

①

CHI Systems, Inc.
100 Burns Place
Goleta, CA 93117
(805) 984-8868

OK -
DTIC

85 - 0219

AD-A221 570

CHI-5 ARRAY PROCESSOR

SUPPORT

Final Technical Report

April 1985

PRINCIPAL INVESTIGATOR:
PROJECT SCIENTISTS:

Dr. Glen J. Culler
Dr. Judith B. Bruckner
Thomas W. Fuller
Virginia R. Grant
Dr. Michael McCammon
Dr. Jean A. Nisbet

S
MAY 15 1990
E
D

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.

90 05 14 138

DO NOT REMOVE
> *Z0AAAAAA52910916*

INTRODUCTION

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 UNIBUS¹ or QBUS¹ 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.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Date _____	
Approved _____	
Special Agent in Charge	
Dissemination _____	
A-1	

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 MACLNK 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 accommodate 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 multiplier operations, are specified using an infix notation, e.g.

$$\begin{aligned} & X * V \\ & MPL + U \rightarrow U \\ & MPLMPR + TU \rightarrow UV. \end{aligned}$$

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

$$\begin{aligned} & X * Y:PP \text{ (Unsigned multiplication)} \\ & MPLMPR + TU :FG \rightarrow UV \text{ (Use F and G adders instead of G and H adders).} \end{aligned}$$

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.

$$\begin{aligned} & V \rightarrow W \text{ (uses YBS if available, otherwise the H adder)} \\ & T \rightarrow U \text{ (uses either the F or G adder)} \\ & T:G \rightarrow U \text{ (uses the G adder)} \end{aligned}$$

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 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 playn (p) character and 48 speech data bits packed into 8 characters, will be transmitted from the CHI-5.

Playn (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.,
CHI-5 MICRO-PROGRAMMING REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q1, CHI Systems Inc., Goleta, California, April 1982.
2. Fuller, T. W.,
CHI-5 SIMULATOR REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q2, CHI Systems Inc., Goleta, California, August 1982.
3. Grant, V. R.,
CHI-5 MACRO-PROGRAMMING REFERENCE MANUAL, Quarterly Technical Report, MDA 903-82-C-0136-Q3, CHI Systems Inc., Goleta, California, October 1983.
4. Culler, G. J. ,
ACOUSTICAL ARRAY PROCESSOR DESIGN, Final Technical Report, MDA 903-78-C-0313, CHI Systems Inc., Goleta, California, June 1983.

APPENDIX A

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

andl.mic	fsqrt.mic
anm255.mic	fsubt.mic
asline.mic	initchk.mic
ave.mic	intsrv.mic
bldpop.mic	latred.mic
blkshifts.mic	ldfx.mic
coding.mic	ldstf.mic
daops.mic	logpwr.mic
decim.mic	movecx.mic
divbyy.mic	moverd.mic
doint.mic	moves.mic
dop.mic	multl.mic
dotxy.mic	ormod.mic
exec.mic	power.mic
extrem.mic	radian.mic
fadd.mic	randoms.mic
fcos.mic	record.mic
fdivs.mic	rmoves.mic
fexp.mic	save.mic
fft1.mic	sched.mic
fft2.mic	score.mic
fftpass.mic	sflt.mic
fftrl.mic	shortops.mic
filter.mic	smvxyd.mic
finv.mic	stepn.mic
float.mic	stkops.mic
flog.mic	tsttol.mic
flts.mic	upchan.mic
fltsops.mic	xymadd.mic
fmult.mic	xymoves.mic

```

TITLE ANDL
ENTRY ANDL
EXT EXMAC
EXPAND
NOLIST
SYMBOL

```

```

"MACRO ANDL, X(A,DESC) DESC=X(A,N
" X(X(A-1) = X(X(A+1) .AND. Y(X(A+1) for 1=0,....,N-1

```

```

ANDL: Y->J, X->YA, X->YC
XC->XA, DEC J, Y->V
ANDLP: X AND V->V, INC YA
V->X, INC XA, Y->V, DEC J,
IF J>0 GOTO ANDLP
READ, GOTO EXMAC
END

```

```

"V=first mask
"X .AND. Y->V
"store result
"Repeat
"exit

```

```

TITLE ANN255
ENTRY M255TOL, LTOM255
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

```

```

"MACRO M255TOL, DA DOP=DY->W Read 32 words and convert
"MACRO LTOM255, DA DOP=DY->W D/A output

```

```

EQU DI='12'O, ANALOG='10'O, S='8225'D, SIN2=4

```

```

M255TOL: W->DA(DI)
ANALOG->DEV
'30'D->J
-1->M
IO XOR W->V, S->YL
V->U, '16'O AND V->V
'10000'O->XL, '10000'O*V:PP
'17'O AND U->V, O->T, U:G
ML+T+CCO:F->U, '16'D-V:G,
'16'D*V:H->V
SHIFT(U)*V:PP,
IF G<0 GOTO NEG
'33'D->U
MR-U->U, IO XOR W->V
U*YL:ER, 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+CCO:F->U, '16'D-V:G,
'16'D*V:H->V
SHIFT(U)*V:PP,
IF G<0 GOTO NEGL
'33'D->U
MR-U->U
U*YL:ER
NOOP
ML->U
WRITE U, GOTO OPSEQ
'33'D->U
U-MR->U, IO XOR W->V, GOTO L2
'33'D->U
U-MR->U, GOTO L3

```

```

"set up output address
"select ANALG IO
"Count-1
"1's compliment mask
"first input
"V=16'n
"shift to get n
"V=m, test sign
"u=n+1
"v=m+16
" (m+16)*2** (n+1)
"sign neg?
"2*16.5
"U=VAL,V=next pt
"VAL*SCL,save code
"clean off 16'n
"shift to get n
"store result,V=m
"test sign,done?
"U=n+1
"V=m+16
" (m+16)*2** (n+1)
"sign negative?
"2*16.5
"VAL
"AS
"NOOP
"store last result
"u=VAL,v=next pt
"2*16.5
"u=VAL,negative

```

```

"MACRO LTOM255, DA DOP=DY->W D/A output

```

```

LTOM255: W->DA(DI)
ANALOG->DEV
READ, '16'D->XL
ABS DX->U, '31'D->J
U* SIN2
'33'D->T, U:G
MR+T->V, '8'D->W, DX->T,

```

```

"select ANALOG IO
"get first value
"do loop 32 times
"2*ABS(L)/S
"2*16.5,no ovf for SCLV
"V=SL, save old L

```

```

SHIFT(SCLV) *V:PP,
M-SCLV->M
XL *M:PP, ABS DX->U
MR->M, '32'D->YL
U *SINV2, T->U
'33'D->T, W *YL:FR,
MR->U, U:H
MR *T->V, '8'D->M, DX->T,
READ
SHIFT(SCLV) *V:PP, ML *U->V,
M-SCLV->M,
IF H<0 GOTO NEGLM
-1 XOR V->V, XL *M:PP
ABS DX->U, MR->M
V->IO, DEC J, IF J>0 GOTO LM2P
COTO OPSEQ
NEGLM: '177'O XOR V->V, XL *M:PP
ABS DX->U, MR->M, GOTO LM2
END
    
```

```

"fetch new
"float SL
"compute N-6
"16*(N-6)
"W=flt M, YL=fix shift
"2*ABS L/S, or ig.val
"fix M
"test sign, u=16*(N-16)
"V=SCL, save old val
"get next
"float M
"compute N-6
"V=unsigned result
"form pos. result
"W=flt M, u=ABS L
"output result
"form neg result
"W=flt M, U=ABS L
    
```

```

TITLE ASLINE
EXPAND
NOLIST
SYMBOL
ENTRY LIMIT, SNDOMD, SNDCHR, ASINT
EXT COTO, OPSEQ
EQU EA='31'O "DA + DBLE

"ASYNCHRONOUS LINE INTERFACE COMMANDS
"DEFINITIONS FOR INTERFACE CONTROL REGISTER BITS
EQU M='77400'O, RESET='100000'O
EQU IEN='140000'O "INT ENABLE
EQU DATA='160000'O "DEFAULT IS CMD
EQU RD='170000'O "READ STATUS/DATA FROM DEV
EQU WRT='174000'O "WRITE CMB TO DEV CMD/DATA
EQU L1='176000'O "SELECT LINE 1
EQU LO='177000'O "SELECT LINE 0
EQU DEN='177400'O "ENABLE AP DATA ONTO CMB

"BAUD RATE CODES FOR LIMIT
" 0000 50
" 0001 75
" 0010 110
" 0011 134.5
" 0100 150
" 0101 300
" 0110 600
" 0111 1200
" 1000 1800
" 1001 2000
" 1010 2400
" 1011 3600
" 1100 4800
" 1101 7200
" 1110 9600
" 1111 19200

" MACRO LIMIT, ARG ARG=BAUD RATE CODES FOR L1, L0
LIMIT: '11'O->DEV, V->U, U->T "MAKE V AVAILABLE
XB=M+RESET+DEN+LO+L1,
19->J
V->IO, "V=RESET, BAUD RATE
V-RESET->V "COUNT FOR PULSE WIDTH
DEC J, "START RESET PULSE
IF J>0 COTO "PREPARE RESET END
V->IO, "HOLD FOR 5 MIC. SEC.
YB=M+DEN+WRT+L1+LO+'116'O, "END RESET
YB->V "MODE SETUP
V->IO, "SET MODE
19->J, 19->W "SET COUNT
T:F->U, U->V, "RESTORE V
XB=M+DEN+L1+LO+'116'O, "SETUP END WRT
XB->T "END WRT FOR MODE
T->IO,
    
```

```

DEC J,
IF J>0 GOTO .
XB=M*DEN*MRT*LO*LI*66'O, "WAIT
XB->IO, "START MRT FOR CMD
M->J "SET UP COUNT
XB=M*DEN*LI*LO*66'O,XB->I "END MRT FOR MODE
I->IO, DEC J, "WAIT
IF J>0 GOTO .
XB=M*DATA*LEN, XB->I, "FETCH NEXT MACRO
READ "EXIT TO NEXT MACRO
I->IO, EXEC MACRO

* MACRO SNDCHAR,ARG ARG=LINE CODE, U = CHAR
SNDCHR: V->M, M:H->V, "SAVE V
GOTO SNDC

* MACRO SNDCMD, LINECODE U = CTRL CHAR
SNDCMD: M - DATA:H -> V, "FORM CTRLS
V->M "SAVE V
SNDG: '11'O->DEV, "ADDRESS LINE UNITS
V * U ->V "ADD COMMAND
V -> IO, "SEND ADDRESS
YB=DEV*MRT, "PREPARE MRT
YB + V -> V "START MRT PULSE
V->IO, "PREPARE MRT END
V - MRT -> V "FETCH NEXT MACRO
READ, V->I, M->V "END MRT, EXIT
I->IO, EXEC MACRO

EQU RCVMSK='2400'O "RCVR RDY BITS FOR LI & LO

"MACRO ASINT, DA(LINE DESCS), A(RCV ROUTINE), A(TR ROUTINE)
" DOP = DY->DA, DBLE
" RETURNS V=A(LINE DESCRIPTOR), U=INPUT CHAR AND
" BRANCHES TO THE TR OR RCV ROUTINE IF TRRDY
" OR RBDY IS ACTIVE. IF NO LINES READY,
" CONTINUE IN LINE.
" LINE DESCRIPTOR FORMAT:
" 0: MASK FOR TRRDY AND RBDY BITS FOR LINE
" 1: LINE CODE=M*DATA*LG*LEN (IF INTERRUPTS USED)
" 2: A(NEXT CHARACTER PAIR IN D). CHARACTER COUNT
" 4: CURRENT CHARACTER, A(OUTPUT COMPLETE ROUTINE)
" 6: OUTPUT ACTIVE FLAG, A(INPUT ROUTINE)
ASINT: 2->J "NUMBER OF LINES
NEXT: DA->U, 8: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 OPSEQ "NO LINES READY

LOOP: '11'O->DEV, DX->V "V=TRRDY/RBDY MASK
IO AND V -> V, RD->M "TRRDY OR RBDY?

```

```

RCVMSK AND V
IF H=0 GOTO NEXT,
DY->V
IF H=0 GOTO TRRDY,
V->IO, V:M:H->V
V->IO, V:M:H->V,
'377'O->M
U->V, V->T, READ(EA)
IO AND M -> M
T->IO, M->U
DX->M, GO TO GOTO

TRRDY: READ(EA), U->V
DY->M, GO TO GOTO
END
"RDY?
"NEITHER READY FOR LINE
"ONLY TRRDY
"ADDRESS LINE
"SET CODE MASK
"CHAR CODE MASK
"V=A(LINE DESC)
"CLEAN OFF CHAR
"CHAR -> U, END RD
"EXIT TO RCV
"V=A(LINE DESC)
"EXIT TO TR

```

TITLE AVE
 ENTRY AVE
 EXT EXMAC
 EXPAND
 NOLIST
 SYMBOL

"MACRO AVE, XVA(DESC) DESC = C, N
 " Y(YA+j) = Y(YA+j-1) + C * (Y(YA+j+r) - Y(YA+j)), j=1,...,N
 " Y(0) is not changed

AVE: Y->J, X*MB:FR, INC XA, INC YA
 X->YA, Y->YC
 Y->V, INC YA
 MA*Y:FR, INC YA(YC)
 " Do Loop for j=0,...,N
 AVEL: MA*Y:FR, DEC YA(YC+1)
 V->Y, INC YA(2), V-ML->V
 ML*V->V, MA*Y:FR,
 INC YA(YC), DEC J,
 IF J>0 GOTO AVEL
 READ, GOTO EXMAC
 END

"set CNT,C
 "Y=Y(YA)
 "V=Y(YA)
 "Y(YA+1)*C
 "Y(YA+j+1+r)*C
 "new Y(YA+j)
 "C*Y(YA+j+1)
 "j+1->j
 "exit

TITLE BLDPOP
 ENTRY BLDPOP
 EXT EXMAC
 EXPAND
 NOLIST
 SYMBOL

"MACRO BLDPOP, CNT, YA(CHNLBLS)+1, XA(POPTAB)
 " DOP = DY->J, READ
 "Assume CHANNEL BLOCKS in Y PAD = MREC, PA, PB, PC
 "Output POPTAB in X PAD = PA, PD, PE, PF, PG, PH

BLDPOP: DX->YA, DY->XA, DEC J
 Y->U, INC YA
 LOOP: U->X, INC XA, Y+U->U,
 Y:H->V, INC YA
 U->X, INC XA, Y+U->U,
 Y->W, INC YA
 U->X, INC XA, W+V->U, INC YA
 U->X, INC XA, W->U
 V->X, INC XA, Y->U, INC YA,
 DEC J, IF J>0 GOTO LOOP
 READ, GOTO EXMAC
 END

"adjust count
 "U=PA
 "PA->X, PB->V
 "PD=PA, PB->U
 "PD->X, PC->W
 "PE=PA, PB+PC->U
 "PF->X, PE=PB+PC->U
 "PE->X, PC->U
 "PC->X
 "PB->X,next PA->U
 "DO CNT BLS
 " exit

TITLE BLKSHFTS
 ENTRY NORMX, NORMY, SCALEX, SCALEY, LDMA, LDMA,
 MULFRC, MULTFXC, MULTYXC, MULTYXC
 EXT EXPMAC, ONESTEP
 EXPAND
 NOLIST
 SYMBOL

```
"MACRO NORMX, XVA(DESC) DOP=10 DESC=>YA, XA
"
" Computes X*2**U->Y for N X's, 0<=U<16
"MACRO MULTXC, XVA(DESC) Computes MA*X->Y
"MACRO NORMY, XVA(DESC) DOP=10 DESC = XVA, N
" Computes (XY*2**U)->XY for N values, 0<=U<16
"MACRO MULTYXC, XVA(DESC) Computes (XY*MA)->XY
"MACRO SCALEY, XVA(DESC) DESC = XVA, N
" Computes (XY*2**U)->XY for N values -16<=U<0
"MACRO MULFRC, XVA(DESC) Computes XY*MA->XY
"MACRO SCALEX, XVA(DESC) DOP=10 DESC = YA, XA
"
"Computes X*2**U:FR->Y for N values -16<=U<0
" This is twice the inverse of NORMX
"MACRO MULTFXC, XVA(DESC) Computes X*MA:FR->Y
"MACRO LDMA, ARG DOP = 0,2,4,6
" Puts ARG ln MA (signed)
NORMX: SHIFT(U)*MB:PP
MULTXC: Y->JA, INC YA, X->U
MA*X:P2, INC XA, DEC J,
U->YC
MB->U, MA*X:P2, INC XA,
YC->YA
U->Y, INC YA, MB->U,
MA*X:P2, INC XA, DEC J,
IF J>0 GOTO .
READ, GOTO EXPMAC
"MACRO NORMY, XVA(DESC) DOP=10 DESC = XVA, N
" Computes (XY*2**U)->XY for N values 0<=U<16
"MACRO MULTYXC, XVA(DESC) Computes (XY*MA)->XY
NORMY: SHIFT(U)*MB:PP
MULTYXC: Y->J, X->YA, X->XC
MA*Y:PP, INC YA, XC->XA,
DEC J
MA*X:P2, INC XA, DEC J
MA*Y:PP, MLR->UV, DEC YA
V->Y, INC YA(2), MR->U->U,
MA*X:P2, DEC XA
U->X, INC XA(2), MLR->UV,
MA*Y:PP, DEC YA,
DEC J, IF J>0 GOTO NKYLP "repeat N times
READ, GOTO EXPMAC
"MACRO SCALEY, XVA(DESC) DESC = XVA, N
```

```
" computes XY*2**U->XY, for N values -16<=U<0
"MACRO MULFRC, XVA(DESC) Computes XY*MA->XY
SCALEY: SHIFT(U)*MB:PP
MULFRC: X->XC, X->YA, Y->J
XC->XA, DEC J, MA*Y:PP,
INC YA
MA*X:P2, INC XA, 3->YC
O->U, ML:H->V, MA*Y:PP,
INC YA
MLR->UV:GH->UV, MA*X:P2,
DEC XA
MULFRC: U->X, V->W, O:F->U, ML->V,
MA*Y:PP, DEC YA(2),
INC XA(2)
W->Y, MLR->UV:GH->UV,
INC YA(YC), MA*X:P2,
DEC XA, DEC J,
IF J>0 GOTO MFLRP
READ, GOTO EXPMAC, INC XA(2)
"leave XA=YA
"MACRO SCALEX, XVA(DESC) DOP=10 DESC = YA, XA
"
" Computes X*2**U:FR->Y for N values -16<=U<0
"MACRO MULTFXC, XVA(DESC) Computes X*MA:FR->Y
SCALEX: SHIFT(U)*MB:PP
MULTFXC: Y->XA, X->T, INC YA
Y->J, T->YA
DEC J, MA*X:FR, INC XA
MA*X:FR, INC XA
ML->W, MA*X:FR, INC XA
W->Y, INC YA, ML->W,
MA*X:FR, INC XA,
DEC J, IF J>0 GOTO .
READ, GOTO EXPMAC
"MACRO LDMA, ARG DOP=0,2,4,6 Puts ARG ln MA (signed)
LDMA: DISABLE
W*MB:22
READ, GOTO ONESTEP
END
```

```
"Will not serve ints
"set up MA
"EXPAC, ENABLE INTS
```

```

TITLE CODING
ENTRY ENCODE, DECODE
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

```

```

"ENCODE, CNT, YA(LIST), XA(A(CODE TABLE))
"DECODE, CNT, YA(A(CODE TABLE)), XA(CODES LIST)
EQU DI='12'0

```

```

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

```

```

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

```

```

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

```

```

TITLE DAOPS
ENTRY STUDA, LDU DA
INTRLV
EXT OPSEQ
EQU DA=10

```

```

STUDA M->DA(DA)
      WRITE U, GOTO OPSEQ
LDUDA M->DA(DA)
      READ
      DX->U, GOTO OPSEQ
      END

```

```

"DESTINATION
"STORE U IN D(DA)
"SOURCE ADDRESS
"FETCH D(DA)->U

```

TITLE DECIM
 ENTRY DECIM, SIZECHK
 EXT OPSEQ
 EXPAND
 NOLIST
 SYMBOL

"DECIM - FFT DECIMATE ROUTINE
 "MACRO DECIM, D(DESC) DOP=0,2,4,6
 " Assumes a descriptor: COUNT, DA(DATA)
 " K, RB In D
 " L, K, and RB are filled in by FFT1
 "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: M->DA(DI:D)
 READ, O+U->U
 DX->V, DY->DA(DI:D)
 SHIFT(SCLV)*4:PP
 O->T, O->W, V->J
 T->XC, T->YA, MR->V
 DEC J, XC->XA, DCHV+T->U,
 READ
 LOOP: MR+V:H->V, U->XA, U->YC,
 DX->X, ABS XB->U
 DY->Y, ABS YB->U, YC->YA,
 U OR W->W
 U OR W->W, DCHV+T->U,
 READ, DEC J,
 IF J<0 GOTO LOOP
 M->V, GOTO OPSEQ

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

SIZECHK: 1->M
 SCLV-W->M, 1:G->V
 U->DA(DI:D), W-V:H->V
 IF H<0 GOTO SCALE,
 SHIFT(V)*MB:PP,
 CLR XA, CLR YA
 READ, RTN
 SCALE: READ, W:H->V, X->XL,
 O->U, -2->W
 V->U, DX:H->V, READ (DI),
 INC DA(U)
 V->J, MA*XL:FR, INC XA,
 DX-U->U
 MA*Y:FR, INC YA, U->T, DEC J
 MA*X:FR, DEC XA, ML->U
 U->X, INC XA(2), ML->U,
 MA*Y:FR, DEC YA
 U->Y, INC YA(2), ML->U,

MA*X:FR, DEC XA,
 DEC J, IF J<0 GOTO SCLP
 WRITE T, INC DA(M),
 CLR XA, CLR YA
 READ (DI:D)
 RTN
 END

"shift X(j+2)
 "j+1->j, loop
 "store new scl

```
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, 0->W, 0->V
```

```
U-Y->U, U->I
```

```
LOOP: WM+WM+LOCO:CH->WM
```

```
IF G<0 GOTO Z,
```

```
U+U->U
```

```
NEXT: DEC J, IF J>0 GOTO LOOP,
```

```
U-Y->U, U->I
```

```
I->I
```

```
IF G>0 GOTO INC, READ
```

```
V->U, W->V, EXEC MACRO
```

```
INC: VT+OM:CH->UV, EXEC MACRO
```

```
Z: I+I->U, GOTO NEXT
```

```
END
```

```
"this usually fails
```

```
"Z+VM+ (subt OK)
```

```
"subt no good?
```

```
"double new rem.
```

```
"done all bits?
```

```
"next trial subt
```

```
"wait for result
```

```
"rem > .5?
```

```
"no, exit
```

```
"inc result, exit
```

```
"double rem
```

```
TITLE DOINT
ENTRY DOINT
EXT OPSEQ
EXPAND
NOLIST
SYMBOL
```

```
"CMD MACRO DOINT, INT# - push int routine on stk if pending
```

```
EQU STK='10'0, DI='12'0
```

```
DOINT: W->DEV, W->DA(DI)
```

```
CLR SBIT
```

```
IF STAT=0 GOTO OPSEQ, READ
```

```
WRITE (STK) DX
```

```
GOTO OPSEQ
```

```
END
```

```
"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
DOP1: DX->W, DX->T, READ
DOP2: DY->W, DX->T
DOP3: DX->T, DY->W, READ
DOP4: DY->XA, XA->XC,
      IF J>0 PSB=XW, CONT
DOP5: DY->XA, XA->XC,
      IF J>0 PSB=XNR, CONT
DOP6: DX->YA, YA->YC,
      IF J>0 PSB=YW, CONT
DOP7: DX->YA, YA->YC,
      IF J>0 PSB=YNR, CONT
DOP10: DX->YA, DY->XA
DOP11: DX->YA, DY->XA, READ
DOP12: DY->XA
DOP13: DY->XA, READ
DOP14: DX->YA
DOP15: DX->YA, READ
DOP16: DY->J
DOP17: DY->J, READ
DOP20: NOOP
DOP21: NOOP
DOP22: DY->DA(DI)
DOP23: DY->DA(DI:D)
DOP24: DY->U->W
DOP25: DY->U->W, READ
DOP26: DY->V->W
DOP27: DY->V->W, READ
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
DOP37: DY->J, READ, CONT
LOOP:  YB=DY, YB=DY, READ, PSVWRT
      XB=DX, YB=DY, READ, PSLMRT
      XB=DX, YB=DY, DEC J, PSRWRT
      IF J>0 PSB=LOOP, CONT, READ
      EXEC MACRO
      "CNT-1->W
      "COTO RDPS
      "set CNT-1
      "DY->PSV
      "DXY->PSL
      "DXY->PSR
      "DONE?
      "next macro

```

```

RDPS: W->J, DX->W, CONT
      W->DA(DI:D), CONT
      PSVRD, WRITE, XB=0
      PSLRD, WRITE, DEC J
      PSRRD, WRITE, IF J>0 PSB=RDLP
      READ (EA), CONT
EXMAC: EXEC MACRO
XW: X->T, X->W, XC->XA
XNR: X->W, X->T, XC->XA, READ
YW: Y->W, YC->YA
YNR: Y->W, YC->YA, READ
      END

```

```

"ARG1->J
"DEST address, DBLE
"O,PSV->DLDR
"PSL->DLDR
"PSR->DLDR, done?
"fetch next macro
"X(ARG1)->T,W
"X(ARG1)->W,T, next word
"Y(ARG1)->W
"Y(ARG1)->M, next word

```

```

TITLE DOTXY
ENTRY DOTX, DOTX
EXT EXMAC
EXPAND
NOLIST
SYMBOL

```

```

"MACRO DOTX, XYA(DESC) DOP=LXYA
"
" DESC-> K, N X, Y
" BA, FA
"MACRO DOTX, XYA(DESC) DOP=LXYA
"
" DESC-> K, N X, Y
" BA, FA

```

```

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

```

```

LOOP: MLAB+UV:GH->UV, X*Y,
      INC XA, INC YA, DEC J,
      IF J>O GOTO LOOP
      GOTO EXMAC

```

```

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

```

```

DOTXL: MLAB+UV:GH->UV, X*X, INC XA,
      DEC J, IF J>O GOTO DOTXL
      GOTO EXMAC
END

```

```

TITLE EXEC
ENTRY RTN, EXEC, END, HOSTI, RUN, INTRTN
EXT GOTO, INI, CALL, SETP7
EXPAND
NOLIST
SYMBOL

```

```

"MACRO EXEC, xxx DOP=NOOP
"MACRO RTN, -1 DOP=DY->W
"MACRO END, -1 DOP=DY->W
"MACRO RUN, -1 DOP=NOOP
"MACRO HOSTI, PORT DOP=0,2,4,6
"MACRO INTRTN, -1 DOP=0,2,4 Enable interrupts and RTN

```

```

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

```

```

EXEC: CMD->DEV
      DA->W
      CLR SBIT, GOTO CALL
      READ (STK), INC DA(W)
      READ, INC DA(O)
      DX->W, GOTO GOTO
      READ (STK), INC DA(W)
      READ, INC DA(O)
      DX->W, YB=DA(EA)
      W->DA, GOTO INI
      CALL SETP7, ENABLE
      DY->W, GOTO RTN
      HST->DEV
      NOOP
      IF STAT=1 GOTO.
      W->DEV, READ (EA)
      INT HOSTI, EXEC MACRO
      INTRTN: ENABLE, GOTO RTN
      END

      "select CMD port
      "rest of CMD
      "save old EA
      "DEC STK PTR
      "fetch address
      "D (STK) ->W
      "back up STK
      "fetch old EA
      "FA=EA
      "restore EA
      "set up CMD PORT
      "resume execution
      "address HST
      "wait for status valid
      "wait for AITN CLR
      "set PORT
      "send Interrupt

```

TITLE EXTREM
ENTRY EXTREM
EXT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO EXTREM XVA(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 call 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->M, Y->J, INC XA, INC YA
X->YA, Y->XA, DEC J
1*1, YA->YC, X->U, INC YA,
INC XA
X-U:G
U-X:G
IF G<O GOTO DNO, M-MR->M,
X->U, INC XA
X-U:G, GOTO UP1
IF G<O GOTO MAX, X-U:G
UP1: IF H>O GOTO UPO, M-MR->M,
U->Y, X->U, INC XA
YA->YC, YC->YA, READ,
GOTO EXIT
MAX: XA->XC, INC YA
XC->V, W:H
V->Y, INC YA, U-X:G,
DEC J, IF J>O GOTO DNI
YA->YC, YC->YA, GOTO EXIT
DNO: IF G<O GOTO MIN, U-X:G
DNI: IF H>O GOTO DNO, M-MR->M,
U->Y, X->U, INC XA
YA->YC, YC->YA, READ,
GOTO EXIT
MIN: XA->XC, INC YA
XC->V, W:H
V->Y, INC YA, X-U:G,
DEC J, IF J>O GOTO UP1
UP1: YA->YC, YC->YA
EXIT: YC->Y, READ, GOTO EXMAC
END

TITLE FADD
ENTRY FADD
NOLIST
EXPAND
SYMBOL
EXT FLOAT2

"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->M, INC YA, Y->U, U->T
XB*Y:PP, XB=1, W+O:H, INC YA
SHIFT(M)*MB:PP, O-W->M,
IF H<O GOTO SY
SHIFT(W)*V:PP, U:H->W,
IF H=O GOTO NOSH, MR->U
IF SHIFT SMALL GOTO SUV
MA*T:P2, Y->V
IF SHIFT OVF GOTO FLOAT2,
Y:G->U, U:H->V, V->T,
DEC YA(2)
GASN*T:GCO:F->U, ML+V:G->V,
GOTO TCU
SUV: V+HCO->U, ML-U:H->V, GOTO ADD2
NOSH: Y+T:HCO->U, U+V:H->V,
DEC YA(2), GOTO FLOAT2
SY: IF SHIFT SMALL GOTO NORM,
MA*Y:P2, U+W->M
IF SHIFT OVF GOTO FLOAT2,
T:G->U, V:H, DEC YA(2)
TUV: GASN*T:GCO:F->U, ML+V:G->V
NORM: T+HCO->U, ML+V:H->V
ADD2: ML*AR+UV->UV, GOTO FLOAT2,
DEC YA(2)
END

"set J=R, W=input count
"list addresses
"make MR=1
"get X(-1)
"going up or down
"down case
"decreasing
"U=X(O)
"X(1)-X(O)
"X(j)<x(j-1)?
"dec N, loop?
"X(j)=Y, X(j+1)=U
"save r ptr
"skip MACRO
"get index
"V=index
"save in list
"continue if room
"no room, done
"X(j) > X(j-1)?
"dec N, loop?
"X(j)=Y, X(j+1)=U
"save r ptr
"skip MACRO
"save index
"V=index
"save index in list
"continue if room
"no room, save ptr
"save ptr at front

```

TITLE FCOS
ENTRY FCOS, FSIN
EXT RADIAN, AQUAD, BQUAD, CQUAD, DQUAD
EXPAND
NOLIST
SYMBOL

```

```
EQU TMOBPI='1772'0
```

```

FCOS: '2'0:W:H->W, 0->T, V->X, INC XA
      U:G, V:H, TMOBPI->RA, U->X, DEC XA
      W->Y, W-'36'0:H
      IF CH=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 CQUAD
        GOTO AQUAD
        IF H=0 GOTO BQUAD
        GOTO DQUAD
        1->W, 0->V
        '40000'0->U
        GOTO FINISH

```

```
"COS 0=1
```

```

FSIN: '2'0:W:H->W, 0->T, V->X, INC XA
      U:G, V:H, TMOBPI->RA, U->X, DEC XA
      W->Y, W-'36'0:H
      IF CH=0 GOTO ZERO
      IF H>0 GOTO TOOBIG
      CALL RADIAN
      '3'0:U->U

```

```

GOTO QTEST
ZERO: '100000'0->W

```

```

TOOBIG: YB->U, YB:H->V, YB->W, DEC YA,
        YB='7777'0

```

```

FINISH: RTN
        END

```

```

TITLE FDIVS
ENTRY FDIVS, SFLOAT2
LISTOBJ
SYMBOL

```

```

"MICRO Subroutine SDIV
" Computes X/Y / U/M, U>0
" Assumes a 128 word table of inverse values:
" TAB(1) = 128/(1+128) in ROM
"MICRO Subroutine SFLOAT2
" Assumes test for U=0 performed in G adder

```

```
EQU INVTAB='1000'0
```

```

FDIVS: U*'256'D:FR
        Y-W->W, XB->T,
        ML->V, 1+W->W
        '128'D*V:PP, ML+T->V
        V->RA, INVTAB-V:C
        RL:RL:FR, U-MR->U
        U+U->U, 0->T,
        ML+T->T
        T+U:FR
        RL->U
        U-ML->U, 0:H->V, GOTO DIVE
        SS: IF G=0 GOTO ONE,
            RL-U->U, 0:H->V
        DIVZ: X*U:FR
            NOOP
        ML->U, GOTO SFLOAT2
        ONE: X->U, 0:H->V, RTN

```

```

"MICRO SUBROUTINE SFLOAT2
"Assumes test for u=0 performed in G adder

```

```

SFLOAT2: ABS U->V, MA+U:P2,
          '100000'0->T
          IF G=0 GOTO Z, W-SCLV->M,
            SHIFT(SCLV)*MB:P2
          IF SHIFT SMALL GOTO FIN, T->U
          '48'D->U, ML XOR U->V
          U+W->W, V->U, RTN
          MR->U, RTN
          T->W, 0->U, RTN
          END

```

```
"ABS(result)
```

```

"result D
"adjust scale
"branch if no overflow
"correct mantissa sign
"correct scale, exit
"normalized mantissa
"true zero

```

```
"Computes table index
"SCL=SCL(N)-SCL(D)
```

```

"INC SCL
"shift index back
"fetch TVAL, test i=0
"TVAL*TVAL, U=EPS
"U=2EPS
"D close to 1.0?
"T=TVAL*TVAL
"2E*TVAL*TVAL
"TVAL
"TVAL-2E*TVAL*TVAL
"D=1.0?
"1-2EPS
"W*INV(D)

```

```

"result in U, float
"Q=, DONE

```

```

TITLE FEXP
ENTRY FEXP
EXT FLOAT2
EXPAND
NOLIST
SYMBOL

```

```

"MICRO SUBROUTINE FEXP Floating point exponential
" Input data is in uv/w
" Flnd exp(n) = exp((2**a)*M) = (2**SCL)*MAN
"Uses two tables in ROM:
" the first is 32 bits = 1/(2ln2), with point at boundary
" the second is 45 words = EXP2B(n) = exp(n/64), n=0,44

```

```

EQU TBASE='1310'O, LN2='1301'O, INVLN2='1366'O

```

```

FEXP:

```

```

INVLN2->RA
RR*V:PP, O->T, W-'15'O:H
RL*MB:PP, U:G, 2*W:H->W
RR*U:P2, ML->U, W:H,
IF H>O GOTO OVEREL
ML*P + OU:CH->UV, RL*MB:P2,
LN2->Y
ML*P + UV:CH->UV, Y->RA,
GASN+T*CCO:F->T,
IF H<O GOTO RSHIFT
ML*P+TU:FG->TU, SHIFT(W)*V:PP
MA*U:PP, '7777'O->V
MA*U:P2, ML+HCO:G->V, MR*V:H
ML*P+OV:GH->UV, O->T, XB*XB:PP,
XB->XL, XB='400'O
ML*P+TU:FG->TU
ML*U->V, RL*V:PP,
'200'O->YL
RR*U:PP, RR->V, V:H->W
O*ML-RLV:CH->UV, XB=TBASE, XB->T,
RL*MB:PP
O*ML+UV:GH->UV, W->Y
ML*P+UV:CH->UV, '400'O->W
U*YL:2P, U AND W, '777'O->W
U AND W->V, V->YL, V->W,
XL*XL:PP
T*ML:G->U, O->T,
IF H=O GOTO EPSL
U*ML:G->U, V-'1000'O:H->V

```

```

CONT:

```

```

V*YL:2P, V->XL
U->RA, '7777'O->U
ML*P+OU:CH->UV, MA*XL:22,
GASN+T*CCO:F->T
ML*P+UV:CH->UV, ML+HCO:G->V,
MR*V:H, '52525'O:YL:PP
O->T, 1->YL
XB*U*ML:CH->UV, V->YL,
XL*YL:PP, XB='100000'O

```

```

EPSL:

```

```

"EL*ER, EL->XL
"fetch EXPTB
"EL*EL
"2(EL*ER)->UV
"E*E/2->UV
"EL/3
"begin 1+E/3
"((E*E/3)R->YL

```

```

ML*P+UV:GH->UV, O->T,
IF G<O GOTO MTBL
U*YL:PP
MR->U, W:H->V, '7777'O->W
T*CCO:F->T, ML+V+HCO:G->V,
MR*V:H, Y->W
XB*ML+UV:GH->UV, O->T,
XB='100000'O
RL*O:H
RL*V:PP, '7777'O->Y
T:G, U:H, RR*U:PP
ML*P+UV, RL*MB:PP
IF H=O GOTO FLOAT2
ML*P+UV:GH->UV, T*CCO:F->T
ML*U+CCO:F->U, MR+U+HCO:G->V,
Y*V:H, XB=O, XB->T
T:G, U:H GOTO FLOAT2
RSHIFT: ML*P+TU:FG->TU, '20'O*W:H->V
SHIFT(W)*U:PP, O->U
IF H<O GOTO RBIG,
MA*U:P2
ML->V, O->T, XB='400'O,
XB*XB:PP, XB->XL
ML*P+OV:GH->UV, GASN+T:F->T,
GOTO CONT
RBIG: SHIFT(V)*MB:P2, T->U,
O->T, '400'O->XL
IF SHIFT SMALL GOTO ONLYT,
O:H->V, XB=U, XB:G, GASN+T:F->T,
XL*XL:PP
'40000'O->U
ML->W, RTN
ONLYT: GASN*ML+TO:FG->UV, GOTO CONT
OVEREL: IF G>O GOTO POSINE
O->U, O:H->V
'100000'O->W
RTN
POSINE: YB='7777'O, YB->U, YB:H->V, YB->W
DONE: RTN
END

```

```

" (1+E/3) ->UV
"E->UV
"E+E*E*(1+E/3)/2->UV
"SCL->W
" (1+E)*E*E*(1+E/3)/2->UV
"begin EXPTB (E TERM)
"MANTISSA>O
"shift u
"S+2<-16
"shift T
"shift T>16 right
"-32<S+2<-16
"S+2<-32
"#->1
"M>0?
"#->+lnf

```

```

TITLE FFT1
ENTRY FFT1
EXT OPSEQ, SIZECHK
EXPAND
NOLIST
SYMBOL

```

"MICRO FFT1, DA(DESC) DOP=0,2,4,6

EQU DI='12'0

```

FFT1:  M->U, CALL SIZECHK
        DX='100000'0:PP
        READ, INC DA(0)
        WRITE DXB, YB=1, ML->V
        WRITE VYB, YB='512'D
        V->J, CLR XA, CLR YA
        Y:G->V, INC YA, INC XA,
           '100000'0*MB, 0->M,
           DEC J
L:      Y->V->U, MA*X:FR, V-Y:H->V,
        DEC YA, DEC XA
        U->Y, ABS YB->U, INC YA,
           X->I, INC XA
        V->Y, ABS YB->V, INC YA,
           U OR W->M, ML+T:F->U
        V OR W->M, U->X,
           ABS XB->U, DEC XA,
           T-ML->T
        U OR W->M, T->X,
           ABS XB->U, INC XA(2)
        U OR W->M, INC XA, INC YA,
           Y->V, DEC J,
           IF J=0 GOTO L
        M->V, GOTO OPSEQ
END

```

```

"shift data if needed
"count*0.5
"fix scale
"set SCALE, L=1
"set K, RB
"set lop count
"V=Y(0)
"MA=-1.0
"MSIZE
"U=Y(2j)+Y(2j+1)
"V=Y(2j)-Y(2j+1)
"store new Y(2j)
"store new Y(2j+1)
"size new Y(2j)
"size new Y(2j+1)
"store new X(2j+1)
"store new X(2j+1)
"T=X(2j)+X(2j+1)
"store new X(2j)
"size new X(2j)
"V=Y(2j+1)
"j+1->j, loop
"V=SIZE, EXIT

```

```

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

```

FFT2:  M->U, CALL SIZECHK
        READ, '100000'0->XL
        DX->I, DY->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->M,
           IF G=0 GOTO LAST
        V->RA, CALL FFTPASS
        DA('14'0)->U, INC XA(XC),
           INC YA(YC-1)
        U-1->U, YC->J
        U->DA, 0->V,
           IF G=0 GOTO LP
        DA(EA)->V
        -2+V->V
        V->DA, M:H->V, GOTO OPSEQ
        V->RA, CALL FFTPASS
        M->V, GOTO OPSEQ
END

```

```

"size data, U=A(DESC)
"XL=0.5
"SCALE, L
"V=new L
"L->YA, XC
"U=new K
"XA=YC=L
"update L, V=new RB
"store new K, RB
"set CNT=L, U=K-1
"save in F(14)
"set RC
"last pass?
"RA=0, do group
"get k
"XYA(next group)
"dec K, J=L
"New K->F(14)
"do next group
"get EA
"back up
"set EA to repeat
"RA=0, do last pass
"V=size, done

```

```

TITLE FFTPASS
ENTRY FFTPASS
EXPAND
NOLIST
SYMBOL

```

```
"SUBROUTINE FFTPASS
```

```
" W = SIZE DATA
```

```
" XA = A(X2), XC=L, YC=L, YA=A(Y2), YC=L
```

```
" J=L, RA=0, RC=512/L
```

```
"Assumes a 512 doubleword table in ROM of roots of unity:
```

```
" RL(1) = -COS(2pi*1/1024)
```

```
" RR(1) = -SIN(2pi*1/1024)
```

```
FFTPASS: RL*X:FR, '77777'0->YL,
```

```
DEC XA(XC)
```

```
X*YL:FR, 0->T, INC XA(XC)
```

```
DEC J
```

```
RR*Y:FR, T-ML->T, Y->YL,
```

```
'77777'0->XL
```

```
RR*X:FR, ML-T->T,
```

```
INC YA
```

```
T-ML->U, ML-U:H->V,
```

```
0->T, RL*YL:FR,
```

```
Y->YL, DEC YA(YC+1),
```

```
DEC XA(XC)
```

```
U->X, ABS U->U, INC XA(XC),
```

```
ML-T->T, INC RA,
```

```
XL-Y:FR
```

```
V->X, ABS V->V, INC XA,
```

```
U OR W->W, T-ML->T
```

```
V OR W->W, ML-T:F->U,
```

```
ML-T->V, RL*X:FR,
```

```
DEC XA(XC)
```

```
U->Y, ABS U->U, INC YA(YC),
```

```
XL*X:FR, 0->T,
```

```
INC XA(XC)
```

```
V->Y, ABS V->V, INC YA(Y)
```

```
U OR W->W, T-ML->T,
```

```
RR*YL:FR
```

```
V OR W->W, ML-T->T,
```

```
ML-T->U, RR*X:FR,
```

```
DEC J, IF J>0 GOTO LOOP
```

```
"U=X1-COS*X2-SIN*X2+
```

```
"size new Y2
```

```
"T=X1+COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1-COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1+COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1-COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1+COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1-COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1+COS*X2
```

```
"store new Y1
```

```
"size new Y2
```

```
"T=X1-COS*X2
```

```

TITLE FFTRL
ENTRY FFTRL
EXT OPSEQ, SIZECHK
EXPAND
NOLIST
SYMBOL

```

```
EQU DI='12'0
```

```
"MACRO FFTRL, DA(DESC) DOP=0,2,4,6
```

```
" FFTRL is the final pass of a Real FFT. To perform a
```

```
" Real FFT of order 2N on X1, i=0,...,2N-1, first do a Complex
```

```
" FFT of order N on Zj, R1(Zj)=X(2j) and IM(Zj)=X(2j+1),
```

```
" j=0,...,N-1. The final RLFFT pass computes the first N+1
```

```
" bins of the complex spectrum using the formulae:
```

```
" Let Z(j)=X(j)+iY(j)
```

```
" Z(0)'=X(0)+Y(0), Z(N)'=X(0)-Y(0)
```

```
" For j=1,...,N/2:
```

```
" x1=X(j)+X(N-j), x2=Y(j)+Y(N-j)
```

```
" y1=Y(j)-Y(N-j), y2=X(N-j)-X(j)
```

```
" X(j)' = x1*SIN(t)*y2 + COS(t)*x2
```

```
" X(N-j)' = x1*SIN(t)*y2 - COS(t)*x2
```

```
" Y(j)' = y1*SIN(t)*x2 + COS(t)*y2
```

```
" Y(N-j)' = y1*SIN(t)*x2 - COS(t)*y2
```

```
" The SCALE is decreased by one.
```

```
FFTRL: W->U, CALL SIZECHK
```

```
'100000'0->YL
```

```
Y->U, DX-XC, DX->YA, READ
```

```
DX->T, DY->J, READ, YA->YC,
```

```
CLR YA
```

```
DY*YL:PP, U+U->U,
```

```
O:H->V
```

```
READ, INC DA(-4),
```

```
GASN+T->T, -4+U:G,
```

```
V->XL
```

```
0->RA, T->V, X->T,
```

```
YA->YC, YC->YA,
```

```
ML->W
```

```
WRITE V
```

```
W->RC, T+T->V
```

```
V->T, Y->V, XL*XL
```

```
RL-U:FR, GOTO START
```

```
X-T->U, Y:H->V, X->T,
```

```
RR-U:FR, YA->YC, YC->YA
```

```
U->XL,
```

```
Y+V->U, T-ML->T,
```

```
XA->XC, XC->XA, INC YA
```

```
RL-U:FR, W->Y, DEC YA
```

```
RR*MB:FR, V-Y:H->V,
```

```
DEC YA, T-ML->T,
```

```
ML-T->U
```

```
RL*XL:FR, YA->YC, YC->YA,
```

```
T-ML->T, ML-U->U
```

```
ML-V->V, ML-V->W,
```

```
T->X, INC XA
```

```
"U=A(DESC), size data
```

```
"YL=-1.0 OR .5
```

```
"XC=YA=N, U=Y(0)
```

```
"T=SCALE,
```

```
"L=N/2->J
```

```
"RB*0.5
```

```
"X2=2Y(0)->U
```

```
"F=DA(SCALE)
```

```
"DEC SCALE
```

```
"XL=0
```

```
"Limit RA, T=X(0)
```

```
"YA=N, YC=0, W=RC
```

```
"update SCALE
```

```
"set RC, V=X1
```

```
"T=X1, V=Y, O=0
```

```
"-COS*X2
```

```
"U=Y2, V=Y1
```

```
"T=X2, YA=N-K
```

```
"-SIN*Y2, XL=Y2
```

```
"U=x2, T=x1
```

```
"XA=K, YA=N-K+1
```

```
"-COS*x2, store Y2'
```

```
"-SIN*x2, V=Y1
```

```
"x1+SIN*Y2
```

```
"x1-SIN*Y2
```

```
"-COS*Y2, YA=N-K
```

```
"T=X1, U=X2'
```

```
"y1-SIN*x2, y1-SIN*x2
```

```
"store X1'
```

```

V-ML->V, W-ML->W, X->T,
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
END

```

```

"V=Y1', W=Y2'
"T=X1+, -1.0*X1+
"store X2', Y1'
*exit, K=N-K

```

```

TITLE FILTER
ENTRY FILTER
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

```

```

"MACRO FILTER, XYA(DESC) DESC = N, XA(SOURCE)

```

```

" "
" " Also have filter coefficients and filter memory
" " at xya(desc)-1 to -m stored as:

```

```

" XYA-1 KM, BM-1
" XYA-2 KM-1, BM-2

```

```

" "
" XYA-M K1, B1, BO

```

```

" " N and XA are destroyed, the Bj are updated.
" " The source is replaced by the filter output.

```

```

" F=SRCEJ

```

```

" " for m=M-1->0, F=F-Km+1*Bm
" " Bm+1=Bm+Km+1*F

```

```

" " Output = F, BO = F

```

```

FILTER: Y->XA, 1->V

```

```

NEXT: X->T, XA->XC, DY->YC, DY->XA

```

```

X-V:H->V, U->Y, YC->YA

```

```

V->X, DEC XA, DEC YA, 3->YC

```

```

IF H<0 GOTO OPSEQ, X*Y:FR,

```

```

X->XL, Y->W, DEC XA,

```

```

INC YA(2)

```

```

Y->J, DEC YA(YC)

```

```

T-ML->T, W:H->V, DEC J,

```

```

X*Y:FR, Y->W, DEC XA,

```

```

INC YA

```

```

XL*T:FR, DEC J

```

```

T-ML->T, DEC YA(2),

```

```

X*MB, INC XA

```

```

ML*V->U, W:H->V, X->XL,

```

```

DEC XA(2), MA*Y:FR,

```

```

Y->W, INC YA(YC)

```

```

U->Y, DEC YA(2), XL*T:FR,

```

```

DEC J, IF J>0 GOTO LOOP

```

```

T->U, XC->XA, 1->W

```

```

U->Y, INC YA, ML*V->U,

```

```

W:H->V, T->X, INC XA,

```

```

GOTO NEXT

```

```

END

```

```

"point at source

```

```

"T=SRCE, XC=PTR

```

```

"DEC N store B1

```

```

"New N

```

```

"done?

```

```

"KM*BMM-1

```

```

"set J=M, Y=BM-2

```

```

"SRC-KB, V=BM-1

```

```

"KM-1*BM-2

```

```

"KM*F

```

```

"F=F-KM+1*BM

```

```

"set MA=KM-1

```

```

"BM+1=BM+KM+1*F

```

```

"KM-1*BM-2

```

```

"store BM+1

```

```

"EM*F, M-1->M

```

```

"T=U=F

```

```

"store BO, U=B1

```

```

"store F in output

```

```

TITLE FINV
ENTRY FINV
EXPAND
NOLIST
SYMBOL

```

```

"MICRO SUBROUTINE FINV
" Compute INV(UV/W), ABS (U) >= .5
" Assumes a 128 word table of inverse values:
" TAB(1) = 128/(1+128), only 16 bit values are needed

"Algorithm for INVERSE (D)
" Let D=D1/256 + EPS such that
" 128 <= D1 <= 256 and -1/512 <= EPS <= 1/512
" Look up TVAL(D1-128) = 128/D1 (16 bit accuracy)
" Compute .5ID1 = TVAL-2EPS*TVAL (16 bit accuracy)
" Then D*ID1 = 1+E and ABS (E) < 2**-16.
" Compute .5ID2 = .5ID1*(2-ID1*D), (16x32 bit multiples)
" The result, ID2, satisfies D*ID2 = 1-E**2
" The output scale is W' = 1-W.

```

```

EQU INVTAB='1000'0

```

```

FINV:  U*'256'D:FR, U->T, U->XL
        U:G
        W->YL, V->W
        XB=INVTAB-'128'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-MR->U, O->T, V*MB
        UM+UM:GH->UV, ML+T->T
        T:G, T*U:FR
        '40000'0->T, U:H->V, MR->U
        IF G>0 GOTO DIVE, W:H
        U-ML->U, W:H
        '7777'0-V->U, W:H
        U*W:2P
        MA*XL:22, IF H=0 GOTO Z
        OO-MLMR:CH->UV,
        T-GASN-1*CCO:F->T
        TU-MLMR:CH->UV, MA+V:2P,
        O->T, '17777'0->W
        MA+V:2P, '777'0->V
        MA+U:22, T:F->U,
        MLMR+VM:CH->VM,
        '8'D->XL
        MLMR+UV:CH->UV,
        GASN+T*CCO:F->T,
        1*YL, 1->W
        MLMR+TU:FC->TU, XL+V:PP,
        IF G<0 GOTO CORRECT
        MA+U:PP, T-'10000'0:G,

```

```

        W-MR->W
        MA+T:P2, ML:H->V, O->U, T->X
        MLMR+UV:GH->UV,
        IF G<0 GOTO EXIT
        '140000'0->U
        1+W->W, O->V
        MR+U->U, RTN
        EXIT: '128'D*U, V-ML->V
        NEG:  V->RA, -1->V
        RL+RL:FR, RL:G,
        GOTO COM
        Z:  O->U, O->V, GOTO C
        CORRECT: TU-GASN:FG->TU,
        W-MR->W
        MA+U:PP, T:G,
        GOTO FIN
        OF:  O->V, U-V->W, RTN
        END

"check for 1.0, 1-SCALE
"shift top
"jump if not 1.0

"will halve mantissa
"lnc SCL to compensate
"get top, exit
"shift index back, RA (-TVAL)
"fetch -TVAL
"compute TVAL+TVAL
"test for TVAL=-1
"TUV=-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 0

```

```

FLOAT: '31'D->M, U:G, V:H
FLOAT2: ABS U->V, MA*V:PP
IF CH=0 GOTO Z, W-SCLV->M,
IF SHIFT SMALL GOTO FIN,
MA*U:P2
IF SHIFT OVFL GOTO BIG,
MLMR->UV, O->T
U:G, V:H, GOTO FLOAT2
Z: O->U, O:H->V, '100000'O->M
EXIT: MR+U->U, RTN
BIG: MLMR+TU:FG->UV,
'48'D-M->M
'100000'O XOR U->M, M->U
U:H->M, M->U, RTN
END

```

```

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

```

```

TITLE FLOG
ENTRY FLOG
EXT FLOAT2
EXPAND
NOLIST
SYMBOL

```

```

"MICRO SUBROUTINE FLOG Floating point Logarithm
" Input data is in UV/W
"Uses two tables in ROM: the first is four words:
" 2*256, ---
" 2.5*256, RA(ln(2))
" 3*256, RA(ln(2.5))
" ---, RA(ln(3))
" the second is 67 words = ln((n*192)/128), n=0,64
" ln(2.5), ln(3)

```

```

EQU THRFRTH='60000'O, TWOTHRDS='52525'O, INVTAB='1000'O,
PTRTAB='1304'O, LOGTAB='1201'O, LN2L='54271'O,
LN2R='5774'O

```

```

FLOG: PTRTAB->RA
1->RC
U:G, XB->T, XB=THRFRTH,
M->YL, O->W
A: U-T:G, XB->T, XB=TWOTHRDS
U-T:G, U:H, IF G>O GOTO T1
M-U:V:GH->UV, XB->T, XB=THRFRTH
IF H<O GOTO A, T:G
O-T->U, T->M, GOTO FLOAT2
T1: IF G<O GOTO T1, T:G,
RL*V:PP, INC RA
MA*U:PP, RR->YC
MLMR->UV, XB->T,
XB=INVTAB-'128'D
MLMR+TU:GH->UV
U->RA, O->T, XB+U->U,
XB=LOGTAB-INVTAB-'64'D
RL*W:PP
RR*V:PP, '77777'O->M, O->T
RL*MB:PP, MLMR+OW:GH->VM
T:OCO:F->U, U->RA,
MLMR+VM:GH->VM
MLMR+UV:GH->UV
UV*UV:GH->UV
"Compute D*D/2
U*V:PP, U->T, V->U
MA*T:PP, '37777'O->M, O->V
OML*VM:GH->VM
VM*VM:GH->VM
ML*V:HCO:G->V, MR*W:H,
'77777'O->M
"Compute 64D**3, D, and D**3/3
T*V:PP
MA*256'D:PP
U*MB:PP, O->T, RR->M,

```

```

"first multiplier
"test for M>0
"save SCL in YL
"M:75
"M:2/3, M<O, M>0?
"negate M
"M/=0?
"float lg neg log
"find segment
"start C*MR
"C*MR->UV
"table base
"U=RR(TINV)
"fetch TINV
"U=RA(TLOG)
"INVR*EPSL
"INVL*EPSL
"fetch TLOG
"UV=128D
"UV=256D
"DL*DR, TU=256D
"DL*DL
"cross term
"double it
"V=D*D/2
"RND for D**3
"D*D**2/2
"shift DL
"shift DR

```

```

MLMR-IV:CH->UV, MR+HCO:G->U, MR+M:H
U+253, O:FR
OML+UV:CH->UV, RL+1:PP
OML+UV:CH->UV, LN2R+YL, O->I
MRV+UM:CH->VM, YC->RA
RL-V->V, LN2L->XL
VRR-OM:CH->VM,
XL+MB
MLMR-VM:CH->VM, CASN-I-1+GCO:F->U
MLMR+UV:CH->UV
MA+V:PP, ABS U->V
SHIFT(SCLV)+MB:PP, M:H->V,
O->I, SCLV->W
IF SHIFT SMALL GOTO SM,
MA+V:PP
MA+U:P2, 1->W, MLMR->UV
OML+UV:CH->UV
MR+U:HCO:G->U, V:H, GOTO FLOAT2
MA+U:P2, ML-T:F->U,
MB->V, '16'D-W->M
OML+UV:CH->UV
MR+U->U, RTN
END
    
```

SM:

```

"U=64D**3
"DL-D**2/2
"64D**3/192
"UV=D-D**2/2
"D-D**2/2+D**3/3
"start LN(2)R+S
"sub L2 from it
"LJ=LN(C)-L(2)
"start LN(2)*S
"S*LN(2)R-LN(C)
"FLOG+2**31
"start shift
"DSCL->M
"shift <=15?
"shift bottom
"SCL=1
"middle shifted
"bottom shifted
"top shifted
"shift top
"middle shifted
"bottom shifted
"top shifted,exit
    
```

```

TITLE FLTS
ENTRY FADDS, FSUBTS, FMULTS
EXT SFL0AT2
EXPAND
NOLIST
SYMBOL

"Assume OP1 mantissa in U, SCL in W;
" OP2 mantissa in X, SCL in Y

"MICRO SUBROUTINE FADDS
FADDS: Y-W:H->V, O->I
SHIFT(V)*X:P2, T-V:G->V
IF H>O GOTO SUV, T-V:G,
SHIFT(V)*U:P2
SUM: IF G>O GOTO SFL0AT2, U:H->V,
ML+U->U
X+V->U, GOTO SFL0AT2
Y->W, X->U, GOTO SUM

"MICRO SUBROUTINE FSUBTS
FSUBTS: Y-W:H->V, O->I
SHIFT(V)*X:P2, T-V->V
IF H>O GOTO SSUV,
SHIFT(V)*U:P2
IF G>O GOTO SFL0AT2, U:H->V,
ML-U->U
X-V->U, GOTO SFL0AT2
Y->W, X->U
U-ML->U, GOTO SFL0AT2

"MICRO SUBROUTINE FMULTS
FMULTS: X+U:FR
Y+W->M
ML->U, GOTO SFL0AT2
END

TITLE FLTS
ENTRY FADDS, FSUBTS, FMULTS
EXPAND
NOLIST
SYMBOL

"Assume OP1 mantissa in U, SCL in W;
" OP2 mantissa in X, SCL in Y

"MICRO SUBROUTINE FADDS
FADDS: Y-W:H->V, O->I
SHIFT(V)*X:P2, T-V:G->V
IF H>O GOTO SUV, T-V:G,
SHIFT(V)*U:P2
SUM: IF G>O GOTO SFL0AT2, U:H->V,
ML+U->U
X+V->U, GOTO SFL0AT2
Y->W, X->U, GOTO SUM

"MICRO SUBROUTINE FSUBTS
FSUBTS: Y-W:H->V, O->I
SHIFT(V)*X:P2, T-V->V
IF H>O GOTO SSUV,
SHIFT(V)*U:P2
IF G>O GOTO SFL0AT2, U:H->V,
ML-U->U
X-V->U, GOTO SFL0AT2
Y->W, X->U
U-ML->U, GOTO SFL0AT2

"MICRO SUBROUTINE FMULTS
FMULTS: X+U:FR
Y+W->M
ML->U, GOTO SFL0AT2
END

"man1*man2
"SCL1*SCL2->SCL
"result ->U
    
```

```

TITLE FLTSOPS
ENTRY SADD, SSUBT, SMULT
EXT FADDS, FSUBTS, FMULTS, 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 SADD, ARG DOP=0.2,4 OP2 at XY(ARG)
" Computes X/Y + U/V -> U/V

```

```

SADD: I->YA, W->XA, V->W, "XYA(OP2)
CALL FADDS "do FADD
W->V, READ, GOTO EXMAC "exit

```

```

"MACRO SSUBT, ARG DOP=0.2,4 OP2 at XY(ARG)
" Computes X/Y - U/V -> U/V

```

```

SSUBT: I->YA, W->XA, V->W, "XYA(OP2)
CALL FSUBTS "do FSUBT
W->V, READ, GOTO EXMAC "exit

```

```

"MACRO SMULT, ARG DOP=0.2,4 OP2 at XY(ARG)
" Computes X/Y * U/V -> U/V

```

```

SMULT: I->YA, W->XA, V->W, "XYA(OP2)
CALL FMULTS "do FMULT
W->V, READ, GOTO EXMAC "exit
END

```

```

TITLE FMULT
ENTRY FMULT
EXT FLOAT2
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+W->W, DEC XA, DEC YA "SCL1+SCL2
X*V:2P, 0->T "ML2*MR1
Y*U:P2, 0->U "MR2*ML1
X*MB:22, INC XA, INC YA, "ML2*ML1
GASNML:TU:FG->TU, "fix SCL for FRAC MULTI
I+W->W "add MR2*ML1->UV
GASNML:TU:FG->UV "add ML1*MR1, float
MLMR+UV:GH->UV,
GOTO FLOAT2
END

```

```

TITLE FSQRT
ENTRY FSQRT
EXIT FLOATZ
EXPAND
NOLIST
SYMBOL

```

```

"MICRO SUBROUTINE FSQRT Floating Square Root
" Input data is in UV/W
"Uses two tables in ROM:
" the first is 128 words = SQRTAB(n)=art(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 SQBASE='1367'O, IBASE='1000'O, N128='200'O

```

```

FSQRT: U*'1000'O:2P, U:G
V->U, W->Y, '177'O AND U->V
IF G>O GOTO CONT, ML->V, V->T
O->U, O:H->V, RTN
O->T, ML->W
YB*W->M, YB=IBASE-N128
W->RA, XB*W->M,
XB=SQBASE-IBASE
RR*U:PP
RL*V:PP, '77777'O->V
T*GCO:F->T, MLMR*OV:CH->UV,
MA*U:PP
OML*TU+HCO:FG->TU, MR*V:H,
W->RA
MLMR*TU:EG->UV, O->W,
'100000'O->T
UV*UV:CH->UV
TW-UV:CH->UV, U->X,
V->Y, U->XL,
U*V:PP
O->T, '77777'O->W
MLMR*OW:CH->UV, XL*XL:PP,
U->XL, V->YL
OML*TU+HCO:FG->TU, MR*W:H
MLMR*TU:EG->UV
U*YL:PP, '40000'O->T
XL*V:PP, '77777'O->W
MLMR*OW:CH->UV, MA*U:PP
OML*TU+HCO:FG->TU, MR*V:H
MLMR*TU:EG->UV, O->M,
'100000'O->T
UV*UV:CH->UV
XY-UV:CH->UV, DEC YA
RR*U:PP, O->T, '17777'O->W
RL*V:PP, 4->XL
MLMR*OW:CH->UV, MA*U:PP
T*GCO:F->T, MLMR*UV:CH->UV,
'100000'O*Y:P2, Y->W
MLMR*TU:EG->TV, XL*V:PP,

```

```

"move u 9 left
"mask 9 lead 0
"M<O?
"M<=O, O->UV
"N->W
"E=UV
"fetch ITAB
"begin ITAB+E=D

"fetch SQBASE
"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*D/2*(1-D)
"SQRTB*(D term)

```

```

"begin S/2

```

```

'00001'O AND W:H
ML->W, SQBASE->RA
TV*TV:CH->UV, O->T,
IF H>O GOTO SODD
ML*V:H->V, U:G, GOTO FLOATZ
ML*V->V, 1+W->W, RR*U:PP
RL*V:PP
MLMR->UV, MA*U:PP, O->T
OML*TU+HCO:FG->TU, MR*V:H
MLMR*TU:EG->UV
UV*UV:CH->UV, GOTO FLOATZ
END

```

```

SODD:

```

```

"test S odd
"INT(S/2)->W
"SQRTB*(D term)

"art(1/2)*art(M)

```

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 XYA->--,SCL
 " XYA-1-> ML, MR
 " Computes Second operand - First operand

FSUB: Y-W->W, DEC YA, DEC YA,
 1*Y
 Y->YL, 0->I
 IF H=O GOTO SUV, T-W->W,
 SHIFT(W)*YL:PP
 IF SHIFT SMALL GOTO NORM,
 MA*X:P2, MR+W->W
 IF SHIFT OVEL GOTO FLOAT2,
 U:G, V:H, U->I,
 INC YA, INC YA

GASML-TV:EG->UV,
 V:H, GOTO FLOAT2
 NORM: IF H=O GOTO NOSH,
 TML-UV:GH->UV
 ML*MR+UV:GH->UV, INC YA,
 INC YA, GOTO FLOAT2
 NOSH: XY-UV:GH->UV, INC YA,
 INC YA, GOTO FLOAT2

" SCL2 > SCL1, will shift MLR1
 SUV: SHIFT(M)*V:PP, MR->W
 IF SHIFT SMALL GOTO NS,
 MA*U:P2, Y->V
 IF SHIFT OVEL GOTO FLOAT2,
 X->U, V:H, X->I,
 INC YA, INC YA
 TU-GASML-TV:EG->UV,
 V:H, GOTO FLOAT2
 NS: XV-OML:GH->UV
 UV-ML*MR:CH->UV, INC YA,
 INC YA, GOTO FLOAT2
 END

TITLE INltCHK
 ENTRY INIT, INI, 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,
 CMBS='30'0,
 DI='32'0,
 STK=0, "(really 10)
 EA='31'0,
 STKBS='12'0,
 HST='12'0,
 DIN=0,
 DOUT=1,
 PNL5W='17'0,
 MACBASE='140000'0

CHK: PNL5W->DEV
 MACBASE->W
 IF STAT=0 GOTO INI
 GOTO GOTO, ENABLE
 LOC CHECK+'10'0
 INIT: GOTO START, 0->V, CLR S
 START: V->DA(DIN)
 V->DA(DOUT)
 STKBS->W
 W->DA(STK), GOTO CHECK
 CALL SETP7
 IF STAT=0 GOTO BRSC
 CMBS->W, DISABLE
 W->DA(DI)
 READ, GOTO EXMAC
 HST->DEV
 CMBS->W
 W->DA(CMD)
 IF STAT=1 GOTO .
 CMD->DEV
 INT HOST, RTN
 END

"Address SW
 "entry point
 "HOST INIT
 "start
 "set PSA
 "set PORT 0
 "set PORT 1
 "Init STK
 "set up CMD port
 "wait for CMD
 "point at CMD buffer
 "fetch first MACRO
 "select HST STATUS
 "set up CMD port
 "wait for ATTN clear
 "DEV=CMD PORT
 "Interrupt HOST

TITLE INTSRV
ENTRY INTSRV, CALL, GOTO, OPSEQ
EXT CMD
EXPAND
NOLIST
SYMBOL

"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, 0->W,
IF STAT=1 GOTO CMD

1->T
W->DEV, W->DA(DI), T+W->W
INTLOOP: CLR SBIT, W->DEV, T+W->W
IF STAT=0 GOTO INTLOOP, READ
DX->W, GOTO CALL
"W=EA(INT routine)

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

CALL: DA(EA)->U, U->T
WRITE (STK) U, T->U
"save old EA,U
"EA->STK

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

GOTO: W->DA(EA)
OPSEQ: READ(EA)
EXEC MACRO
END
"set new EA
"fetch next MACRO
"do it

TITLE LATRED
ENTRY LATRED
EXT EXMAC
EXPAND
NOLIST
SYMBOL

"MACRO LATRED, XYA(DESC) DESC = K, N
BA, FA

" For j = 0, ..., N
" F(EA+j) = F(EA+j) + K*B(BA+j)
" B(BA+j) = B(BA+j) + K*F(EA+j)

LATRED: Y->J, X*MB:22, INC XA, INC YA
X->YA, Y->XA, DEC J
MA*Y:FR, INC YA
MA*X:FR, X->T, INC XA
ML*T->T, W:H->V, MA*Y:FR,
ML*V:H->V, Y->W, INC YA
T:G->U,
MA*X:FR, X->T,
DEC XA, 3->YC
U->X, INC XA(2), V->U,
ML*T->T, W:H->V,
MA*Y:FR, Y->W,
DEC YA(2)
U->Y, INC YA(YC), ML*V->V,
T:F->U, MA*X:FR,
X->T, DEC XA, DEC J,
IF J>0 GOTO LTRLP
READ, GOTO EXMAC
END
"set J=N, MA=X
"set BA,FA
"K*B(0)
"K*F(0), T=F(0)
"T=F(0), V=B(0)
"K*B(1), W=B(1)
"V=B(0), U=F(0)
"K*F(1), T=F(1)
"set up YA INC
"F(j)=X, U=B(j)
"T=F(j+1), V=B(j+1)
"K*B(j+2), W=B(j+2)
"B(j)=Y
"V=B(j+1), U=F(j+1)
"K*F(j+2) <T=F(j+2)
"j+1->j
"exit

LTRLP:


```

STF16: U->I, DA(16)->U
        W->DA, GOTO STF
STF17: U->I, DA(17)->U
        W->DA, GOTO STF

STF:   XB=U, YB=U, WRITE, YB->DA,
        T->U, GOTO OPSEQ
END

```

```

"get file value
"set destination
"get file value
"set destination

"F->D, restore F
"restore U, exit

```

```

TITLE LOGPWR
ENTRY LOGPWR
EXT FLOAT2, FLOG, OPSEQ
EXPAND
NOLIST
SYMBOL

"MACRO LOGPWR, XYA(DESC)  DESC=XYA(DATA),N
"  U = SCALE OF DATA
"  Converts 32 bit data to floating point,
"  computes the logarithm, 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
LOOP:   Y:H->V, X->U
        CALL FLOG
        W-'7'0->W
        SHIFT(W)*V:PP
        MA*U:P2
        ML:H->V, 0->U
        ML*U:V:GH->UV, Y->W
        U->X, V->Y, INC XA, INC YA,
        DEC J, IF J>0 GOTO LOOP
        GOTO OPSEQ
END

```

```

"save SCL, float
"compute log
"compute shift
"shift V
"shift U
"shifted V
"result in UV, get SCL
"store result

"data pt in UV, SCL=W
"save SCL, float
"compute log
"compute shift
"shift V
"shift U
"shifted V
"result in UV, get SCL
"store result

```

```

TITLE MOVECX
ENTRY MOVECX
EXT EXMAC
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
NOOP                          "wait for DEC J
V->X, INC XA, DEC J,          "store val
READ, T->V, GOTO EXMAC
END

```

```

TITLE MOVERD
ENTRY MOVERD
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

```

```

EQU DI='12'0

```

```

"MACRO MOVERD, RA, DA, N DOP=1,3,5,7

```

```

MOVERD: DY->J                  "COUNT
W->RA, DX->W                  "set SOURCE
W->DA(DI:D), DEC J           "set DEST
I->RC, U->I                  "INC=1
RL->U                          "get RL
LOOP: WRITE URR, INC RA, DEC J, "store RLR
      IF J>0 GOTO LOOP
      T->U, GOTO OPSEQ
END                             "exit

```

```

TITLE MOVES
ENTRY MOVEDD, MOVEXD, MOVEXD, SETD,
SMVXD, SMVXD
EXT OPSEQ
EXPAND
NOLIST
SYMBOL

```

```

"MACRO MOVEDD, DA(SOURCE), DA(DEST), COUNT DOP=DY->M, READ
"MACRO MOVEXD, DA(SOURCE), YA(DEST), COUNT DOP=DY->M, READ
"MACRO MOVEXD, DA(SOURCE), XA(DEST), COUNT DOP=DY->M, READ
"MACRO MOVEXD, XA(SOURCE), DA(DEST), COUNT DOP=DY->XA, READ
"MACRO MOVEXD, YA(SOURCE), DA(DEST), COUNT DOP=DY->XA, READ
"MACRO SMVXD, DA, XA, CNT DOP=1,3,5,7
"MACRO SMVXD, DA, YA, CNT DOP=1,3,5,7
"MACRO SETD, ARG, xxx, DA DOP=1,3,5,7

```

```

EQU DI='12'0

```

```

MOVEDD: DY->J, V->I
DEC J, W->DA(DI), DX->V
DDL: READ, V->DA, I+W->M,
I+V->V
DEC J, IF J>0 GOTO DDL,
XB=DX, WRITE,
W->DA
GOTO OPSEQ, I->V
"save V
"V=dest DA
"fetch word, DA=DEST
"inc both DA's
"store word
"DA=source
"restore V, done
"MACRO MOVEXD, DA(SOURCE), YA(DEST), COUNT DOP=DY->M, READ
MOVEXD: DY->J, DX->YA
DEC J, W->DA(DI)
"set DA
"fetch first word
"first d->v, fetch second
"store Y(j), D(j+1)->v
"branch CNT-1 times
"done
GOTO OPSEQ

```

```

"MACRO MOVEXD, DA(SOURCE), XA(DEST), COUNT DOP=DY->M, READ
MOVEXD: DY->J
W->DA(DI), DEC J, DX->W
"set COUNT
W->XA, READ
"set DA
DX->X, READ, INC XA,
"D->X, fetch next
DEC J, IF J>0 GOTO
"branch CNT-1 times
GOTO OPSEQ

```

```

"MACRO MOVEXD, XA(SOURCE), DA(DEST), COUNT DOP=DY->XA, READ
MOVEXD: DY->J, DX->W
"set COUNT
MOVEXD: W->DA(DI), DEC J
NOOP
"set DA
WRITE X, INC XA, DEC J,
"wait for count
IF J>0 GOTO
"branch CNT-1 times
GOTO OPSEQ

```

```

"MACRO MOVEXD, YA(SOURCE), DA(DEST), COUNT DOP=DY->YA, READ

```

```

MOVEXD: DY->J, DX->W
"set count
MYXD: W->DA(DI), DEC J
"set DA
V->I, Y->V, INC YA
"first Y
WRITE V, Y->V, INC YA,
"V->, Y->V
DEC J, IF J>0 GOTO
"branch CNT-1 times
GOTO OPSEQ, I->V
"resetre V, exit

```

```

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

```

```

"MACRO SMVXD, DA, XA, CNT DOP=1,3,5,7
SMVXD: DX->XC, DY->J
"set J
XC->XA, GOTO MYXD2
"set XA

```

```

"MACRO SMVXD, DA, YA, CNT DOP=1,3,5,7
SMVXD: DX->YA, DY->J
"set YA, J
GOTO MYXD1

```

```

"MACRO SETD, ARG, xxx, DA DOP=1,3,5,7
SETD: DY->DA(DI), U->I, V->U,
W:H->V
"set DEST, ARG->V

```

```

WRITE V, U:H->V, I->U,
"ARG->D
GOTO OPSEQ
"exit
END

```

```
TITLE MULTL
ENTRY MULTL
EXT EXMAC
EXPAND
NOLIST
SYMBOL
```

```
"MACRO MULTL, XYA(DESC)  DESC = XYA, L
```

```
MULTL:  Y->J, X->YA, X->XC
```

```
XC->XA, DEC J
```

```
X*Y:22, INC XA, INC YA
```

```
NOOP
```

```
MULTLL: X*Y:22, DEC XA, DEC YA,
```

```
MLMR->UV
```

```
U->X, V->Y, INC XA(2),
```

```
INC YA(2), DEC J,
```

```
IF J>0 GOTO MULTLL
```

```
READ, GOTO EXMAC
```

```
END
```

```
"set up J,YA
```

```
"set up XA
```

```
"start first
```

```
"store result
```

```
"repeat L times
```

```
TITLE ORMOD
ENTRY ORMOD
EXT EXMAC
EXPAND
NOLIST
SYMBOL
```

```
"MACRO ORMOD, A(DESC)  DESC = XA, L
```

```
" Computes OR(ABS Xi)  i=0,...,L-1
```

```
" Returns U = Number of leading zeros, V = OR
```

```
ORMOD:  Y->J, X->XC
```

```
XC->XA, O->U, O:H->V
```

```
ABS X->U, U OR V->V,
```

```
INC XA, DEC J,
```

```
IF J>0 GOTO ORLP
```

```
SCLV->U, READ, GOTO EXMAC
```

```
END
```

```
"set up J
```

```
"first value 0
```

```
"Do loop L+1 times
```

```
"leading zero cnt
```

TITLE POWER
 ENTRY POWER
 EXT EXMAC
 EXPAND
 NOLIST
 SYMBOL

"MACRO POWER, XVA(DESC) DESC = XVA, N

```
POWER: Y->J, X->XC, X->YA
XC->XA, J->XC, DEC J
X*X, INC XA
Y*Y, INC YA, O->I
MLT->T, MR->U, X*X, INC XA
MLMR*TV:GH->VM, Y*Y,
DEC YA, O->I
LOOP: W->Y, INC YA(2), MLT->T,
V->X, INC XA(XC).
MR->U, X*X, DEC XA(2)
MLMR*TV:GH->VM,
Y*Y, DEC YA, O->I,
DEC J, IF J>0 GOTO LOOP
READ, GOTO EXMAC
END
```

```
"set COUNT
"set XA INC
"XO*YO
"Y1*Y1
"X1*X1
"XO*YO+YO*YO
"Y1*Y1
"store P(j)R
"X(j+1)*X(j+1)
"store P(j)L
"Y(j+2)*Y(j+2)
```

TITLE RADIAN
 ENTRY RADIAN, AQUAD, BQUAD, CQUAD, DQUAD
 EXT FMULT, FADD, FLOAT2
 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 M->UV, s+2->V, v->XO, u->X1, s+2->YO,
 " XA, YA->, and 2/pi has been fetched

"RADIAN returns these stored XY values to the calling subroutine
 " for the determination of the proper quadrant for sin and cos,
 " with XA, YA->8, n->u, and TRG(128-n) ->tw; 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) XVA->6
" sin(quado) = cos(quad3)
"BQUAD: cosD*TRG(128-n) + sinD*(-TRG(n)) XVA->6
" sin(quadi) = cos(quado)
"CQUAD: cosD*(-TRG(n)) + sinD*(-TRG(128-n)) XVA->6
" sin(quad1) = cos(quadi)
"DQUAD: cosD*(-TRG(128-n) + sinD*TRG(n)) XVA->6
" sin(quad3) = cos(quad2)
```

EQU PIDTWO='1771'O, TWODPI='1772'O, TRASE='1570'O

```
RADIAN: RR*V:PP, W:H, O->I
RL*MB:PP, INC YA
MLMR:GH->UV, RR*U:P2,
IF H<O GOTO RSHIFT
MLMR*OU:GH->UV, T*CO:F->T,
RL*MB:P2, V->XL,
PIDTWO->Y, DEC YA
MLMR*UV:GH->UV, GASN*T*CO:F->T,
SHIFT(Y)*XL:PP, INC YA
MLMR*TV:FG->TV, MA*V:PP,
IF SHIFT SMALL GOTO LSM
MA*U:PP, ML->M
MA*T:P2, MLMR*OM:GH->VM,
'200'O->YL
MLMR*OV:GH->UV, Y->RA, DEC YA
MR*U:G->U, V*YL:PP,
'777'O AND V->V
EPSL: '400'O AND V, '400'O*400'O:PP,
W->YL, '400'O->XL
U->X, ML->U, '77777'O->W
IF H=O GOTO CONT
ML*U:G->U, V-'1000'O:H->V
CONT: RR*YL:PP, U->Y, O->I
RL*YL:PP, ML->M
MLMR*OM:GH->VM, RL*MB:P2
GASNML*TV:HCO:FG->TV, MR*W:H
"begin 2/pi*M
"YA=1
"YA=0
"begin shift
"YA=1
"shift v
"shift u
"shift t
"128=YL
"fetch pi/2, YA=0
"V*128
"mask 7 lead 0
"INTQ-XO,N=U
"ROUND
"begin E*pi/2
"N=YO,E=.VM
```

```

MLMR-TV:EG->UV, O->W,
  INC XA, INC YA
U->G, V:H, CALL FLOAT2
U->X, V->Y, INC XA, INC YA
W->Y
DELTA:
XBXB-UV:GH->UV, XB=0
CALL FMULT
INC XA, INC YA, W-1->W
V->Y, U->X, INC XA, INC YA
W->Y, DEC XA(2), DEC YA(2)
CALL FMULT
XB+V:PP, XB='52525'0
MA+U:P1, '77777'0->T
MLMR-OV:GH->UV, O->T
MLMR-UV:GH->UV, CASN+T+GCO:F->T
MLMR-TU:GH->UV
CALL FADD
W->Y, 1->W, DEC XA, DEC YA
U->X, V->Y, O->V,
  INC XA(2), INC YA(2)
'40000'0->U
INC XA, INC YA, CALL FADD
W->Y, '11'0->T, DEC XA, DEC YA
U->X, V->Y, DEC XA(2), DEC YA(2)
DEC XA, DEC YA, YB->W, T->U,
  YB=TBASE
Y->W->M, Y->Y, Y->V, INC YA(2)
SHIFT(SCLV)*YL:PP, X->T,
  U-SCLV->V, INC XA(2),
  INC YA(2)
W->RA, XB+W->W, XB=TBASE+'200'0,
  INC XA(2), INC YA
MR->U, 1+V:H->V, W->Y,
  INC XA(2), INC YA
RR+U:PP, T->X
RL+MB:PP, '77777'0->W
ML+HCO:G->V, MR+W:H, V->W
MLMR-OV:GH->UV, CALL FLOAT2
X->T, W->Y, DEC XA, DEC YA
U->X, Y->W
V->Y, XB+W:H->V, XB:G,
  XB=TBASE, INC XA(2),
  INC YA(2)
V->RA, W:H->V, '11'0->U,
  INC XA, INC YA
SHIFT(SCLV)*V:PP, U-SCLV->W
T->X, 1+W->W
MR->U, '77777'0:H->V
RR+U:PP
RL+MB:PP
ML+HCO:G->V, MR+V:H
MLMR-OV:GH->UV, CALL FLOAT2
X->T, W->Y, DEC XA, DEC YA
U->X, V->Y, T:G->U, U->T,
  INC XA, INC YA, RTN
DQUAD: OO-UV:GH->UV, DEC XA(2), DEC YA(2)

```

```

CALL FMULT
W->Y, DEC XA, DEC YA
U->X, V->Y, INC XA(2), INC YA(2)
X->W, Y:H->V, INC XA, INC YA
Y->W, DEC XA(2), DEC YA(2)
DEC XA(2), DEC YA(2), CALL FMULT
INC XA(2), INC YA(2), CALL FADD
DEC XA(2), DEC YA(2), GOTO DONEC
DEC XA(2), DEC YA(2), CALL FMULT
W->Y, DEC XA, DEC YA
U->X, V->Y, INC XA(2), INC YA(2)
OO-XY:GH->UV, INC XA, INC YA
Y->W, DEC XA(2), DEC YA(2)
DEC XA(2), DEC YA(2), CALL FMULT
INC XA(2), INC YA(2)
OO-XY:GH->UV, INC XA, INC YA
Y->W, DEC XA(2), DEC YA(2),
  CALL FMULT
DEC XA(2), DEC YA(2), CALL FADD
DEC XA(2), DEC YA(2), RTN
AQUAD: DEC XA(2), DEC YA(2)
W->Y, DEC XA, DEC YA
U->X, V->Y, INC XA(2), INC YA(2)
X->W, Y:H->V, INC XA, INC YA
Y->W, INC XA(2), INC YA(2),
  CALL FMULT
DEC XA(2), DEC YA(2), CALL FADD
DEC XA(2), DEC YA(2), RTN
DONEA: MLMR+OU:GH->UV, T+GCO:F->T,
  RL+MB:P2, PIDTMO->Y, DEC YA
MLMR-UV:GH->UV, '200'0->YL,
  CASN+T+GCO:F->T
MLMR+TU:FG->TU, 5->XL,
  5+W:H
SHIFT(Y)+U:PP, INC YA(2)
MA+T:P2, DEC YA
ML->W, Y->RA, DEC YA, T:G->U
MLMR+OV:GH->VM, '777'0->T
T AND V->V, O->U, V+YL:PP,
  IF G<O GOTO RQ3
GOTO EPSL
RQ3: 3->U
GOTO EPSL
DELELT: X->V, O->T, INC XA
X->U, W-2->W, DEC XA, DEC YA
T->X, '0'0->Y

```

```

"XYA=4
"COS*TRG(128-N)
"COS PROD=XY34
"XYA=6
"XZA=4
"XYA=2
"XYA=4
"XYA=2
"XYA=4
"COS*TRG(128-N)
"XYA=5
"-TRG(N)
"XYA=4
"XYA=2
"XYA=4
"XYA=2
"-TRG(128-N)
"XYA=2
"SIN PROD=XY12
"XYA=5
"-TRG(N)
"XYA=4
"XYA=2
"XYA=0
"XYA=4
"XYA=2
"SIN PROD=XY12
"XYA=4
"XYA=6
"XYA=4
"XYA=2
"YA=0
"128=YL
"shift u, YA=2
"shift t, YA=1
"fetch pl/2
"MASK 7 LEAD 0
"XA=1
"S=W, XYA=0
"O=N=INTQ

```

```

Q3: IF G>0 GOTO QO
    3->X
    '200'0->Y
QO: INC XA, INC YA, U:G, V:H,
    U->X, V->Y, INC XA, INC YA
    W->Y, GOTO DELTA

LSM: MA:U:PP
    MA:T:P2, ML->W, '200'0->YL
    MLAR+OW:CH->VW, '777'0->I,
        Y->RA, DEC YA
    MLAR+OV:CH->UV
    T AND V->V, V*YL:PP, GOTO EPSL
    END

```

```

"INTQ=3=XO
"N=128=YO
"XYA=2
"shift u
"shift t,128=YL
"YA=0
"fetch p1/2

```

```

TITLE RANDOMS
ENTRY RANDOMS
EXPAND
NOLIST
SYMBOL

"MACRO RANDOMS, XYA(DESC)  DESC = R(-1), RM
"                               SCL, N
"                               XA
"   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:FR
        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
"YL=RM
"set J, XL=SCL
"V=R(0)
"start R(0)*RM
"start R(0)*SCL
"J=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)

```

TITLE RECORD
 ENTRY RECORD
 EXT EXMAC
 EXPAND
 NOLIST
 SYMBOL

"MACRO RECORD XYA (CHNLBLK) CHNLBLK = AREC, MREC
 " "
 " "

EQU EXBS='1420'0,
 EXPEND='1440'0,
 PMAX='127'D,
 PMIN='36'D,
 DF='12525'0,
 BF='31460'0,
 DTABS='1420'0

RECORD: U-X->U, INC XA,
 EXBS*MB:PP
 U-X->U, U->M, INC XA(2)
 MA*1:PP
 U*X:FR, DEC XA
 IF G<0 GOTO NOTBLNK,
 MR->U
 EXIT: READ, GOTO EXMAC
 NOTBLNK: ML*U->U, EXPEND->T
 U->XA, XA->XC, U-T:G,
 '100000'0->XL
 X*Y:FR, XC->XA, V->U
 XL*XL:P1, X*M:H->V,
 IF G<0 GOTO CHKTH
 NOOP

CHKTH: U-ML:G, PMAX:H->V,
 MA*V:PP
 PMIN->T
 IF G<0 GOTO EXIT, V-ML:G
 U->Y, INC YA, ML-T:G,
 ML:H->V, DF*MB:PP
 MA*V:FR, V->X, Y->U,
 DEC XA(2)
 IF G<0 GOTO UPMAX
 M->Y, INC YA, X*M:H->V,
 INC XA(2)
 IF G<0 GOTO CONT
 MA*PMIN:FR, PMIN->X
 Y->M, XB=DTABS-'6'0,
 U->Y, INC YA, ML*T->T,
 BF*MB:FR
 M->Y, T->YA
 V->X, ML->U, Y:H->V, INC XA
 U->X, INC XA(2), READ
 V->X, EXEC MACRO

"16 word table in Y
 "limit
 "Max PAV value
 "Min PAV value
 "Damping table mult
 "Blanking multiplier
 "8 word table in X

"ACUR-AREC
 "base of EXP DECAY TABLE
 "U=ADISP, M=new PA
 "EXPBS*1
 "start DMP*ADISP
 "ADISP>0?
 "U=EXBS
 "no, exit
 "U=EXPXA
 "fetch EXPT value
 "EXPXA-EXPEND
 "MREC*EXPT,U=MCUR
 "MA=0.5,V=PAV+PA
 "EXPXA<EXPEND?

"MCUR-THRESH
 "start 0.5*(PAV+PA)
 "PMAX-PAV
 "MCUR-.MREC.PAV-PMIN
 "V=PAV
 "U=old PA
 "PAV*DF
 "PAV>PMAX?
 "new PA->Y, V=ACUR
 "PAV>PMIN?
 "no, PMIN->PAV
 "M=old PB
 "PA'->PB, T=YA (DMP)
 "compute BLNK
 "PB'->PC, fetch DMP
 "ACUR->AREC, V=DMP
 "new BLNK
 "new DMP

UPMAX: X->T, INC XA(2), PMAX->V
 GOTO CONT, MA*V:FR, V->X,
 M->Y, M+T->V, INC YA
 END
 "T=AREC, pt at PAV
 "DF*PMAX, PMAX->PAV
 "PA->Y, V=ACUR

```

TITLE RMVXY
ENTRY RMVXY, RMVXX
EXT EXMAC
EXPAND
NOLIST
SYMBOL

```

```

"MACRO RMVXY, XA, YA, N X(XA+j)--Y(YA-j);N
"MACRO RMVXX, YA, XA, N Y(YA+j)--X(XA-j);N

```

```

RMVXY: DX->YA, DY->J
DEC J, X->N, INC XA
NOOP
N->Y, DEC YA, X->W, INC XA,
DEC J, IF J>0 GOTO .
READ, GOTO EXMAC

RMVXX: DX->XC, DY->J
DEC J, XC->XA, V->W
Y->V, INC YA
V->X, DEC XA, Y->V, INC YA,
DEC J, IF J>0 GOTO .
N->V, READ, GOTO EXMAC
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

```

```

EQU EA='31'0

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

```

TITLE SCHED
 ENTRY SCHED
 EXT OPSEQ
 EXPAND
 NOLIST
 SYMBOL

```

"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='12'0, STK='10'0

SCHED: V->T, M:H->V, DA(STK)->W
        YB=BOS+1, YB->DA, W-YB->W
        W->J, O->W
        "get current ptr
        "N=CNT-1
        "set counter
        "fetch entry
        "V=prev stk entry
        " move entry up

SLP: DX->V, READ, INC DA(0)
      WRITE V, DEC J,
      IF J>0 GOTO SLP
      T->V, GOTO OPSEQ
  END
  
```

TITLE SCORE
 ENTRY SCORE
 EXT EXMAC
 EXPAND
 NOLIST
 SYMBOL

```

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

"MACRO SCORE, xxx
" This program is entered at SCORE, below

EQU BIPTR='174'0, POPTAB=BIPTR+4,
    CNDEND=POPTAB+'36'D, CAND=CNDEND+1,
    CNDPTR=CAND+1, BTAB=CNDPTR+1, CTAB=BTAB+2

TOP: V->Y, U-X:G, INC XA, DEC YA
LOOP: U->Y, XA->XC, U-X:G, INC XA,
      "CNDPTR->Y, CAND-CTAB(1)
      "U-X(j), j->XC, CAND->Y
      IF G<0 GOTO LOOP
      "repeat if CAND>CTAB(j-2)
      XC->YA, O->V, BIPTR->XA
      "fetch DPN, pt at BIPTR save
      BTAB->X, BTAB->T, INC XA(2)
      "Init. BIPTR
      Y->U, V->X, INC XA
      3->J, T->YA
      "PANE=O, U=DPN
      "Y=BIAS, 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->YC
        X->V->V, DEC XA, Y->W, YC->YA
        V->X, Y+V->U, Y-V:H->V,
        "V=old PANE
        "V=new PANE, W=BIAS
        "U=UB, V=LB
        DEC YA(2)
        "LB->MR, V=init count
        V*Y, INC XA(2), W:H->V
        "T=Pj, UB-Pj
        SETUP: X->T, U-X:H, INC XA
              "H=UB-Pj, Pj->t
              X->T, U-X:H, INC XA,
              "G=LB-P(j-1)
              MR-T:G, GOTO CNTLP
              "H=UB-Pj, Pj->T
              RSET: X->T, U-X:H, INC XA,
                   "G=LB-P(j-1)
                   MR-T:G,
                   "done?
                   CNTLP: X->T, U-X:H, INC XA,
                          IF G<0 GOTO FIN
                          "U=UB-Pj, Pj->T
                          MR-T:G,
                          "P(j-2)>UB?
                          IF H<O GOTO CNTLP
                          "may be in, inc cnt
                          X:G, DEC XA(2), Y+V:H->V,
                          "if P(j-2)>LB count it
                          IF G<O GOTO SETUP
                          "restart UB-Pj-1
                          X->T, U-X:H, INC XA,
                          "dec count
                          V-Y->V, GOTO RSET
                          "U=1, W=count
                          FIN: BIPTR->XC, Y->U, V:H->W,
                               INC YA
  
```

```

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

" ENTRY POINT FOR SCORE and for each candidate PITCH

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

```

```

"score count
"V=CAND
"update bias ptr
"CAND->WINNER, COUNT->SCORE
"Y=BIAS, X=PAGE
"done?

```

```

"limit value
"fetch XA(CAND)
"CNDPTR-CNDEND
"U=new CAND
"V=new CNDPTR, CAND-CTAB (0)
"have a candidate

```

```

TITLE SFLT
ENTRY SDIV, SELT
EXT 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 SELT, SCL. Converts UV*2**SCL to short FP.
SELT: U:G, V:H, CALL FLOAT2
      W->V, READ, GOTO EXMAC
      "test value, float
      "V=SCL, exit

"SDIV, ARG DOF=0,2,4 OP2 at XY(ARG)
" Computes X/Y / U/V -> U/V
SDIV: T->YA, W->XA, V:H->W,
      CALL FDIVS
      W:H->V, READ, GOTO EXMAC
      END
      "XYA(OP2)

```

TITLE SHORTOPS
 ENTRY LDUX, LDVY, LDUI, LDUY, LDV, LDVY, LDUVY, LDUVX, LDUVY,
 STUX, STYX, STVX, STUY, STUVY, STUXY, STUXI, STUYI,
 STVD, STVD, LDY, LDY, LDY, LDY, LDY, LDY, LDY,
 ADDU, ADDUX, SUBU, SUBAU, SUBV, SUBAV, ADDV,
 ADDVY, ADDAY, ADDAX, MULTU, MULTV, FMULTU, FMULTV,
 IFUEQ, IFUGI, IFULI, IFVLI, IFLY, IFYLI, IFALI

EXT EXMAC, OPSEQ, COTO
 EXPAND
 NOLIST
 SYMBOL

"A collection of short operations involving UV and an argument

EGU 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->XA, READ, COTO 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 LDUVY, DA DOP=0 (DA must be even) D->UV
 " (also could have DOP=3 for inline ARG)
 LDUVY: DX->U, DY:H->V, READ, COTO EXMAC

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

"MACRO STY, YA DOP=15 V->Y(YA)
 STY: 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

"MACRO STUXI, YA DOP=14 U->X(Y(YA))
 STUXI: Y->XA, READ, COTO STUX

"MACRO STUYI, YA DOP=14 U->Y(Y(YA))
 STUYI: Y->YC
 YC->YA, READ, COTO STUY

"MACRO STUD, DA DOP=22 U->D(DA)
 STUD: WRITE U, COTO OPSEQ

"MACRO STVD, DA DOP=22 V->D(DA)
 STVD: WRITE V, COTO OPSEQ

"MACRO STUVD, DA DOP=23 UV->D(DA) DA must be even
 STUVD: WRITE UV, COTO OPSEQ

"MACRO LDY, ARG, XA, DOP=1,3,5,7 ARG->Y(YA)
 LDY: DX->XA, READ, U->T, V->U, W:H->V
 V->X, U:H->V, T:G->U, EXEC MACRO "ARG->X

"MACRO LDY, ARG, YA, DOP=1,3,5,7 ARG->Y(YA)
 LDY: DX->YA, READ

LDY: W->Y, EXEC MACRO

"MACRO LDXY, XYA, XVAL, YVAL DOP=11
 " XVAL->X(XYA), YVAL->Y(XYA)

LDXY: DX->X, DY->Y, READ, COTO EXMAC

"MACRO LDUVY, ARG DOP=0,2,4,6 Y(ARG+V)->U
 LDUVY: V+W:H->V, V->W "base + index
 V->YA, W->V, READ, COTO LDUY "point at Y value

"MACRO ADDU, ARG DOP=1,3,5,7 U+ARG->U
 ADDU: W+U->U, EXEC MACRO

"MACRO ADDUX, XA DOP=13 X+U->U
 ADDUX: X+U->U, EXEC MACRO

"MACRO ADDV, ARG DOP=1,3,5,7 V+ARG->V
 ADDV: W+V->V, EXEC MACRO

"MACRO ADDVY, YA DOP=15 Y(YA)+V->V
 ADDVY: Y+V->V, EXEC MACRO

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

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

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

"MACRO SUBAV, ARG DOP=1,3,5,7 ARG-V->V

```

SUBAV:  W-V->V, EXEC MACRO
"MACRO ADDAY, ARG, YA, XXX DOP=1,3,5,7
ADDAY:  DX->YA
        Y-W->W, READ, GOTO LDII

"MACRO ADDAX, VAL, XXX, XA 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 MULTU, ARG DOP=0,2,4,6 U*ARG->UV
MULTU:  U-W:22, GOTO MPUV

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

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

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

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

"MACRO IFYLT, ARG, YA, BRANCH DOP=1,3,5,7
" IF Y(YA)<ARG, GOTO BRANCH
IFYLT:  DX->YA
        Y-W:H
        LIT: NOOP
        DY->W, READ, IF H<0 GOTO GOTO "wait for test value
        EXEC MACRO

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

"MACRO IFUEQ, VAL DOP=0,2,4,6
" IF U=ARG, DO NEXT MACRO
IFUEQ:  U-W:H
        NOOP
        READ, IF H=0 GOTO EXMAC
        READ, GOTO EXMAC

```

```

"MACRO IFULT, VAL DOP=0,2,4,6
" IF U<ARG DO NEXT MACRO
IFULT:  U-W:H
        NOOP
        READ, IF H<0 GOTO EXMAC
        READ, GOTO EXMAC

"MACRO IFUGT, ARG DOP=0,2,4,6
" IF U>ARG DO NEXT MACRO
IFUGT:  W-U:H, GOTO ULT

"MACRO IFVLT, ARG DOP=0,2,4,6
" IF V<ARG DO NEXT MACRO
IFVLT:  V-W:H, GOTO ULT
        END

```

"skip one doubleword

```
TITLE SMXYD
ENTRY SMXYD
INTRLV
EXT MVXYD1
```

```
SMXYD DX->XC, DX->YA, DY->J
XC->XA, GOTO MVXYD2
END
```

```
"SET YA AND COUNT
"SET XA, GOTO MOVE
```

```
TITLE STEP
ENTRY STEP, IFEA, IFA, INCD, ONESTEP
EXT OPSEQ, EXEC, GOTO
EXPAND
NOLIST
SYMBOL
```

```
"CMD MACRO IFA, ARG, VAL, N DOP=1,3,5,7
"CMD MACRO IFEA, EAVL, xxx, N DOP=1,3,5,7
"CMD MACRO STEP, N DOP=DY->W
" MACRO INCD, ARG, DA, EAVL DOP=1,3,5
" D(DA) *ARG->D, IF RESULT \=0 GOTO EAVL
```

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

```
IFA: DX-M:H, GOTO COMP
IFEA: DA(EA)-H:H
COMP: YB=DA(DI)
STEP: IF H=0 GOTO EXEC, DY->W
U->T, V->U, -1*W:H->V,
-1->W
READ (DI), INC DA(W)
IF H<0 GOTO EXEC, WRITE V,
U->V, T:F->U
READ (EA)
ONESTEP: EXEC MACRO, ENABLE "do one MACRO
```

```
"MACRO INCD, ARG, DA, EAVL DOP=1,3,5
" D(DA) *ARG->D, If result \=0 goto EAVL
```

```
INCD: DX->W
W->DA(DI), T->U, U->T
READ, DY->W
DX+U->U, READ, INC DA(-1)
WRITE U
IF C=0 GOTO OPSEQ, T->U
GOTO GOTO
END
"W=DA
"save U, U=ARG
"fetch D, W=EAVL
"D*ARG, back up DA
"update D
"is result=0
"no, branch
```

```
"VAL-ARG
"EA-EAVL
"EA=DA
"CNT->H, DONE?
"save U,V
"DEC CNT
"DEC FILE(DA)
"update CNT, done?
"restore U,V
```

TITLE STKOPS
 ENTRY LDSTED, S1STED, STKADD, POPSTK,
 STKMULT, SAVSTK, UVTOSTK, STKDIV, STKLOG
 EXT FADD, OPSEQ, EXMAC, FLOAT, FMULT, FINV, FSUB, FLOG
 EXPAND
 NOLIST
 SYMBOL

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

"FLOATING POINT STACK FORMAT

" X , Y
 " -- , SCL
 " ML , MR for first number
 " -- , SCL2
 " ML , MR for second number
 " -- , SCL for IOS number (XYA points here)
 " U=ML , V=MR for IOS number

"MACRO LDSTED, A(FP WORD) DOP=DY->F (DA)

LDSTED: DEC YA, DEC YA, READ
 DX->I, READ, V->Y, DEC YA
 DX->V, READ
 DX->W, READ (EA)
 U->X, DEC YA, I->U, M->Y,
 EXEC MACRO

"MACRO ST1STED, DA DOP=DY->F (DA)

ST1STED: WRITE U, Y->U
 WRITE V
 WRITE U
 POPSTK: INC YA, INC YA, READ (EA)
 X->U, Y:H->V, INC YA,
 INC YA, EXEC MACRO

"MACRO SAVSTK, DA push STK and save ptr

SAVSTK: DEC YA, DEC YA, Y->W
 U->X, V->Y, DEC YA, DEC YA
 XA->XC, M->Y
 WRITE XC, GOTO OPSEQ

"MACRO UVTOSTK, A..A DOP=0,2,4 FLOAT UV->TOS

UVTOSTK: I->YA, M->XA,
 CALL FLOAT
 M->Y, READ, GOTO EXMAC

"MACRO STKADD , xxx DOP=NOOP

STKADD: Y->W, INC YA(2), INC YA(2),
 CALL FADD
 M->Y, READ, GOTO EXMAC

"MACRO STKSUB , xxx DOP=NOOP

STKSUB: Y->W, INC YA(2), INC YA(2),
 CALL FSUB
 M->Y, READ, GOTO EXMAC

"MACRO STKMULT, xxx DOP=NOOP
 STKMULT: Y->W, INC YA(2), INC YA(2),
 CALL FMULT
 M->Y, READ, GOTO EXMAC

"point at ARG2
 "save new SCL

"MACRO STKDIV, xxx DOP=NOOP

STKDIV: Y->W, INC YA(2), INC YA(2),
 CALL FINV
 CALL FMULT
 M->Y, READ, GOTO EXMAC

"point at numerator
 "do divide

"save new SCL

STKLOG: Y->W, CALL FLOG

M->Y, READ, GOTO EXMAC
 END

```

TITLE ISTIOL
ENTRY ISTIOL
EXT EXMAC
EXPAND
NOLIST
SYMBOL

```

```

"MACRO ISTIOL, ARG
" IF LARG - U1 < V, DO NEXT MACRO, ELSE SKIP
ISTIOL: U-W->U, U->I
V-U:G, V-U:H, I:F->U
IF G<O GOTO HTEST
IF G>O GOTO EXMAC, READ
READ, GOTO EXMAC
HTEST: IF H>O GOTO EXMAC, READ
READ, GOTO EXMAC
END

```

```

"U-ARG->U
"V-U, V-U In CH
"do next macro
"else skip

```

```

TITLE UPCHAN
ENTRY UPCHAN
EXT EXMAC
EXPAND
NOLIST
SYMBOL

```

```

"MACRO UPCHAN XYA(CHNLBLK) CHNLBLK = AREC, A(UPDESC)
" ACUR In U
" MCUR In V
"UPDATE DESC In XY = BLNK, A(EXPTAB)
" DMP, PAV
" , PS
EQU EXPLEN='17'0
UPCHAN: U-X->U, YA->XC, Y->XA,
Y->YC
U-X->U, U->W, INC XA,
YC->YA, YA->YC
X-U:ER, XC->XA
IF G>O GOTO NOTBLNK,
Y->U, YC->YA, INC XA
EXIT: READ, GOTO EXMAC
NOTBLNK: ML-U->U, EXPLEN->I
MA*X:ER, U->XA, T-ML:G
X*MB:ER, XC->XA, V->U,
INC YA
Y->V, INC XA,
IF G<O GOTO UPDATE
U-ML:G
NOOP
UPDATE: U->X, DEC XA, V->U
W->Y, INC YA
X*W:H->V, Y->W
V->X, U->Y, INC YA, READ
W->Y, EXEC MACRO
END
"ACUR-AREC
"XA=YC=A(UPDESC)
"U=A disp, W=PA'
"Y=A(EXTAB)
"start DMP*ADISP
"ADISP>0?
"U=EXBS
"U=EXPXA
"MB=MREC, EXPLEN-DMP*ADISP
"PREC*EXPT, U=MCUR
"V=PA
"ADISP=DMP>EXPLEN?
"MCUR-TRESH
"MCUR too small
"MCUR->MREC
"U=PA, PA'->PA
"V=ACUR, W=PB
"ACUR->AREC, PA->PB
"PB->PC, done

```


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
MOVEXYD XYSAVE, xywk, 5 "RESTORE XYMEM
RESTORE INTSVE "AND REGS
INTRTN "EXIT
rcvint STVX ldesca "SAVE A(LINE DESC)
MOVEXYD 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
GOTO store "NOT READY?
LDX 1, lflag "SET TO STATE ONE
LDU '47'O "ENABLE TXRDY
SNDCMD lcode "SET UART CTRL
GOTO store "UPDATE LINE DESC
trint STVX ldesca "SAVE A(LINE DESC)
MOVEXYD ldesca, ldesc, 4 "GET DESC.
IFALT lflag, 1, trdn "No activity
LDUDA X[stringd]
ADDAX 1, X[stringd] "UPDATE POINTER
SNDCMR lcode "SEND CHAR IN U
TLYY -1, Y[stringd], store "DEC COUNT
CALL trxt "COMPLETION ROUTINE
store SMVXYD ldesca, ldesc, 4 "UPDATE DESC
GOTO aslrsp2 "GO CHECK FOR MORE
trdn LDU '46'O
SNDCMD lcode "DISABEL TRDY INT

```

```

        GOTO      aslrsp2
"TOP LEVEL ASL INPUT
RCVINT  IFUEQ    playn
        GOTO      inpars          "Parcel?
        IFUEQ    addr
        GOTO      saddr          "SET ADDRESS?
        IFUEQ    data
        GOTO      data1          "DATA BLOCK?
        IFUEQ    freqn
        GOTO      freqn1        "ANAL START?
        IFUEQ    stop
        GOTO      stop1         "Anal stop?
        IFUEQ    outd
        GOTO      outd1         "DATA OUT?
        RTN              "IGNORE OTHERS
"ENTERED FROM CODEPARS THROUGH SENDOUTP
OUTQASL LDU      PARLINE
        IFUEQ    0
        RTN              "NO OUTPUT NOW
        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     XY[16], A[OUTBUF], 9  "A(DATA), COUNT
outp    LDXY     XY[17], 1, A[asldne]  "FLAG, COMPLETION EXIT
        SMVXYD   [2]V, XY[16], 2      "FILL IN LINE DESC
        LDUDA    [1]V          "GET LINE CODE
        LDV      [0]U          "PUT IT IN V
        LDU      '47'O          "ENABLE TXRDY
        SNDCMD   [0]V          "SEND IT
        RTN              "DONE
"INTERRUPT RESPONSE ROUTINE
asldne  LDU      0
        STUX     lflag
        INCD     0, POCNT, send      "MORE PARS IN QUEUE
        RTN              "NO
"CONVERT ONE PACKED PARCEL TO BASE 64 CODES
OUTASL  MOVEDY   POPTR, BA, 3        "GET NEXT PARCEL
        INCD     -1, POCNT, NEXT1    "DEC PAR CNT
NEXT1   LDU      POPTR            "GET PTR
        ADDU     PRLen            "UPDATE IT
        IFUEQ    POLIM
        LDU      POBSE            "WRAP IF NEEDED
        STUO     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
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		
parinc	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	ldesca	"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[rexit]	"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	"UPDATE QUEUE COUNT
end2	LDU	PRIIN	"GET INPUT PTR
	IFUEQ	PRIPTR	
	SCHED	SYNINP	"WAS QUEUE EMPTY?
	ADDU	PRLEN	"UPDATE PTR
	IFUEQ	PRILIM	
	LDU	PRIBSE	"WRAP IF NECESSARY
	STUD	PRIIN	"STORE PTR
	IFALT	PRICNT, prmax-1, ends	"QUEUE NOT FULL?
	SETD	'23'O, xchr	"MUST SEND XOFF
	SCHED	sxo	
ends	LDU	RCVINT	"TOP LEVEL CHAR INPUT
	STUDA	[7]V	"SET IN LINE DESC
pcex	SMVXYD	[rdesc]V, ldesc, 2	"SAVE RCV DESC
	RTN		"EXIT
"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	
	LDUDA	[4]U	"GET FLAG
	IFUEQ	0	
	GOTO	chkx	"LINE IDLE?
	SCHED	sxo	"NO, TRY LATER
	RTN		
chkx	LDU	xchr	
	LDV	1	"ONE CHAR TO SEND
	STUVXY	XY[16]	
	LDV	PARLINE	"A(LINE DESC)
	GOTO	outp	
freqn1	SETD	POIN, POPTR	"SET QUEUE EMPTY
	SETD	0, POCNT	
	STVD	PARLINE	"SET 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	
	GOTO	doout	"LINE FREE?
	SCHED	coutd	"NO, CHECK LATER

RTN

```
doout  MOVEDY  BOS, BA, 3          "GET 3 WORDS
        INCD   3, BOS, doout2     "UPDATE ADDRESS
doout2  LDU    data                "CODE FOR BLOCK
        LDV   doutb+9            "END OF BUFFER
        CALL  out3               "BUILD OUTPUT CHARS FOR 3 WORDS
        LDV   datline            "GET A(LINE DESC)
        LDXY  XY[16], A[doutb], 9 "ADDRESS, COUNT
        GOTO  outp               "SEND BLOCK

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
```