPTC
        INTEGER TE,   TQTAP,TQC,  TQCI, TBTI
        INTEGER TBT2, TBT3, TBQ0, TBQ1, TBQ2, TBQ3
        INTEGER TCTAP,TCC,  TCRF1,TCRF2,TCRF3,TCRF4
        INTEGER TCRF5,TCRF6,TCRF7,TCRF8,T5
        INTEGER TOFSI
        INTEGER RTAP, RBWC1,RBWC2,RBWC3,RBWC4
        INTEGER RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2
        INTEGER RPTC
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TIPTC
        INTEGER RBTFA,RRTFB,RSNFA,RSNFB,RIPTC
        INTEGER RUN
        INTEGER ADPO, TSRPO,TBTPO,TMPO, RNPO
        INTEGER RBTPO,RSNPO,DAPO, TADFDC
        INTEGER TMFFC,RMFOC,RSMNR,TFRCTR
        INTEGER RFRCTR,     RLSCTR
        INTEGER RONHK,RSYNC,RBOFO,RNOCOR
        INTEGER RERSIN,     VSTATE
C
C       COMMON BLOCK - BUFFER AND SCALAR IDS
C
        COMMON /BSIDS/
     ;TADBA,TADBB,TADRC,TSRA, TSRB, THAMW,
     ;TWSR, TLPCR,TACV, TRFER,TCQRF,
     ;TCRF, TQRF, TLPC, TSR,  TSRM, TSRF,
     ;TSRL, TINFB,TINF1,TINF, TLPFB,TBEB,
     ;TBE1, TBE,  TBEZ, TWSRZ,
     ;TPAC, TBPCP,
     ;TAPR, TARDL,TBRDR,TSNKA,TSNKB,TSNKC,
     ;TBTA, TBTB, TBTC,
     ;TMDMA,TMDMB,
     ;RBTA, RBTB,
     ;RMDMA,RMDMB,RMDMC,RSSPF,RSSSS,RSRA,
     ;RSRB, RRFB, RBR,  RBE,  RHBEB,RHBE1,
     ;RHBE, RLPU, RLPU1,RLPU2,RLPU3,RUBE,
     ;RUBE1,RUBE2,RUBE3,RRND, RUPB0,RUPB1,
     ;RUPB, RHPU, RHUP, RFES, RRF1, RSFM,
     ;RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ;TDCN, TFSZI,TKTHR,TPTC,
```

```
     ;TE,   TQTAP,TQG,  TQGI, TBT1,
     ;TBT2, TBT3, TBQA, TBQ1, TBQ2, TBQ3,
     ;TCTAP,TCG,  TCRF1,TCRF2,TCRF3,TCRF4,
     ;TCRF5,TCRF6,TCRF7,TCRF8,T5,
     ;TDFSI,
     ;RTAP, RBWC1,RBWC2,RBWC3,RBWC4,
     ;RBWM1,RBWM2,RBWM3,RBWM4,RPC1, RPC2,
     ;RPTC,
     ;TSRFA,TSRFB,TBTFA,TBTFB,TIPTC,
     ;RBTFA,RBTFB,RSNFA,RSNFB,RIPTC,
     ;RUN,
     ;ADPO, TSRPO,TBTPO,TMPO, RMPO, RSNPO,
     ;RBTPO,DAPO, TADFDC,     TMFFC,RMFDC,
     ;RSNNR,TFRCTR,     RFRCTR,     RLSCTR,
     ;RONHK,RSYNC,RBOFO,RNOCOR,
     ;RERSIM,     VSTATE

C
C       COMMON BLOCK -
C       BUFFER ADDRESSES AND SIZES NEEDED FOR INIT OF NON-SNAP BUFFERS
C       BY DCA96I (WRITTEN BY DCA96C)
C
        COMMON /BAS/
     ;ATADBC,STADBC,
     ;ATBTC, STBTC,
     ;ARLPU, SRLPU,
     ;ARSNKC,SRSNKC,
     ;ATBROL,STBROL,
     ;ATMDMA,STMDMA,
     ;ATMDMB,STMDMB,
     ;ARDABA,SRDABA,
     ;ARDABB,SRDABB
C
CCCCCCCC END OF DCACOM.FOR CCCCCCCC
C
C       COMMON BLOCK -
C       FUNCTION LIST NAMES
C
        COMMON /FLS/
     ;DCARTS,DCALP,DCAFFT,DCATIM,DRSTRT,
     ;ANLZA,ANLZB,SYNZA,SYNZB
C
C
        DATA Q/'Q'/
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       LOOP THRU SYSTEM UNTIL USER SAYS TO QUIT.
C
C
```



```
1       TYPE 2
2       FORMAT(' BBN 9600 BPS MAP-300 VOCODER SYSTEM'/)
C
 5      CONTINUE
C
        TYPE 10
10      FORMAT(' CONFIGURING MAP BUFFERS AND SCALARS...')
        CALL DCA96C
C
        TYPE 20
20      FORMAT(' INITIALIZING MAP BUFFERS AND SCALARS...')
        CALL DCA96I
C
        TYPE 30
30      FORMAT(' PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS
                                          **')
        CALL DCA96F
C
        TYPE 40
40      FORMAT(' EXECUTING DCA96 SYSTEM...')
        CALL DCA96E
C
        TYPE 100
100     FORMAT(' ENTER "Q" TO QUIT, ANY OTHER CHAR. TO RESTART SYSTEM:
                                        **$)
        ACCEPT 101,ICHAR
101     FORMAT(A1)
C
C
        IF (ICHAR .NE. Q) GOTO 5
C
C
C
C
C
        STOP
        END
```

```
C [BBN-TENEXD]<KFIELD>DCA96S.FOR.2, 17-Dec-79 15:21:18, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE SQWAV(OUTBUF,ISIZE,NCYCLE,AMP)
C
C       PURPOSE
C               COMPUTES A SQUARE WAVE OF AMPLITUDE "AMP",
C               WITH "NCYCLE" CYCLES OCCURRING IN "ISIZE" SAMPLES.
C
C       PARAMETERS
C               OUTBUF - (REAL) OUTPUT ARRAY
C               ISIZE - (INTEGER) NUMBER OF OUTPUT POINTS
C                       (MUST BE A MULTIPLE OF "NCYCLE")
C               NCYCLE - (INTEGER) NUMBER OF SQUARE WAVE CYCLES
C               AMP - (REAL) SQUARE WAVE AMPLITUDE
C
C
        SUBROUTINE SQWAV(OUTBUF,ISIZE,NCYCLE,AMP)
C
        REAL OUTBUF(1)
C
        IBLKSZ = ISIZE/NCYCLE
C
        DO 10 I=0,NCYCLE-1
C
        DO 20 J=1,IBLKSZ/2
   20   OUTBUF((I * IBLKSZ) + J) = AMP
C
        DO 30 J=(IBLKSZ/2)+1,IBLKSZ
   30   OUTBUF((I * IBLKSZ) + J) = -AMP
C
   10   CONTINUE
C
C
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE HAMMNG(OUTBUF,ISIZE)
C
C       PURPOSE
C               GENERATES HAMMING WINDOW OF SIZE "ISIZE" (INTEGER)
C               INTO REAL ARRAY "OUTBUF".
C
C
        SUBROUTINE HAMMNG(OUTBUF,ISIZE)
C
        REAL OUTBUF(1)
C
        DARG = 3.1415926535 * 2./ISIZE
        DO 10 I=1,ISIZE
        OUTBUF(I) = .54 - .46*COS(DARG*(I-1))
   10   CONTINUE
```



```
C
        RETURN
        END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE LPFHPF(OUTBUF,NZRPD,NCOEFS,IFLAG)
C
C       PURPOSE
C               GENERATES LOW (IFLAG=0) OR HIGH (IFLAG=1) PASS
C               FILTER COEFFICIENTS FOR SYMMETRIC FILTER OF SIZE
C               "NCOEFS", FROM 38 PRE-GENERATED FILTER COEFFS.
C               PAD FILTER WITH "NZRPD" ZEROS ON RIGHT AND LEFT.
C
C       PARAMETERS
C               OUTBUF - (REAL) OUTPUT ARRAY
C               NZRPD - (INTEGER) NUMBER OF ZEROS TO BE PADDED
C                       ON RIGHT AND LEFT
C               NCOEFS - (INTEGER) SIZE OF "OUTBUF"
C                       (NCOEFS = 75 + (2*NZRPD))
C               IFLAG - (INTEGER) 0 => LPF
C                                 1 => HPF
C
C       NOTES
C               INTERNAL ARRAY "COEF" IS LEFTMOST 38 COEFFICIENTS
C               FOR 75 POINT SYMMETRIC LPF.
C               TO GENERATE LPF COEFFICIENTS, FLIP "COEF" ABOUT
C                THE 38TH COEFFICIENT.
C               TO GENERATE HPF COEFFICIENTS, NEGATE ALL 38
C                ELEMENTS OF "COEF", FLIP IT, AND ADD 1 TO
C                CENTER (38TH) COEFFICIENT.
C
C
        SUBROUTINE LPFHPF(OUTBUF,NZRPD,NCOEFS,IFLAG)
C
        REAL OUTBUF(1),COEF(38)
        DATA COEF/
     ; .003278332,  -.01239776,   .0019428,    .005104966,
     ; .003773232,  -.0062597876,-.00432255,  -.005019879,
     ;-.001160612,   .004424945,  .006697484,  .002892782,
     ;-.00435413,   -.008619823, -.005261945,  .003793186,
     ; .01072995,    .00841496,  -.002464284, -.01288288,
     ;-.01262172,    .000004605,  .01499785,   .01022254,
     ; .004183214,  -.01694727,  -.02600564,  -.01126482,
     ; .0186138,     .0388134,    .02430349,  -.01989256,
     ;-.06136104,   -.05554319,   .02069518,   .1466779,
     ; .2644006,     .3123665/
C
C
        ILPF = 0
        IHPF = 1
```

```
C
      IF (NZRPD .LE. 0) GOTO 15
      DO 10 I=1,NZRPD
      OUTBUF(I) = 0.
      OUTBUF(NCOEFS - (I-1)) = 0.
   10 CONTINUE
C
   15 DO 20 I=1,37
      IF (IFLAG .EQ. ILPF) OUTBUF(NZRPD+I) = COEF(I)
      IF (IFLAG .EQ. ILPF) OUTBUF((NCOEFS-NZRPD)-(I-1)) = COEF(I)
      IF (IFLAG .EQ. IHPF) OUTBUF(NZRPD+I) = -COEF(I)
      IF (IFLAG .EQ. IHPF) OUTBUF((NCOEFS-NZRPD)-(I-1)) = -COEF(I)
   20 CONTINUE
C
      IF (IFLAG .EQ. ILPF) OUTBUF(((NCOEFS-1)/2)+1) = COEF(38)
      IF (IFLAG .EQ. IHPF) OUTBUF(((NCOEFS-1)/2)+1) = 1.-COEF(38)
C
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE RANDOM(OUTBUF,ISIZE)
C
C     PURPOSE
C             GENERATES 'ISIZE' GAUSSIAN RANDOM NUMBERS
C             (AVG=0, STANDARD DEV=1) INTO OUTPUT ARRAY 'OUTBUF'.
C
C
      SUBROUTINE RANDOM(OUTBUF,ISIZE)
C
      REAL OUTBUF(1)
C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
C     MODIFIED 12/79 BY KFIELD TO USE PRE-CALCULATED RANDOM
C     SEQUENCE.  THE RANDOM SEQUENCE CONTAINED IN ARRAY
C     'RANVAL' WAS COMPUTED BY THE COMMENTED-OUT CODE LISTED
C     BELOW.  THE CURRENT ROUTINE RETURNS MIN0(ISIZE,60)
C     GAUSSIAN RANDOM NUMBERS INTO OUTPUT ARRAY 'OUTBUF'.
C
C*    STDV = 1.
C*    AV = 0.
C*    ISEED = 0
C*    JSEED = 0
CC*
C*    DO 10 I=1,ISIZE
C*    CALL GAUSS(ISEED,JSEED,STDV,AV,X)
C*    OUTBUF(I) = X
C*  10 CONTINUE
CC*
```



```
C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
      REAL RANVAL(60)
      DATA RANVAL/
     ;-2.7444408,-0.0094934, 1.4597659, 1.2063489, 0.4131660,
     ; 1.4147344, 0.0214019, 1.6560664, 1.3033938, 0.2715330,
     ; 1.7663431, 0.2781010,-1.2192678,-2.2564795,-0.0446711,
     ;-0.8386793,-0.4878235, 0.8258138,-0.3383965,-0.7929077,
     ; 1.9772177, 0.3260298,-0.2290912, 0.1292825, 0.4678478,
     ; 0.8642869,-1.0185137, 1.4542589,-0.9114408, 0.0732004,
     ; 1.8040791,-0.4791102,-0.2436576,-0.4021111,-0.4380717,
     ;-0.5194597,-0.7993069, 1.0959602,-0.6092229,-0.7029214,
     ; 1.9636006, 0.2375574,-1.7611332,-0.2531296, 0.3995585,
     ;-0.2947278,-0.1330595, 0.4187880,-0.3244419,-0.0057702,
     ; 2.8482389, 1.7141938,-0.2289071, 0.4120498,-0.4314938,
     ;-1.1530452, 1.4781532, 0.0788455, 0.2259036, 0.0434405/
C
C
      DO 20 I=1,MIN0(ISIZE,60)
      OUTBUF(I) = RANVAL(I)
   20 CONTINUE
C
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE GAUSS
C
C     PURPOSE
C             COMPUTES A NORMALLY (GAUSSIAN) DISTRIBUTED
C             RANDOM NUMBER WITH A GIVEN MEAN AND STANDARD
C             DEVIATION.
C
C     USAGE
C             CALL GAUSS(IX,JX,S,AM,V)
C
C     DESCRIPTION OF PARAMETERS
C             IX -    INTEGER VARIABLE INITIALLY SET TO 0
C                     AND THEREAFTER CONTAINING INTEGER NUMBER
C                     GENERATED BY THE SUBROUTINE FOR USE ON THE
C                     NEXT ENTRY TO THE SUBROUTINE.
C             JX -    INTEGER VARIABLE INITIALLY SET TO 0
C                     AND THEREAFTER CONTAINING INTEGER NUMBER
C                     GENERATED BY THE SUBROUTINE FOR USE ON THE
C                     NEXT ENTRY TO THE SUBROUTINE.
C             S -     THE DESIRED STANDARD DEVIATION OF THE
C                     NORMAL DISTRIBUTION.
C             AM -    THE DESIRED MEAN OF THE NORMAL DISTRIBUTION.
C             V -     THE VALUE OF THE COMPUTED NORMAL RANDOM VARIAB
```

```
C
C       REMARKS
C               THIS SUBROUTINE USES RAND WHICH IS MACHINE SPECIFIC
C
C       SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C               RAND
C
C       METHOD
C               USES 12 UNIFORM RANDOM NUMBERS TO COMPUTE
C               NORMAL RANDOM NUMBERS BY CENTRAL LIMIT THEOREM.
C               THE RESULT IS THEN ADJUSTED TO MATCH THE GIVEN
C               MEAN AND STANDARD DEVIATION.  THE UNIFORM
C               RANDOM NUMBERS COMPUTED WITHIN THE SUBROUTINE
C               ARE COMPUTED BY RAN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
        SUBROUTINE GAUSS(IX,JX,S,AM,V)
        A = 0.0
        DO 50 I=1,12
        Y = RAN(IX,JX)
   50   A = A + Y
        V = (A - 6.0)*S + AM
        RETURN
        END
```