CHI-5 ARRAY PROCESSOR
SUPPORT
Final Technical Report
April 1985

PRINCIPAL INVESTIGATOR:
Dr. Glen J. Culler

PROJECT SCIENTISTS:
Dr. Judith B. Bruckner
Thomas W. Fuller
Virginia R. Grant
Dr. Michael McCammon
Dr. Jean A. Nisbet

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3625. Contract MDA 903-82-C-0136.

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

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.
The CHI-5, a low-cost, generalized array processor, was developed under an earlier DARPA contract, MDA903-78-C-0313 as part of an effort to develop low-cost packet speech hardware. The CHI-5 was used successfully by CHI Systems, Inc. and SRI International for digital voice coding using linear predictive coding (LPC) for transmission through packet-switched networks.

In support of the DARPA program in packetized speech, the CHI-5 could be used not only for the basic LPC algorithms, but also to support development and demonstration of algorithms for degraded speech environments, and integration of voice and data transmissions. In addition, the CHI-5 could serve as a low-cost compute server, attached to a host computer, to support circuit simulation in programs developing other low-cost speech hardware.

For the CHI-5 to be useful to a variety of DARPA contractors involved in these programs, it was necessary to provide generally usable program development software for the CHI-5 and to make additional processors available. The CHI-5 host interface needed to be revised to make it possible to connect it to host computers with a standard device interface. Also, a standard asynchronous interface was needed to allow direct connection to communication media and for other simple, low bandwidth interface applications. This contract, MDA-903-82-C-0136, was established to meet these needs by providing ten additional CHI-5 processors with the necessary interfaces and providing spare parts and support for them. It also provides for the development of program development software written in a portable language so that it could be used on computers available at the DARPA contractors who would be using the CHI-5s.

This report describes the hardware and software which was developed during this contract. Section 1 is a description of the hardware enhancements of the CHI-5 and the construction of the additional ten units. Section 2 describes the program development software, including a micro-code assembler and linker, a micro-instruction simulator, and a macro-language assembler and linker. Section 3 describes the interface program to support control and data transfer protocols for use of the serial interface for digital voice, data and control transfers. The appendix contains the micro-code developed for the system and a listing of the serial interface control program.
1. HARDWARE ENHANCEMENTS

In order to make the CHI-5 processor generally useful in a variety of environments, three external interfaces were required. The first of these is a parallel interface to a controlling host computer. This interface allows for rapid transfer of data through DMA access to the data memories of both the host and the CHI-5. It also provides for direct control of the CHI-5 from the host computer by providing for initialize and interrupt controls. The parallel host interface was part of the original design of the CHI-5 and its predecessor, the LPCAP. However, the host side of this interface was originally developed using custom cards which plugged into either a UNIBUS' or QBUS' chassis of a DEC computer. The interface had to be redesigned to use standard interface cards available from DEC and others.

The second interface, also a part of the CHI-5 as originally designed, provides for simultaneous input and output of analog data. This analog interface supports speech data input and output at 8Khz sampling. No changes were required in this interface.

The final external interface supports a pair of asynchronous serial lines, each operating at programmable rates up to 19.2 K baud. These serial interfaces, although they support relatively slow transfer rates, allow connection of the CHI-5 to almost any computer or terminal as well as a large variety of other equipment, and allow communication over inexpensive, long-distance lines. This serial interface was not provided in the original design, but could be accommodated within the existing I/O and interrupt architecture of the CHI-5.

1.1. Parallel Host Interface

The parallel host interface was redesigned to use signals available from the DR11-B or DRV11-B, interface cards for the UNIBUS and QBUS, respectively. These cards support the host half of a direct memory transfer, either from or to the CHI-5, maintaining the word count and host memory address counter. The CHI-5 host interface card (HIF) uses a three bit function code provided by these interfaces to select one of eight address registers for the CHI-5 data memory address. Upon completion of the transfer, an interrupt signal is generated by the DEC interface card for the host computer, and by the HIF for the CHI-5.

Firmware in the CHI-5 uses a function code of 7 to indicate that the data transferred to the CHI-5 is a command, and processes the command immediately. Since each function code can have a separate address in CHI-5 memory for its data, several processes can be set up at once in the CHI-5, each waiting for a signal that their data buffer has been transferred to initiate the next stage in their processing. Digital voice coding, for example, uses two buffers, one holding LPC parameters which have been computed from analog input for transmission or further processing by the host, the other holding parameters received from the host which are to be used by the CHI-5 to synthesize speech for analog output.
The interrupt signals from completion of data transfers are used to initiate parts of the LPC analysis and synthesis programs.

The data transfers to or from the host computer must always be started by the host, who has control over the interface card. However, the program running in the CHI-5 must be able to signal the host when it has data for it, or needs more data to continue, if it is to be able to do real-time processing with speech data. The DMA transfer interface cards available from DEC do not provide for an asynchronous interrupt signal from the attached device, so a second card, the DR11-C or DRV11, is used to provide for this interrupt, as well as a status code which reflects which buffer is ready for a transfer. In addition, the DR11-C provides a means for the host to send an initiate signal to reset the CHI-5 when desired.

1.2. Asynchronous Serial Interface

The asynchronous serial interface of the CHI-5 includes two separate RS232-C standard ports, one configured to connect to a modem or as a terminal to another computer (DTE) and the other configured to connect a terminal to it directly (DCE). The transfer rates of each port can be set separately to one of sixteen speeds. The serial interface uses USART devices which assemble and disassemble characters as they are received or transmitted serially. For each character, an interrupt is generated to the CHI-5, which must then take any input character or provide the next character for transmission. The circuits required for the asynchronous interface are packaged on the arithmetic control unit card of the CHI-5 processor and are connected by a ribbon cable to two D-25 female connectors mounted on the back of the chassis.

1 UNIBUS and QBUS are trademarks of Digital Equipment Corporation (DEC)
2. PROGRAM DEVELOPMENT SOFTWARE

The CHI-5 architecture supports two levels of instruction decoding. The lowest level is a wide micro instruction with separate fields within the instruction to control each part of the hardware for one fixed instruction clock period. Micro instructions are fetched from a program memory, separate from the data memory. Each instruction word is 80 bits long. The program memory includes 2048 words of programmable read only memory, for fixed firmware, and 1024 words of writeable memory. Programming at the micro instruction level allows maximum utilization of the computer, but it requires a fairly detailed understanding of the architecture.

The second type of instruction is called a macro instruction. A macro instruction contains an operation name and a list of operands. It is fetched from the same memory used for data, and is one or more 32 bit doublewords long. Each macro instruction causes the execution of a sequence of micro instructions. A standard macro instruction set is supported using firmware in micro programs in ROM. Application specific macro instructions can be defined by writing their micro programs.

To support this two level programming environment, two separate assemblers and a micro instruction simulator are provided:

CHI5ASM
A FORTRAN-77 program which generates relocatable, microcode modules for the CHI-5. Separately assembled modules may be linked together by the program CHI5LINK to form a single program module. The CHI-5 MICRO-PROGRAMMING REFERENCE MANUAL provides a detailed description of the CHI-5 microinstruction set, micro assembler and linker.

CHI5SIM
A FORTRAN-77 program which simulates the operation of the CHI-5 hardware at the micro instruction level. This simulator can use the program module generated by CHI5LINK. The CHI-5 SIMULATOR REFERENCE MANUAL describes the simulator program in detail.

MACASM
A FORTRAN-77 program which generates relocatable, macrocode modules. Separately assembled modules from MACASM may be linked together with program modules containing microcode from CHI5LINK by the program MACLINK to create a executable load module. The CHI-5 MACRO PROGRAMMING REFERENCE MANUAL provides a detailed description of the macro assembler and linker.

2.1. CHI-5 Micro Assembler and Linker

The microcode assembler allows micro routines to be constructed using symbolic labels, expressions to generate numeric values and mnemonic keyword based definitions for the micro instructions. Pseudo-operations are supported to allow definition of entry points to the routine, references to externally defined symbols,
and to define labels representing the value of an expression. Other pseudo-
operations control listing and output options for the assembler.

The language chosen for specification of the micro instructions is as high level as
is compatible with full expression of the capabilities of the machine. Each CHI-5
micro instruction is composed of one or more individual operations. An operation
is described in terms of an operator with zero, one or two operands and possibly
a destination. The operands, operators and destinations correspond to elements
of the CHI-5 hardware. Each operation can involve several control fields of the
CHI-5 micro instruction, and there is often more than one way to perform a given
operation. The assembler automatically selects the hardware elements needed to
perform the operation specified and generates the proper control fields, although
the programmer can explicitly state what hardware elements are to be used when
necessary. It also keeps track of the usage of the control fields and attempts to
select alternate data paths or arithmetic elements in order to successfully accom-
modate the operations specified for the micro instruction. If it cannot succeed, it
reports the fields where a conflict in usage has occurred.

The syntax of the individual operations has been chosen to be as natural and
simple as possible. Two operand operations, such as most adder and all multi-
plier operations, are specified using an infix notation, e.g.

\[ X \times V \]
\[ MPL + U \rightarrow U \]
\[ MPLMPR + TU \rightarrow UV. \]

Specific selection of hardware elements or options to operations are given by fol-
lowing the operation by a colon (:) and the option, e.g.

\[ X \times Y:PP \] (Unsigned multiplication)
\[ MPLMPR + TU :FG \rightarrow UV \] (Use F and G adders instead of G and H adders).

Data transfer operations are specified by giving the source, a right arrow, and the
destination. Whenever possible, busses will be used in preference to adders, but
if a bus is already in use, or the only path is through an adder, the adder will be
used automatically. However, if a path through a specific adder is required, as
when a test is to be performed on the value being moved, the adder can be
specified.

\[ V \rightarrow W \] (uses YBS if available, otherwise the H adder)
\[ T \rightarrow U \] (uses either the F or G adder)
\[ T:G \rightarrow U \] (uses the G adder)

Single operand operations, such as INC or DEC, use an operator, operand syntax:

\[ INC XA; CLR S; GOTO label1. \]

Operations requiring no operands are specified by their mnemonics alone:

\[ INT HOST; RTN; READ. \]

The assembler output is a relocateable object module containing the information
needed to combine it with other similar modules into a load module. CHI5LINK
is an interactive program, also written in FORTRAN-77, which combines these object modules, resolving references between separate modules, to build a single load module with instructions located at fixed absolute addresses in the program memory. CHI5LINK also produces a symbol table, giving the entry point address associated with each microprogram. The symbol table and load module are used by the simulator for testing of the microprograms. The symbol table is used by the macro assembler to allow it to assign operation values to macro instructions which will use these microprograms. If the load module is linked to load into writeable program memory, the macro linker can include the load module in its data memory image to make it available for loading into program memory under control of the macro program.

2.2. CHI-5 Microinstruction Simulator

The CHI-5 simulator is an interactive program, written in standard FORTRAN-77, which simulates the operation of a CHI-5 processor. The simulator maintains a functional model of the CHI-5. This model is composed of variables, known as "state variables", which hold values corresponding to values held by the elements in a real CHI-5. A set of commands is available to the user with which the values of state variables may be manipulated. Values may be changed directly, or through the simulated execution of CHI-5 instructions.

The simulator may be run either interactively from a terminal, or as a batch job using a file of commands. Its image of the state of the simulated CHI-5 can be examined, and saved or restored to files. A data file represents the state of the data memories, including the array memories X and Y, the table memory R and the main data memory D. A program file contains the state of the program memory; it is often the output of the micro program linker. A state file holds the simulator state variables, including the contents of the simulated CHI-5 registers. During the simulation of CHI-5 instructions, selected state variables can be traced. These variables are written to a logging file each time an instruction is executed.

The simulator supports the analog input and output devices as real-time I/O by maintaining a clock for analog I/O that 'ticks' once every 500 processor instructions during simulation. This corresponds to the 8 KHz analog sampling rate used by the real device. Analog input and output use files which contain the values for input or hold the result. Host DMA transfer is simulated by LOAD and STORE commands for D-memory without affecting the DA registers, S bits or interrupts. The S bits are set by the user to simulate interrupts when desired.

2.3. CHI-5 Macro Assembler and Linker

At the most basic level, the CHI-5 executes micro instructions, with each micro instruction specifying parallel operations for individual hardware elements. Control of the system, however, including applications, interrupt routines, and commands from a host processor, are specified in a higher level language, the instructions of which are called macro instructions.
A macro instruction consists of an operation code and a list of operands. The macro instructions are fetched from D memory for execution. Each macro involves the execution of two micro programs. The first, called the mode, generally fetches the first operand into a register and reads the next doubleword, if any, of the instruction from D memory. It is usually consists of only one or two instructions and is always located in read only program memory; the upper 5 bits of the operation code specify which mode program is to be used. The second micro program performs the actual operation and may be any length; the lower 11 bits of the operation code give the starting address of the program.

The macro assembler generates macro programs for execution on the CHI-5. In order to do this it must know about the set of instructions which will be available in microprogram memory. This information is provided by macro instruction definition statements which associate a macro instruction with an entry point in a microprogram load module and describes its operands. The assembler uses this information to check the use of each instruction and can generate the proper mode for many instructions depending on the location of the first operand.

In order to allow the assembler to generate the proper mode for variable mode instructions, and check that the operands used are correct for the particular instruction, the assembler associates a type with each label used. It also allocates space in X, Y and D memories if desired and supports data statements for initializing D memory variables.

The output of the CHI-5 macro assembler is a relocateable object file. These files are combined using the macro linker, MACLNK, to make a load module file which is ready for transfer into D memory and execution. The linker is an interactive program similar to CHI5LINK, the micro program linker. It also supports the inclusion of microcode from a CHI5LINK output file in the load module and automatically generates a macro subroutine which will load the file into program memory when it is called.

3. SERIAL LINE CONTROL PROGRAM

As part of the packet speech effort, CHI has implemented a preliminary version of the "NSC Low-Rate Vocoder Interface" to provide a means of connecting the CHI-5 to hosts via its RS-232 serial interface. The CHI version of this protocol provides for transfer of both speech and data between the CHI-5 and an external processor, as well as limited control over the vocoder. This protocol has been used with the CHI-5 at the majority of the contractor sites where the processors have been delivered.

The serial protocol encodes both speech and data into characters for transfer over the serial interface using ASCII character codes between "space" and "_" to represent six bits of information by adding the code for "space" as a number to the six bits of data to get the code to send. This uses half of the available 128 codes. The first 32 codes are not used for information transfer, since they are conventionally used for control information. 31 of the last 32 codes,
corresponding to ASCII characters "", "a-z", "{", "|", "}" and "$" are used to define control information within the protocol. The CHI implementation uses six control characters to provide the following functions:

Address (a)
Set the data transfer buffer address to the value given by the following three characters.

Data (d)
Store the data which follows in D memory, at the data transfer buffer address. Eight characters provide enough data for three 16-bit words. The data transfer continues into consecutive addresses in the buffer until another protocol control character is received.

Output data (o)
Convert three 16-bit words at the current buffer address to eight characters and transmit them. The buffer address is then incremented by three so subsequent output data requests will cause the following data to be sent.

Freqn (f)
Start analysis of speech received on the A/D interface. Each 20.5 msec a parcel of speech parameters, consisting of a plauo (p) character and 48 speech data bits packed into 8 characters, will be transmitted from the CHI-5.

Plauo (p)
Synthesize and output through the D/A interface speech using the following eight characters as data for one 20.5 msec frame.

Stop (s)
Stop analysis of speech. Quit sending speech parameters.

In addition to the protocol control characters, the CHI implementation used the ASCII control characters XON and XOFF to limit the rate at which parameters are received to the real time analog output rate. It also recognizes XON and XOFF on input and suspends or resumes transmission of speech and data parameters as requested.

This protocol is implemented in the CHI-5 by an interrupt routine which responds to interrupts from the serial lines and two subroutines accessed by the vocoder programs for parameter sending and receiving. Appendix B contains a listing of the macro language program for this protocol.
REFERENCES

1. Bruckner, J. B.,

2. Fuller, T. W.,

3. Grant, V. R.,

4. Culler, G. J.,
APPENDIX A

Listings of the following CHI-5 microprogram modules are included here:

andl.mic
andm255.mic
asline.mic
ave.mic
bdldpop.mic
blkshifts.mic
coding.mic
daops.mic
decim.mic
divbyy.mic
doint.mic
dop.mic
dotxy.mic
dotxy.mic
exec.mic
extrm.mic
fadd.mic
fcos.mic
fdivs.mic
fexp.mic
fft1.mic
fft2.mic
fftpass.mic
fftlay.mic
filter.mic
finv.mic
float.mic
flog.mic
flts.mic
fltsops.mic
fmult.mic
fsqrt.mic
fsubt.mic
initchk.mic
intsrv.mic
latred.mic
ldfx.mic
ldstf.mic
logpwr.mic
moveex.mic
moverd.mic
moves.mic
multl.mic
ormod.mic
power.mic
radian.mic
randoms.mic
record.mic
removes.mic
save.mic
sched.mic
score.mic
sfft.mic
shortops.mic
smvxyd.mic
stepn.mic
stkops.mic
tsttols.mic
upchan.mic
xymadd.mic
xymoves.mic
"MACRO ANDL, XYA(DESC) DESC=XYA,N
" X(XYA-1) = X(XYA-1) .AND. Y(XYA-1) for i=0, ..., N-1

ANDL:  Y->J, X->YX, X->YCY
XH->XH, DEC J, Y->V
ANDLP: X AND V->V, INC YA
V->X, INC XH, Y->V, DEC J, 
IF J>0 GOTO ANDLP
READ, GOTO EXMAC
END

"MACRO M255TOL, DA DOF=DY->W Read 32 words and convert
"MACRO LTM255, DA DOF=DY->W D/A output

EQU DI='12'O, ANALOG='10'O, S='8225'D, SINV2=4

M255TOL: W->DA(D1)
ANALOG->DEV
'30'D->J
'1'->W
IO XOR W->V, S->YL
V->U, '160'O AND V->V
'10000'O->XL, '100000'OU:PP
'17'O AND U:V, U->, U:G
LOOP: ML:T+OCC:F->U, '16'D:V:G,
'16'D:V:H->V
SHIFT(U)V:PP;
 IF G:O GOTO NEG
 '33'D->U
MR:U->U, IO XOR W->V
L2: U:YL:FR, V->U
XL:V:PP, O->T
ML:T+F->T, '17'O AND U->V
WRITE T, U:G, O->T, DEC J,
 IF J>0 GOTO LOOP
ML:T+OCC:F->U, '16'D:V:G,
'16'D:V:H->V
SHIFT(U)V:PP;
 IF G:O GOTO NEGL
 '33'D->U
MR:U->U
VAL
L3: U:YL:FR
NOOP
ML:->U
WRITE U, GOTO OPSEQ
NEG: '33'D->U
U:MR->U, IO XOR W->V, GOTO L2
NEGL: '33'D->U
U:MR->U, GOTO L3

LTM255: W->DA(D1)
ANALOG->DEV
READ, '16'D->XL
ABS DX->U, '31'D->J
U:SINV2
'33'D->T, U:G
MR:T->V, '8'B->W, DX->T.

"select ANALOG IO
"get first value
"do loop 32 times
"2ABS(L)/S
"2*16.5,no ovf for SCLV
"V-SL, save old L
READ
SHIFT(SCLV)+V:PP,
  W-SCLV->W
XL*W:PP, ABS DX->U
MR->W, '3' D-XYL
LMLP: U*SINV2, T->U
  '3' D-XYL, W-YL:FR,
  MR->U, U: H
MR+V, '8' D-XYL, DX->T,
READ
SHIFT(SCLV)+V:PP, ML+U->V,
  W-SCLV->W
  IF H=0 GOTO NEGLM
-1 XOR V->V, XL*W:PP
ABS DX->U, MR->W
LM2: V->IO, DEC J, IF J=0 GOTO LMLP
  GOTO OPSEQ
NEGLM: '177' O XOR V->V, XL*W:PP
ABS DX->U, MR->W, GOTO LM2
END

"asline. mic"

TITLE ASLINE
EXPAND
NLIST
SYMBOL
ENTRY LIMIT, SNDCMD, SNDGIM, ASINT
EXIT GOTO, OPSEQ
EQU EA='31'0
  "DA + DBLE

"ASYNCHRONOUS LINE INTERFACE COMMANDS
DEFINITIONS FOR INTERFACE CONTROL REGISTER BITS
EQU M='77400'0, RESET='1000000'0
EQU IEN='140000'0  "INT ENABLE
EQU DATA='160000'0  "DEFAULT IS CMD
EQU RD='170000'0  "READ STATUS/DATA FROM DEV
EQU WR='174000'0  "WRITE CMD TO CMD/DATA
EQU LI='176000'0  "SELECT LINE 1
EQU LO='177000'0  "SELECT LINE 0
EQU DEN='177400'0  "ENABLE AP DATA ONTO CMD

"BAUD RATE CODES FOR LIMIT
  "0000  50
  "0001  75
  "0010  110
  "0011  135.5
  "0100  150
  "0101  200
  "0110  250
  "0111  300
  "1000  600
  "1010  1200
  "1011  1800
  "1100  2400
  "1110  3600
  "1111  4800
  "1111  7200
  "1111  9600
  "1111  19200

  "MACRO LIMIT, ARG  ARG=BAUD RATE CODES FOR LI,LO
LIMIT: '11'0-DEV, V->U, U->T  "MAKE V AVAILABLE
  XB=M-RESET:DEN+L0+L1, XB OR W->V
  "V=RESET, BAUD RATE
  "COUNT FOR PULSE WIDTH
  "START RESET PULSE
  "PREPARE RESET END
  "HOLD FOR 5 MICSEC.
  "END RESET
  V->IO,
  YB=M-REN+WR+L1+L0+116',0.
  "MODE SETUP
  "SET MODE
  "SET COUNT
  V->IO,
T+V, U->V,
  "RESTORE V
  "SETUP END WR
T->IO,
  "END WR FOR MODE
```
DEC J,
IF J>0 GOTO . "WAIT
XB=H-DEC+WBX'=L1+660, 
XB->IO, "START WB FOR CMD
W+J "SET UP COUNT
XB=H-DEC+L1+660.XB->T 
T->IO, DEC J, "END WB FOR MODE
IF J=0 GOTO "WAIT
XB=DATA+IEN, XB->T,
READ "FETCH NEXT MACRO
T->IO, EXEC MACRO "EXIT TO NEXT MACRO

* MACRO SNDCHAR, ARG=LINE CODE, U=CHAR

SNDCHR: V->W, W:H->V, "SAVE V
GOTO SND

* MACRO SNDCMD, LINECODE U=CTRL CHAR

SNDCMD: W - DATA:H -> V, "FORM CTBLS
V ->W "SAVE V
SND: '110' -> DEV, "ADDRESS LINE UNITS
V + U ->V "ADD COMMAND
V -> IO, "SEND ADDRESS
YB-DEC+WBX, YB+V ->V "PREPARE WB
V->IO, "START WB PULSE
V-WRT -> V "PREPARE WB END
READ, V->T, W->V "FETCH NEXT MACRO
T->IO, EXEC MACRO "END WB, EXIT

EQU RCVMSK='24000'0 "RCVR RDY BITS FOR L1 & L0

"MACRO ASINT, DA(LINE DESC), A(RCV ROUTINE), A(TR ROUTINE)

: DCF = DLY->DA, DBLE
: RETURNS V=A(LINE DESCRIPTOR), E=INPUT CHAR AND
: BRANCH TO THE TR OR RCV ROUTINE IF TRDY
: OR RDY IS ACTIVE. IF NO LINES READY,
: CONTINUE IN LINE.
: LINE DESCRIPTOR FORMAT:
: 0: MASK FOR TRDY AND RDY BITS FOR LINE
: 1: LINE CODE=H-DATA-Ln-Len (IF INTERRUPTED USED)
: 2: A(NEXT CHARACT PAIR IN D). CHARACTER COUNT
: 4: CURRENT CHARACTER, A(OUTPUT COMPLETE ROUTINE)
: 6: OUTPUT ACTIVE FLAG, A(OUTPUT ROUTINE)

ASINT: 2+J "NUMBER OF LINES
NEXT: DA->U, B:H->V "A(LINE DESC) ->U
DEC J, IF J=0 GOTO LOOP, "BRANCH IF MORE
READ, INC DA(V) "FETCH MASK/CODE
READ(EA) "SKIP OVER EXITS
GOTO OFSEQ "NO LINES READY

LOOP: '110'->DEV, DX->V "V=TRDY/RDY MASK
IO AND V -> V, RD->W "TRDY/RDY?
```

**ave.mic**

```
TITLE AVE
ENTRY AVE
EXIT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO AVE, X,YA(DESC) DEC = C, N
  Y = YA, r
  Y(AA+j) = Y(AA+j-1) + C *(Y(AA+j-r) - Y(AA+j)), j=1,...,N
  Y(0) is not changed
AVE:
  Y->J, X*MS:FR, INC XA, INC YA
  Y->YA, Y->YC
  Y->V, INC YA
  MA*Y:FR, INC YA(YC)
  Do Loop for j=0,...,N
  Y(AA+j+1)*C
  "new Y(AA+j)
  "new Y(AA+j-1)
  C*(Y(AA+j-2))
  J+1->J
  "exit
READ, GOTO EXMAC
END
```

**bldpop.mic**

```
TITLE BLDPNP
ENTRY BLDPNP
EXIT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO BLDPOP, CNT, YA(CHNLBLKS)+1, XA(POPTAB)
  DOP = DY->J, READ
  "Assume CHANNEL BLOCKS in Y PAD = MREC, PA, PB, PC
  "Output POPTAB in X PAD = PA, PD, PF, PE, PC, PB
BLDPNP:
  DX->YA, DY->X, DEC J
  Y->U, INC YA
  U->X, INC XA, Y+U->U, INC YA
  LOOP: U->X, INC XA, W+V->V, INC YA
        U->X, INC XA, W->U
        U->X, INC XA
        V->X, INC XA, Y->U, INC YA
        "DO CNT BLKS
        DEC J, IF J>0 GOTO LOOP
        GOTO EXMAC
"adjust count
"U=PA
"PA->X, PB->V
"PD=PA+PB->U
"PD=PA+PB+PC->U
"PB->X, PE=PB+PC->U
"PC->X
"exit
```

TITLE BLKSHIFTS
ENTRY NORMX, NORMY, SCALEXY, SCALEX, LDMA,
MULFRC, MULTERRC, MULTXC, MULTYX
EXIT EXMAC, ONESTEP
EXPAND
NOLIST
SYMBOL

"MACRO NORMX, XYA(DESC) DOP=10 DESC=YA, XA
" Computes X*2**U-XY for N X's, 0<=U<16
"MACRO MULTXC, XYA(DESC) Computes MA*X>Y
"MACRO NORMXY, XYA(DESC) DOP=10 DESC=XYA, N
" Computes (XY*2**U)-XY for N values, 0<=U<16
"MACRO MULTYC, XYA(DESC) Computes (XY*MA)-XY
"MACRO SCALEXY, XYA(DESC) DESC=XYA, N
" Computes (XY*2**U)-XY for N values -16<=U<0
" This is twice the inverse of NORMXY
"MACRO MULTFRG, XYA(DESC) Computes X*MA:FR->Y
"MACRO LDMA, ARG DOP = 0,2,4,6
" Puts ARG in MA (signed)

NORMX: SHIFT(U) *MB:PP
" Set MA-2**U
MULTXC: Y->XA, INC YA, X->U
Y->J, MA*X=P2, INC XA, DEC J,
" set first mult
U->Y, M*X=P2, INC XA,
MR->U, MA*X=P2, INC XA,
" first result
U->Y, INC YA, MR->U,
MA*X=P2, INC XA, DEC J,
" set J=0 GOTO.
IF J=0 GOTO .
READ, GOTO EXMAC

"MACRO NORMXY, XYA(DESC) DOP=10 DESC=XYA, N
" Computes (XY*2**U)-XY for N values 0<=U<16
"MACRO MULTYX, XYA(DESC) Computes (XY*MA)-XY

SCALEXY: SHIFT(U) *MB:PP
" Set up MA
MULTFRG: X->XC, X->YA, Y->J
XC->X, DEC J, MA+Y:PP,
" set up XA
MA*X=P2, INC XA, J->Y
INC YA,
INC YA,
0->U, ML:H->V, MA+Y:PP,
" UV=shifted RH
" set MA
MLMR+UV:GH->UV, MA*X=P3,
DEC XA
MFRPL: U->X, V->W, 0:F->U, M->V,
MA+Y:PP, DEC YA(2)
INC XA(3)
W->Y, MLMR+UV:GH->UV,
INC YA(2), MA*X=P2,
DEC XA, DEC J,
" shift RH(3)
IF J=0 GOTO MFRPL
READ, GOTO EXMAC, INC XA(2)
" leave XA=YA

"MACRO SCALEX, XYA(DESC) DOP=10 DESC=YA, XA
" Computes X*2**U:FR->Y for N values -16<=U<0
"MACRO MULTFRG, XYA(DESC) Computes X*MA:FR->Y

LDMA: DISABLE
" Will not serve ints
W:MB=22
READ, GOTO ONESTEP
END
READ, GOTO EXMAC, ENABLE INTS

"MACRO LDMA, ARG DOP=0,2,4,6 Puts ARG in MA (signed)
coding.mic

TITLE CODING
ENTRY ENCODE, DECODE
EXIT OPSEQ
EXPAND
NOLIST
SYMBOL

"ENCODE, CNT. YA(LIST), XA(A(CODE TABLE))
"DECODE, CNT. YA(A(CODE TABLES)), XA(CODES LIST)

EQU DI="12'0

ENCODE: DEC J, DX->YA, DY->XA
X->W
ENLP: W->DA(DI), INC XA
READ, 1->W, Y->U
DX->G, READ, 1:H->V
READ, DX->G
READ, DX->G, W+V:H->V,
IF G=0 GOTO .
V->Y, INC YA, X->M,
DEC J, IF J=0 GOTO ENLP
GOTO OPSEQ

"DESTINATION
"STORE U IN D(DA)

LDUDA W->DA(DA)
READ
DX->U, GOTO OPSEQ
END

"SOURCE ADDRESS
"FETCH D(DA) -> U

DECODE: DX->YA, DY->XA, DEC J
X->W, INC XA
Y+->W, INC YA,
MA+40000'0
W->DA(DI)
READ, X->W
DX->MB:FR, Y+->W, INC YA
READ, W->DA
DCLP: DX->MB:FR, INC XA
ML+U, READ, X->W, DEC XA(2)
ML+U+U, DX->MB:FR
Y+->W, INC YA
U->X, INC XA(2), READ, W->DA,
DEC J, IF J=0 GOTO DCLP
GOTO OPSEQ
END

DAOPS.mic

TITLE DAOPS
ENTRY STUDA, LDUDA
INTRLV
EXIT OPSEQ
EQU DA=10

STUDA W->DA(DA)
WRITE U, GOTO OPSEQ

"SOURCE ADDRESS
"FETCH D(DA) -> U
**DECIM** - FIT DECIMATE ROUTINE

*MACRO DECIM, D(DESC) DOP=0,2,4,6

* Assume a descriptor: COUNT, DA(DATA)
* SCALE, L
* X, RB in D
* L.E. and RB are filled in by FIT1

**SUBROUTINE SIZECHK.** Checks V, U=A(DESC)
* If V>2**13, shifts data right by 1 or 2 and adjusts SCALE

**EQU DI='12'0**

```
DECIM:  W->DA(DI:D)
READ, U->V
W->V
DEC J, FROM XA, DCSV+J->U,

LOOP:  MB:V:H->V, U->XA, U->Y,
      DY->V, ABS XB->U
      if J>0 goto LOOP
      V=COUNT, F=A(V
      "X=COUNT, F=A(DATA)
      U->W. V->J
      T=XC. T->YR, MR->V
      M=2**(16-r)
      DEC J, XC->XA, DOM+Y->U,
      READ

       Inc ptr
      "next dest, load real
      "load imaginary
      "size real
      "size imaginary
      "decrement ptr
      "fetch next pair
      "V=Max data, exit

      "V=SIZE
      "SCLV=SIZE-2
      "set to read DESC, shift # is 1 more
      "shift needed
      "set RA for BS
      "point at data
      "XL=X(0), get count
      "dec for file
      "V=COUNT
      "V=COUNT
      "shift X(0)
      "new scale
      "shift Y(0)
      "shift X(1), U=New X(0)
      "store X(j), U=Y(j)
      "shift Y(j+1)
      "store Y(j+1)
      "store X(j+1)
```

TITLE DIVBYY
ENTRY DIVBYY
EXPAND
Nolist
SYMBOL

"MACRO DIVBYY, YA DOP=DX->YA
" Computes U/Y to 32 bit accuracy in UV
" 0 <= U <= Y

DIVBYY: '31'D->J, O->W, O->V
U->U, U->T
LOOP: VM+VW+LOC0:GH->VM
IF G<0 GOTO Z,
U+U>U
NEXT: DEC J, IF J>0 GOTO LOOP,
U->U, U->T
1->T
IF G<0 GOTO INC, READ
V->U, W->V, EXEC MACRO
"no, exit
INC: VT+GW:GH->UV, EXEC MACRO
Z: T+T=T+U, GOTO NEXT
"double rem
END

"CMD MACRO DINT, INT8 - push int routine on stk if pending
EQU STK='10'0, DI='12'0

DINT: W->DEV, W->DA(DI)
CLR SRIT
IF STAT=0 GOTO OPSEQ, READ
WRITE (STK) DX
GOTO OPSEQ
END

TITLE DINT
ENTRY DINT
EXIT OPSEQ
EXPAND
Nolist
SYMBOL

"int, vector
"clear S if set
"exit if not set
"push routine on STK
"done
**TITLE DOP**
**ENTRY EXMAC**
**EXPAND**
**NOLIST**
**SYMBOL**

EQU DI='12'0, EA='31'0

*These ops start at location 0*

DOP0: DX→W, DX→T

"D'(ARG1)→W,T"

DOP1: DX→W, DX→T, READ

"D'(ARG1)→W.T, next word"

DOP2: DY→W, DX→T

"ARG1→W.T"

DOP3: DX→T, DY→W, READ

"ARG1→W.T, next word"

DOP4: DY→XA, XA→XC, if J>0 PSB=XW, CONT

"select XA, branch"

DOP5: DY→XA, XA→XC, if J>0 PSB=XMR, CONT

"select XA, branch"

DOP6: DX→YA, YA→YC, if J>0 PSB=YM, CONT

"set YA, branch"

DOP7: DX→YA, YA→YC, if J>0 PSB=YMR, CONT

"set YA, branch"

DOP10: DX→YA, DY→XA

"set XYA=ARG"

DOP11: DX→YA, DY→XA, READ

"set XYA=ARG, next word"

DOP12: DY→XA

"set XA"

DOP13: DY→XA, READ

"set XA, next word"

DOP14: DX→YA

"set YA"

DOP15: DX→YA, READ

"set YA, next word"

DOP16: DY→J

"ARG1→J"

DOP17: DY→J, READ

"ARG1→J, next word"

DOP20: NOOP

DOP21: NOOP

DOP22: DY→DA(D1)

"ARG1→FILE"

DOP23: DY→DA(D1:D)

"ARG1→FILE, DBLE"

DOP24: DY→U→W

"ARG1→U→W"

DOP25: DY→U→W, READ

"ARG1→U→W, next word"

DOP26: DY→V→W

"ARG1→V→W"

DOP27: DY→V→W, READ

"ARG1→V→W, next word"

DOP30: NOOP

DOP31: NOOP

DOP32: NOOP

DOP33: NOOP

DOP34: NOOP

DOP35: NOOP

*PS→D and D→PS routines*

*These can only be reached via DOP or IF J>0 PSB=

DOP36: DY→W, READ, if J>0 PSB=RDPS, CONT

"CNT-1→W"

DOP37: DY→J, READ, CONT

"GOTO RDPS"

LOOP: YB=DX, YB=DY, READ, PSWMT

"DY→PSV"

X8=DX, YB=DX, DEC J, PSWMT

"DXXY→PSL"

YB=DX, YB=DY, DEC J, PSWMT

"DXXY→PSR, DEC CNT"

IF J>0 PSB=LOOP, CONT, READ

"DONE?"

EXEC MACRO

"next macro"
dotxy.mic

TITLE DOTXY
ENTRY DOTXY, DOTX
EXIT EXMAC
EXPAND
NOLIST
SYMBOL

**MACRO DOTXY, X,Y (DESC) DOP=LDXYA**  
`DESC` -> `X, N` `X, Y`
`BA, FA`

**MACRO DOTX, X,Y (DESC) DOP=LDXYA**  
`DESC` -> `X, N` `X, Y`
`BA, FA`

DOTXY:  
`Y -> J, INC XA, INC YA, 0 -> U, 0:H -> V, 0:O`
`Y -> XA, X -> YA`

LOOP:  
`MLMR->UV:GH->UV, X*Y, INC XA, INC YA, DEC J, IF J>0 GOTO LOOP`
`MLMR->UV:GH->UV, READ, GOTO EXMAC`

DOTX:  
`Y -> J, INC XA, INC YA, 0 -> U, 0:H -> V, 0:O`
`Y -> XA`

DOTXL:  
`MLMR->UV:GH->UV, X*X, INC XA, DEC J, IF J>0 GOTO DOTXL`
`MLMR->UV:GH->UV, READ, GOTO EXMAC`

END

exec.mic

TITLE EXEC
ENTRY RTN, EXEC, END, HOSTI, RUN, INTXM
EXIT GOTO, IN1, CALL, SETP7
EXPAND
NOLIST
SYMBOL

**MACRO EXEC, XXX DOP=NOOP**
**MACRO RTN, -1 DOP=DY->W**
**MACRO END, -1 DOP=DY->W**
**MACRO HOSTI, PORT DOP=0,2,4,6**
**MACRO INTRIM, -1 DOP=0,2,4 Enable interrupts and RTN**

EQU CMD='70, STK='10'0, EA='31'0, HST='12'0

EXEC:  
`CMD->DEV`  
"select CMD port"
`DA->W`
`CLR SBIT, GOTO CALL`
`RTN: READ(STK), INC DA(M)`
`READ, INC DA(0)`
`DEC STK PTR`  
"fetch address"
`D(STK)->W`  
"back up STK"
`END: READ(STK), INC DA(M)`
`READ, INC DA(0)`
`DX->W, YS=DA(EA)`
`FA=EA`  
"fetch old EA"
`W->DA, GOTO IN1`  
"restore EA"

RUN:  
`CALL SETP7, ENABLE`
`DY->W, GOTO RTN`
`resume execution`

HOSTI:  
`HST->DEV`  
"address HST"
`NOOP`
`wait for status valid`
`IF STAT=1 GOTO, W->DEV, READ (EA)`  
"wait for ATN CLR"
`INT HOST, EXEC MACRO`  
"set PORT"

INTXM:  
`ENABLE, GOTO RTN`  
"send Interrupt"
MACRO EXTREM XA(DESC) Desc=N,R
" YA(output), XA(input)
" Searches a list of N-1 points in X for extreme
" values. For each extreme found, the value of
" the point and its biased index are recorded in Y.
" Neither end point of the list can be an extreme.
" The YA of the next open cell in the list is
" recorded at the beginning of the list.
" If R extremes are found, the search terminates
" and the next macro doubleword is executed.
" If less than R extremes are found, the next
" macro doubleword is skipped.

EXTREM: X->W, Y->J, INC XA, INC YA
X->YA, Y->YA, DEC J
I+1, YA->YC, X->U, INC YA,
INC XA
X:UG
U:X:G
IF G-O GOTO DNO, W-MR->W,
X->U, INC XA
X:UG, GOTO UP1
UP0: IF G-O GOTO MAX, X:UG:
UP1: IF H-O GOTO UP0, W-MR->W,
U->Y, X->U, INC XA
YA->YC, YC->YA, READ,
GOTO EXIT
MAX: XA->XC, INC YA
XC->V, W:W
V->Y, INC YA, U:XC,
DEC J, IF J>0 GOTO DNI
YA->YC, YC->YA, GOTO EXIT
DNO: IF G-O GOTO MIN, U:X:G:
DNI: IF H-O GOTO DNO, W-MR->W,
U->Y, X->U, INC XA
YA->YC, YC->YA, READ,
GOTO EXIT
MIN: XA->XC, INC YA
XC->V, W:W
V->Y, INC YA, U:XC,
DEC J, IF J>0 GOTO UP1
YA->YC, YC->YA
EXIT: YC->Y, READ, GOTO EXMAC
END

MICRO SUBROUTINE FADD
" Assumes first number has its mantissa in UV, scale in W
" Second number is in Y with its scale in Y(YA), ML in Y(YA+1),
" MR in Y(YA+2)
" Returns with Y and YA unchanged

FADD: Y->W, INC YA, Y->U, U->T
XB*Y:PP, XB=1, W:0,H, INC YA
"MB=ML2, test for S1>S2
SHIFT(M) MB:PP, O->W,W,
"shift(ML2), S1>S2->W
IF H<0 GOTO SY
SHIFT(W) V:PP, U:0, H->W,
"shift(ML1), SL2>W
IF H=0 GOTO NOSY, MR->U
IF SHIFT SMALL GOTO SVU
MA+T:P2, Y->V
IF SHIFT VF GOTO FLOAT2,
Y->U, U:0, V->Y, V->T,
DEC YA(2)
GASN+T=GCO:F->U, ML:V:G->V,
GOTO TUV
"add shifted ML to M2
NOSY: Y->HCO, U->V, H->V,
DEC YA(2), GOTO FLOAT2
SY: IF SHIFT SMALL GOTO NORM,
"shift ML2
MA+P2, U:0,W
IF SHIFT VF GOTO FLOAT2,
T:G->U, V:V, DEC YA(2)
GASN+T=GCO:F->U, ML:V:G->V
"add shifted ML2 to M1
TUV: U:G, V:H, GOTO FLOAT2
NORM: T:HCO->U, ML:V:H->V
"test mantissa for zero
ADD2: ML:MR+UV->UV, GOTO FLOAT2,
DEC YA(2)
END
TITLE FCOS
ENTRY FCOS, FSIN
EXT RADIAN, AQUAD, QQUAD,quaD, QQUAD
EXPAND
Molist
SYMBOL
EQU TWCODP=\'1772\'0

FCOS: \'20\'W:W->W, 0->T, V->X, INC XA
U:G, V:H, TWCODP->RA, U->X, DEC XA
W->Y, W\'-36\'0:U
IF GH=0 GOTO ONE
IF H=0 GOTO TOOBIG
CALL RADIAN

QTEST: \'00001\'0 AND U
\'00002\'0 AND U, T->U
IF H=0 GOTO EVEN, DEC XA(2), DEC YA(2)
IF H=0 GOTO QQUAD
GOTO AQUAD

EVEN: IF H=0 GOTO EQUAD
GOTO AQUAD

ONE: L->W, 0->V
\'40000\'0->U
GOTO FINISH

FSIN: \'20\'W:W->W, 0->T, V->X, INC XA
U:G, V:H, TWCODP->RA, U->X, DEC XA
W->Y, W\'-36\'0:U
IF GH=0 GOTO ZERO
IF H=0 GOTO TOOBIG
CALL RADIAN
\'30\'0->U
GOTO QTEST
ZERO: \'100000\'0->W
GOTO FINISH

TOOBIG: YB->U, YB->X, YB->W, DEC YA,
YB=\'77777\'0
FINISH: RTN
END

TITLE FDIVS
ENTRY FDIVS, SFLATA2
LISTOBJ
SYMBOL

\"MICRO Subroutine SDIV\"
\"Computes X/Y / U/W , U>0\"
\"Assumes a 128 word table of inverse values:\n\"TAB(1) = 128/(1.128) in ROM\"
\"MICRO Subroutine SflATA2\"
\"Assumes test for U=0 performed in G adder\n
EQU INVTAB=\'1000\'0

FDIVS: U='256'D:FR
Y=W->W, XB->T,
X=INVTAB-'128'D
ML->V, 1=W->W
'128'DV:FP, ML->T->V
V->RA, INVTAB:V:G
RL*RL:FR, U->MR:U
R=U-U, G->T,
IF G=0 GOTO SS
D close to 1.0?
T=+T:VAM
T+T:FR
RL->U
U=ML->U, 0:H->V, GOTO DIVZ
T+T=+VAM
SS: IF G=0 GOTO ONE
GOTO DIVZ
D=\'1\'0?
T=+T:VAM
T+T:FR
RL->U
ML->U, 0:H->V, GOTO SFLATA2
ONE: X->U, 0:H->V, RTN
FINISH: RTN
END

\"MICRO SUBROUTINE SFLATA2\"
\"Assumes test for u=0 performed in G adder\n
SFLATA2: ABS U->V, MA:U:P2,
\'100000\'0->T
IF G=0 GOTO Z, W:SCLV->W,
SHIFT(SCLV)*:HB:P2
IF SHIFT SMALL GOTO FIN, T->U
'40'D->U, ML XOR U->V
U+W->N, V->U, RTN
FIN: MR->U, RTN
Z: T->W, D->U, RTN
END

\"ABS(result)\n\"result D
\"branch if no overflow
\"correct mantissa sign
\"correct scale, exit
\"normalized mantissa
\"true zero
"MICRO SUBROUTINE FEXP Floating point exponential

* Input data is in u/v
* Find exp(n) = exp((2**n)*H) = (2**SCL)*MAN

"Uses two tables in ROM:
* the first is 32 bits = 1/(2in2), with point at boundary
* the second is 45 words = EXPTB(n) = exp(n/64), n=0.44

EQU BASE=1310'd, LNZ='1301'd, INV=136'd

FEXP: INVL3>RA
RR+V:PP, O->T, W='15'O:H
RLMB:PP, U:G, 2W=H->W
RR+U:PP, ML->U, W,H,
IF H=O GOTO OVERFL
MLMR+OU:GH->UV, RL=MB:PP, LNZ->Y
MLMR+UV:CH->UV, Y->RA,
GASN+T:OCC,F->T,
IF H=O GOTO RSHIFT
MLMR+TU:FG->TU, SHIFT(W)V:PP
MA+U:PP, '77777'0->V
MA+P:3, ML+HCG:O->V, MB+V:H,
MLMR+UV:CH->UV, O->T, X=XP:PP,
X=XL, X='4000'O
MLMR+TU:FG->TU
CONT: ML+T:Y, RB+V:PP,
'200'O->YL
RR+U:PP, RR+V, V:H->W
OML-RL:GH->UX, XB=BASE. X=->T.
MLMR+OU:GH->UV, W->Y
MLMR+UV:CH->UV, '4000'O->W
U+YL:2P, U AND W, '7777'O->W
U AND W->V, V=YL, V->W,
XL+XP:PP
T+ML:G->U, O->T,
IF H=O GOTO EPSL
U+ML:G->U, V-'1000'O'H->V
"store scl->YO
"uv*ln2+(FRAC-1)
"u*64, round?
"mask 7 lead O
"ER->VL,E=W
"TBASE(-N)-U
"round n*1/u,E=1/64->v

EPSL: V+YL:2P, V=XL
U+RA, '7777'O->U
MLMR+OU:GH->UV, MA+XL:22,
GASN+T:OCC,F->T
GASN+T:OCC,F->U, ML+HCG:O->V,
MLMR+UV:CH->UV, MA+XL:22,
"2*(EL)X->UV
"E=2->UV
"EL/3
"XBO+TMLX:GH->UV, V=YL,
"begin 1/3
"(E=2)/3R->YL
TITLE FFT1
ENTRY FFT1
EXT OPSEQ, SIZECHK
EXPAND
NOLIST
SYMBOL

"MICRO FFT1, DA(DESC) DOP=0,2,4,6
EQU DI='12'0

FTP1: W->U, CALL SIZECHK
"shift data if needed
"count=0.5
DX='1000000'0:PP
READ, INC DA(0)
WRITE DXYB, YB=1, ML->Y
WRITE VYB, YB='152'0
V->J, CLR XA, CLR YA
Y:0->V, INC YA, INC XA,
'1000000'0+MB, 0->W, DEC J
L:
Y:V->U, MA*X:FR, V:Y:U->V,
DEC YA, DEC XA
U->Y, ABS YB->U, INC YA,
X->T, INC XA
V->Y, ABS YB->V, INC YA,
U OR W->W, ML+T->U
V OR W->W, U->X,
ABS XB->U, DEC XA,
T:ML->T
U OR W->W, T->X,
ABS XB->U, INC XA(3)
U OR W->W, INC XA, INC YA,
Y->V, DEC J,
IF J=0 GOTO L
W->V, GOTO OPSEQ
END

TITLE FFT2
ENTRY FFT2
EXT FFTPASS, SIZECHK, OPSEQ
EXPAND
NOLIST
SYMBOL

"MACRO FFT2, DA(DESC) DOP=0,2,4,6
" Repeated step of FFT does FFTPASS log(COUNT)-1 times
" Upon exit, SCALE word in DESCRIPTOR is updated.
" The fft result is in DATA PAD starting at 0
EQU EA='31'0, DI='12'0

FTP2: W->U, CALL SIZECHK
"size data, U=A(DESC)
READ, '1000000'0->XL
DX->T, TY->V, READ
XL*DX:PP, V*V->V
MA*DY:PP, V->XC, V->YA
ML->U, READ, INC DA(-4),
XC->XA, YA->YC
WRITE TV, ML->V
WRITE UV, IF G=0 GOTO OPSEQ
YC->J, U:1->U
U->DA('14'0)
V->RC, 0->V, 0->W,
"save in F(14)
"set RC
"last pass?
LP:
V->RA, CALL FFTPASS
DA('14'0)->U, INC XA(XC),
INC YA(YC-1)
U:1->U, YC->J
U->DA, 0->V,
"New X->F(14)
"do next group
"get EA
"back up
V->DA, W:->V, GOTO OPSEQ
LAST: V->RA, CALL FFTPASS
W->V, GOTO OPSEQ
END
"SUBROUTINE FFTPASS"
"W = SIZE DATA"
"XA = A(X), XC = L, YA = A(Y), YC = L"
"J-L, RA-O, RC = 512/L,"
"Assumes a 512 doubleword table in ROM of roots of unity:"
"RL(1) = - COS(Z2pI/1024)"
"RR(1) = - SIN(Z2pI/1024)"

"FTTPASS: RL*FR, '77777'0->YL,"
"DEC XA(XC),"
"X*YL:FR, 0->T, INC XA(XC),"
"DEC J,"
"RR*FR, T->ML->T, Y->YL,"
"RL*FR, T->ML->T, Y->YL,"
"DEC XA(XC),"
"O->T, RL*YL:FR,"
"V->X->Y->X, DEC YA(YC-1),"
"DEC XA(XC),"
"U->X, ABS U->U, INC XA(XC),"
"ML:U->T, INC RA,"
"RL*YL:FR,"
"V->X, ABS V->V, INC XA,"
"U OR W->W, T->ML->T,"
"V OR W->W, ML:U->T,"
"DEC XA(XC),"
"U->Y, ABS U->U, INC YA(YC),"
"ML:*FR, 0->T,"
"INC XA(XC),"
"V->Y, ABS V->V, INC YA(YC),"
"V OR W->W, ML:U->T,"
"DEC J, IF J=0 GOTO LOOP"
"U=X1-COS*Y2-SIN*Y2"
V=ML->V, W=ML->W, X->T,
1
XA->XC, XC->XA,
X*YL:FR
U->X, DEC XA, V->Y, INC YA,
INC RA, DEC J,
IF J>0 GOTO LOOP
GOTO OPSEQ

"V=Y1', W=Z1'
"T=x1r, -1.0*x1r
"store X2', Y1'
"exit, K=N-K

TITLE FILTER
ENTRY FILTER
EXIT OPSEQ
EXPAND
NOLIST
SYMBOL

"MACRO FILTER, XA(DESC) DESC = N, XA(SOURCE)
"-
"Also have filter coefficients and filter memory
"at XA = DESC = -m stored as:
" XA-1 XM, BM-1
" XA-2 XM, BM-2
"... ...
"XA-M XM, BO
"N and XA are destroyed, the Bj are updated.
"The source is replaced by the filter output.
"F=SRCEj
"for m:XM-1->0, F=F-XM-1*BM
"  Bm+1=BM*XM+1*F
"Output = F, BO = F

FILTER: Y->XA, 1->V
NEXT: X->T, XA->XC, DY->YJ, DY->XA
X:V=H->V, U->Y, YC->YA
V->X, DEC XA, DEC YA, 3->YC
IF H=O GOTO OPSEQ, X:Y:FR,
"done?
X:Y=XL, Y->W, DEC XA,
INC YA
Y->J, DEC YA(YC)
T:ML->T, W:V->V, DEC J,
X:Y:FR, Y->W, DEC XA,
INC YA
XL+T:FR, DEC J
LOOP: X:ML->T, DEC YA(YC),
X*MB, INC XA
ML+V->U, W:V->V, X:YL, DEC XA(YC),
MA+Y:FR,
Y->W, INC YA(YC)
U->Y, DEC YA(YC), XL+T:FR,
DEC J, IF J>0 GOTO LOOP
"set MA=KM-1
"set BM+1=BM+KM+1*F
"set MB=BM+KM+1*F
"set XM=XM+1*F
"set Y=U
"store BM+1
"store 0M-1
"store BO, U=BM-1
W:V->V, T:ML->V, INC XA,
GOTO NEXT
END
TITLE FINV
ENTRY FINV
EXPAND
NOLIST
SYMBOL

"MICRO SUBROUTINE FINV"
"Computes INV(U/V), ABS(U) >= .5"
"Assumes a 128 table of integer values:
"TAB(i) = 128/(1+128), only 16 bit values are needed"

"Algorithm for INVERSE(D)"
"Let D=OD1/256 + EPS such that
128 <= D <= 256 and -1/512 <= EPS <= 1/512"
"Look up TVAL(OD1-128) = OD1/128 (16 bit accuracy)
Compute .SID1 = TVAL+2EPS*PRAV+TVAL (16 bit accuracy)
Then D*ID1 = 1/E and ABS(E) < 2*16.
Compute .SID2 = .SID1+(2*ID1*ID1), (16x32 bit multiplies)
The result, ID2, satisfies D*ID2 = 1/E*2
The output scale is W = 1/W.

EQU INVTAB='1000'O

FINV:
U4256'D:FR, U->T, U->XL
UG
W:YL, V->W, XB:INVTAB1128'D,
XB:V
IF G=0 GOTO NEG, ML->U
'128'D+U, ML:V->H->V,
T:G
V->RA, 1->V
RL:FR, IF G=0 GOTO OF
T:G, T:FR
'40000'O->T, U:V->H, ML:R->U
IF G=0 GOTO DIVE
U:ML->U, W:H
'77777'O->V->U, W:H
DIVE:
UW:3P
MA:XL:22, IF H=0 GOTO Z
00:MR:GH->UV.
T:GASNV1000GIF-T
C:
TU:MLR:GH->UY, MA:V:2P,
O->T, '177777'O->W
MA:V, 2P, '77777'O->V
MA:U, 22, T:R->U,
MLR:GW:GH->VW,
'8'D->XL
MLR:UV:GH->OY,
GASNV:T:GCO:F->T,
1:YL, 1->W
MLR:TU:FC->TU, XL:V:PP,
IF G=0 GOTO CORRECT
MA:U:PP, T10000'O:G;

"IN ROM"

W=MR->W
FIN:
MLR:UV:GH->UY,
IF G=0 GOTO EXIT
'140000'O->U
1W->W, O->V
EXIT:
MA:U-2:U, RTX
NEG:
'128'D-2:U, V:ML->V
V->RA, 1->V
RL:RL:FR, RL:G,
OOTO COM
Z: 0->U, 0->V, GOTO C
CORRECT: TU:GASNV:FG->TU,
W:MR->W
MA:U:PP, T:G,
OOTO FIN
OF: 0->V, U-V->W, RTN
END

"check for 1.0.1-SCALE"
"shift top"
"jump if not 1.0"
"will have mantissa"
"Inc SCL to compensate"
"Get top, exit"
"shift index back, RA(-TVAL)"
"fetch -TVAL"
"compute TVAL*TVAL"
"test for TVAL-=1"
"TV=1.5"
"DEC TU"
"1-SCALE"
"shift middle part"
"set G NEG"
"make special 0"
TITLE FLOAT
ENTRY FLOAT, FLOAT2
EXPAND
NOLIST
SYMBOL

"MICRO SUBROUTINE FLOAT, UV = INTEGER, result scale in W
"MICRO SUBROUTINE FLOAT2, UV/W an UNNORMALIZED NUMBER
" Must have tested mantissa for O

FLOAT2: ABS U->V, MA->V:PP
     IF GH=0 GOTO Z, W->SCLV->W, 
     SHIFT(SCLV)*HB:PP 
     IF SHIFT SMALL GOTO FIN, 
     MA=U:PP 
     IF SHIFT OVEL GOTO BIG, 
     MLMR->UV, O->T
     MR->U, '16D+M'->W 
     U:G, V:H, GOTO FLOAT2 
FIN: MLLR=UV, GOTO EXIT
Z: O->U, O:H->V, '10000000'-W
EXIT: MR->U, RTN
BIG: MLMR=UV, PG->UV, 
     '48D+M'->W
     '10000000' XOR U->W, W->U 
     U:W, W->U, RTN
END

"Scale for int.test val
"ABS(U)-V, save V
"value=0
"adjust scale
"float 0-15 places 
"shift MS 
"overflow?
"shifted MS
"shifted 16+ places
"float more
"least part
"make FP0
"Normalized MS
"down shifted mantissa
"correct scale
"invert sign bit 
"put in place

"first multiplier
"test for M=0
"save SCL in YL 
A: U:T,G, XB=T, XB=THRESH, 
    W->YL, O->W
    M: 75
U:T,G, U:H, IF G=0 GOTO T1 
    WW+UV=GH+UV, XB=T, XB=THRESH 
    IF H=0 GOTO A, T:G 
    O>T->U, T->W, GOTO FLOAT2 
TI: IF G=0 GOTO T1, T:G, 
    RLV=PP, INC RA 
    MLMR=UP, RR->YCY 
    MA=U:PP, RR->YCH 
    XL=INVBX+128'D 
    XB=INVZ (ZCY) 
    MLMR=GH+UV 
    U->RA, O->T, XB=U+U, 
    XB=LOGTAB-INVZ '64'D 
    RLV=PP 
    RR=PP, '77777777'-W, O->T 
    RL=MB:PP, MLMR=OV:GH+UW 
    T:G=I:U, U->RA, 
    MLMR=WX:GH+V 
    MLMR=UV:GH+UV 
    UV=128D 
    UV=256D 
"Computes d/d/2 
U:V:PP, U->T, V->U 
    MA*PPP, '37777777'-W, O->V 
    OML=V:GH=VV 
    WM=V:WH+VW 
    GL=V:H, O->V, MR=WH, 
    '77777777'-W
"Computes 4d/3, 3, and d**3/3
24:PP 
    MA*256'D:PP 
    UMB:PP, O->T, RR->W,
"U=64D*3
"D-L**2/2
"64D**3/192
"UV-D**2/2
"D-D**2/2*D**3/3
"start LN(2)R'S
"add TLOG.fetchLN(2)
"sub LN from it
"LN(2)(C)-L(2)
"start LN(2)'S
"S*LN(2)R-LN(C)
"FLOG+2**31
"start shift
"DBCM->W
"shift <-15?
"shift bottom
"SCL'1
"middle shifted
"bottom shifted
"top shifted
"shift top
"middle shifted
"bottom shifted
"top shifted
"result ->U
" Assume OP1 mantissa in U, SCL in W;
" OP2 mantissa in X, SCL in Y

"MICRO SUBROUTINE FADDS
FADDS: Y-W:V->V, 0->T
SHIFT(V)*X:P2, T-V:G->V
IF H-O GOTO SSV, T-V:G,
SHIFT(V)*U:P2
SUM: IF G-O GOTO SFLOAT2, U:H->V,
ML-U->U
X+V->U, GOTO SFLOAT2
SUV: Y->W, X->U, GOTO SUM

"MICRO SUBROUTINE FSUBIS
FSUBIS: Y-W:V->V, 0->T
SHIFT(V)*X:P2, T-V:V
IF H-O GOTO SSV,
SHIFT(V)*U:P2
IF G-O GOTO SFLOAT2, U:H->V,
ML-U->U
X-V->U, GOTO SFLOAT2
SSUV: Y->W, X->U
U-ML-U, GOTO SFLOAT2

"MICRO SUBROUTINE FMULTS
FMULTS: X:U:FR
Y+U->W
"SCL1:SCL->SCL
"result ->U
TITLE FMULT
ENTRY FMULT
EXT FLOAD2
EXPAND
NOLIST
SYMBOL

"MICRO SUBROUTINE FMULT  ML1,MR1 in UV, SCL1 in W
ML2,MR2 in XY(XYA-1), SCL2 in Y(XYA)

FMULT:  Y+M->W, DEC XA, DEC YA
       X*V:2P, 0->T
       Y*U:2P, 0->U
       X*MB:22, INC XA, INC YA,
       GASML+TU:FG->TU,
       1+M->W
       GASML+TU:FG->UV
       MLMR:UV:GH->UV,
END    GOTO FLOAT2
       add MR2*ML1->UV
       add ML1*MR1, float

"SHORT FLOATING POINT OPERATIONS. All operations assume
the first operand is in U/V, U=Mantissa, V=Scale.
The second operand is in XY at the location given
by the argument. X=Mantissa, Y=Scale

"MACRO SADD, ARG DOP=0,1,4  OP2 at XY(ARG)
" Computes X/Y = U/V -> U/V
SADD: T->YA, W->XA, V->W,
      CALL FADDS
      "do FADD
      W->V, READ, GOTO EXMAC
      "exit

"MACRO SSUBT, ARG DOP=0,1,4  OP2 at XY(ARG)
" Computes X/Y - U/V -> U/V
SSUBT: T->YA, W->XA, V->W,
       CALL FSUBTS
       "do FSUBT
       W->V, READ, GOTO EXMAC
       "exit

"MACRO SMULT, ARG DOP=0,1,4  OP2 at XY(ARG)
" Computes X/Y * U/V -> U/V
SMULT: T->YA, W->XA, V->W,
       CALL FMULTS
       "do FMULT
       W->V, READ, GOTO EXMAC
       "exit
"MICRO SUBROUTINE FSQRT Floating Square Root"
"Input data is in UV/W"
"Uses two tables in ROM:
"the first is 128 words = SQRTab(n)=sqrt(n/256), n=128, 255
"the second is 128 words = INVTAB(n)=128/n, n=128, 255
"If M is negative, zero is returned in UV, and the original
"scale in W"

EQU SQRBASE='1367'0, IBASE='1000'0, M128='200'0

FSQRT: U*1000'0:2P, U:O
V->U, W->V, '177'0 AND U->V
IF G>0 GOTO CONT, ML->V, V->T
0->U, 0->V, RTN
CONT: O->T, ML->W
YB=W-W, YB=IBASE-N128
W->RA, XW=W-W, XB=SQRBASE-IBASE
RRU+PP, '777777'0->V
RLV:PP, '777777'0->V
T+CCO:F->T, MLMR+UV:GH->UV,
MA+PP
OML+TU:HC0:FO->TU, MR+V:H,
W->RA
MLMR+TU:FG->UV, O->W,
'100000'0->T
UV+UV:GH->UV
TW+UV:GH->UV, U->X,
V->Y, U->X,
UV:PP
O->T, '777777'0->W
MLMR+O:GH->UV, XL*XL:PP,
U->XL, V->YL
OML+TU:HC0:FO->TU, MR+V:H
MLMR+TU:FG->UV
U:Y+PP, '40000'0->T
XLV:PP, '777777'0->W
MLMR+O:GH->UV, MA+PP
OML+TU:HC0:FO->TU, MR+V:H
MLMR+TU:FG->UV, O->W,
'100000'0->T
UV+UV:GH->UV
XY+UV:GH->UV, DEC YA
RR+PP, 0->T, '177777'0->W
"move u 9 left
"mask 9 lead 0
"H=0?
"H=0, 0->UV
"W=W
"E=UV
"fetch ITAB
"begin ITAB*E=D
"fetch SQRBASE
"D/2->UV
"double it
"(1-D)->UV
"DL->X, XL, DR->Y
"DL*DR
"(1-D)L->XL
"(1-D)R->YL
"2*DL*DR->TU
"D*D/2->UV
"(D*D/2)+(1-D)
"D*D/2+(1-D)/2-1/2
"-UV
"D*D/2+(1-D)-1
"1-D*D/2+(1-D)
"SQRRT*(D term)
"begin S/2
fsub.mic

TITLE FSUB
ENTRY FSUB
EXT FLOAT2
EXPAND
NOLIST
SYMBOL

"MICRO SUBROUTINE FSUB"
"Assume one operand in U, V, W = ML, MR, SCL"
"Second operand in XY PAD, current X-Y = --, SCL"
"Computes second operand - first operand"

FSUB:  Y-W->W, DEC XA, DEC YA, INC Y
        Y->YL, 0->T
        IF H=0 GOTO SUI, T- W->W, SHIFT(W)*YL:PP
        IF SHIFT SMALL GOTO NORM, MA*X:X, MR+W->W
        IF SHIFT OVEL GOTO FLOAT2, U:V, V:U, INC XA, INC YA
        GASNLG:TV:FQ->UV.
        V:U, GOTO FLOAT2
        NORM: IF H=0 GOTO NOSH.
        IF Y-W->T:UV:GH->UV
        INC YA, GOTO FLOAT2
        NOSH: XY-UV:GH->UV, INC XA, INC YA, GOTO FLOAT2

"SCL2 > SCL1, will shift MIR1
SCL1: SHI T(W)*Y:PP, MR+W
IF SHIFT SMALL GOTO NS,
MA*-X=P, Y->V
IF SHIFT OVEL GOTO FLOAT2, U:V, V:U, INC XA, INC YA
TU:GASNLG:TV:GH->UV.
V:U, GOTO FLOAT2
NS: XY-ML:GH->UV, INC XA, INC YA, GOTO FLOAT2
END

initchk.mic

TITLE INITCHK
ENTRY INIT, IN1, CMD, SETP7
EXT EXMAC, GOTO
EXPAND
NOLIST
SYMBOL

"INIT is the initialization routine to be burned into ROM for CHI-5"

EQU
CMD='7'0,
GMBS='30'0,
DI='32'0,
STK=0, "(really 10)
EA='31'0,
STBS='12'0,
HST='12'0,
DIN=0,
DOUT=1,
PPLSW='17'0,
MACBASE='140000'0

CHECK: PNLSW->DEV
        MACBASE->M
        IF STAT=0 GOTO IN1
        GOTO GOTO, ENABLE:
        LOC CHECK='10'0
INIT: GOTO START, D- V, CLR S
START: V-DA(DIM)
        V-DA(DOUT)
        STBS->M
        W-DA(STK), GOTO CHECK
INIT: CALL SETP7
BRSC: IF STAT=0 GOTO BRSC
CMD: GNBS- W, DISABLE
        W-DA(DI)
        READ, GOTO EXMAC
SETP7: HST->DEV
        GNBS->W
        W-DA(CMD)
        IF STAT=1 GOTO .
        CMD->DEV
        INT HOST, RTN
END
INTSRV -- INTERRUPT HANDLER, CALL, GOTO

EQU CMD='7'0, HST='12'0, STK='10'0, EA='31'0, DI='12'0

INTSRV: GOTO FLIH, DA->W
FLIH: CMD->DEV, DISABLE
W->2->W
W->DA, O->W,
IF STAT=1 GOTO CMD
1->T
W->DEV, W->DA(DI), T+W->W
IF STAT=0 GOTO INTLOOP, READ
DX->W, GOTO CALL

"MACRO CALL, EA DOP is DY->W or D(DX)->W for indirect"

"set PSA, get EA"
"CMD first"
"back up EA"
"Restore file"
"CMD INT?"

INTLOOP: CLR SBIT, W->DEV, T+W->W
O->S(j), DEV=j+1, W=j+1
IF STAT=1 GOTO INTLOOP, READ
W=EA(INI routine)

"MACRO GOTO, EA DOP is DY->W or D(DX)->W for indirect"

CALL: DA(EA)->U, U->T
WHITE (STK) U, T->U
EA->STK

"set new EA"
"fetch next MACRO"
"do it"
TITLE LDFX
ENTRY LDFX
EXT OPSEQ
EXPAND
NOLIST
SYMBOL
EQU DAX='170'

LDFX: W->DA(DAX)
GOTO OPSEQ
END

"set index file cell

"MACRO LDFj, ARG
j=0,...,6 DOP=0,2,4,6
"MACRO STfj, DA
j=0,...,17 DOP=0,2,4,6

LDF0: W->DA(0)
GOTO OPSEQ
LDF1: W->DA(1)
GOTO OPSEQ
LDF2: W->DA(2)
GOTO OPSEQ
LDF3: W->DA(3)
GOTO OPSEQ
LDF4: W->DA(4)
GOTO OPSEQ
LDF5: W->DA(5)
GOTO OPSEQ
LDF6: W->DA(6)
GOTO OPSEQ

STF0: U->T, DA(0)->U
W->DA, GOTO STF
"set destination
STF1: U->T, DA(1)->U
W->DA, GOTO STF
"get file value
STF2: U->T, DA(2)->U
W->DA, GOTO STF
"set destination
STF3: U->T, DA(3)->U
W->DA, GOTO STF
"set destination
STF4: U->T, DA(4)->U
W->DA, GOTO STF
"get file value
STF5: U->T, DA(5)->U
W->DA, GOTO STF
"set destination
STF6: U->T, DA(6)->U
W->DA, GOTO STF
"get file value
STF7: U->T, DA(7)->U
W->DA, GOTO STF
"set destination
STF10: U->T, DA(10)->U
W->DA, GOTO STF
"get file value
STF11: U->T, DA(11)->U
W->DA, GOTO STF
"set destination
STF12: U->T, DA(12)->U
W->DA, GOTO STF
"get file value
STF13: U->T, DA(13)->U
W->DA, GOTO STF
"set destination
STF14: U->T, DA(14)->U
W->DA, GOTO STF
"get file value
STF15: U->T, DA(15)->U
W->DA, GOTO STF
"set destination
ldstf.mic

STF16:  U->T, DA(16)->U  "get file value
W->DA, GOTO STF      "set destination
STF17:  U->T, DA(17)->U  "get file value
W->DA, GOTO STF      "set destination
STF:    XB=U, YB=U, WRITE, YB->DA,  "T->D, restore F
         T->U, GOTO OPSEQ      "restore U, exit
END

logpwr.mic

TITLE LOGPWR
ENTRY LOGPWR
EXIT FLOAT2, FLOG, OPSEQ
EXPAND
MOLIST
SYMBOL

"MACRO LOGPWR, X,YA(DESC) DESC=X,YA(DATA),N
" U = SCALE OF DATA
" Converts 32 bit data to floating point,
" computes the logarithms, and fixes the
" result with a scale of 7. The result is
" stored in place in XY.

LOGPWR:  U->W, Y->J, X->YA, X->XC  "W=SCALE
         XC->XA, DEC J
         W:'7:0->W  "data pt in UV, SCL=W
         CALL FLOG
         W:17:0->W  "save SCL, float
         "compute log
         V:PP        "compute shift
         MA:PP:U    "shift V
         ML:H->V, 0->U  "shifted V
         ML@R:UV:CH->UV, Y->W
         "result in UV, get SCL
         U->X, V->Y, INC XA, INC YA,  "store result
         DEC J, IF J>0 GOTO LOOP
         GOTO OPSEQ
         END
movecx.mic

TITLE MOVECX
ENTRY MOVECX
EXIT EXHAC
EXPAND
NOLIST
SYMBOL

"MACRO MOVECX, VAL, XA, LEN DOP=1,3,5,7

MOVECX: DY->J, DX->XC
       "set up J
       XC->XA, DEC J, W->V, V->T
       "save V
       HOOP
       "wait for DEC J
       V->X, INC XA, DEC J,
       IF J>0 GOTO .
       READ, T->V, GOTO EXHAC
       END

moverd.mic

TITLE MOVERD
ENTRY MOVERD
EXIT OPCODE
EXPAND
NOLIST
SYMBOL

EQU DI='12'0

"MACRO MOVERD, DA, NA, DOP=1,3,5,7

MOVERD: DY->J
       "COUNT
       W->RA, DX->W
       W->DA(DI:D), DEC J
       "set DEST
       1->BC, U->T
       "INC-1
       LOOP: RL->U
       "get RL
       WRITE URR, INC RA, DEC J,
       IF J>0 GOTO LOOP
       T->U, GOTO OPSEQ
       "exit
       END
TITLE MOVE
ENTRY MOVE, MOVEY, MOVEX, MOVEXD, SHYDX, SHYDY
EXIT OPSEQ
EXPAND
NOLIST
SYMBOL

"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set count" "first X" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set DA" "V->, Y->V" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "DEC J, if J>0 GOTO" "GOTO OPSEQ, T->V" "resotre V, exit"

"These macros have their parameters reversed so" "DA can be specified indirectly.

"MACRO SMYDX, DA, YA, CNT DOP=1,3,5,7" "set J" "set YA"
"MACRO SMYDX, DA, YA, CNT DOP=1,3,5,7" "set J, XC->XA, GOTO MOVXD" "set XA"
"MACRO SMYDX, DA, YA, CNT DOP=1,3,5,7" "set YA, J" "GOTO MOVY1"

"MACRO SETD, ARG, XXX, DA DOP=1,3,5,7" "set DEST, ARG->V" "W,H,V" "ARG->D"
"MACRO SETD, ARG, XXX, DA DOP=1,3,5,7" "set DEST, ARG->V" "W,H,V" "ARG->D"
"MACRO SETD, ARG, XXX, DA DOP=1,3,5,7" "set DEST, ARG->V" "W,H,V" "ARG->D"

"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set count" "first X" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set DA" "V->, Y->V" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "DEC J, if J>0 GOTO" "GOTO OPSEQ, T->V" "resotre V, exit"

"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set count" "first X" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "set DA" "V->, Y->V" "branch CNT-1 times"
"MACRO MOVE, DA(SOURCE), DA(DEST), COUNT DOP=DY->W, READ" "DEC J, if J>0 GOTO" "GOTO OPSEQ, T->V" "resotre V, exit"
"MACRO MULTIL, XYA(DESC)  DESC = XYA, L
MULTIL:  Y->J, X->YA, X->XC
         XC->XA, DEC J
         X*Y:22, INC XA, INC YA
         NOP
MLTLL:  X*Y:22, DEC XA, DEC YA,
         MAR->UV
         U->X, V->Y, INC XA(2),
         INC YA(2), DEC J,
         IF J=0 GOTO MLTLL
         READ, GOTO EXHAC
END

"MACRO ORMOD, A(DESC)  DESC = XA, L
" Computes OR(AABS X1)  i=0,...,L-1
" Returns U = Number of leading zeros. V = OR
ORMOD:  Y->J, X->XC
         XC->XA, 0->U, 0:H->V
         "set up J
ORLP:   ABS X->U, U OR V->V,
         INC XA, DEC J,
         IF J=0 GOTO ORLP
         SCLY->U, READ, GOTO EXHAC
         "leading zero cnt
END
"MACRO POWER, XXY (DESC)  DESC = XXY, N"

POWER:  Y->J, X->XC, X->YA  "set COUNT"
X->XA, J->JC, DEC J  "set XA INC"
Y->Y, INC YA, 0->T  "YO+YO"
ML->T->T, MR->U, X->X, INC XA  "X1*X1"
MLMR+TU:GH->WV, Y->Y, INC XA  "X0*X0+YO*YO"
DECI YA, 0->T  "Y1+Y1"
LOOP:  W->Y, INC YA(2), ML->T->T,  "store P(j)R"
MR->U, X->X, DEC XA(2)  "store P(j)L"
V->X, INC XA(XC)  "MEM+TU:GH->WV,"
"(j+1)"*X(j+1)  "Y(j+2)"*Y(j+2)
DEC J, IF J=0 GOTO LOOP
READ, GOTO EDMAC
END

"MACRO RADIAN, QAUD, BQUA, CQQU, DQQU"
ENTRY RADIAN, AQUAD, BQUAD, CQUAD, DQUAD
EXT FMULT, FADD, FLOATA2
EXPAND
NOLIST
SYMBOL

"Calculates quadrant, table fetch n, delta, sinD, cosD,  "TRG(n), TRG(128-n), and quadrant formulas for FSIN  "and FCOS in subroutine FCOS"
"RADIAN assumes H->U, v->X, u->X, v->X, u->X, v->X,  "and TRG(128-n) ->twv; sinD is in XY1,Y2;
"cosD is in XY3,Y4; TRG(n) is in XY5,Y6; TRG(128-n) is in XY7,Y8"

"AQUAD: cosD*TRG(n) + sinD*TRG(128-n)  XYA->6"
"sin(qad0) = cos(qad3)  XYA->6"
"BQUAD: cosD*TRG(128-n) + sinD*(-TRG(n))  XYA->6"
"sin(qad1) = cos(qad0)  XYA->6"
"CQUAD: cosD*(-TRG(n)) + sinD*(-TRG(128-n))  XYA->6"
"sin(qad2) = cos(qad1)  XYA->6"
"DQUAD: cosD*(-TRG(128-n)) + sinD*sinD*TRG(n)  XYA->6"
"sin(qad3) = cos(qad2)  XYA->6"

EQU  PSTD=’1771’O, TWDOPF=’1772’O, TRBASE=’1570’O

RADIAN:  RR+VF, PP, W, H, 0->T  "begin 2/pi*M"
R+MBB:PP, INC YA  "YA=1"
ML/MR:GH->UV, RR+U:PL  "begin shift"
MLMR+OU:GH+UV, T+GCOF+T,  "SHIFT(Y)"*XLP:PP, INC YA  "SHIFT(X)"
RL+MBB:PP, V->XU  "begin shift"
PIDTWO-Y, DEC YA  "SHIFT(Y)"
MLMR+OU:GH+UV, GASHT+GCOF+T,  "begin shift"
"SHIFT(Y)+2"*XLP:PP, INC YA  "SHIFT(X)"
MLMR+TU:GF->TU, MA+PP  "begin shift"
"SHIFT SMALL GOTO LSM"
MA+PP, ML->W  "shift u"
MA+TF2, ML+MB:GH->WV  "shift t"
"200O->YL  "128->YL"
MLMR+OU:GH+UV, Y->RA, DEC YA  "fetch pi/2, YA=0"
MR+U:O+U, V+VLP:PP  "mask 7 lead 0"
"77777'0 AND V->V"
EPSL:  '400'0 AND V, '400'0*400'0:PP,  "INTQ=XO, N=U"
"U->X, ML->U, '77777'0->W  "INTP=XO, N=U"
IF H=V GOTO CONT  "ROUND"
CONT:  RR+VLP:PP  "begin E+pi/2"
RL+VLP:PP, U->Y, O->T  "N=Y; E invention V"
radian.mic  Page 4

IF Q>0 GOTO QO
Q3:  3->X
'200'0->Y
QO:  INC XA, INC YA, UG, V:HM,
     CALL FLPAT1
     U->X, V->Y, INC XA, INC YA
W->Y, GOTO DELTA
XYA=2

LSM:  MA*U:PP
MAT: P3, ML->W, '200'0->YL
MLMB+ON:GH->VM, '777'0->T,
Y->RA, DEC YA
MLMB+OV:GH->UV
T AND V->V, V*YL:PP, GOTO EPSL
END

random.mic  Page 1

TITLE RANDOMS
ENTRY RANDOMS
EXPAND
NOLIST
SYMBOL

"MACRO RANDOMS, XVA(DESC) DESC = R(-1) * RM
   SCL, N
"  Computes Rj = R(j-1) * RM (Modulo 2**16)
"  and X(XA+j) = Rj*SCL:FR for j=0,...,N

RANDOMS: X*Y:PP, Y->YL, XA->XC,
           INC XA, INC YA
           X->XL, Y->J, INC YA
           MR->V, DEC J, Y->XA
           V*YL:PP
           V*XL:ER
RLP:    MR->V, V->X
           V*YL:PP, ML->U
           U->X, INC XA, V*XL:FR,
           DEC J, IF J>0 GOTO RLP
           XC->XA, READ
           T->X, EXEC MACRO
           END

"start R(-1)*RM
"Y=RM
"set J, XL=SCL
"V=R(0)
"start R(O)*RM
"start R(O)*SCL
"T=R(j), V=R(j+1)
"U=X(j)
"store X(j)
"j+1->j, do N times
"pt at R(-1) save
"save R(N-1)
"MACRO RECORD XYA(CNMLBNL) CNMLBNL = AREC, MREC
= BLNK, PA
= PAV, PB
= DMP, PC

EQU EXBS='1420'0,
EXPEND='1440'0,
\"16 word table in Y\"
PMAX='127'0,
\"limit\"
PMIN='36'0,
\"Max PAV value\"
DF='12525'0,
\"Min PAV value\"
BF='31460'0,
\"Damping table mult\"
DIABS='1420'0,
\"Blanking multiplier\"
\"8 word table in X\"

RECORD: U-X->U, INC XA,
EXBS*HB:PP
U-X->U, U->W, INC XA(2)
MA=1:PP
U+X:FR, DEC XA
IF G>O GOTO NOTBLNK,
IF 0->U
EXIT: READ, GOTO EXMAC
"ACUR-AREC
"base of EXP DECAY TABLE
U=ADISP, W=new PA
EXBS=1
\"start DMP=ADISP\"
ADISP=0?
\"U=EXBS\"
EXIT
NOTBLNK: ML-U->U, EXPEND->T
U->XA, XA->XC, U->T:G,
\"fetch EXP value\"
\"100000'0->XL\"
\"EXPX->EXP\"
MREC=EXP, U=MCI
\"MREC=EXP, U=MCI\"
MA=0.5, V=PAV+PA
\"MREC=EXP, U=MCI\"
U=EXBS
\"EXBS=1, EXPIND\"
MCI=THRESH
\"start 0.5*(PAV+PA)\"
PMIN->T
IF G<O GOTO EXIT, V-ML:G
U->Y, INC YA, ML->T:G,
\"PMAX-PAV\"
ML->V, DF*HB:PP
\"PMIN=PAV\"
\"MREC-PAV,DMC=PMIN\"
V=PAV
\"V=PAV\"
U=old PA
\"U=old PA\"
PAV*DF
\"PAV=DAV-PA\"
\"PAV=PAV?\"
\"PAV=PMAX?\"
\"PAV=PMAX?\"
W->Y, INC YA, X+M:W->V,
\"PAV=PAV?\"
INC XA(2),
\"PAV=PMAX?\"
IF G<O GOTO UMAX
\"PAV=PAV?\"
MA*PMIN:FR, PMIN->X
\"no, PMIN->PAV\"
\"V=old PB\"
\"MCI=THRESH\"
\"V=old PB\"
\"V=old PB\"
\"V=old PB\"
CONT: Y->M, XB=DTABS-'6'0,
XB->T, DEC XA(2)
U-Y, INC YA, ML->T: T,
\"compute BLNK\"
BF*HB:FR
\"compute BLNK\"
W-Y, T->YA
\"compute BLNK\"
V-X, ML->U, Y:H->V, INC XA
\"new BLNK\"
U-X, INC XA(2), READ
\"new BLNK\"
V-X, EXEC MACRO
TITLE RMVYX
ENTRY RMVYX, RMVYX
EXT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO RMVYX, YA, XA, N  X(XA+j) .--> Y(YA+j);N
"MACRO RMVYX, YA, XA, N  Y(YA+j) .--> X(XA-j);N

RMVYX:  DX->YA, DY->J
DEC J, X->W, INC XA
"set YA,J
W->X, DEC YA, X->W, INC XA,
DEC J, IF J>0 GOTO .
READ, GOTO EXMAC
"W-X(XA+0)
NOOP
"X(+J)->Y(-J)
"W-X(+J), J+1->J
READ, GOTO EXMAC
"set XA, save V
Y->V, INC YA
"V=Y(+0)
READ, GOTO EXMAC
"store X[-j]
"V=Y(+J), J+1->J
W->V, READ, GOTO EXMAC
"exit
END

TITLE SAVE
ENTRY SAVE, RESTORE
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

"MACRO SAVE, DA: DOP=ARG->DA, DBLE U.V, XA, YA->D
"MACRO RESTORE, DA: DOP=ARG->DA, DBLE D->U, V.XA, YA

SAVE:  XA->XC, YA->YV, WRITE UV
       WRITE XCYC, GOTO OPSEQ
RESTORE: READ
         DX->U, DY->V, READ
         DX->XC, DY->YV, READ (EA)
         XC->XA, YC->YA, EXEC MACRO
END
A-

**MACRO SCHED, A(Routine)**  
DOP=0, 2, 4, 6  
Add ROUTINE to subroutine stack at bottom +1  
Must not be used in to level routine

**EQU**  
**.BOS=12'80, EXT=10'0**

**SCHED:**  
V->T, W:V, DA(STK) -> W  
W->J, 0->W  
READ, INC DA(W), GOTO SLPE  
SLPE:  
DX->V, READ, INC DA(0)  
1: Move entry up  
IF J->0 GOTO SLPE  
T->V, GOTO OPSEQ  
END

**MACRO SCORE, xxx**  
This program computes a "best choice" pitch value  
and its "score" using the Gold-Rabiner algorithm.  
It uses an input table of 36 pitch values  
picked from the list of extremes  
in the low-passed speech data, along with the  
pitch selected for the previous frame. Each of 6  
out pitch candidates is checked using four successively  
larger windows.

**EQU**  
**BIPTR=17'40, BOPTAB=BIPTR+4,**  
**CNENDEL=BOPTAB+36'D, CAND=CNENDEL+1,**  
**CNPTR=CAND+1, BTAB=CNPTR+1, CTAB=CTAB+2**

**TOP:**  
V->Y, U:X:G, INC YA, DEC VA  
**CNENDEL->Y,CAND-CNTAB(1)**  
**U(X):J->XC,CAND->Y**  
**IF G=0 GOTO LOOP**  
**REPEAT IF CAND=CNTAB(J-2)**  
**XC->YA, Q->V, BIPTX->XA**  
**FETCH DPN, pt at BIPTX save**  
**BTAB->X, BTAB->T, INC XA(2)**  
**INIT. BIPTX**  
**Y->U, V=XT, INC XA**  
**PANE=0, U=DPN**  
**3->J, T=YA**  
**Y=BIA, set J**  
**U->X, DEC XA, IF G=0 GOTO SCORE**  
**DPN->X, CAND no good?**

**The following loop is repeated four times with**  
**windows = DPN, 2*DPN, 3*DPN, and 4*DPN.**  
**Assume X = PANE, Y = BIAS**

**NXTPN:**  
X->V, INC XA, CAND->YX  
V->V, Y=V, DEC XA, Y=V, Y=V  
**V=old PANE**  
**V=new PANE, W=BIAS**  
**U=UB, V=LB**  
**DEC YA(2)**  
**V=VY, INC XA(2), W:V->V**

**SETUP:**  
X=V, U:X:H, INC XA  
**T=PB, UB=PB**  
**H=UB=PB, PJ->T**  
**MR:T, GOTO CNTLP**

**RES:**  
C=V, U:X:H, INC XA  
**H=UB=PB, PJ->T**  
**G=LB=PB(J-1)**  
**IF G=0 GOTO FIN**  
**G=0, GOTO CNTLP**

**CNTLP:**  
C=V, U:X:H, INC XA  
**H=UB=PB, PJ->T**  
**P(J-2)=UB**

**X=0, DEC XA(2), Y:V:H->V,**  
**IF G=0 GOTO SETUP**  
**X=0, U:X:H, INC XA,**  
**V=VY, GOTO RES**

**FIN:**  
**BIPTX->XC, T=V, V=V, W=V, INYA**
score.mic Page 2

Y=W: INC XA, XC->XA
X=U->U, Y=H->V, DEC XA
U->X, INC XA, IF H>0 GOTO NOTBST
V->X, W->Y
NOTBST: INC XA, U->YA, DEC J, IF J>0 GOTO NEXTN

"ENTRY POINT FOR SCORE and for each candidate PITCH"

SCORE: CNDEND->W
CNDPTR->YA
Y->XA, Y=W:H, '6':G->V
X->U, CTAB->XA
Y+V=H->V, U=X:G, INC XA, IF H<0 GOTO TOP
READ, GOTO EXMAC
END

sflt.mic Page 1

"TITLE SFLT"
ENTRY SDIV, SFLT
EXIT FDIVS, FLOAT2, EXMAC
EXPAND
NOLIST
SYMBOL

"SHORT FLOATING POINT OPERATIONS."
"All operations assume the first operand is in U/V,"
"U=Mantissa, V=Scale."
"The second operand is in XY at the location given"
"by the argument. X=Mantissa, Y=Scale"

"MACRO SFLT, SCL. Converts UV*2^SCL to short FP."

SFLT: U:G, V:H, CALL FLOAT2
"test value, float
W->V, READ, GOTO EXMAC
"V=SCL, exit

"SDIV, ARG DOP=0,2,4 OP2 at XY(ARG)
"Computes X/Y / U/V -> U/V
SDIV: T->XA, W->XA, V:H->W,
CALL FDIVS
W:H->V, READ, GOTO EXMAC
END"
"A collection of short operations involving UV and an argument"

EQU EA='31'0

"MACRO LDUX, XA DOP=13 X(XA)--U
LDUX: X--U, EXEC MACRO"

"MACRO LDVY, YA DOP=15 Y(YA)--V
LDVY: Y--V, EXEC MACRO"

"MACRO LDUI, YA DOP=14 X(Y(YA))--U
LDUI: Y--X, READ, GOTO LDUX"

"MACRO LDUY, YA DOP=15 Y(YA)--U
LDUY: Y--U, EXEC MACRO"

"MACRO LDU, ARG DOP=1,3,5,7 ARG--U
LDU: W--U, EXEC MACRO"

"MACRO LDV, ARG DOP=1,3,5,7 ARG--V
LDV: W--V, EXEC MACRO"

"MACRO LDUVY, XYA DOP=11 XY(XYA)--UV
LDUVY: X--U, Y:H--V, EXEC MACRO"

"MACRO LDUVD, DA DOP=0 (DA must be even) D--UV
" (also could have DOP=3 for inline ARG)
LDUVD: DX--U, DY:H--V, READ, GOTO EXMAC"

"MACRO STUX, XA DOP=13 U--X(XA)
STUX: U--X, EXEC MACRO"

"MACRO STVY, YA DOP=15 V--Y(YA)
STVY: V--Y, EXEC MACRO"

"MACRO STVX, XA DOP=13 V--X(XA)
STVX: V--X, EXEC MACRO"

"MACRO STUY, YA DOP=15 U--Y(YA)
STUY: U--Y, EXEC MACRO"

"MACRO STUVY, XYA DOP=11 UV--XY(XYA)
STUVY: U--X, V--Y, EXEC MACRO"
SUBAV: W-V->V, EXEC MACRO

"MACRO ADDAD, ARG, YA, XXX DOP=1,3,5,7
ADDAD: DX->YA
Y+M->W, READ, GOTO LDIV

"MACRO ADDAX, VAL, XXX, YA DOP=1,3,5,7 X(XA)+ARG->X
ADDAX: DY->XA
X+W:H->V, V->W, READ
V->X, W->V, EXEC MACRO

"MACRO MULT, ARG DOP=0,2,4,6 U*ARG->UV
MULT: U*W:22, GOTO MPVU

"MACRO MULTV, ARG DOP=0,2,4,6 V*ARG->UV
MULTV: V*W:22
MPVU: READ(EA)
MLMR->UV, EXEC MACRO

"MACRO FMULTU, ARG DOP=0,2,4,6 U*ARG:FR->U
FMULTU: U*W:FR
MFU: READ(EA)
ML->U, EXEC MACRO

"MACRO FMULTV, ARG DOP=0,2,4,6 V*ARG:FR->V
FMULTV: V*W:FR
MFV: READ(EA)
ML->V, EXEC MACRO

"MACRO TLNY, ARG, YA, BRANCH DOP=1,3,5,7
" Y(YA)+ARG->Y, IF Y\_=0 GOTO BRANCH
TLNY: DX->YA
Y+M->W
W->Y
DY->W, READ, IF H=0 GOTO EXMAC
W->DA, GOTO OPSEQ
"set new EA

"MACRO IFULT, ARG, YA, BRANCH DOP=1,3,5,7
" IF Y(YA)=ARG, GOTO BRANCH
IFULT: DX->YA
Y+M->H
"wait for test value
LIT: NOOP
DY->W, READ, IF H=0 GOTO GOTO
"set new EA

"MACRO IFALT, ARG, VAL, BRANCH DOP=1,3,5,7
" IF ARG<VAL, GOTO BRANCH
IFALT: W-DX: H, GOTO LIT

"MACRO IFUQ, VAL, DOP=0,2,4,6
" IF U=ARG, DO NEXT MACRO
IFUQ: U-W: H
NOOP
READ, IF H=0 GOTO EXMAC
READ, GOTO EXMAC
"skip one doubleword
TITLE SMXYD
ENTRY SMXYD
INTRLY
EXT MVXYD2

SMXYD DX->XC, DX->YA, DY->J
    "SET XA AND COUNT
XC->XA, GOTO MVXYD2
    "SET XA, GOTO MOVE
END

TITLE STEPN
ENTRY STEPN, IFEA, IFA, INC, ONSTEP
EXT OPSEQ, EXEC, GOTO
EXPAND
NOLIST
SYMBOL

"CMD MACRO IFEA, ARG, VAL, N DOP=1, 3, 5, 7
"CMD MACRO IFEA, EAV, xxx, N DOP=1, 3, 5, 7
"CMD MACRO STEPN, N DOP=DY->W
  MACRO INC, ARG, DA, EAV, DOP=1, 3, 5
  D(DA+ARG->D, IF RESULT \=-0 GOTO EAVAL

EQU EA='31'0, DI='12'0

IFEA: DX-W:H, GOTO COMP
      "VAL-ARG
IFEA: DA(EA)=W:H
      "EA-EAVAL
COMP: YB=DA(DI)
      "FA=DA
      IF H=0 GOTO EXEC, DY->W
      "CNT->W, DONE?
STEPN: U->T, V->U, T->W:V->W,
      "save U,V
      -1->W
READ (DI), INC DA(W)
      "DEC CNT
      IF H=0 GOTO EXEC, WRITE V,
      "update CNT, done?
READ (EA)
      "restore U,V
ONESTEP: EXEC MACRO, ENABLE
      "do one MACRO

"MACRO INC, ARG, DA, EAV, DOP=1, 3, 5
  D(DA+ARG->D, IF result \=-0 goto EAVAL

INC: DX->W
      "W=DA
      W->DA(DI), T->U, U->T
READ, DY->W
      "fetch D, W=EAVAL
      DX:U->U, READ, INC DA(-1)
      "D=ARG, back up DA
WRITE U
      "update D
      IF G=0 GOTO OPSEQ, T->U
GOTO GOTO
      "is result=0
      "no branch
END
"FLOATING POINT STACK FORMAT"  
  
  "X , Y    
  "SCL  
  "ML , MR for first number  
  "- , SCL2  
  "ML , MR for second number  
  "- , SCL3  
  "U=ML , V=MR for TOS number  

"MACRO LDSTED, A (FP WORD) DOP=DI->F (DA)  
LDSTED:  DEC DX, DEC YA, READ   
        DX->T, READ, Y->Y, DEC YA   
        "ML->T, OLD MR->Y   
        "MR->V, fetch SCL   
        "SCL->W   
        U->X, DEC EA, T->U, M->Y, 
        EXEC MACRO   
        "SCL->Y, do next  

"MACRO STSTED, DA DOP=DI->F (DA)  
STSTED:  WRITE U, Y->U   
        WRITE V   
        WRITE U   
        "store ML   
        "store MR   
        "store SCL   

"POPSTK:  INC XA, INC YA, READ (EA)   
          X->U, Y:V->V, INC XA, 
          INC YA, EXEC MACRO   

"MACRO SAVSTK, DA push STK and save ptr DOP=DI->F (DA)  
SAVSTK:  DEC DXA, DEC DAY, Y->W   
         "W=SCL   
         U->X, V->Y, DEC XA, DEC YA   
         "STK PTR->X  
         WRITE XC, GOTO OPSEQ   

"MACRO UVSTSTK, A, A DOP=0, 2, 4 FLOAT UV->TOS  
UVSTSTK:  T->YA, W->XA, 
          CALL FLOAT   
          W->Y, READ, GOTO EXHMAC   
          "save SCL  

"MACRO STADD, XXX DOP=NOOP  
STADD:  Y->W, INC XA(2), INC YA(2), 
         CALL FADD   
         W->Y, READ, GOTO EXHMAC   
         "do add   
         "save new SCL  

"MACRO STSUB, XXX DOP=NOOP  
STSUB:  Y->W, INC XA(1), INC YA(2), 
         CALL FSUB   
         W->Y, READ, GOTO EXHMAC   
         "do subtract   
         "save new SCL  

"MACRO STMULT, XXX DOP=NOOP  
STMULT:  Y->W, INC XA(2), INC YA(2), 
         "point at ARG2   
         CALL FMULT   
         "save new SCL  

"MACRO STDIV, XXX DOP=NOOP  
STDIV:  Y->W, INC XA(2), INC YA(2), 
         "point at numerator   
         "do divide   
         CALL FINV   
         "save new SCL  

STLOG:  Y->W, CALL FLOG   
         W->Y, READ, GOTO EXHMAC  
         END
TITLE TSTTOL
ENTRY TSTTOL
EXT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO TSTTOL, ARG
" IF LARG - U1 < V, DO NEXT MACRO, ELSE SKIP
TSTTOL: U-W->U, U->Y
V->U, V->H, T:F->U
IF G=0 GOTO HTST
IF G=0 GOTO EXMAC, READ
READ, GOTO EXMAC
ELSE SKIP
HTST: IF H=0 GOTO EXMAC, READ
READ, GOTO EXMAC
END

TITLE UPCHAN
ENTRY UPCHAN
EXT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO UPCHAN XYA(CHNLBLK) CHNLBLK = AREC, A(UPDESC)
" ACUR in U MREC, PA
" MCUR in V PB
" "UPDATE DESC in XY = BLNK, A(EXPTAB)
" DMP, PAV
" EQU EXPLEN='17'0

UPCHAN: U-X->U, XA->XC, Y->XA,
Y->YC
U-X->U, U->W, INC XA,
YC->YA, YA->YC
XU:FR, XC->XA
IF G=0 GOTO NOTBLNK,
Y->U, YC->YA, INC XA
NOTBLNK: ML+U->U, EXPLEN->T
MA*X:FR, U-XA, T-MI:G
X+MB:FR, XC->XA, V->U,
INC YA
Y->V, INC XA,
IF G=0 GOTO UPDATE
U-MI:G
NOOP
IF G=0 GOTO EXIT
UPDATE: U->X, DEC XA, V->U,
W->Y, INC YA
X+M:H->V, Y->W
V->X, U->Y, INC YA, READ
W+Y, EXEC MACRO
END
xymadd.doc

TITLE XYMADD
ENTRY XYMADD, YMADDX
EXIT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO XYMADD, XYA(DESC) DOP=10 DESO = M, L
"MACRO YMADDX, XYA(DESC) DOP=10 DESC = M, L

XYMADD: Y->J, X+MB:22,
INC XA, INC YA
Y->XA, X->YA, DEC J
MAC:FR, INC XA
Y->W, INC YA
X:Y=ML:W->V, MAC:FR,
Y->W, DEC YA, INC XA
V->Y, INC XA(2), DEC J,
IF J<0 GOTO XMAYL
READ, GOTO EXMAC

XMAYL: Y->J, X+MB:22, INC XA, INC YA
Y->XA, X->YA, DEC J
MAC:FR, INC YA
X->T, INC XA
XMAYL: ML:T->U, MAC:FR, X->T,
DEC XA, INC YA
U->X, INC XA(2), DEC J,
IF J<0 GOTO XMAYL
READ, GOTO EXMAC
END

xymoves.doc

TITLE XYMOVES
ENTRY MOVEXYD, MOVEDXY, MOVEXY, MOVEYX, MVXYD2
EXIT OPSEQ, EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO MOVEXYD, DA(SOURCE), XA(DEST), COUNT DOP=DX->W, READ
"MACRO MOVEXYD, XA(SOURCE), DA(DEST), COUNT DOP=DX->W, READ
"MACRO MOVEXY, XA, Y, COUNT DOP=TY->X, READ
"MACRO MOVEYX, XA, X, COUNT DOP=DY->X, READ

EQU DI='12'0

MOVEXYD: DX->YA, DY->J
W->DA(DI:D), DEC J, DX->W
W->XA, READ
"set COUNT
"set DA
"set XA, fetch word
"set COUNT-1 times
"branch COUNT-1 times
GOTO OPSEQ
"EXIT

"MACRO MOVEXYD, XA(SOURCE), DA(DEST), COUNT DOP=DX->XAXA, DX->YA, READ
MOVEXYD: DY->J, DX->W
MVXYD2: W->DA(DI:D), DEC J
"set COUNT
"set DA
"set COUNT
"set COUNT-1 times
"next pair
"branch COUNT-1 times
GOTO OPSEQ

"MACRO MOVEXY, XA, YA, COUNT DOP=DY->X, READ
MOVEXY: DY->J, DX->Y
"set DA, COUNT
DEC J
X->W, INC XA
W->Y, INC YA, X->W, INC XA
DEC J, IF J<0 GOTO .
READ, GOTO EXMAC
"set COUNT
"set COUNT
"save, first word
"store word, get next
"branch COUNT-1 times

"MACRO MOVEYX, YA, XA, COUNT DOP=DX->YA, READ
MOVEYX: DY->J, DX->X
"set count
DEC J, W->XA
V->Y, INC YA
V->X, INC XA, Y->V, INC YA,
DEC J, IF J<0 GOTO .
READ, GOTO EXMAC
"set COUNT
"set COUNT
"store word, get next
"branch COUNT-1 times

END
APPENDIX B

"ASLDATA - INPUT AND OUTPUT IN BASE 64
TITLE ASLDATA
LISTOBJ
LISTXT
ENTRY ASLDRSP, RLSEIAL, RCVINT, OUTQASL, OUTASL

DECLARE X: ldesca='1760'O, lflag='1763'O, outptr='1761'O
DECLARE XY: ldesc='1761'O, stringd='1762'O, xywk='1760'O,
         rexits='1764'O, accum='1762'O
DECLARE Y: rcvxt='1764'O, lcode='1761'O, trxt='1763'O,
         shifter='1761'O

INSERT DMEMEQ2.ins
INSERT EQANAD.ins

ASLDRSP
SAVE INTSVE
MOVEXYD xywk, XYSAVE, 5 "SAVE XYMEM USED
aslrsp2 ASINT LINEO, rcvint, trint "DETERMINE TYPE
MOVEDXY XYSAVE, xywk, 5 "RESTORE XYMEM
RESTORE INTSVE "AND REGS
INRTIN "EXIT
rcvint STVX ldesca "SAVE A(LINE DESC)
MOVEDXY ldesca, ldesc, 4 "GET LINE DESC
IFUEQ '23'O
GOTO noint "XOFF?
IFUEQ '21'O
GOTO intok "XON?
IFULT '40'O
GOTO aslrsp2 "IGNORE OTHER CTRL
CALL rcvxt "CALL ROUTINE
GOTO aslrsp2 "GO CHECK FOR MORE
noint ADDAX -2, lflag "CHANGE ACTIVITY FLAG
GOTO store "UPDATE LINE DESC
intok LDUX lflag "GET ACTIVITY FLAG
IFUEQ 0
GOTO aslrsp2 "IGNORE XON IF IDLE
ADDU 2 "UPDATE FLAG
STUX lflag 
IFULT 1 "NOT READY?
GOTO store "SET TO STATE ONE
LDX 1, lflag "ENABLE TXRDY
LDU '47'O "SET UART CTRL
SNDCMD lcode "UPDATE LINE DESC
GOTO store "SAVE A(LINE DESC
trint STVX ldesca "GET DESC.
MOVEDXY ldesca, ldesc, 4 "No activity
IFALT 1flag, 1, trdn "UPDATE POINTER
LDUDA X[stringd]
ADDA X, X[stringd] "SEND CHAR IN U
SNDCHR lcode "Dec count
TLYY -1, Y[stringd], store "COMPLETION ROUTINE
CALL trxt "UPDATE DESC
store SMVXYD ldesca, ldesc, 4 "GO CHECK FOR MORE
GOTO aslrsp2 "UPDATE
trdn LDU '46'O "DISABLE TRDY INT
SNDCMD lcode
GOTO aslrsp2

"TOP LEVEL ASL INPUT
RCVINT IFUEQ playn "Parcel?
GOTO inpars
IFUEQ addr
GOTO saddr "SET ADDRESS?
IFUEQ data
GOTO datal "DATA BLOCK?
IFUEQ freqn
GOTO freqnl "ANAL START?
IFUEQ stop
GOTO stopl "Anal stop?
IFUEQ outd
GOTO outdl "DATA OUT?
RTN "IGNORE OTHERS

"ENTERED FROM CODEPARS THROUGH SENDOUTP
OUTQASL LDU PARLINE
IFUEQ 0 "NO OUTPUT NOW
RTN
LDUDA [4]U "GET FLAG
IFUEQ 0
GOTO chkq "LINE IDLE?

send SCHED OUTQASL
RTN
chkq INCD 0, POCNt, setp "HAVE PARS?
RTN "NO
setp CALL OUTASL "BUILD CHARS FOR PARCEL
LDV PARLINE "V=A(LINE DESC)
LDXY X[16], A[OUTBUF], 9 "A(DATA),COUNT
outp LDXY X[17], 1, A[asldne] "FLAG, COMPLETION EXIT
SMVXYD [2]V, X[16], 9 "FILL IN LINE DESC
LDUDA [1]V "GET LINE CODE
LDV [0]U "PUT IT IN V
LDU '47'0 "ENABLE TXRDY
SNDCMD [0]V "SEND IT~
RTN "DONE

"INTERRUPT RESPONSE ROUTINE
asldne LDU 0
STUX 1flag
INCD 0, POCNt, send "MORE PARS IN QUEUE
RTN "NO

"CONVERT ONE PACKED PARCEL TO BASE 64 CODES
OUTASL MOVEXY POPTR, BA, 3 "GET NEXT PARCEL
INCD -1, POCNt, NEXT1 "DEC PAR CNT
NEXT1 LDU POPTR "GET PTR
ADDU PRLN "UPDATE IT
IFUEQ POLIM
LDU POBSE "WRAP IF NEEDED
STUDA POPTR "STORE NEW PTR
LDU playn "START OF PARCEL CODE
LDV OUTBUF+9
out3 STVX FA "SET POINTER
STUDA [-9]V "SET CODE CHAR
LDVY BA+2 "LAST 16 BITS
CALL next "OUTPUT C7
CALL next "OUTPUT C6
CALL next "C5 LEFT
STUX FA+1 "SAVE IT
LDVY BA+1 "NEXT 16 BITS
MULTV 4 "GET 2 BITS IN U
IFULT 0
ADDU 4
"CLEAR ANY SIGN EXT.
ADDUX FA+1 "COMBINE WITH FIRST 4 BITS
STUDA FA "OUTPUT C5
CALL next "OUTPUT C4
CALL next "OUTPUT C3
CALL next "C2 LEFT
STUX FA+1 "TEMP SAVE
LDVY BA "FIRST 16 BITS OF PARCEL
MULTV 16 "GET 4 BITS IN U
IFULT 0
ADDU 16 "CLEAR ANY SIGN EXT.
ADDUX FA+1 "COMBINE WITH FIRST 2 BITS
STUDA FA "OUTPUT C2
CALL next "OUTPUT C1
next ADDAX -1, FA "DEC PTR
MULTV 64 "6 BITS
IFULT 0
ADDU 64 "CLEAN OFF SIGN EXT.
ADDU '40'0 "ADD ZONE
STUDA FA "OUTPUT
RTN

saddr "SET ADDRESS FOR DATA TRANSFER
LDU BOS "A(BUFFER)
LDXY rexits, A[ends], A[parinc] "END WORD, CHAR EXITS
GOTO indat

data1 LDU BOS "DESTINATION ADDRESS
LDXY rexits, A[rtn], A[parinc] "WORD, CHAR EXITS
GOTO indat

inpars "PARAMETER INPUT COLLECTION
LDU PRIIN "A(BUFFER)
LDXY rexits, A[end], A[parinc] "WORD, CHAR EXITS
GOTO indat

indat STUDA [rdesc]V "SET A(BUFFER)
LDU 1
STUDA [rdesc+1]V "INITIAL SHIFTER
LDU 0
STUDA [rdesc+3]V "CLEAR ACCUM.
SMVXYD [6]V, rexits, 1 "STORE NEW RCV EXITS
RTN

rtn IFUGT '137'0
GOTO chke "HAVE CODE?
MOVEDXY [rdesc]V, ldesc, 2 "GET RCV DESC.
SUBU '40'0 "REMOVE ZONE FROM CHAR
MULTU shifter "SHIFT INTO PLACE
ADDVY Y[accum] "ADD ON OLD
STUVXY accum "SAVE RESULT
LDUY shifter "GET SHIFTER
MULTU 64 "MAKE IT 6 MORE PLACES
STVY shifter "STORE IT
LDV ldesc "RECOVER A(LINE DESC)
IFUEQ 0
GOTO pcex "RIGHT WORD NOT FULL?
STUY shifter "NEW SHIFTER
LDUY Y[accum] "GET WORD
STUDA outptr "OUTPUT IT
ADDAX 1, outptr "UPDATE PTR
LDUX X[accum]  "GET RESIDUAL BITS
STUY Y[accum]  "SAVE IN RIGHT HALF
GOTO X[rexis]  "END OF WORD PROCESS

chke CALL ends  "RESET INPUT EXIT
GOTO RCVINT  "GOT PROCESS CODE

end LDUY shifter  "GET SHIFTER
IFUGT 1
GOTO pcex  "NOT DONE YET?
INCD 1, PRICNT, end2

end2 LDU PRIN  "UPDATE QUEUE COUNT
IFUEQ PRIPTR  "GET INPUT PTR
SCHED SYNINP  "WAS QUEUE EMPTY?
ADDU PRLIN  "UPDATE PTR
IFUEQ PRILIM  "WRAP IF NECESSARY
LDU Pribse  "STORE PTR
STUD PRIN  "QUEUE NOT FULL?
IFALT PRICNT, prmax-1, ends  "MUST SEND XOFF

ends LDU RCVINT  "TOP LEVEL CHAR INPUT
STUDA [7]V  "SET IN LINE DESC
pcex SMVXYD [rdesc]V, ldesc, 2
RTN  "SAVE RCV DESC

"INPUT WHEN PARCEL REMOVED FROM QUEUE
RLSEIAL LDU PRICNT
IFUEQ prmax-2
GOTO sxon  "WAS QUEUE FULL?
RTN

sxon SETD '21'O, xchr  "WILL SEND XON
sxo LDU PARLINE  "GET FLAG
LDUDA [4]U
IFUEQ 0
GOTO chkx  "LINE IDLE?
SCHED sxo  "NO, TRY LATER
RTN

chkx LDU xchr  "ONE CHAR TO SEND
LDV 1
STUVXY XY[16]
LDV PARLINE  "A(LINE DESC)
GOTO outp

frequ1 SETD POIN, POPTR  "SET QUEUE EMPTY
SETD 0, POCNT  "SET OUTPUT LINE
STVD PARLINE  "CLEAR OUTPUT LINE
RTN

stop1 SETD 0, PARLINE  "CLEAR OUTPUT LINE
RTN

outd1 STVD datline  "SAVE A(LINE DESC)
coutd LDV datline  "GET A(LINE DESC)
LDUDA [4]V  "GET FLAG
IFUEQ 0
coto doout  "LINE FREE?
sched coutd  "NO, CHECK LATER
RTN

doout MOVEDY BOS, BA, 3
     INCD 3, BOS, doout2
     "GET 3 WORDS
     "UPDATE ADDRESS
     "CODE FOR BLOCK
     "END OF BUFFER
     "BUILD OUTPUT CHARS FOR 3 WORDS
     "GET A(LINE DESC)
     "ADDRESS, COUNT
     "SEND BLOCK

doout2 LDU data
     LDV doutb+9
     CALL out3
     "END OF BUFFER
     "BUILD OUTPUT CHARS FOR 3 WORDS
     "GET A(LINE DESC)
     "ADDRESS, COUNT
     "SEND BLOCK

     LDV dateline
     LDXY XY[16], A[doutb], 9
     GOTO outp

rdesc EQU LINEOR-LINEO
datline EQU OUTBUF+9
doutb EQU datline+1
xchr EQU doutb+9
prmax EQU 20

" LINE DESCRIPTOR FORMAT
" 0: TXRDY+RCVRDY MASK, LINE CODE
" 2: A(NEXT OUTPUT CHAR), COUNT
" 4: LINE FLAG, OUTPUT DONE EXIT
" 6: RECEIVE WORD EXIT, RECEIVE CHAR EXIT
"

RECEIVE DESCRIPTOR FORMAT
" 0: A(NEXT BUFFER SPACE), MULTIPLIER
" 2: DOUBLEWORD INPUT ACCUMULATOR
" 4: 4 WORDS UNUSED
END