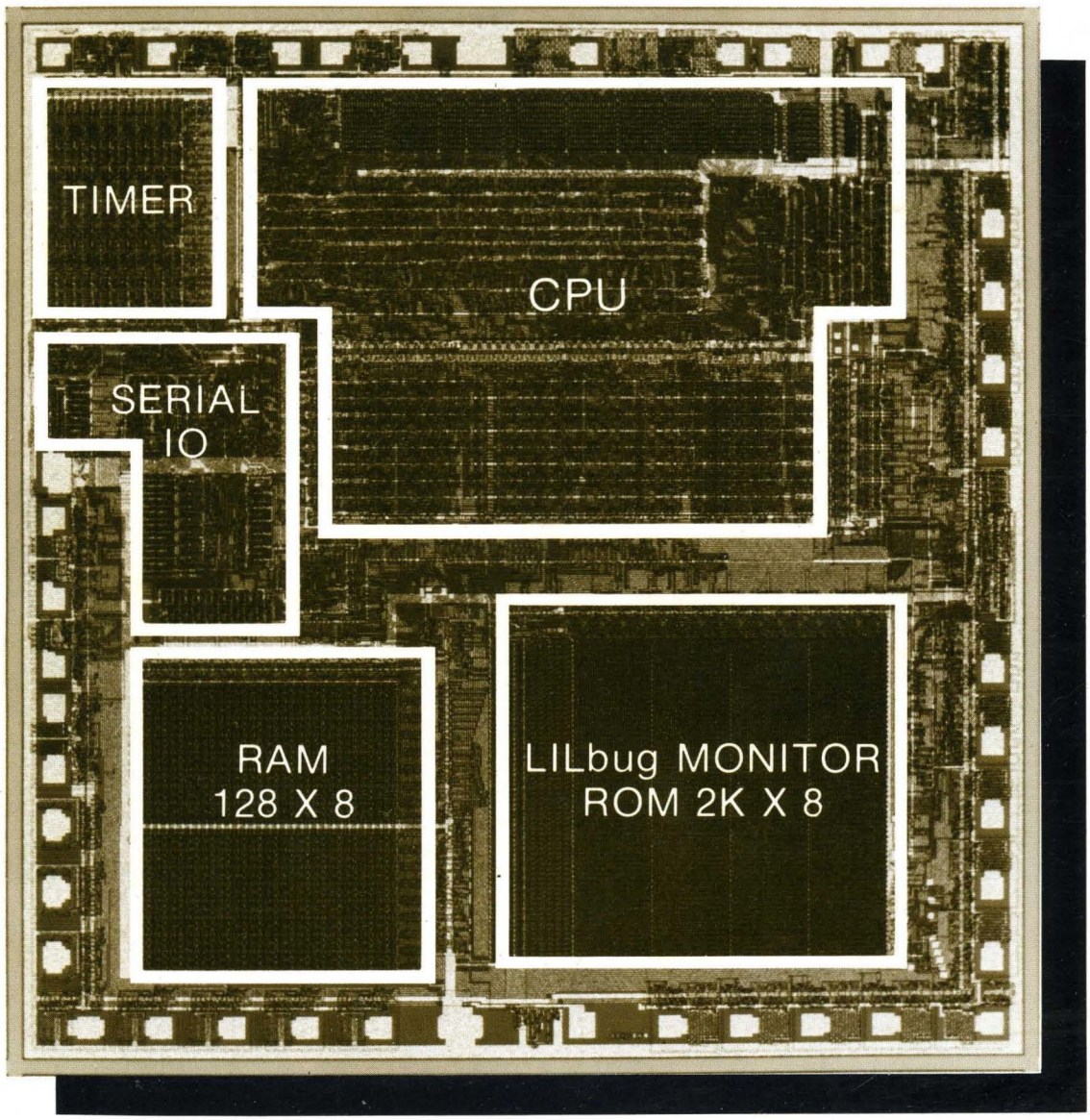
LILbug

MONITOR FOR THE MC6801L1



MOTOROLA

MOTOROLA

LILbug

MONITOR FOR THE MC6801L1

PREPARED BY
NMOS MICROCOMPUTER SYSTEMS
AUSTIN, TEXAS

Second Edition

LILbug

MONITOR FOR THE MC6801L1

TABLE OF CONTENTS

1.0 MC6801 OVERVIEW

Successful design begins with a clearly stated set of objectives. The principle design objectives driving the MC6801 development have remained constant throughout its evolution: to provide the user with the finest single-chip microcomputer possible today with sufficient flexibility and power to deal with his problems extending into the 1980's. Three clear trends are emerging with respect to future applications: users will continue to strive for minimization of part counts, make increasing use of distributed processing, and automate processes heretofore unimagined. The MC6801 has demanded the highest skills and talents from its designers in meeting these formidable objectives.

1.1 ARCHITECTURAL INNOVATIONS

The MC6801 represents the state-of-the-art in single-chip microcomputer development. The MC6801 MPU has been enhanced from the M6800 with respect to both its capability and internal architecture. In addition to the MPU, the MC6801 has incorporated an on-chip oscillator driver, an on-chip programmable timer, a serial communications capability, two types of memory and 8 modes of operation. The single most noteworthy accomplishment of the architecture is probably the systems integration of these components into a powerful single-chip microcomputer.

The MC6801 microcomputer may be considered an entire family of processors all of which have been integrated into one common design. This approach was taken at the risk of overwhelming the user with the myriad of options available. The advantages to be gained from the approach are significant:

- a single microcomputer can be used in an extremely wide and vastly different variety of applications.
- the user can build upon his experience with the microcomputer without the fear of early obsolescense, and
- the user's investment in learning and support tools may be amortized over a longer period of time than with a number of more custom designs, hence it also will free the user from a heavy burden of inventory.

1.2 SOFTWARE INNOVATIONS

The M6800 programmer will feel very much at ease with the MC6801 instruction set. All of the familiar instructions of the M6800 have been implemented and function in exactly the same manner. In addition to the M6800 instructions, several new and powerful instructions have been added to ease the problems associated with 16-bit arithmetic problems. The A and B-accumulators have been concatenated into a single, double-accumulator (D) for which a full set of 16-bit arithmetic instructions (including load, store, add, subtract, and shift) are included. These new double accumulator instructions have been implemented with all of the addressing modes available for single accumulator operations.

Indexing has been greatly enhanced by adding a "Add B-accumulator to X" instruction along with the capability to both push and pull the index register. A hardware multiply instruction has also been added to further aid in arithmetic problems. Improvements in the internal architecture also have resulted in faster cycle times for many instructions.

The MC6801 MPU may be considered an enhanced M6800 MPU which capitalizes on the user's investment with the M6800 Family of Parts. New users will appreciate the widely acclaimed easy-to-learn instruction set.

1.3 SUMMARY OF FEATURES

Hardware

- M6800 Bus Compatible
- Single 5V Power Supply
- 8-Bit Word Size
- 16-Bit Address Field
- TTL-Compatible Inputs and Outputs
- On-Chip Oscillator/Driver
- On-Chip 16-Bit Dual Function (Input and Output) Programmable Timer
- On-Chip Serial Input/Output
- On-Chip 128 Byte RAM
- On-Chip 2K Byte ROM
- Vectored Priority Interrupts for Timer and Serial I/O
- Four Programmable Input/Output Ports
- Eight Hardware Programmable Modes of Operation
- Mask Option for External Clock Input
- Peripheral Controller Mask Option
- EROM version for all Mask Options
- Mask Relocatable ROM Address
- Mask Relocatable RAM Address
- Programmable External Address Space to 64K
- Multiplexed Address/Data Bus
- Valid Address Strobe
- On-Chip Standby RAM for 64 Bytes
- Vectored Restart
- Maskable Vectored Interrupt
- Separate Non-Maskable Interrupt
- Full Duplex Programmable Serial I/O for Either NRZ or Bi-phase
- Four Baud Rate Programmable Selection for Serial I/O

Software

- MC6800 Upward Compatible Architecture
  - — Two 8-Bit Accumulators
  - — One 16-Bit Index Register
  - — One 16-Bit Stack Pointer
- MC6800 Upward Compatible Instruction Set
  - — All M6800 Instructions Included (Many contain fewer cycles)
  - — M6800 Object Code Compatible
  - — 8 X 8-Bit Unsigned Hardware Multiply
  - — 16-Bit Arithmetic Capabilities (Load, Store, Add, Subtract, Shift)
  - — Additional Index Register Instructions (Push, Pull, Add B to X)

2.0 LILbug MONITOR OVERVIEW

The standard MC6801 ROM pattern contains the LILbug monitor. This ROM monitor is used to evaluate and debug a program under development. The LILbug monitor commands enable the user to:

- Load a program
- Verify that a program was properly loaded
- Dump the program
- Punch (record) the program on tape
- Examine and change data in a memory location (including I/O)
- Calculate the offset for relative addressing

- Examine and change data in the user's program registers
- Insert, display, and remove breakpoints in the program
- Freerun or trace through the user's program

The 6801 LILbug monitor is designed to enable a user to modify the system to meet his specific hardware and software needs. The main areas of flexibility are:

- I/O device independence
- Interrupt vector independence
- Command table expandability

3.0 COMMANDS

3.1 COMMAND FORMAT

The general command format is:

<Command> <Delimiter> [<Arg> <Delimiter> [<Arg>]]

The command name is followed by a delimiter of space, comma, or carriage return. If there are no arguments, carriage return terminates the command; otherwise a space or a comma separates the command from its argument. A space or comma separates arguments from each other. The "." and "/" are special quick commands that do not need carriage returns.

3.2 COMMAND DESCRIPTION

LOAD L

L <Offset>

Load a tape file. Figure 3.2 shows the monitor's tape format. Load with offset is available by specifying the hexadecimal <offset>. The offset is added to the address on tape to get the load address. All offsets are positive, but wrap around memory modulo 64k. For example, to offset a load by minus $1000 locations, use $F000 as offset. An attempt to load into ROM, protected RAM, or non-existent memory terminates the command after printing '?'. A checksum error also prints '?' and terminates. On some tape readers, several extraneous characters are transmitted before the tape reader is turned off. These characters will sometimes print after the prompt on the console.

Example:

```
!2012 600 400 080 300 440 BF0/
2018 00 1A0 7F0 650 EE0
!L
!2012/60 40 08 08 30 44 BF 1A 7F
!3012/42 05 03 03 C0 60 FD FD FF
!L 1000
!3012/60 40 08 08 30 44 BF 1A 7F
```

VERIFY V

V <Offset>

Verify or compare the contents of memory to the tape data. A hexadecimal <offset> can be specified. A checksum error, a bad compare, or non-existent memory prints '?' and then terminates.

Example:

```
!V 1000
!V
!
```

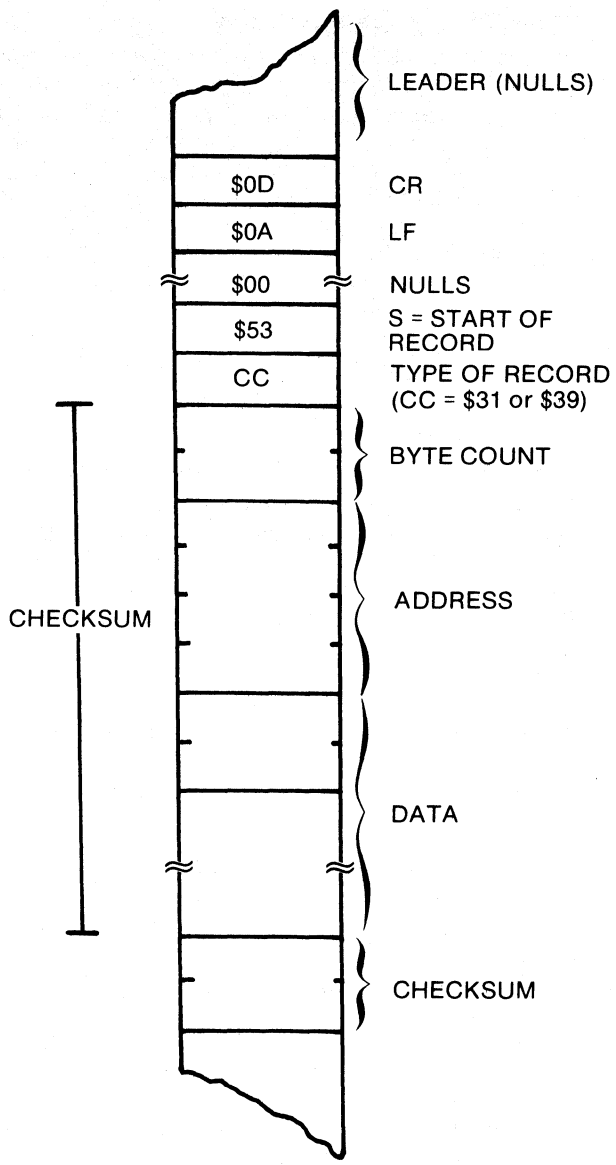


Figure 3.2 — Tape Format

PUNCH P <Adr1>, <Adr2>

Punch or record formatted binary object tape as shown in figure 3.2. Start at <Adr1> and punch through <Adr2>. This routine fails to diagnose an end address smaller than the beginning address. With low speed the values punched to tape may also print to console. High speed may cause some garbage characters to print to the console during the punch.

Example:

```
!P 2012 20A6

:11B2012604008083044BF1A7FEE65EEE4A6044801000589118 19EEF71
:11B202AFFAFB3EEFBD603852031005912FCFFB9BDF5F7BAF7FE051114
:11B2042081812545280EF65B3F7FCEF2FE7169042801 09F0751EFEFDE
:11B205AABFFFF798926C0801A00CB02C4DDA2FFF6FFE5DFB7571 01346
:11B207200815128088 8ABFE85CD8D7DAD6D0040001B8B46BA94DFFF51
:11B208AFFABC9EF4BAF0206C308025117 08FA7F6DFFEEDB2FBB40447D
:10820A2814508629570
:00000001FF
```

DISPLAY D <Adr1>, <Adr2>
D <Adr1>
D

Display contents of memory in hexadecimal followed by the literal ASCII characters. Start at <Adr1> and print thru <Adr2>. Ignore the command if the beginning address is larger than the end address. The command prints blocks of 16 bytes — <Adr1> is masked back to the beginning of a block and <Adr2> is extended to the end of a block. If only one argument is specified, display a block of 48 words surrounding address <Adr1>. If no argument is given display a block of 48 words containing the last location used in memory examine.

Example:

```
!D 2012 2056

     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
2010 00 12 60 40 08 08 30 44 BF 1A 7F EE 65 EE E4 A6   ..`@..0D?.....e..$
2020 04 48 01 00 05 89 11 81 9E EF FF AF B3 EE FB D6   .H.........../3..V
2030 03 85 20 31 00 59 12 FC FF B9 BD F5 F7 BA F7 FE   .. 1.Y...9=..:..
2040 05 11 08 18 12 54 52 80 EF 65 B3 F7 FC EF 2F E7   .....TR...3.../.
2050 16 90 42 80 10 9F 07 51 EF EF AB FF FF 79 89 26   ..B....Q..+....y.&

!D 2045

     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
2030 03 85 20 31 00 59 12 FC FF B9 BD F5 F7 BA F7 FE   .. 1.Y...9=..:..
2040 05 11 08 18 12 54 52 80 EF 65 B3 F7 FC EF 2F E7   .....TR...3.../.
2050 16 90 42 80 10 9F 07 51 EF EF AB FF FF 79 89 26   ..B....Q..+....y.&
```

MEMORY M <Adr>(Space)
<Adr>/

Initiate the memory examine/change function. Print the value at <Adr> and maintain a pointer to that address. <Adr> is 1 to 4 hex digits not counting leading zeroes, and it must be followed by a a delimiter of space or slash. The form <Adr>/ cannot start with characters 'A' through 'F'. If the address would normally start with 'A' thru 'F', zero should precede the address. For example: 0FFFE/ instead of FFFE/. The user may execute a combination of the following sub-commands to modify the current memory location, look at consecutive words of memory, move the memory pointer, or terminate memory examine.

- <DATA> — Enter one byte of data to replace the value at the current pointer. There is no size check on the data. If <DATA> is more than two digits, the right-most byte of data becomes the new value.

- SPACE — Increment the current pointer and print the value at the new address on the same line.

- , — (Comma) Increment the pointer to the next memory location with no print of address or value. This allows insertion of data without seeing current values in memory.

- LF — (Line feed) Increment the pointer to the next memory location, print the address and value at that address on the next line.

- UA — (Up arrow) Decrement the pointer to the previous memory location. Print the new address and its value on the next line.

- / — (Slash) Print the current address and the value on the next line.

- CR — (Carriage return) Terminate the memory examine command.

Any other input terminates the memory examine command.

Example:

```
!M 2000 30                      * MEMORY EXAMINE
!2000/30
!FFFE/                          * Address cannot start with
?                               *   "A" thru "F"
!0FFFE/30 00                    * Zero should precede the address
!2000/3024 3248 3021            * Change 30 to 24 at address 2000
                                * Space moves pointer to address 2001
                                *   where 32 is changed to 48

!2005/201,2,3,4,5               * At location 2005 start inserting
                                *   values 1, 2, 3, 4, and 5

!2005/01 02 03 04 05            * Look at values just inserted

!2005/01                        * Line feed moves pointer to next
2006 02                         *   memory location
2007 03

!2007/03^                       * Up arrow moves pointer to previous
2006 02^                        *   memory location
2005 01

!2005/01/                       * Slash recalls current address
2005 01/                        *   printing address and value
2005 01

!                               * Carriage return terminates Memory
                                *   Examine - prompt indicates ready
                                *   for next command
```

SLASH /

Recall the location last referenced in memory examine and print that address and the value at that address. No carriage return or delimiter is necessary.

Example:

```
!/2005 01* Slash recalls current address,
        *     printing address and value
```

OFFSET O <Adr1> <Adr2> (CR)

Calculate the offset (second byte) of a branch instruction from <Adr1> to <Adr2>. <Adr1> should be the address of the second byte of a branch instruction. <Adr2> is the absolute address for the branch. The command prints the offset if it is within the -127 to +128 byte range, otherwise it prints '?' and returns to main control loop. If the range is valid, <Adr1> is printed and the user can modify that location.

Example:

```
!O 2034 209A
65
2034 0065
!O 2034 2012
DD
2034 65DD
!O 2034 3000
?
!
```

REGISTER R

Display the contents of the program registers and counter. The registers are printed in the format:

P-XXXX X-XXXX A-XX B-XX C-XX S-XXXX

P-

where:

P = program counter
X = index register
A = accumulator A
B = accumulator B
C = condition code register
S = stack pointer
XX = values in registers

Register P is ready for update. New values may be inserted or the register change function can be terminated.

The following inputs are valid:

<DATA> — Insert new value for the current register. There is no validity check on the length of the data, so the right-most byte or double byte, whichever is appropriate for the register becomes the new register value.

SPACE — Print register value unless <DATA> changed its value. Move the register pointer to the next register, printing the mnemonic and a dash.

Any other input characters terminate the register command. The command terminates after examining and/or updating the stack pointer.

Example:

```
!R
P-0000 X-0000 A-00 B-00 C-D0 S-BB
P-2000 X- 0000 A-123456 B- 00 C- D0 S-A0
!R
P-2000 X-0000 A-56 B-00 C-D0 S-A0
```

BREAKPOINT B <Adr>
B-<Adr>
B
B-

Enter a breakpoint at address <Adr> and print current breakpoints. A maximum of four are allowed.
A question mark is printed if 4 breakpoints are already set.

B -<Adr>

Remove the breakpoint at address <Adr> and print the remaining breakpoints.

B

Display all breakpoints.

B-

Remove all breakpoints and show empty breakpoint table.

Examples:

```
!B 2002
2002 0000 0000 0000
!B 2045
2002 2045 0000 0000
!B -2002
0000 2045 0000 0000
!B 1234
1234 2045 0000 0000
!B
1234 2045 0000 0000
!B -
0000 0000 0000 0000
!
```

GO G <Adr>
G

Execute starting from <Adr>. Execution will continue until a breakpoint (3F) is detected.

G

Execute starting at the current program counter setting.

CALL C <Adr>
C

Call and execute a user routine as a subroutine starting at address <Adr>. A return address to the monitor is stored in the users stack. Breakpoints are recognized and trace can be used after initiating a "C" command. When the user "RTS" is executed, the monitor gets control, prints registers and returns to the main calling routine. The user PC then points into the monitor.

C

Same as C <Adr> except start executing from the current program counter.

TRACE T <Hex number>

Trace the specified number of instructions where <hex number> is a maximum of $FFFF. The opcode, program registers, and counter are printed after each instruction. The format of the print line is:

OP-<Op> P-XXXX X-XXXX A-XX B-XX C-XX S-XXXX

where: <Op> = opcode of last instruction executed
P = program counter
X = index register
A = accumulator A
B = accumulator B
C = condition code register
S = stack pointer
XX = value in register

Example:

```
P-2002 X-1001 A-00 B-BF C-D0 S-00A0
!T 3
OP-01 P-2003 X-1001 A-00 B-BF C-D0 S-00A0
OP-01 P-2004 X-1001 A-00 B-BF C-D0 S-00A0
OP-01 P-2005 X-1001 A-00 B-BF C-D0 S-00A0
```

NEXT .

The character '.' (period) with no carriage return will trace the next instruction and print the trace line.

Example:

```
!.
OP-4C P-2006 X-1001 A-01 B-BF C-D0 S-00A0
!.
OP-7E P-2000 X-1001 A-01 B-BF C-D0 S-00A0
```

CONTROL X

Return control to the main command loop. Control 'X' is recognized during display and trace print out, and whenever the monitor is reading console input.

CONTROL W

Halt print and wait until input character signals continuation of print. This command is recognized during trace print and during display memory print. Print is halted until any character is input, then print continues.

HIGHER HI

Set speed to 1200 baud (÷1024), set padding and set bits in on-chip serial I/O RMCR (rate and mode control register). When changing speeds, tape readers and punches should be off-line to prevent accidental read or write during the transition.

HIGHER YET HY

Set speed to 9600 baud (÷128), set zero padding. This is for CRT terminals. Tape recorders and punches should be off-line when changing speeds.

4.0 MONITOR OPTIONS

Since versatility is built into the MC6801, versatility, expandability, and ease of use were the primary design criteria of LILbug. LILbug can be used in either the single chip or expanded modes of the MC6801. It can be configured to use on-chip I/O, standard external devices (e.g., ACIA, VDG, PTM), or user defined external I/O (e.g., keyboards, displays). All interrupt vectors are accessible to the user and can be redefined to point to user written interrupt routines. Furthermore the user can define additional variable length commands or redefine existing ones to suit his needs. This flexible design will hopefully allow LILbug to become a defacto standard for the basic monitor of all MC6801 based systems.

Often versatility is traded for ease of use. This is not true for LILbug. The monitor initializes all variables to convenient defaults so that a user who wishes to become familiar with the MC6801 and use its on-chip I/O need not even be aware of the more esoteric features of LILbug. All he needs to do is build the simple circuit explained in section 5.1, find a RS232 compatible terminal, push reset and his system is up and running.

4.1 IO INDEPENDENCE

Monitor I/O is indirect. All I/O calls are to a device type:

1. single-byte input (console keyboard)
2. single-byte output (console print)
3. high speed print
4. high speed punch/load device

Each device has three routines associated with it:

1. initialization
2. data in/out
3. device termination

Before a device is used by the monitor it will issue a call to the routine to initialize that device. Normal data transfer is done thru the data in/out routines and devices are terminated thru the device termination routine. Data transfers can be of two types: serial or block. In serial transfers the byte of data is passed to or from the data in/out routines in the A-register. For the high speed print and high speed punch/load, the address of a data packet is passed to the in/out routine. See Appendix B routines HSDTA and BSDTA for details of the format of the data packet.

The addresses for these device routines are ordered in a table. The address of the table is stored in 'IOPTR' at RAM location $OFC. The monitor calls an I/O routine by setting register B as an index into the I/O table, and then calling routine IO. Routine IO retrieves the address from the table and does a subroutine call.

The LILbug monitor defaults to the on-chip serial input and output routines for the Silent 700®. The monitor's high speed routine writes blocks of data to the console output device. The default high speed punch/load device routines are for a Silent 700® cassette. The monitor's RESTART code initializes 'IOPTR' to point to its default table. Following is the default I/O table as defined in LILbug.

```
00171                          ***************** IO TABLE ***************
00172                          * ROUTINE IO IS CALLED WITH
00173                          * INDEX INTO IO TABLE CI OR INTO USER IO TABLE
00174                          * IOPTR POINTS TO THE IO TABLE TO BE USED
00175                          * THE INDEX TABLE DEFINES ORDER OF IO ROUTINES IN IO TABL
00176A F85B     F8C8   A CI      FDB     CION,CIDTA,CIOFF
     A F85D     F891   A
     A F85F     F8D1   A
00177A F861     F8D2   A         FDB     COON,CODTA,COOFF
     A F863     F8A8   A
     A F865     F8D1   A
00178A F867     F8D1   A         FDB     HSON,HSDTA,HSOFF
     A F869     FDEC   A
     A F86B     F8D1   A
00179A F86D     FEAF   A         FDB     BSON,BSDTA,BSOFF
     A F86F     FEC3   A
     A F871     FEBA   A

00181                          * THE FOLLOWING ARE INDICES INTO IO TABLE
00182                0000   A CI.ON  EQU     0          INIT INPUT DEVICE
00183                0002   A CI.DTA EQU     2          INPUT A CHAR W/NO WAIT
00184                0004   A CI.OFF EQU     4          DISABLE INPUT DEVICE
00185                0006   A CO.ON  EQU     6          INIT OUTPUT DEVICE
00186                0008   A CO.DTA EQU     8          OUTPUT A CHAR W/PADDING
00187                000A   A CO.OFF EQU     $A         DISABLE OUTPUT DEVICE
00188                000C   A HS.ON  EQU     $C         INIT HIGH SPEED OUTPUT DEVICE
00189                000E   A HS.DTA EQU     $E         OUTPUT BLOCK OF DATA
00190                0010   A HS.OFF EQU     $10        DISABLE HIGH SPEED DEVICE
00191                0012   A BS.ON  EQU     $12        INIT PUNCH/LOAD
00192                0014   A BS.DTA EQU     $14        WRITE DATA BLK TO PNCH/LOAD
00193                0016   A BS.OFF EQU     $16        DISABLE PUNCH/LOAD
00194                          *
```

The user can redefine the I/O by writing his own routines and putting the addresses of the routines in his table. The routine addresses should be in the same order as shown in the previous table. The user may wish to redefine only part of the I/O routines. In this case, he should fill in the table with the monitor's addresses for those routines he does not change. He must then initialize 'IOPTR' with the address of his table.

The program — USERIO (Appendix C) — is an example of redefining the I/O to use an ACIA instead of the on-chip serial I/O. USERIO defines the initialization and character input/output routines using an ACIA. The other I/O routines used are from the monitor.

4.2 INTERRUPT VECTORS

The 6801 modes define internal or external vectors. If the vectors are internal, they are in the LILbug ROM. The monitor is designed to allow any of the ROM vectors except RESTART to be user defined. The interrupt address is retrieved indirectly from a table whose address is in 'VECPTR' at RAM location $OFE. The table must contain addresses or dummy addresses for seven interrupts. The monitor's restart code initializes 'VECPTR' but the user may define his own vector table in ROM or RAM by changing 'VECPTR'.

Word	Vector Table
$0	Serial
2	Timer Overflow
4	Timer Output
6	Timer Input
8	IRQ1
A	SWI
C	NMI

Following is a listing of the vector table defined in LILbug.

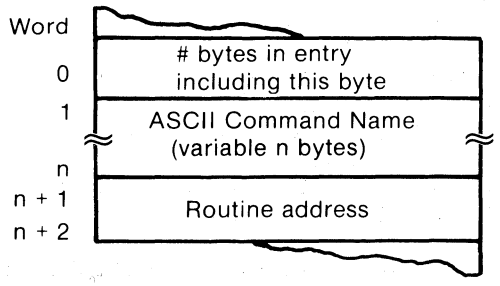
```
01621                          ***************** VECTORS ***************
01622                          * VECTOR INDEPENDENCE
01623                          * ALSO SAVE ON RAM USAGE
01624                          * VECPTR - RAM PNTR TO VECTOR TABLE
01625                          * VECTOR TABLE - ADR OF INTERRUPT VECTORS
01626                          * MAY BE REDEFINED BY USER TABLE IN SAME FORM
01627A FFC8     FCE1   A SERIAL FDB     DUMMY     NOT USED BY MONITOR
01628A FFCA     FCE1   A TIMOVF FDB     DUMMY
01629A FFCC     FCE1   A TIMOUT FDB     DUMMY
01630A FFCE     FCE1   A TIMIN  FDB     DUMMY
01631A FFD0     FCE1   A IRQ1   FDB     DUMMY
01632A FFD2     F821   A SWI    FDB     IN.SWI
01633A FFD4     F803   A NMI    FDB     IN.NMI
01634                          * DUMMY IS AN RTI
```

The monitor defines SWI and NMI. All other interrupts default to an 'RTI' instruction. The LILbug SWI is used for the monitor breakpoint command. The NMI is used by trace and breakpoint. The NMI vector controls the type of hardware tracing, either on-chip timer, or PTM (Programmable Timer Module).

The internal vector default uses the on-chip timer entry. Address $F803 is the NMI vector for trace using the on-chip timer, and address $F800 is for the PTM trace processing. This vector should not be changed after monitor initialization. The initialization defines the timer according to the address in the NMI vector. The chip must be in an expanded mode to use a PTM. SWI and NMI can be defined for other uses, if breakpoint and trace are not used.

4.3 USER-DEFINED COMMANDS

The LILbug monitor has a command table consisting of entries in the form:



The monitor reads a command and does a search of the command table to find a match with the ASCII command name. If a match is found, a subroutine call is made to the routine address following the command name. If no match is found, a question mark is printed. The user can define new commands and have his command table searched prior to the monitor comand table. The address of the command table is stored in 'FCTPTR' at RAM location $OFA. Command names are of variable length. Decimal numbers are not allowed in command names. The characters '/' and '.' are special quick commands that cannot be redefined. A terminating word marks the end of the table. A negative one indicates the monitor table is also to be searched. A negative two means the monitor commands are not being used. Since the user table is always searched first, the user can redefine existing commands. Program USERIO (Appendix C) is an example of redefining LILbug commands. USERIO defines ACIA I/O and redefines the commands, HI and HY, to change output padding for ACIA baud rate change.

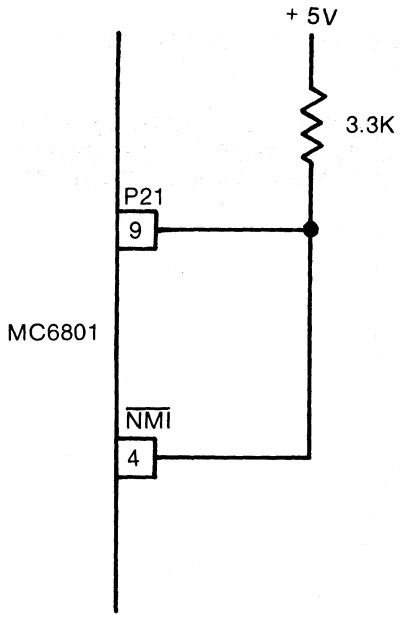
Figure 4.3 is the command table as defined by LILbug.

```
00110                    **********  COMMAND TABLE  **********
00111                    *  THERE MAY BE AN EXTERNAL TABLE OF THE SAME
00112                    *     FORMAT. 'FCTPTR' POINTS TO THE TABLE TO
00113                    *     BE SEARCHED FIRST. THE USER CAN DEFINE
00114                    *     HIS OWN TABLE AND SET FCTPTR.
00115                    *
00116                    *  EACH ENTRY IN THE TABLE IS AS FOLLOWS:
00117                    *      FCB   XXX    XXX=TOTAL SIZE OF ENTRY
00118                    *      FCC   /STRING/ STRING IS THE INPUT STRING
00119                    *      FDB   ADDR     ADDR IS THE ROUTINE ADDRESS
00120                    *
00121                    *  THE LAST ENTRY IS:
00122                    *      -1=END EXTERNAL TABLE,SEARCH INTERNAL TABLE
00123                    *      -2=END OF TABLE(S)
00124                    *
00125                    *  NOTE: AN EXTERNAL FUNCTION TABLE TERMINATED BY
00126                    *    -1, THE INTERNAL TABLE WILL ALSO BE SEARCHED.
00127                    *    IF TERMINATED BY -2, INTERNAL TABLE NOT CHECKED.
00128                    *
00129          F824   A FCTABL EQU    *
00130A F824    04     A        FCB    4          *
00131A F825    42     A        FCC    /B/        *
00132A F826    FB98   A        FDB    BRKPNT
00133A F828    04     A        FCB    4          *
00134A F829    43     A        FCC    /C/
00135A F82A    FC0F   A        FDB    CALL
00136A F82C    04     A        FCB    4
00137A F82D    44     A        FCC    /D/
00138A F82E    FD86   A        FDB    DISPLY
00139A F830    04     A        FCB    4          *
00140A F831    47     A        FCC    /G/
00141A F832    FC11   A        FDB    GOXQT
00142A F834    04     A        FCB    4          *
00143A F835    4C     A        FCC    /L/
00144A F836    FE9A   A        FDB    LOAD
00145A F838    04     A        FCB    4          *
00146A F839    4D     A        FCC    /M/        *
00147A F83A    FA5B   A        FDB    MEMORY
00148A F83C    04     A        FCB    4          *
00149A F83D    4F     A        FCC    /O/
00150A F83E    FAB4   A        FDB    OFFSET
00151A F840    04     A        FCB    4          *
00152A F841    50     A        FCC    /P/
00153A F842    FE72   A        FDB    PUNCH
00154A F844    04     A        FCB    4          *
00155A F845    52     A        FCC    /R/
00156A F846    FB1E   A        FDB    REGSTR
00157A F848    05     A        FCB    5
00158A F849    48     A        FCC    /HI/
     A F84A    49     A
00159A F84B    F8E9   A        FDB    S120
00160A F84D    05     A        FCB    5
00161A F84E    48     A        FCC    /HY/
     A F84F    59     A
00162A F850    F8F1   A        FDB    HY
00163A F852    04     A        FCB    4          *
00164A F853    54     A        FCC    /T/
00165A F854    FC46   A        FDB    TRACE
00166A F856    04     A        FCB    4          *
00167A F857    56     A        FCC    /V/
00168A F858    FEAB   A        FDB    VERF
00169A F85A    FE     A        FCB    -2         *END OF TABLE
```

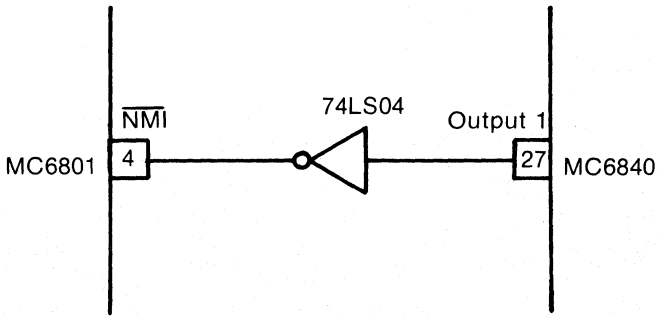
Figure 4.3

4.4 HARDWARE TRACE

Hardware trace permits a user's program to be executed one instruction at a time. Trace and breakpoint functions need this ability. The trace will operate on programs in either RAM or ROM. The circuitry as diagramed below consists of NMI tied to the output of the on-chip timer or to the inverted output of a PTM (Programmable Timer Module). Figure 4.4 shows the output waveform and timing for the on-chip timer and the PTM.



Hardware Trace Using the MC6801 Timer



Hardware Trace Using the PTM

Using the on-chip timer, Port 2 is set for output and the output level bit in the TCSR (Timer Control/Status Register) is set low. The on-chip clock is read. That time is adjusted and stored in the OCREG (Output Compare Register) such that the compare occurs after the first cycle of the next user instruction. After storing the OCREG, the monitor does an 'RTI'. When the compare to the OCREG occurs, the output wave goes low (corresponding to low setting of level bit in TCSR). The low transition causes an NMI. NMI vectors back to the monitor. The output waveform is brought high by setting the level bit in TCSR and by setting OCREG for another compare.

If a PTM is selected for trace instead of the on-chip timer, timer 1 is used. NMI is tied to the inverted output. To execute one user instruction, latches are set such that the output level changes after the first cycle of the user instruction. The PTM output level returns to its initial level after one cycle.

4.5 INITIALIZATION

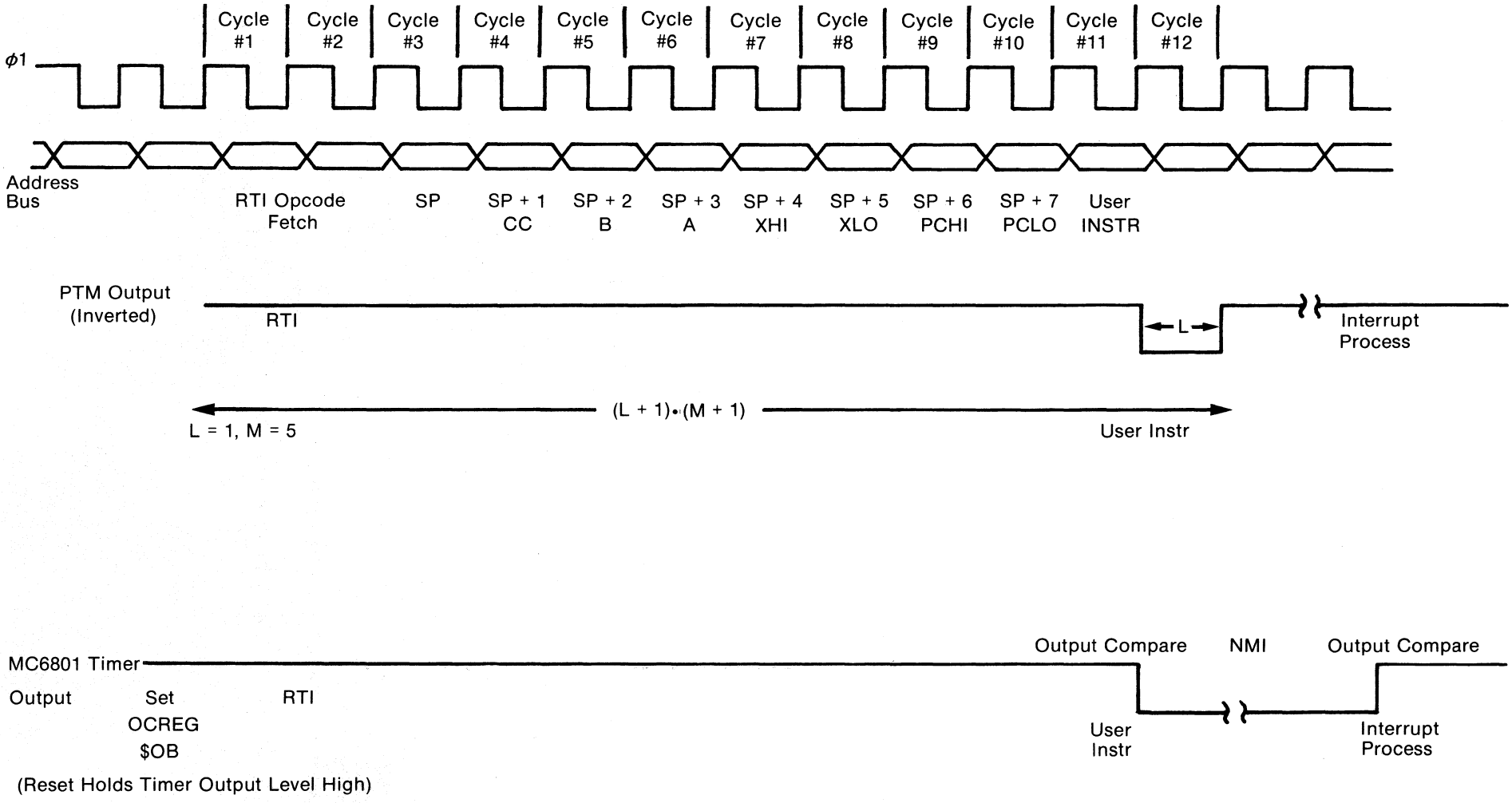
An MC6801 mode that best allocates resources for the user's software must be selected. The MC6801 has eight hardware selectable modes controlling on-chip use of ROM, RAM, interrupt vectors, and I/O ports. Figures 4.5.1 and 4.5.2 are the mode select chart and mode overview.

Modes 2, 3, and 4 cannot use the monitor since the internal ROM is not used. Mode 0 is a test mode designed to check on-chip memory. Mode 7 is minimal having only the on-chip ROM (monitor) and 128 bytes of on-chip RAM. The monitor uses about half of the on-chip RAM. The monitor stack is at address $OCF, and RAM variables are from $ODO to $OFF. The user has the low addresses in RAM. Modes 1, 5, and 6 allow development of large programs since they can have expanded memory. In modes 5 and 6, the user must program port 4 as output to have external memory. Modes 5 and 6 have internal vectors while mode 1 has external vectors. Mode 1 is most easily used with programs that use PTM trace, or programs that redefine I/O devices. For these programs, external devices (and addresses) are necessary. Also, external vectors allow the user to jump to his own initialization routine at RESET.

If the 6801 is in a mode that has internal vectors and LILbug enabled, the RESET vector initializes the monitor's on-chip serial I/O and on-chip timer for hardware trace and breakpoint. The vector table, command table, and I/O tables are initialized. Calls are made to turn on console reader and printer and I/O devices are initialized.

If the user selects a monitor option, such as adding user-defined commands to the command table, a mode with external vectors is usually more convenient. The RESET vector would then jump directly to a user initialization routine. The user needs to set the command table pointer, and then jump into the monitor RESET code such that the table pointer is not redefined to the monitor default. This may cause omitting other initialization which the user routine needs to do prior to entering the monitor. An example is USRCMD in Appendix C. The routine defines a conversion command. The program sets the pointer to the user-command table, and also the pointer to the monitor I/O table. USRCMD then jumps to the monitor code following the monitor setting of the command table pointer

When the user selects mode 1, which allows external vectors with the internal ROM, the MC6801 will input data from the external data bus for the addresses $FFF0 through $FFFF. This function allows the user to define his own vector table in the MC6801 memory map. A full decode of the external memory is not required, due to the architecture of the MC6801, which selects either the internal data bus or external data bus to read the data for these sixteen memory locations.



Hardware Trace
NMI Tied To PTM or On-Chip Timer Output
Figure 4.4

MODE	PC2	PC1	PC0	ROM	RAM	INTERRUPT VECTORS	BUS MODE	OPERATING MODE
7	H	H	H	I	I	I	I	Single Chip
6	H	H	L	I	I	I	MUX (6)	Multiplexed/Partial Decode
5	H	L	H	I	I	I	NMUX (6)	Non-Multiplexed Partial Decode
4	H	L	L	I(2)	I(1)	I	I	Single Chip Test
3	L	H	H	E	E	E	MUX	Multiplexed/No RAM or ROM
2	L	H	L	E	I	E	MUX	Multiplexed/RAM
1	L	L	H	I	I	E	MUX	Multiplexed/RAM & ROM
0	L	L	L	I	I	I(3)	MUX	Multiplexed Test

LEGEND:

I — Internal MUX — Multiplexed L — Logic "0"
E — External NMUX — Non-Multiplexed H — Logic "1"

NOTES:

(1) Internal RAM is addressed at $XX80
(2) Internal ROM is disabled
(3) RESET vector is external for 2 cycles after RESET goes high
(4) Addresses associated with Ports 3 and 4 are considered external in Modes 3 and 4
(5) Addresses associated with Port 3 are considered external in Modes 5 and 6
(6) Port 4 default is user data input; address output is optional by writing to Port 4 Data Direction Register

MC6801 Mode Selects
Figure 4.5.1

OVERVIEW OF MC6801 OPERATING MODES

COMMON TO ALL MODES:

- Reserved Register Area
- I/O Port 1 Operation
- I/O Port 2 Operation
- Timer Operation
- Serial I/O Operation

SINGLE CHIP-MODE 7

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is parallel I/O handshaking port
- Port 4 is parallel I/O port

EXPANDED MEMORY SPACE/NON-MULTIPLEXED BUS-MODE 5

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is 8-bit data bus
- Port 4 is optional 8-bit address bus
- 256 bytes of external memory space
- External memory space select output ($\overline{IOS}$)

EXPANDED MEMORY SPACE/MULTIPLIED BUS-MODES 1, 2, 3, 6

- Port 3 is multiplexed address/data bus
- Port 4 is address bus
- 4 memory space options
  (1) No internal RAM or ROM (Mode 3)
  (2) Internal RAM (Mode 2)
  (3) Internal RAM and ROM (Mode 1)
  (4) Internal RAM, ROM with partial address bus (Mode 6)

TEST-MODES 0 and 4

- Expanded Test-Mode 0
  May be used to test internal RAM and ROM
- Single Chip and Non-Multiplexed Test-Mode 4
  (1) May be changed to Mode 5 without $\overline{RESET}$
  (2) May be used to test Ports 3 and 4 as I/O ports

Figure 4.5.2

4.6 JUMP TABLE

LILbug subroutines and entries frequenty used are in a jump table at the beginning of LILbug. These entries should be used whenever possible to insure compatibility with future versions of the monitor. Following is the jump table as defined in LILbug:

```
 00094A F800                          ORG     #F800
00114

00096                          * JUMP TABLE TO SUBROUTINES
00097A F800 7E FC7E  A EX.NMI  JMP     M.NMI     NMI VECTOR FOR PTM
00098A F803 7E FC79  A IN.NMI  JMP     C.NMI     NMI VECTOR FOR ON-CHIP TIMER
00099A F806 7E F873  A INCHNP  JMP     INCH1     INPUT 1 CHAR W/ NO PARITY
00100A F809 7E F88A  A OUTCH   JMP     OUTCH1    OUTPUT 1 CHAR W/PADDING
00101A F80C 7E FB07  A         JMP     PDATA1    PRINT DATA STRING
00102A F80F 7E FB0E  A         JMP     PDATA     PR CR/LF, DATA STRING
00103A F812 7E FADB  A         JMP     OUT2HS    PR 2 HEX + SP (X)
00104A F815 7E FAD8  A         JMP     OUT4HS    PR 4 HEX + SP (X)
00105A F818 7E FB12  A         JMP     PCRLF     PRINT CR/LF
00106A F81B 7E FADD  A         JMP     SPACE     PRINT A SPACE
00107A F81E 7E F8F6  A STRT    JMP     START     RESTART ADDRESS
00108A F821 7E FD5F  A IN.SWI  JMP     M.SWI     SWI VECTOR
```

Following is a description of the jump table entries and their associated subroutines.

Entry	Address	Description
M.NMI	$F800	NMI entry for hardware trace using a PTM. The LILbug monitor initializes the PTM if the NMI vector is this address.
C.NMI	$F803	NMI entry for hardware trace using on-chip timer. The LILbug monitor initializes the on-chip timer if the NMI vector is this address. This is the default NMI vector.
INCH1	$F806	Call I/O routine CIDTA to input one character into register A. Carry bit is clear until data read, then it is set. INCH1 waits for input, clears parity, and ignores rubouts. OUTCH1 is called to echo input if OUTSW is clear.
CIDTA	----	I/O routine that reads one character of data into register A using on-chip serial I/O. This is a read with no waiting for data. Return with carry bit set if data is read, otherwise carry bit is clear.
OUTCH1	$F809	Call I/O routine to output one character from register A. Saves register B.
CODTA	----	Default LILbug I/O routine called by OUTCH1. Call OUTC to use on-chip serial interface to output one character. Write nulls (padding) after output character. OUTSW and CHRNL control padding. OUTSW is the tape flag; set to 0 if not tape, set to number of nulls to follow character if tape. The upper six bit value of CHRNL is the padding for a carriage return. The low two bit value is the padding for other characters.
PDATA1	$F80C	Print data string pointed to by register X. Last character of data string should be $04. Registers A and X are modified. Call OUTCH1 to print one character at a time.
PDATA	$F80F	Call PCRLF to print carriage return and line feed. Then call PDATA1 to print data string pointed to by register X. Registers A and X are modified.
OUT2HS	$F812	Output two hexadecimal characters pointed to by register X. Then output a space. Modifies register A. Calls OUT2H and SPACE.
OUT2H	----	Print two hexadecimal characters pointed to by register X. Calls OUTHL and OUTHR to convert and print a byte.
OUTHL	----	Convert left four bits of a byte to display code and output. Calls OUTCH1 to print.
OUTHR	----	Convert right four bits of a byte to display code and output. Calls OUTCH1 to print.
OUT4HS	$F815	Call OUT2H and OUT2HS to print four hexadecimal characters pointed to by register X.

PCRLF	$F818	Call OUTCH1 to output a line feed and then a carriage return.
SPACE	$F81B	Call OUTCH1 to output a space.
START	$F81E	Reset entry point. This is the default RESTART vector.
M.SWI	$F821	Default SWI vector in LILbug monitor. This routine processes breakpoints.

5.0 HARDWARE REQUIREMENTS

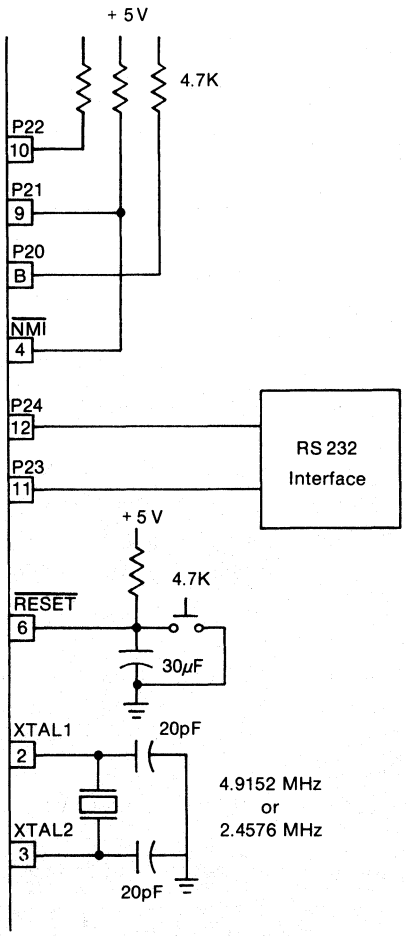
5.1 MINIMUM SYSTEM

The minimum system for the MC6801 requires only a clock generator, serial communication interface, and reset logic. This minimum system is shown in Figure 5.1.

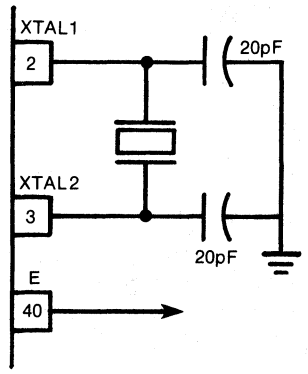
CLOCK— The two connections XTAL1 and XTAL2 can either be used for a parallel resonant crystal or an external clock source as shown in Figures 5.2 and 5.3.

The frequency of the crystal or clock source should be either 4.9152 MHz or 2.4576 MHz to give a standard baud rate for the serial communication interface. The following table shows the baud rates available with these crystals.

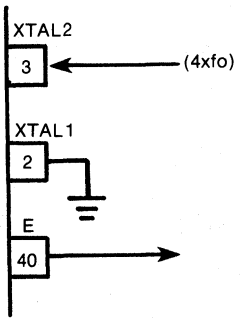
XTAL	4.9152 MHz	2.4576 MHz
$\phi 2$	1.2288 MHz	.6144 MHz
HY	9,600 Baud	4,800 Baud
HI	1,200 Baud	600 Baud
Reset	300 Baud	150 Baud



Minimum System Requirements
Figure 5.1



Connections For A Parallel Resonant Crystal
Figure 5.2



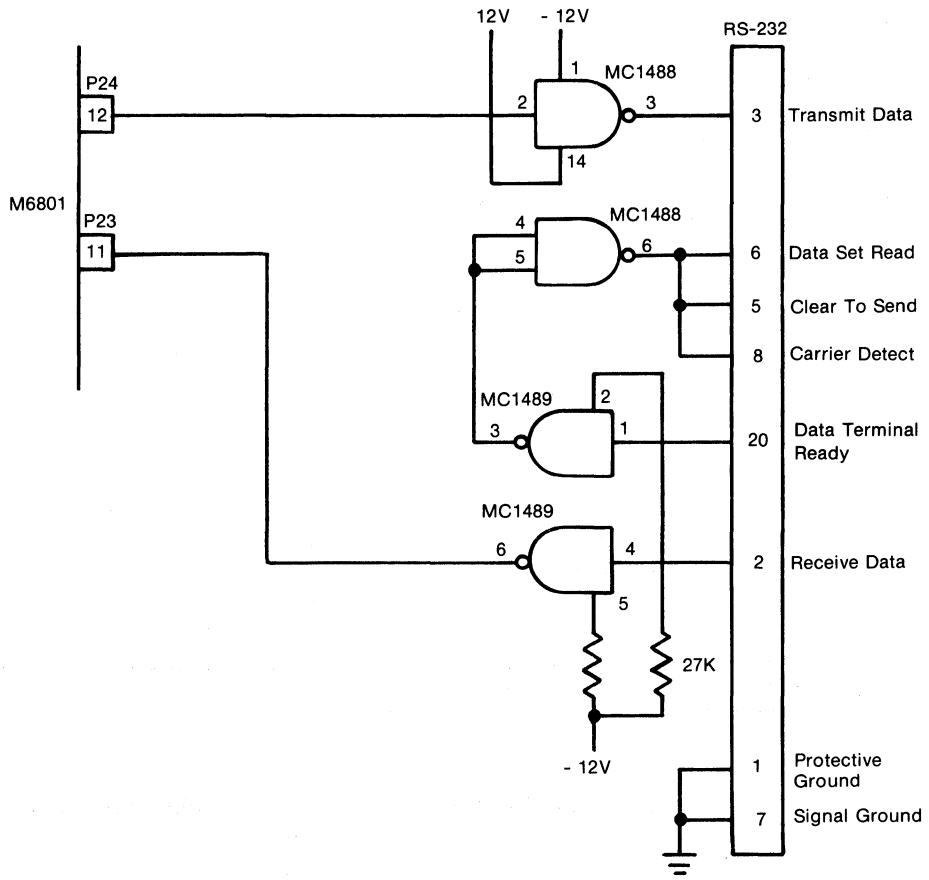
Connections For An External Clock
Figure 5.3

Serial Communication Interface

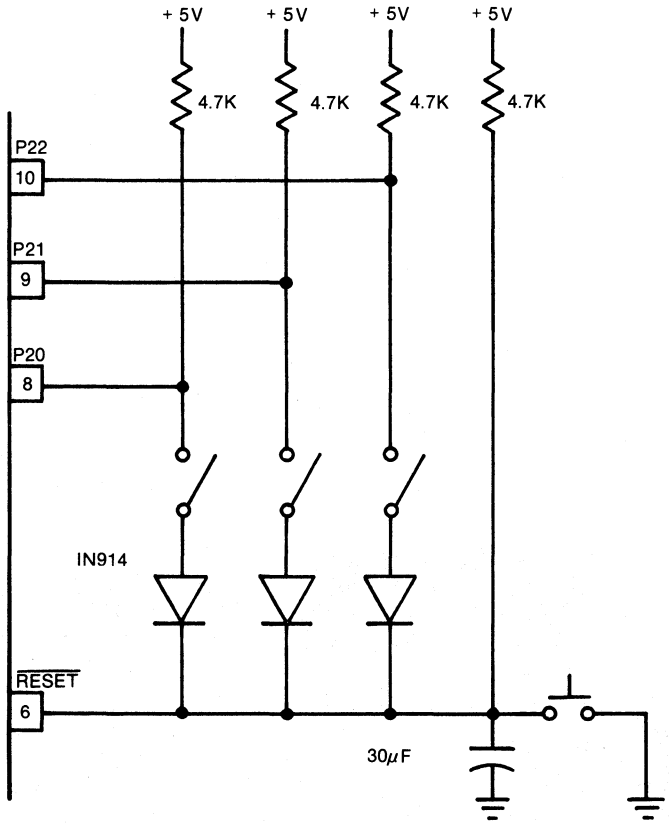
The MC1488 and MC1489A devices provide the system with a RS-232C interface capability. The circuit in Figure 5.4 interfaces the serial input/output of the MC6801 directly to an RS-232C terminal without any additional hardware.

Reset

The Reset Funtion should be performed when power is first applied and if the LILbug Firmware loses program control. The Reset Function is also used to select the mode of operation of the MC6801. The circuit shown in Figure 5.5 is a simple reset function. The diodes are used to select the different modes of operation during reset.



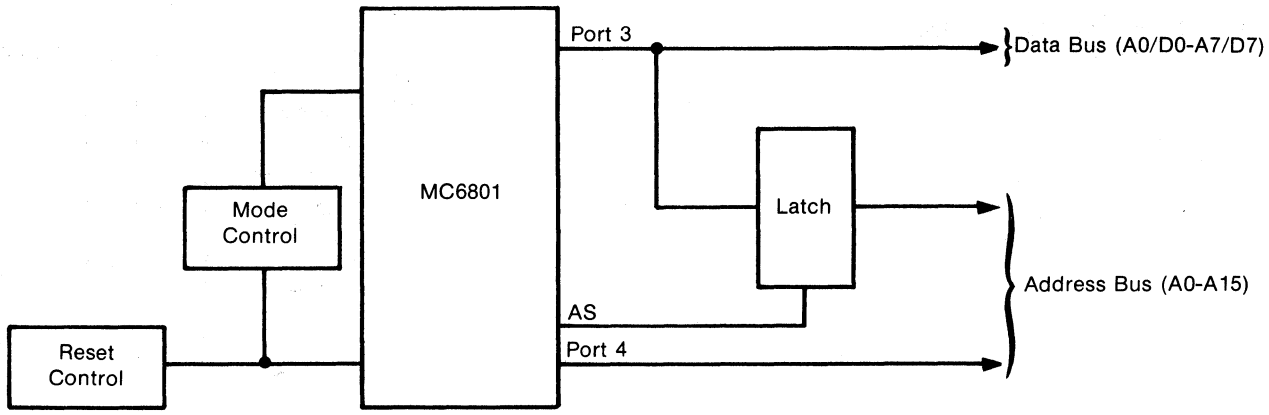
Serial Communication Interface
Figure 5.4



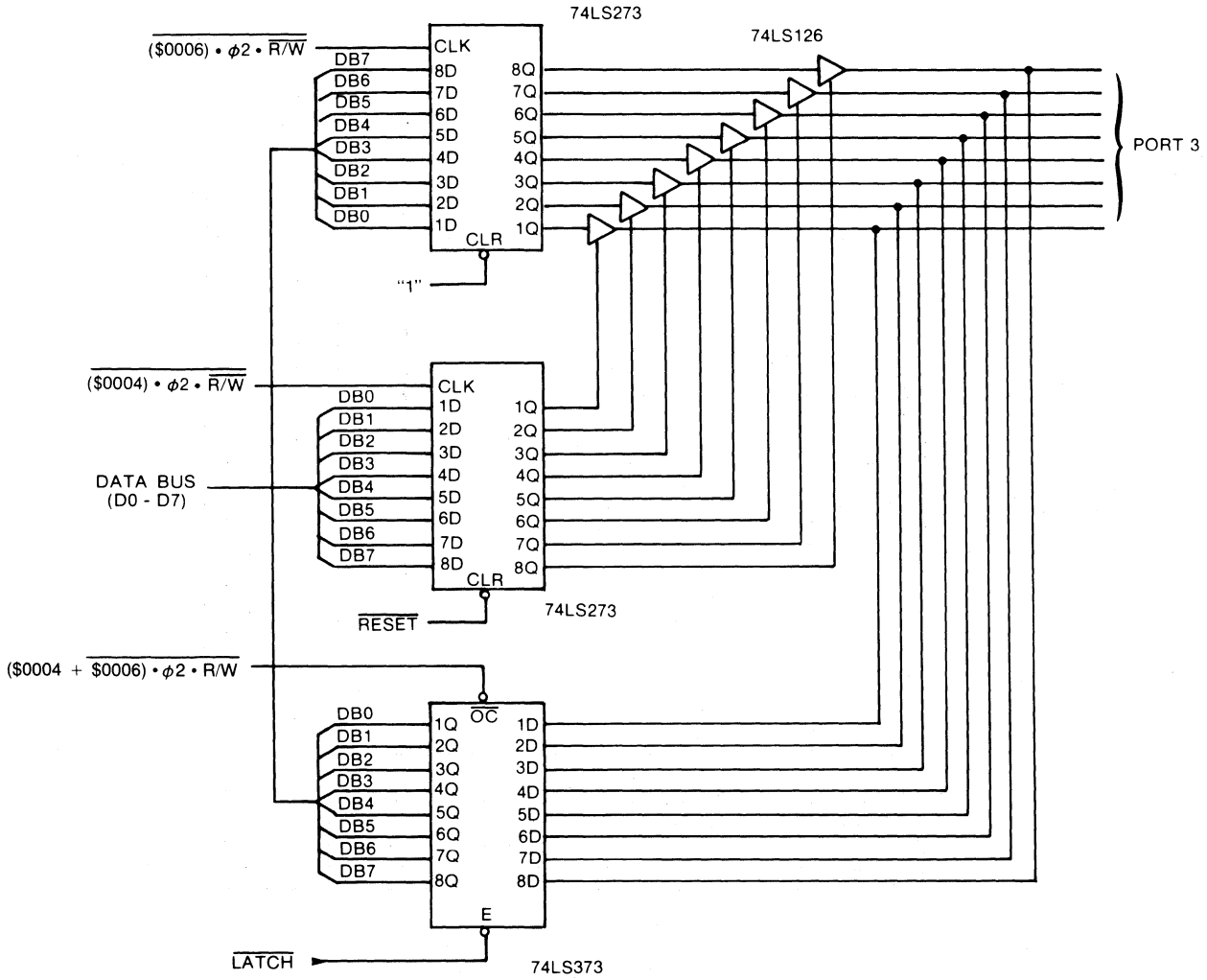
Simple Reset Logic
Figure 5.5

5.2 EXPANDED SYSTEMS

When one of the expanded modes of operation is selected, I/O Port 3 and I/O Port 4 can not be used as I/O lines. A Block Diagram of a Expanded Multiplexed System is shown in Figure 5.6. Port 3 becomes the multiplexed bus for data and address and Port 4 becomes the address bus. Figures 5.7, 5.8, 5.9 and 5.10 show the logic required to replace the I/O Ports when the MC6801 is operated in one of these expanded modes.

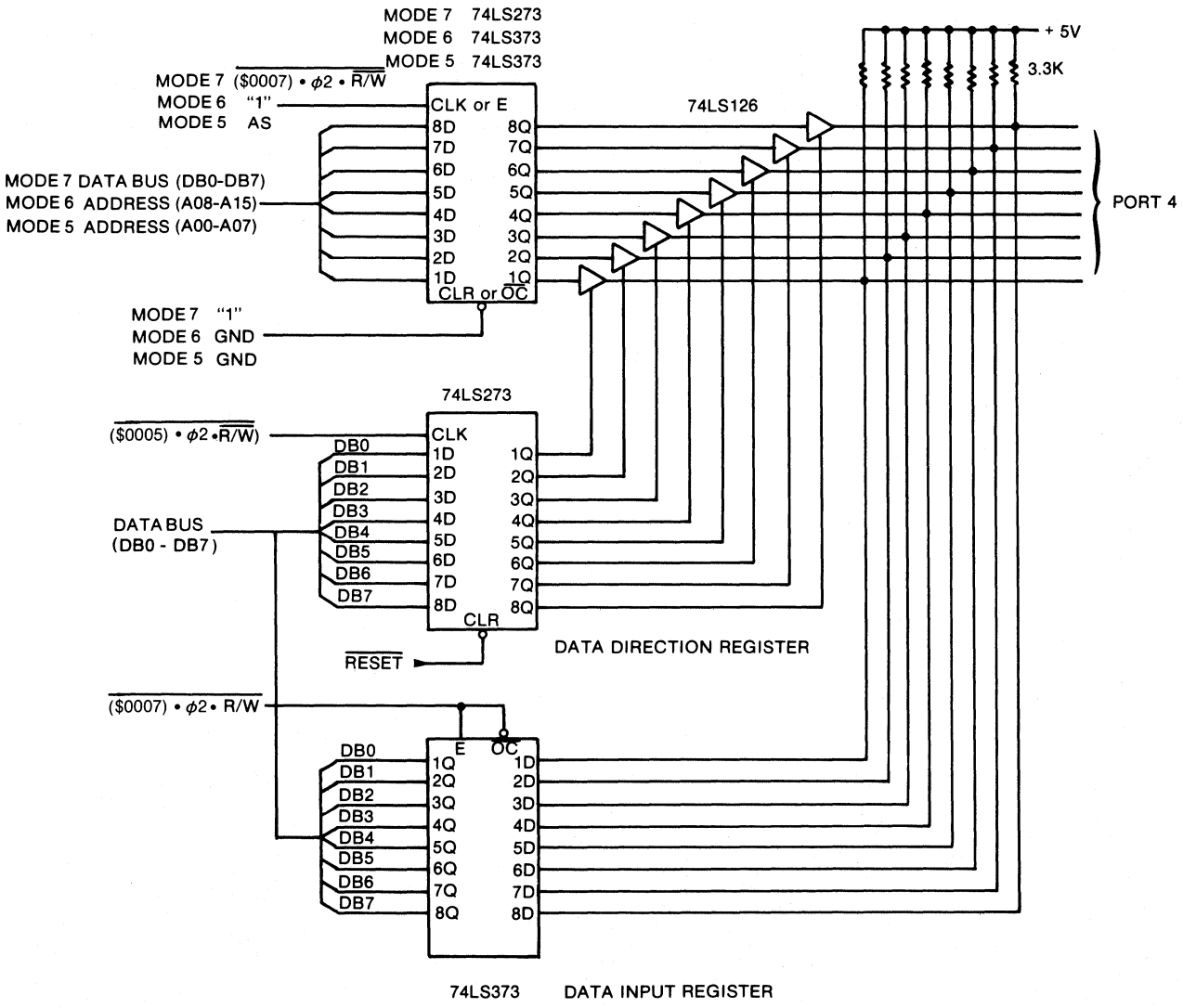


Block Diagram Of A Expanded Multiplexed System
Figure 5.6

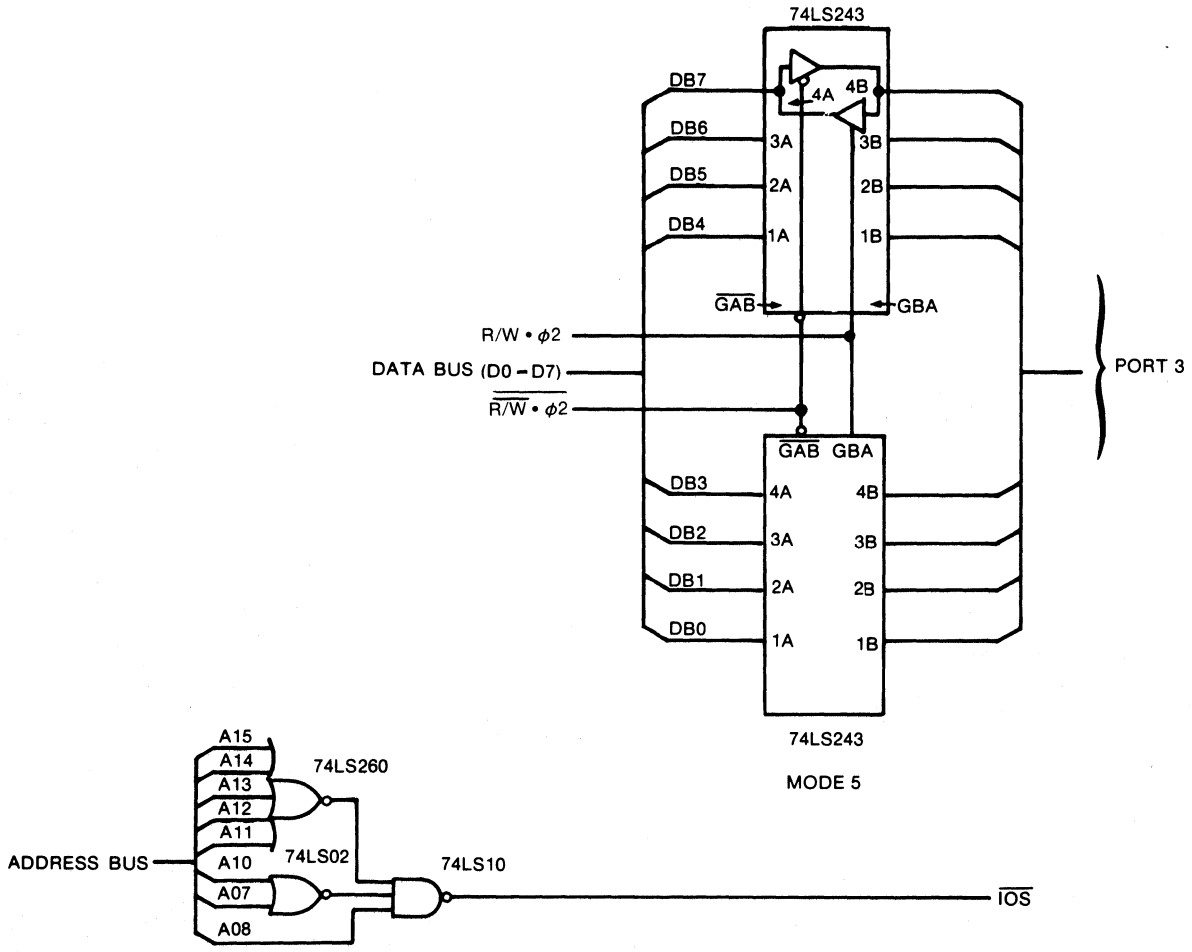


MODE 7

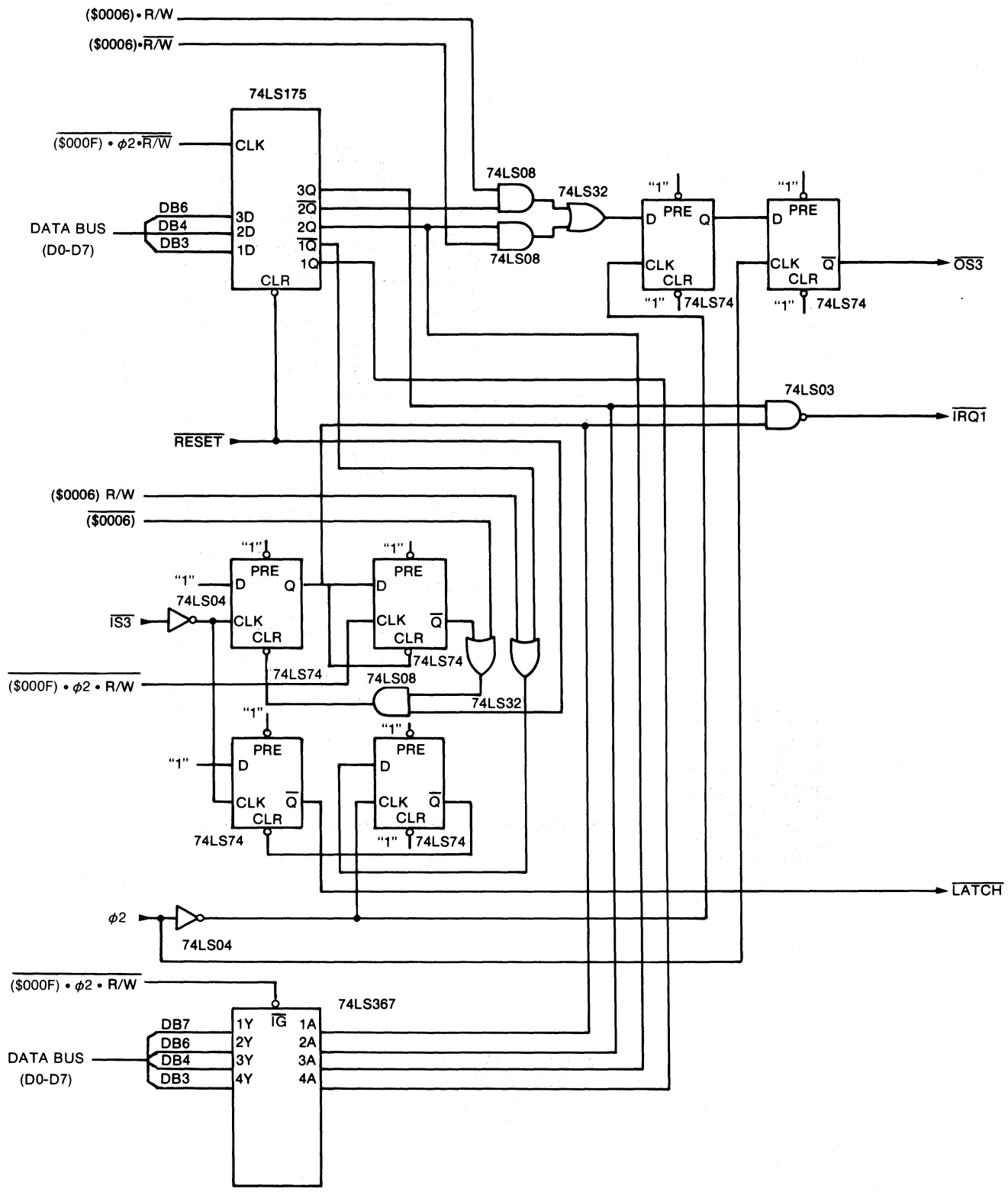
MODE 7 I/O PORT 3
FIGURE 5.7



MODES 5, 6, AND 7 PORT 4
FIGURE 5.8



MODE 5 I/O PORT 3
FIGURE 5.9



I/O PORT 3 CONTROL/STATUS REGISTER
FIGURE 5.10

APPENDIX A

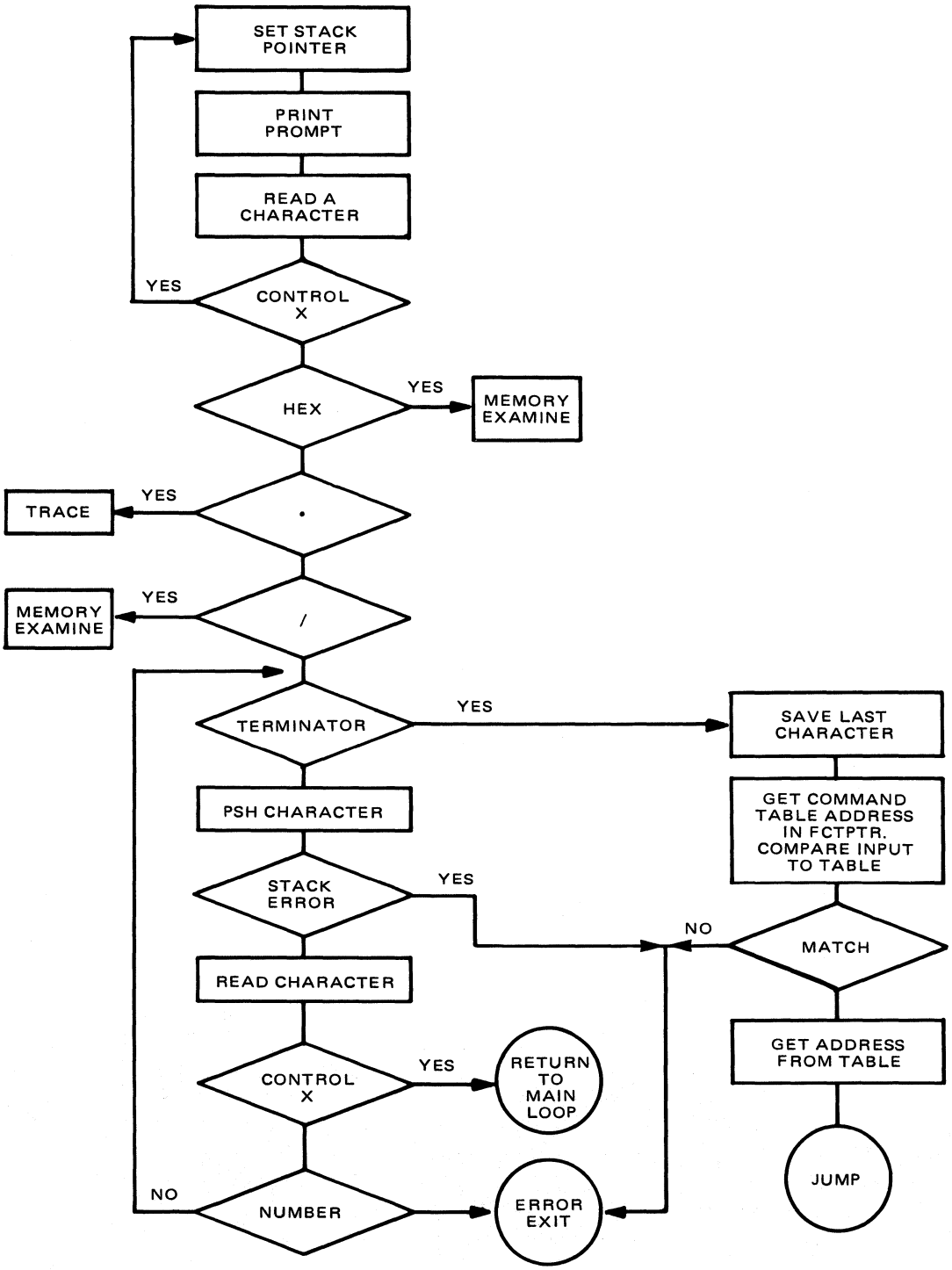
MC6801 MONITOR COMMAND SUMMARY

L [<offset>]	Load a program from tape. Add <offset>, if specified, to load address on tape.
V [<offset>]	Verify that a program was properly loaded. Add <offset>, if specified, to load address on tape.
D [<adr1> [,<adr2>]]	Display memory from <adr1> to <adr2>.
P <adr1> , <adr2>	Punch or record contents of memory from <adr1> to <adr2>.
M <adr>	Set pointer to <adr> and print value at that memory location. Next input: <data> Change one byte in memory to <data> (LF) Increment pointer and print address and value of new pointer. (UA) Up arrow, decrement pointer, print address and value of pointer. ( ) Space, increment pointer and print new value on same line. , Comma, increment pointer with no print of address or value. / Slash, print address and value of current pointer. (CR) End memory examine command.
<adr>/	Same as M <adr>; <adr> must start with 0-9, zero may precede hex address.
/	Print address and value of location last referenced by memory examine.
O <adr1> <adr2>	Calculate relative offset from <adr1> to <adr2> for branch instructions.
B	Display all breakpoint addresses.
B <adr>	Set a breakpoint at address <adr> and display breakpoints.
B-<adr>	Remove the breakpoint at address <adr> and display breakpoints.
B-	Remove all breakpoints and display breakpoints.
G [<adr>]	<Adr>, if specified, becomes the new program counter. Execute from program counter.
R	Display/modify user's MPU registers. Next input: <data> Change value of register to <data> ( ) Space, display register value and move pointer to next register. (CR) Terminate register change command.
.	Period. Trace one instruction. (No carriage return after period).
T [<hex value>]	Trace <hex value> number of commands.
C [<adr>]	<Adr>, if specified, becomes the new program counter. Execute code as subroutine starting at address in program counter. User "RTS" returns to monitor.
HI	Set speed for on-chip I/O to 1200 baud (÷1024).
HY	Set speed for on-chip I/O to 9600 baud (÷128).

APPENDIX B
FLOWCHARTS

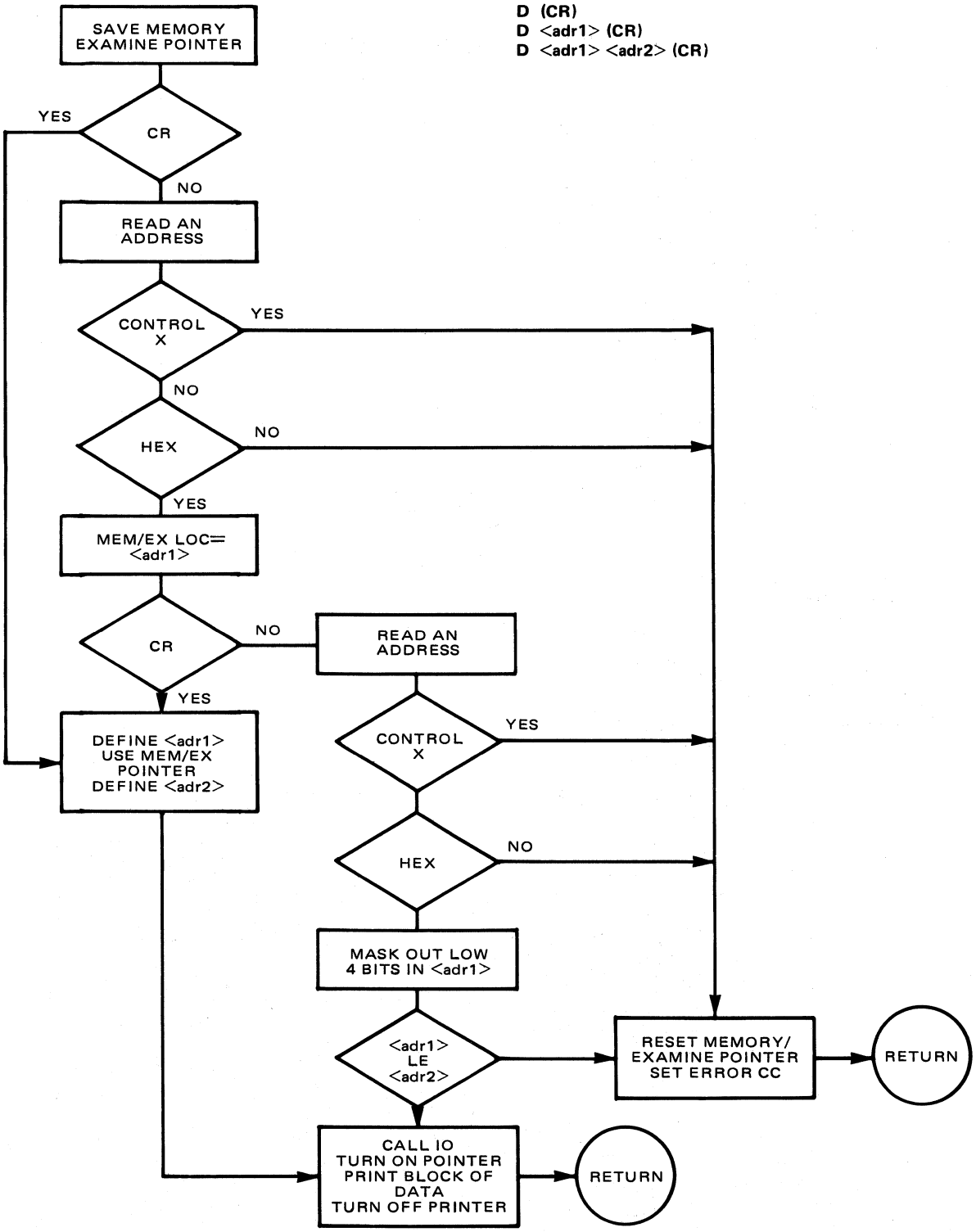
1) **MAIN** — command loop
2) **DISPLAY** — print block of memory command, call HSDTA
3) **HSDTA** — print block of data
4) **VERIFY/LOAD/PUNCH** — initialization for BSDTA call
5) **BSDTA** — bulk store date used by tape commands
6) **MEMORY** — examine/change memory command
7) **PRINT/MODIFY registers** — R command
8) **OFFSET**
9) **BREAKPOINT**
10) **GO/CALL/TRACE** — execution commands
11) **CRTS** — return to monitor after CALL
12) **NMI entry** — breakpoint or trace
13) **SWI** — breakpoint process

MAIN COMMAND LOOP



DISPLAY MEMORY

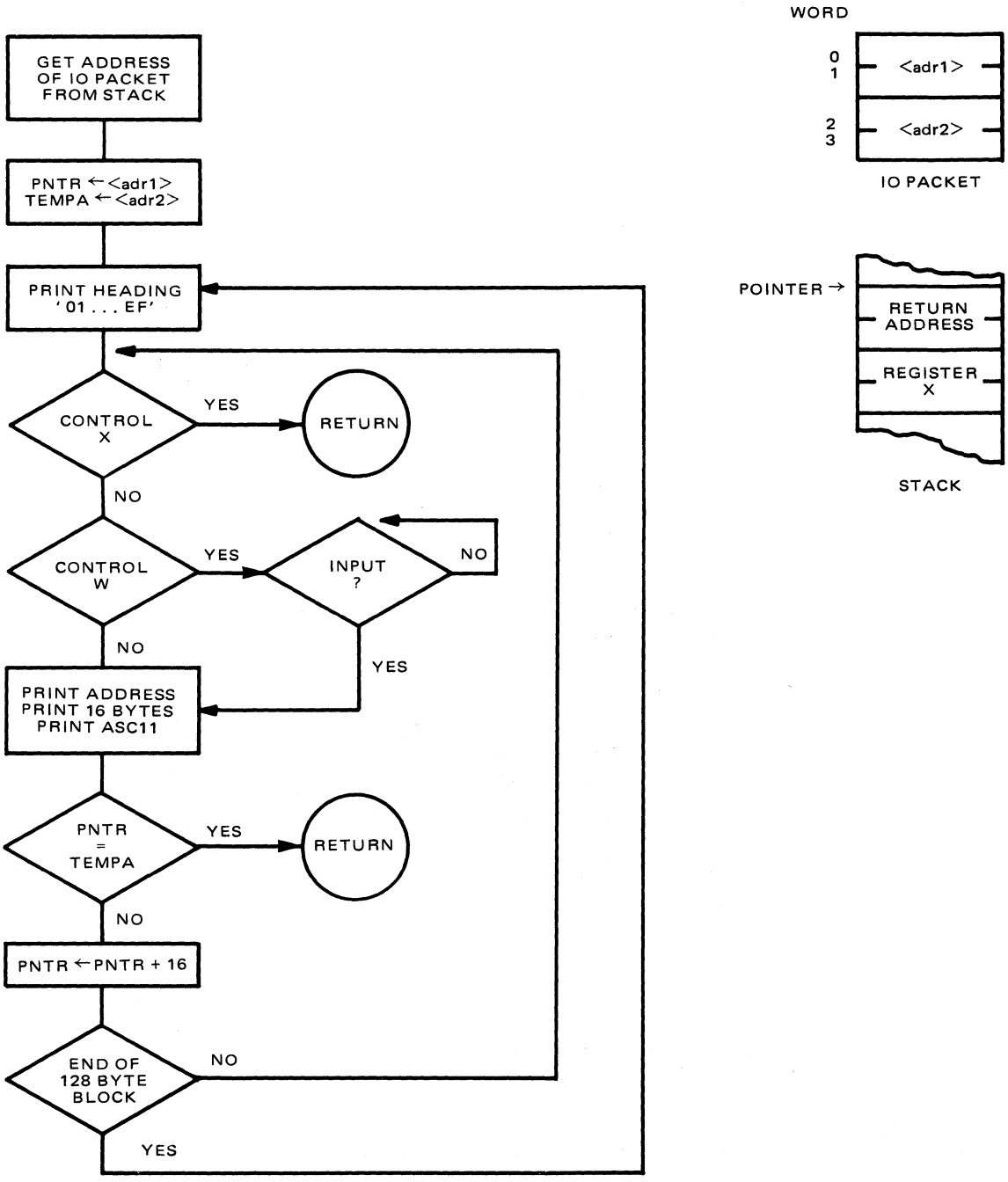
D (CR)
D <adr1> (CR)
D <adr1> <adr2> (CR)



HSDTA (PRINT BLOCK OF DATA)

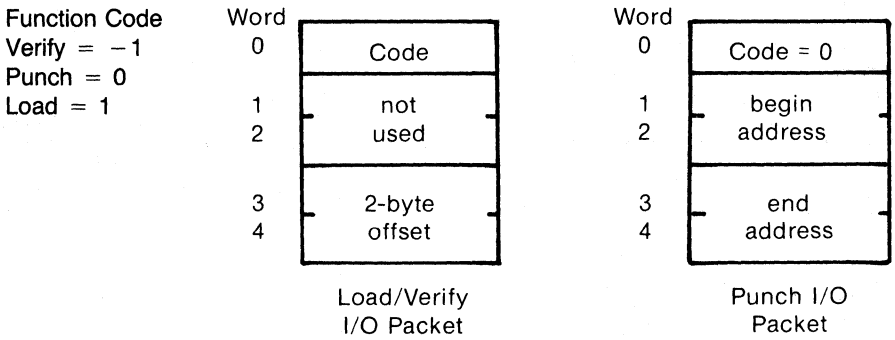
HSDTA is called indirectly through the routine I/O and is the default high speed print routine. The index pointer (register B) for HSDTA is $0E. The I/O Packet contains the start address and the end address. Register X points to the I/O Packet that is transferred to all high speed print routines. The I/O Routine "pushes" register X before calling HSDTA; therefore the address of the I/O Packet follows the return address on the stack.

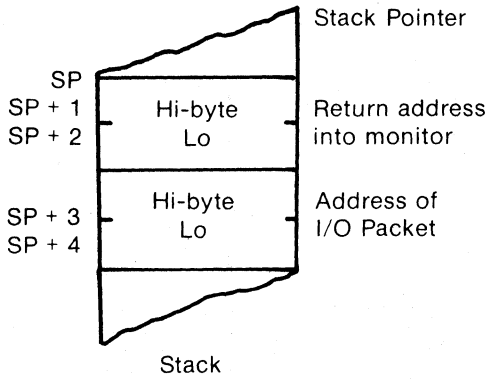
The routine displays a block of memory to the console device by repetitively calling the character output routine.



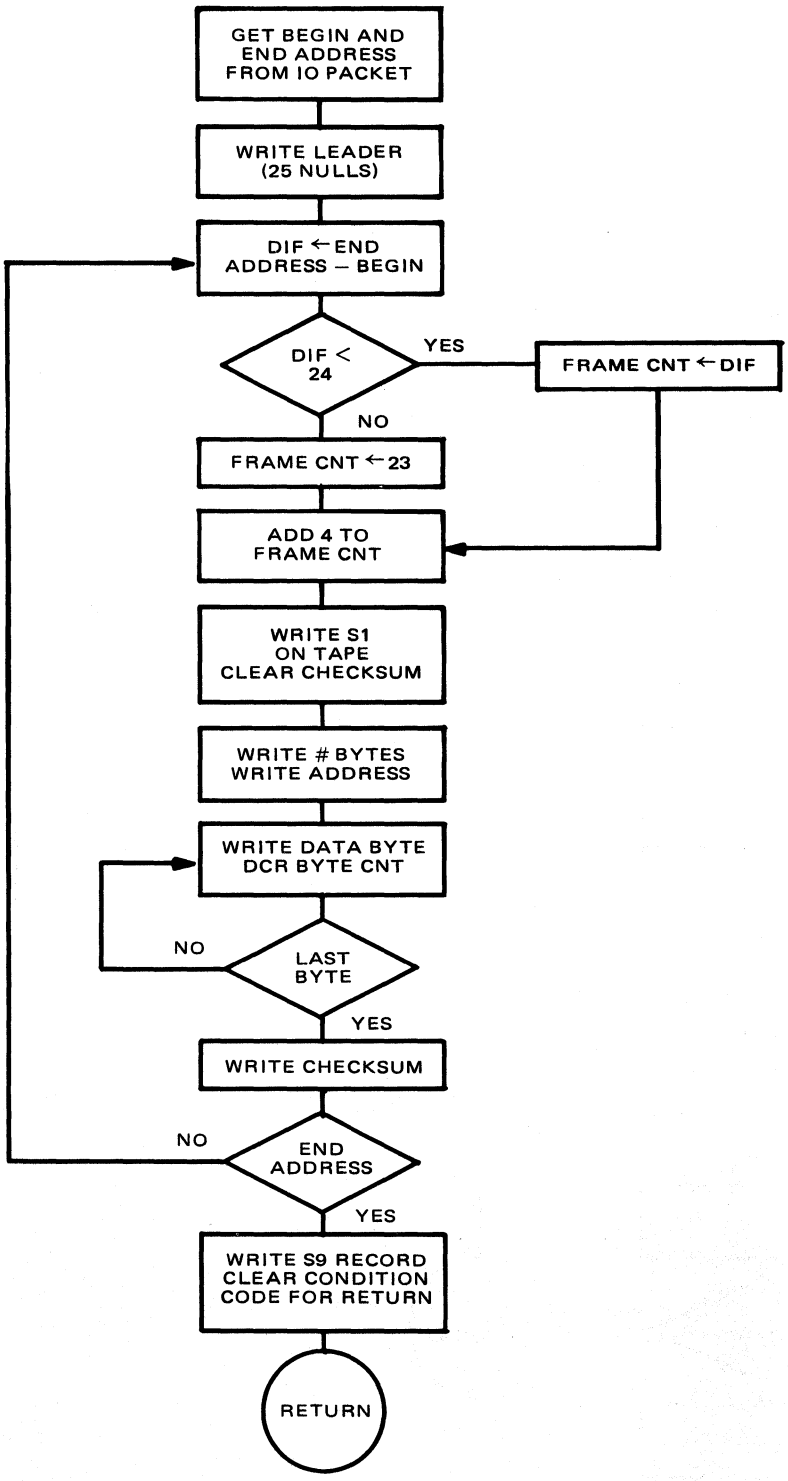
BSDTA (BULK STORE DATA)

BSDTA is a subroutine that reads and writes tape. The monitor's default Load, Punch, and Verify commands use BSDTA. BSDTA is called indirectly by setting the index pointer (register B) to $14 and register X to point to the I/O packet; then call routine IO. Unless the high speed punch/load device has been redefined, the index points to the BSDTA address in the I/O table (defined in the I/O Independence section). The I/O packet passed to all high speed punch/load routines consists of a function code indicating load, punch, or verify. The punch I/O packet has a begin address and an end address. The load and verify routines have a two-byte offset in the I/O packet. The IO routine pushes register X before calling BSDTA; therefore the address of the I/O packet precedes the return address on the stack. This routine calls CIDTA and CODTA to do character input and output.

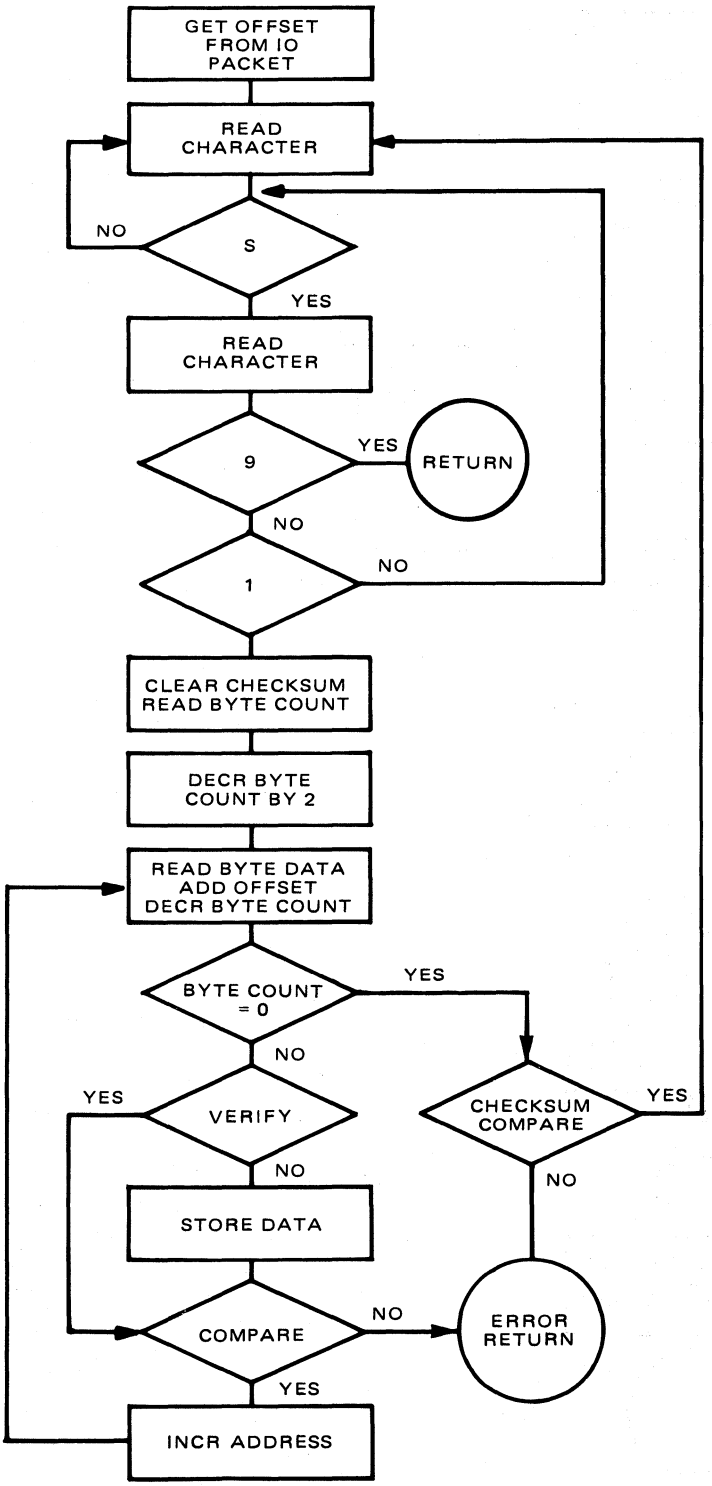




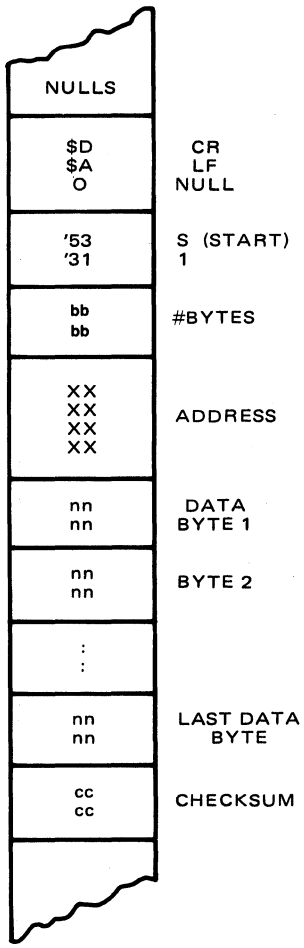
BSDTA
PUNCH


GET BEGIN AND END ADDRESS FROM IO PACKET
WRITE LEADER (25 NULLS)
DIF ← END ADDRESS – BEGIN
DIF < 24
YES
FRAME CNT ← DIF
NO
FRAME CNT ← 23
ADD 4 TO FRAME CNT
WRITE S1 ON TAPE CLEAR CHECKSUM
WRITE # BYTES WRITE ADDRESS
WRITE DATA BYTE DCR BYTE CNT
NO
LAST BYTE
YES
WRITE CHECKSUM
NO
END ADDRESS
YES
WRITE S9 RECORD CLEAR CONDITION CODE FOR RETURN
RETURN

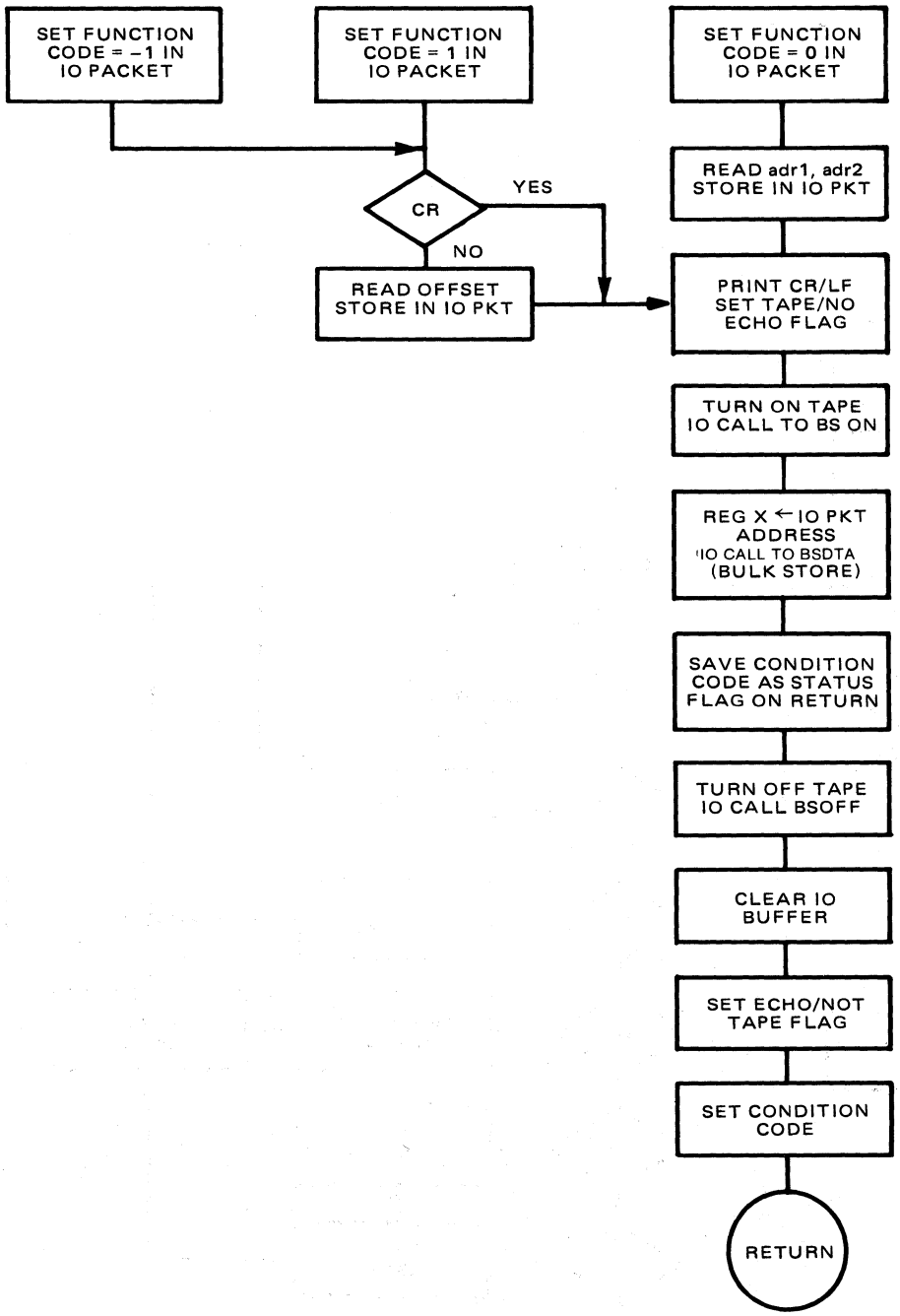
BSDTA
LOAD/VERIFY



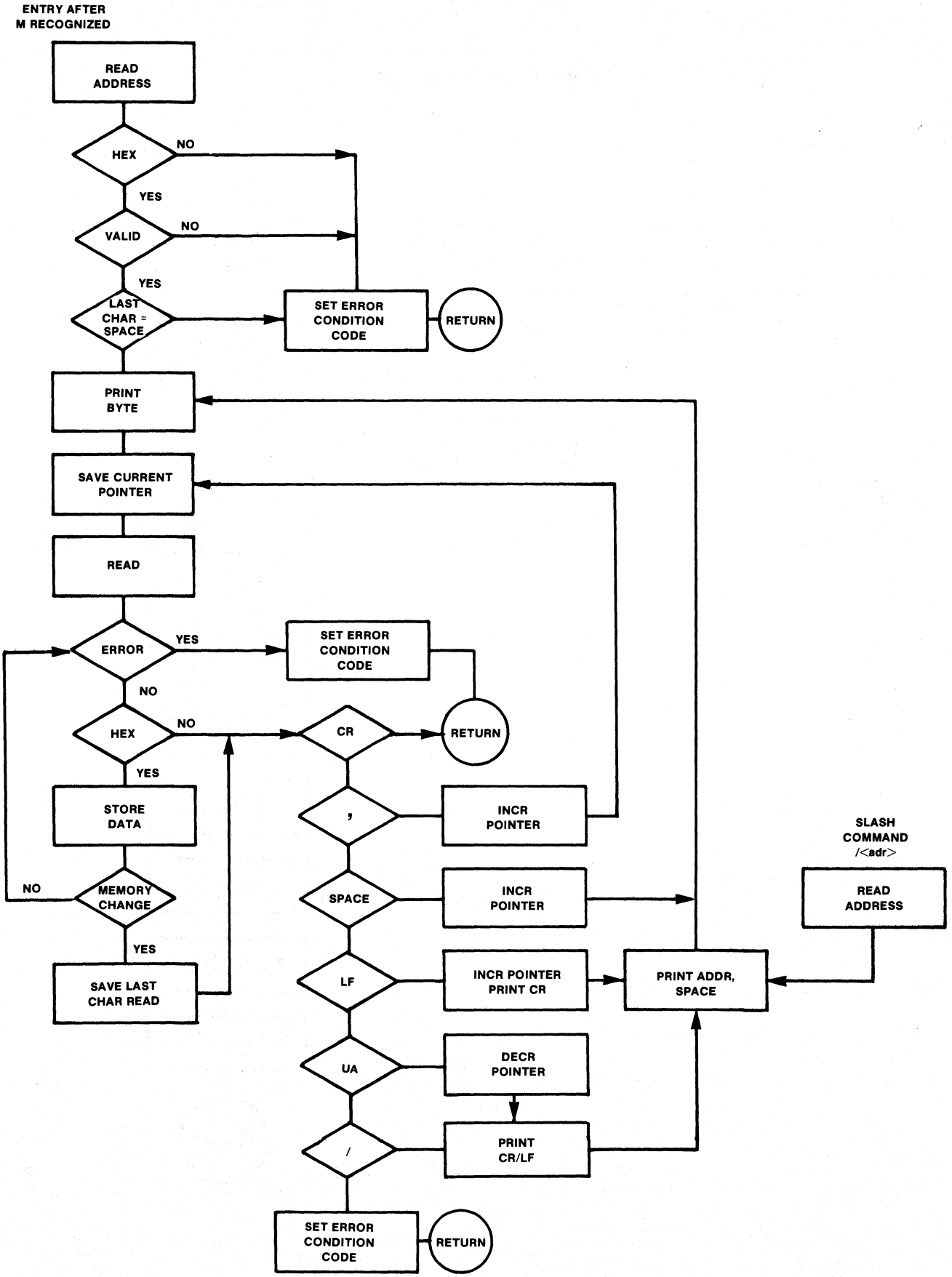
TAPE FORMAT



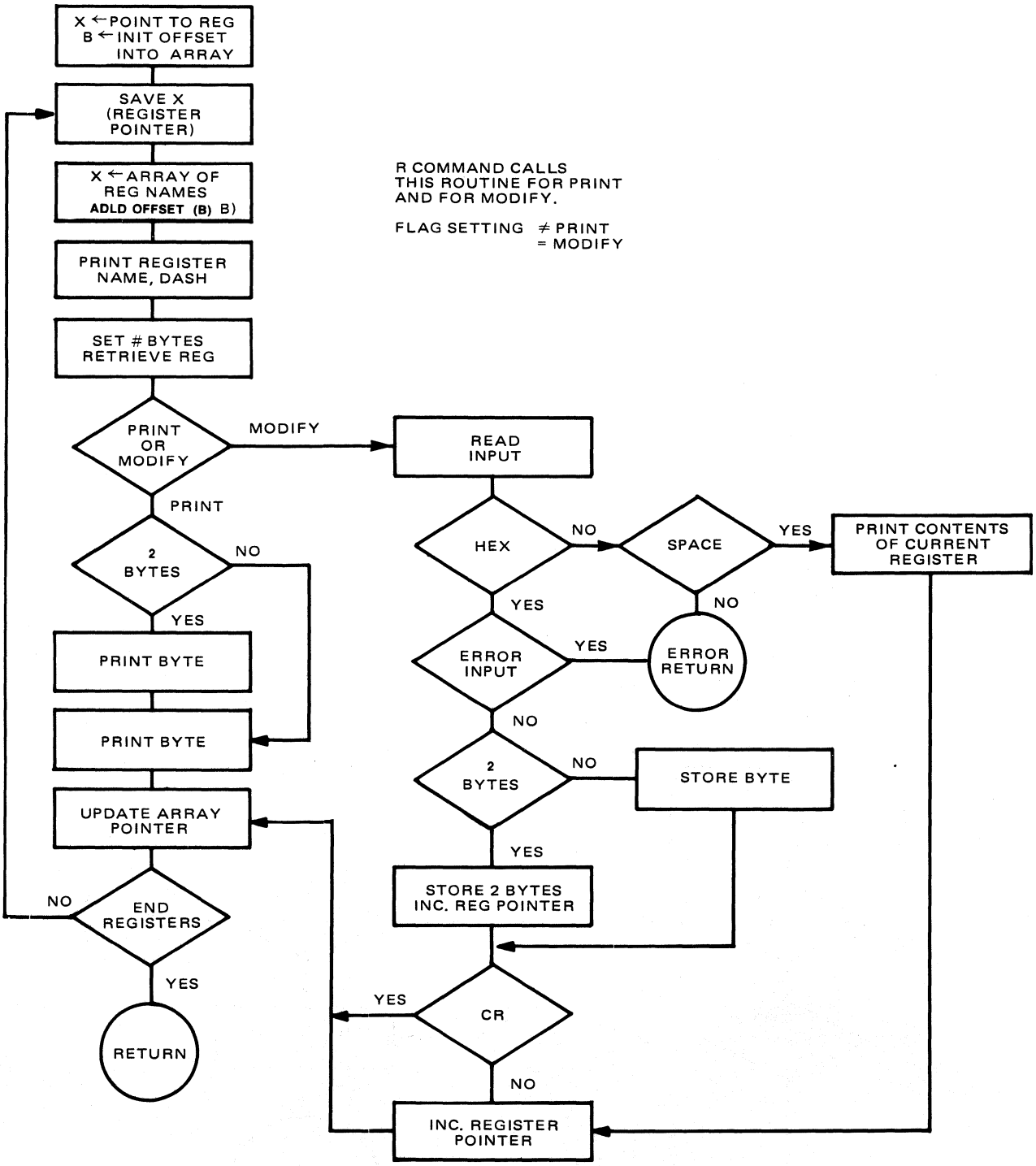
VERIFY LOAD PUNCH



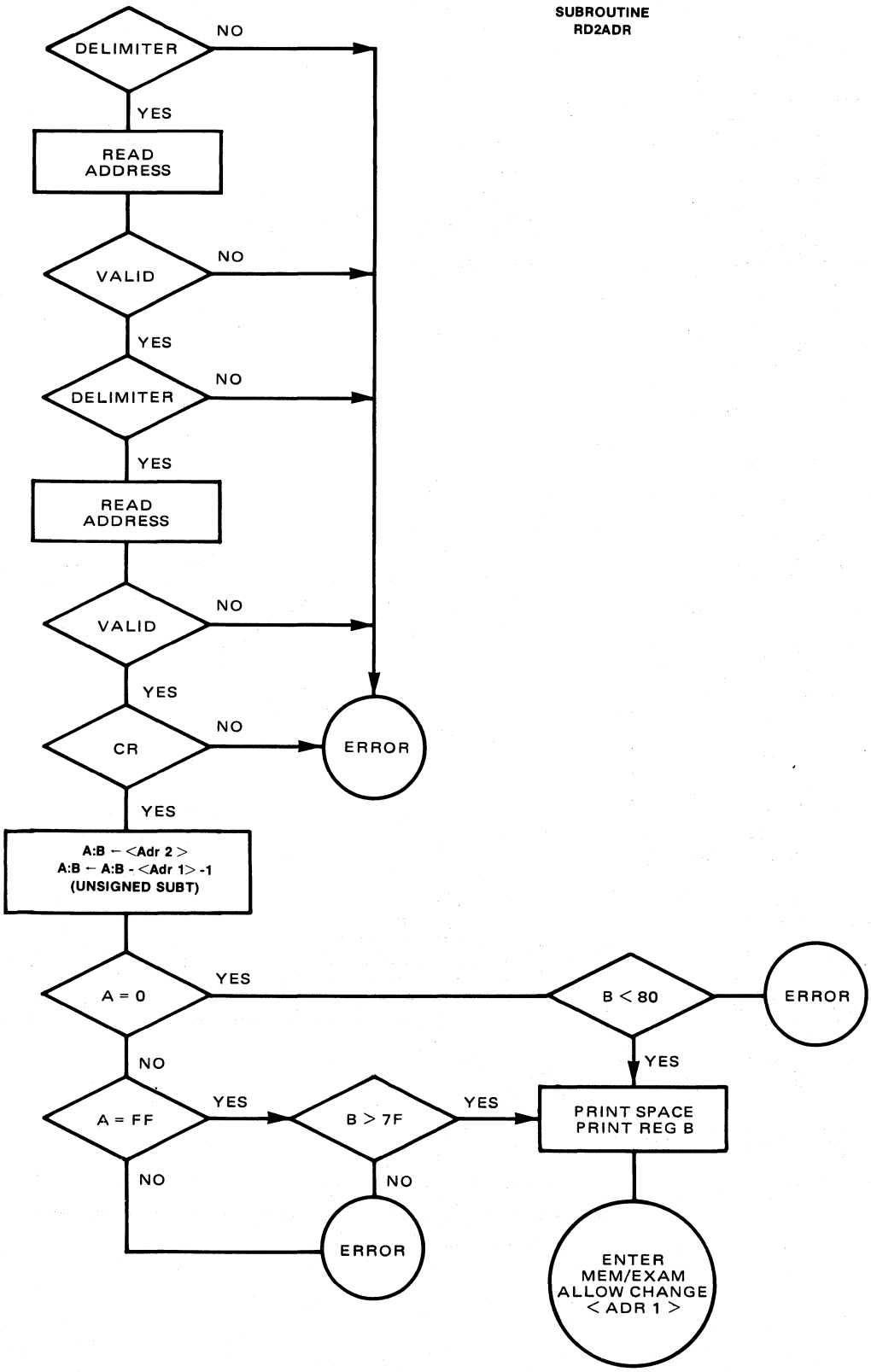
MEMORY EXAMINE/CHANGE



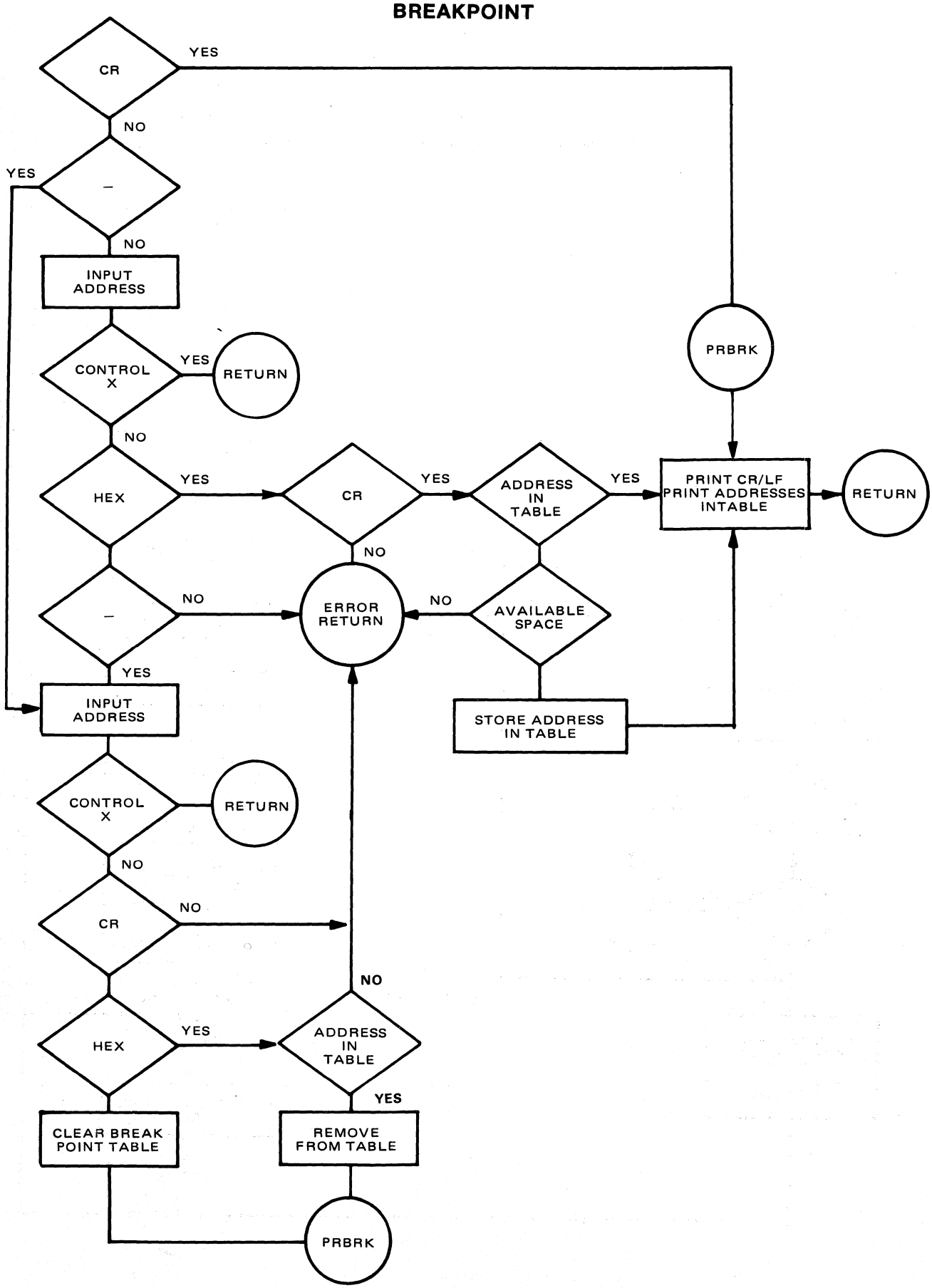
PRINT/MODIFY REGISTERS



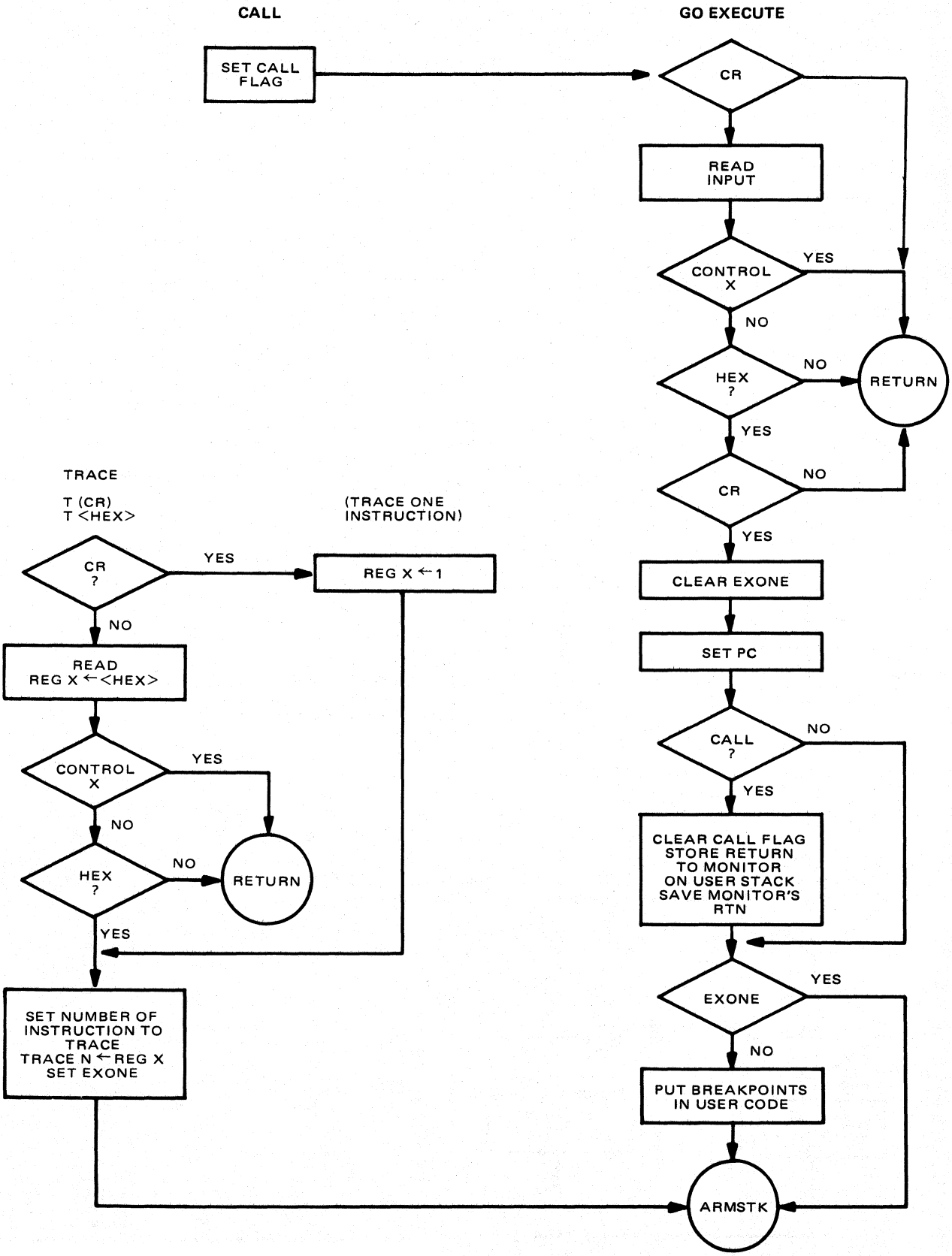
OFFSET


SUBROUTINE
RD2ADR
DELIMITER
NO
YES
READ
ADDRESS
VALID
NO
YES
DELIMITER
NO
YES
READ
ADDRESS
VALID
NO
YES
CR
NO
ERROR
YES
A:B ← <Adr 2 >
A:B ← A:B - <Adr 1> -1
(UNSIGNED SUBT)
A = 0
YES
B < 80
ERROR
YES
NO
A = FF
YES
B > 7F
YES
PRINT SPACE
PRINT REG B
NO
NO
ERROR
ENTER
MEM/EXAM
ALLOW CHANGE
< ADR 1 >

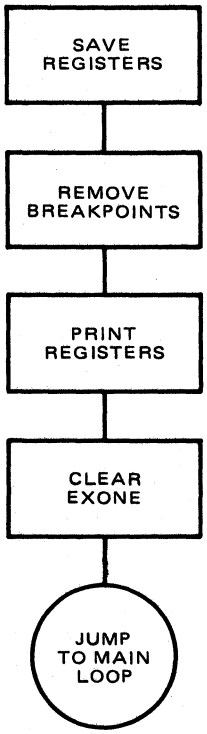
BREAKPOINT


CR
YES
NO
–
YES
NO
INPUT ADDRESS
CONTROL X
YES
RETURN
NO
HEX
YES
CR
YES
ADDRESS IN TABLE
YES
PRINT CR/LF PRINT ADDRESSES INTABLE
RETURN
PRBRK
NO
–
NO
ERROR RETURN
NO
AVAILABLE SPACE
YES
INPUT ADDRESS
STORE ADDRESS IN TABLE
CONTROL X
RETURN
NO
CR
NO
NO
HEX
YES
ADDRESS IN TABLE
YES
CLEAR BREAK POINT TABLE
REMOVE FROM TABLE
PRBRK

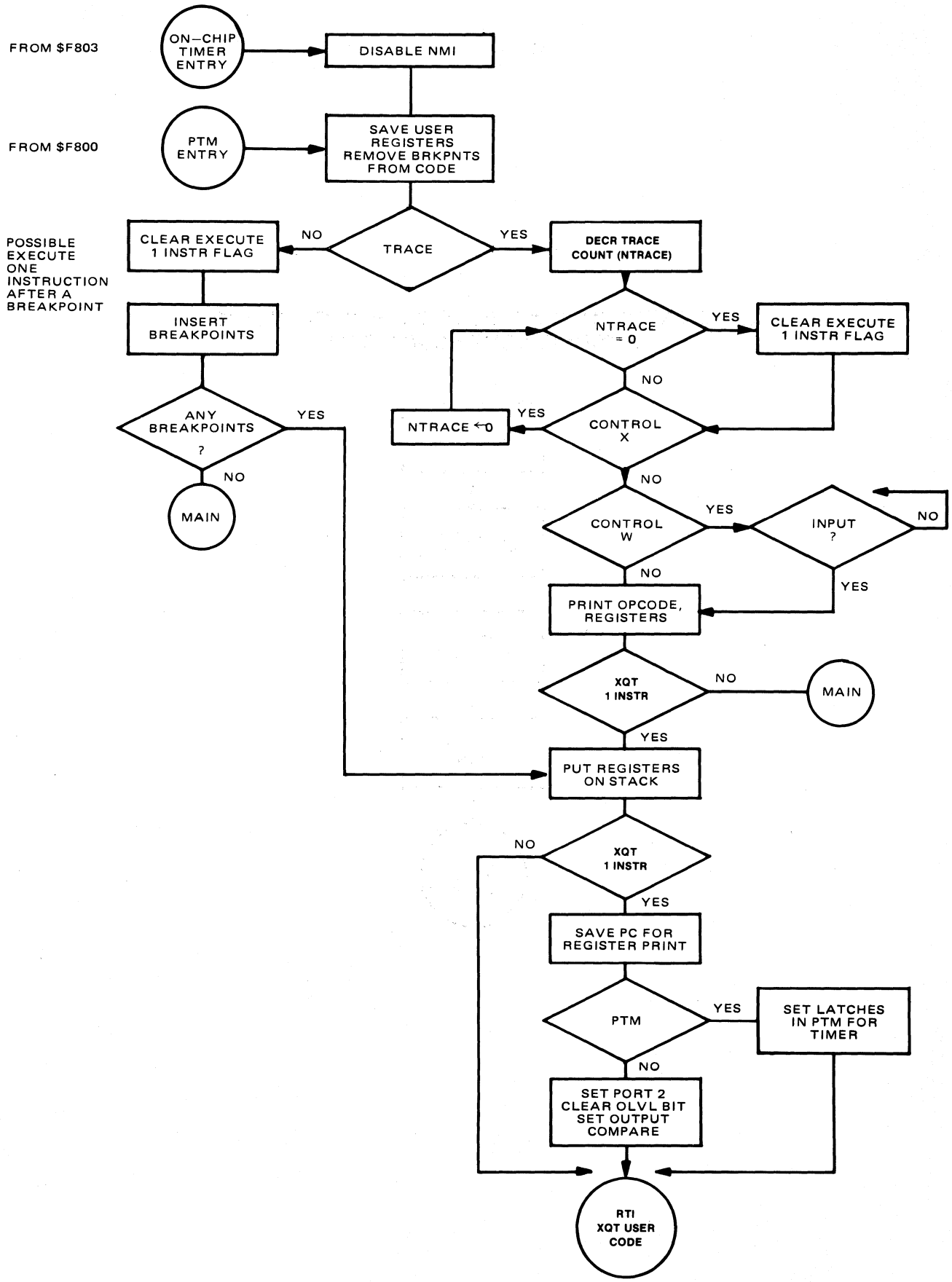
GO/CALL/TRACE



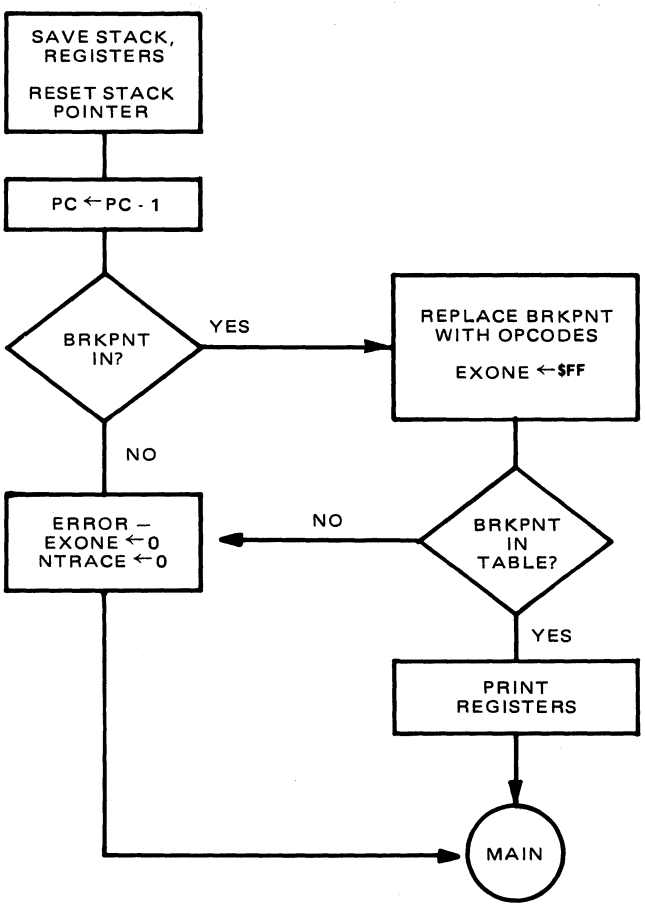
CRTS-ENTRY INTO MONITOR WITH 'C' COMMAND AFTER EXECUTE USER 'RTS'



NMI ENTRIES


FROM $F803
ON–CHIP TIMER ENTRY
DISABLE NMI
FROM $F800
PTM ENTRY
SAVE USER REGISTERS REMOVE BRKPNTS FROM CODE
POSSIBLE EXECUTE ONE INSTRUCTION AFTER A BREAKPOINT
TRACE
NO
YES
CLEAR EXECUTE 1 INSTR FLAG
DECR TRACE COUNT (NTRACE)
INSERT BREAKPOINTS
NTRACE = 0
YES
CLEAR EXECUTE 1 INSTR FLAG
NO
ANY BREAKPOINTS ?
YES
NTRACE ← 0
YES
CONTROL X
NO
NO
MAIN
CONTROL W
YES
INPUT ?
NO
NO
YES
PRINT OPCODE, REGISTERS
XQT 1 INSTR
NO
MAIN
YES
PUT REGISTERS ON STACK
NO
XQT 1 INSTR
YES
SAVE PC FOR REGISTER PRINT
PTM
YES
SET LATCHES IN PTM FOR TIMER
NO
SET PORT 2 CLEAR OLVL BIT SET OUTPUT COMPARE
RTI XQT USER CODE

SWI (BREAKPOINT PROCESSING)



CALLS SUBROUTINES –

MOVSTK - MOVE USER REGISTER FROM STACK TO RAM STORAGE IN MONITOR

SET B - SET BREAKPOINTS (3F) IN USER CODE, SAVING OP CODES.

RBRK - REMOVE BREAKPOINTS FROM USER CODE, REPLACING WITH OP CODES.

APPENDIX C
SAMPLE PROGRAMS

The following routines illustrate the monitor flexibility previously described:

1. User-defined I/O — USERIO redefines console input an output to use an ACIA instead of the on-chip serial I/O. USERIO must run in an expanded mode to enable use of external resources (i.e. ACIA).

2. User-defined vectors in an internal vector mode — OVERFLOW tests timer overflow. An interrupt routine is defined vector table is redefined.

3. User-defined commands — USRCMD adds a command which redefines the monitor's C command. The new command is also referenced as CNVRT$DEC. The command converts a decimal number to hexidecimal.

4. Alternate hardware for trace — USEPTM sets the NMI to indicateuse of a PTM. This must run in expanded mode with external vectors.

```
PAGE  001   USERIO

00001                          OPT    Z01
00002                          NAM    USERIO
00003                       ***** USER PROGRAM TO TEST MONITOR IO INDEPENDENCE
00004                       * CHANGE RESTART VECTOR TO POINT HERE
00005                       * SUBSTITUTE ACIA IO FOR MONOTOR'S SERIAL IO

00007                       * EQUATES INTO MONITOR
00008          00FC  A IOPTR  EQU    $0FC      RAM LOC-PNTR TO IO TABL
00009          F906  A MYNTRY EQU    $F906     ENTRY TO INIT REST OF MONITOR
00010          00F6  A OUTSW  EQU    $0F6      ECHO/TAPE FLAG, TAPE IF #0
00011          00D5  A CHRNL  EQU    $0D5      # NULLS FOR PADDING

00013                       ****************************************
00014                       * USER IO CODE TO INITIALIZE IO TABLE
00015A 3000                    ORG    $3000
00016A 3000 CE 300B  A         LDX    #IOTAB    GET USER IO TABLE
00017A 3003 DF FC    A         STX    IOPTR     DEFINE IO TABLE FOR ACIA
00018A 3005 CE 3068  A         LDX    #CMDTAB   REDEFINE HI & HY
00019A 3008 7E F906  A         JMP    MYNTRY    JUMP TO MONITOR

00021                       * DISPLAY COMMAND (D) USES HS ROUTINES
00022                       * MONITOR'S HS (HIGH SPEED PRINT) WRITES TO
00023                       *   CONSOLE, CALLING OUTCH
00024                       * TAPE COMMANDS USE BS (BULK STORE) ROUTINES
00025                       * BS ROUTINES CALL INCH AND OUTCH  TO READ
00026                       *   AND WRITE TO TAPE
00027                       * INCH AND OUTCH CALL THE READ AND WRITE
00028                       *   ROUTINES DEFINED HERE

00030                       * MONITOR ROUTINES USED IN IO TABLE
00031                       * THE FOLLOWING ADDRESSES WILL CHANGE IF
00032                       * THE MONITOR IS REVISED
00033          F8D2  A COON   EQU    $F8D2     TURN ON CONSOLE OUTPUT
00034          F8D1  A COOFF  EQU    $F8D1
00035          F8D1  A HSON   EQU    $F8D1     TURN ON HI SPEED DEVICE
00036          FDEC  A HSDTA  EQU    $FDEC     OUTPUT TO HI SPEED DEVICE
00037          F8D1  A HSOFF  EQU    $F8D1     TURN OFF HI SPEED DEVICE
00038          FEAF  A BSON   EQU    $FEAF     TURN ON BULK STR DEVICE
00039          FEC3  A BSDTA  EQU    $FEC3     BULK STORE IO
00040          FEBA  A BSOFF  EQU    $FEBA     TURN OFF BULK STR DEVICE

00042                       * ACIA EQUATES
00043          9CF4  A ACIAS  EQU    $9CF4     STATUS WRD
00044          9CF5  A ACIAD  EQU    ACIAS+1   DATA WRD
00045                       ****************************************
00046                       * IO TABLE
00047                       * ADDRESS OF THIS TABLE MUST BE STORED IN IOPTR
00048A 300B    3023  A IOTAB  FDB    AON,READ,AOFF
     A 300D    3030  A
     A 300F    3047  A
00049A 3011    F8D2  A         FDB    COON,PRINT,COOFF
     A 3013    3048  A
     A 3015    F8D1  A
00050A 3017    F8D1  A         FDB    HSON,HSDTA,HSOFF HI SPEED PR  ROUTINES
     A 3019    FDEC  A
     A 301B    F8D1  A
00051A 301D    FEAF  A         FDB    BSON,BSDTA,BSOFF TAPE ROUTINES
     A 301F    FEC3  A
```

```
PAGE  002   USERIO

     A 3021    FEBA  A

00053                       ****************************************
00054                       * USER IO ROUTINES
00055                       * INIT, TURN ON CONSOLE INPUT
00056                       * INITIALIZE ACIA
00057A 3023 CC 0311  A AON     LDD    #$0311    INIT ACIA
00058A 3026 B7 9CF4  A         STAA   ACIAS     MASTER RESET
00059A 3029 F7 9CF4  A         STAB   ACIAS     LEN=8, NO PARITY
00060A 302C 86 10    A         LDAA   #$10      SET PADDING FOR 300 BAUD
00061A 302E 20 45 3075         BRA    HI2

00063                       * READ 1 CHAR W/NO WAIT
00064                       * MONITOR INCH ROUTINE CHECKS C BIT IN CCR
00065                       * IF CLEAR - INCH LOOPS WAITING  FOR INPUT
00066                       * MONITOR 'CHKABT' ROUTINE NEEDS READ W/NO WAIT
00067                       *      TO CHECK FOR 'CONTROL X'
00068A 3030 B6 9CF4  A READ    LDAA   ACIAS
00069A 3033 47                 ASRA             READ BIT SET?
00070A 3034 24 04 303A         BCC    READ2
00071A 3036 B6 9CF5  A         LDAA   ACIAD     READ DATA
00072                       * SET CARRY TO INDICATE DATA READ
00073A 3039 0D                 SEC
00074A 303A 39              READ2  RTS

00076                       * WRITE 1 CHAR - NO PADDING, SAVE B
00077                       * CALLED BY PRINT
00078A 303B 37              WRITE  PSHB
00079A 303C F6 9CF4  A WRITE2 LDAB   ACIAS
00080A 303F 57                 ASRB             WRITE BIT SET?
00081A 3040 57                 ASRB
00082A 3041 24 F9 303C         BCC    WRITE2
00083A 3043 B7 9CF5  A         STAA   ACIAD     WRITE DATA
00084A 3046 33                 PULB
00085          3047  A AOFF    EQU    *
00086A 3047 39                 RTS

00088                       * MONITOR OUTCH ROUTINE CALLS PRINT
00089                       * ADD ROUTINE TO DO PADDING-ASSUME 120CPS
00090A 3048 8D F1 303B PRINT   BSR    WRITE
00091A 304A D6 F6    A         LDAB   OUTSW     NULLS FOR TAPE CR
00092A 304C 26 02 3050         BNE    P1        JMP IF TAPE
00093A 304E D6 D5    A         LDAB   CHRNL     # NULLS FOR CR
00094A 3050 81 0D    A P1      CMPA   #$D       CR?
00095A 3052 27 08 305C         BEQ    P2        JMP IF CR
00096A 3054 81 10    A         CMPA   #$10      DO NOT PAD IF DEL
00097A 3056 27 E2 303A         BEQ    READ2
00098A 3058 C4 03    A         ANDB   #$3       MASK OUT HI 6 BITS
00099A 305A 20 02 305E         BRA    P4
00100A 305C 54              P2     LSRB             REMOVE LOW 2-BIT CNT
00101A 305D 54                 LSRB
00102                       * NO PADDING PER CHAR NEEDED FOR LOWER SPEEDS
00103A 305E 5A              P4     DECB
00104A 305F 2B D9 303A         BMI    READ2     ENOUGH NULLS?
00105A 3061 36                 PSHA
00106A 3062 4F                 CLRA             SET A=NULL
00107A 3063 8D D6 303B         BSR    WRITE
00108A 3065 32                 PULA
```

```
PAGE  003   USERIO

00109A 3066 20 F6 305E          BRA     P4

00111                   ********** USER DEF **********
00112                   * REDEFINE HI AND HY CMD FOR ACIA
00113                   * THE FOLLOWING TABLE IS USED BY THE MONITOR
00114A 3068    05     A CMDTAB FCB     5          # BYTES IN ENTRY
00115A 3069    48     A        FCC     /HI/       CMD NAME (2 BYTES)
     A 306A    49     A
00116A 306B    3073   A        FDB     HI         ADDRESS (2 BYTES)
00117A 306D    05     A        FCB     5          #BYTES IN ENTRY
00118A 306E    48     A        FCC     /HY/       CMD NAME
     A 306F    59     A
00119A 3070    3078   A        FDB     HY         ADDRESS
00120A 3072    FF     A        FCB     -1         END OF TABLE

00122                   * HI & HY SET CHRNL FOR PADDING
00123                   * LOW 2 BITS = # NULLS AFTER CHAR
00124                   * HI 6 BITS = # NULLS AFTER CR

00126                   ********** HI **********
00127                   * SET SPEED FOR 120 CPS
00128                   * SET # NULLS TO PAD CHAR/CR
00129A 3073 86 4F     A HI     LDAA    #$4F
00130A 3075 97 D5     A HI2    STAA    CHRNL
00131A 3077 39                 RTS

00133                   ********** HY **********
00134                   * HIGHER YET - 9600 BAUD FOR CRT
00135                   * SET PADDING TO ZERO
00136A 3078 4F          HY      CLRA
00137A 3079 20 FA 3075          BRA     HI2
00138                           END
TOTAL ERRORS 00000


ACIAD  9CF5  ACIAS  9CF4  AOFF   3047  AON    3023  BSDTA  FEC3  BSOFF  FEBA
BSON   FEAF  CHRNL  00D5  CMDTAB 3068  COOFF  F8D1  COON   F8D2  HI     3073
HI2    3075  HSDTA  FDEC  HSOFF  F8D1  HSON   F8D1  HY     3078  IOPTR  00FC
IOTAB  300B  MYNTRY F906  OUTSW  00F6  P1     3050  P2     305C  P4     305E
PRINT  3048  READ   3030  READ2  303A  WRITE  303B  WRITE2 303C
```



```
PAGE  001   OVFLOW

00001                           NAM     OVFLOW
00002                           OPT     Z01,LLEN=80

00004                   ***** EXAMPLE OF TIMER OVERFLOW INTERRUPT *****
00005                   * REDEFINE VECTOR TABLE USED BY LILBUG
00006                   * THIS VECTOR TABLE IS USED IF THE MC6801 MODE
00007                   *   SELECTED HAS INTERNAL INTERRUPT VECTORS.
00008                   * DEFINE OVERFLOW INTERRUPT, SWI, AND NMI.
00009                   *   THE OTHER INTERRUPTS ARE SAME AS NMI.
00010                   * SET LILBUG VECPTR TO POINT TO VECTOR TABLE
00011                   * ENABLE TIMER OVERFLOW AND INTERRUPT MASK
00012                   * PRINT MESSAGE WITH EACH OVERFLOW INTERRUPT

00014                   * EQUATES INTO MONITOR
00015           F903  A MAIN    EQU     $F903      MONITOR ENTRY
00016           F80F  A PDATA   EQU     $F80F      PR DATA STRING
00017           0008  A TCSR    EQU     $08        CLOCK CONTROL REG
00018           00FE  A VECPTR  EQU     $0FE       PNTR TO VECTOR TABLE

00020                   * MONITOR COMES UP
00021                   * THIS PROGRAM IS LOADED
00022                   * AFTER EACH RESTART, EXECUTE STARTING AT 'BEGIN'
00023                   *    TO SET OVERFLOW INTERRUPT VECTOR

00025A 0080                     ORG     $80
00026A 0080 8E 00C0   A BEGIN   LDS     #$C0       SET STACK POINTER
00027A 0083 96 08     A         LDAA    TCSR       SET ETOI
00028A 0085 8A 04     A         ORAA    #$04
00029A 0087 97 08     A         STAA    TCSR
00030A 0089 CE 009F   A         LDX     #TABLE     SET VECTOR TABLE PNTR
00031A 008C DF FE     A         STX     VECPTR
00032A 008E 0E                  CLI                ALLOW INTERRUPT
00033A 008F 20 FE 008F          BRA     *          WAIT FOR INTERRUPT

00035                   ***** INTERRUPT ROUTINE *****
00036                   * WHEN TIMER GOES FROM $FF TO 0, AN INTERRUPT
00037                   *   OCCURS AND VECTORS TO 'OVFLOW' TO PRINT MSG
00038A 0091 0F          OVFLOW SEI              PREVENT INTERRUPT UNTIL END MSG
00039A 0092 CE 009A   A         LDX     #MSG       GET ADR
00040A 0095 BD F80F   A         JSR     PDATA      GO PRINT
00041A 0098 0E                  CLI                ALLOW INTERRUPT
00042A 0099 3B                  RTI                RETURN

00044A 009A    4F     A MSG     FCC     /OVER/
00045A 009E    04     A         FCB     4

00047                   ****** VECTOR TABLE
00048                   * DEFINE OVERFLOW INTERRUPT, SWI AND NMI
00049                   * VECPTR (ADDRESS $0FE) MUST POINT TO THIS TABLE
00050A 009F    00AB   A TABLE   FDB     NMI        TAKE NMI IF VECTOR HERE
00051A 00A1    0091   A         FDB     OVFLOW     TIMER OVERFLOW
00052A 00A3    00AB   A         FDB     NMI
00053A 00A5    00AB   A         FDB     NMI
00054A 00A7    00AB   A         FDB     NMI
00055A 00A9    F821   A         FDB     $F821      SWI ENTRY - LILBUG BRKPNT
00056A 00AB    F803   A NMI     FDB     $F803      NMI - LILBUG TRACE

00058           0080   A         END     BEGIN
```

```
PAGE  001   USRCMD

00001                          NAM    USRCMD
00002                          OPT    Z01

00004                     ***** EXAMPLE OF USER DEFINED COMMAND
00005                     * EXTERNAL VECTORS SHOULD BE USED
00006                     * RESET VECTOR NEEDS TO BE SET TO COME HERE
00007                     *   SO THAT RESTART WILL RESET USER CMD TABLE
00008                     * ADDRESS OF LABEL 'START' IS THE RESET VECTOR

00010                     * FCTPTR POINTS TO COMMAND TABLE
00011                     * COMMAND TABLE ENTRY CONSISTS OF:
00012                     *   1. NUMBER BYTES IN ENTRY
00013                     *   2. ASCII CHARACTERS FORMING COMMAND NAME
00014                     *   3. ADDRESS OF COMMAND ROUTINE
00015                     * TABLE IS TERMINATED BY:
00016                     *   -1 INDICATING MONITOR TABLE IS ALSO TO BE SEARCHED
00017                     *   OR -2 INDICATING MONITOR COMMANDS ARE NOT USED

00019                     * NOTE: COMMAND LENGTH IS VARIABLE.
00020                     *       DECIMAL NUMBERS ARE NOT VALID IN COMMAND NAME.
00021                     *       . AND / ARE SPECIAL QUICK COMMANDS-CANNOT
00022                     *         BE REDEFINED.
00023                     *       A COMMAND IS TERMINATED BY SPACE, COMMA,
00024                     *          CR, OR LF.



00026                     * ENTRY INTO MONITOR
00027          F90D  A ENTRY  EQU    $F90D    ENTRY TO RESET INITIALIZATION
00028                     * IO TABLE
00029          F85B  A CI     EQU    $F85B    USE SAME IO AS MONITOR

00031                     * MONITOR RAM LOCATION
00032          00FA  A FCTPTR EQU    $0FA     PNTR TO COMMAND TABLE
00033          00FC  A IOPTR  EQU    $0FC     PNTR TO TABLE OF IO ADDRESSES

00035                     * MONITOR ROUTINES USED
00036          F806  A INCH   EQU    $F806    INPUT 1 CHAR INTO REG A
00037          F809  A OUTCH  EQU    $F809    OUTPUT 1 CHAR FROM REG A
00038          F80C  A PDATA1 EQU    $F80C    PR DATA STRG (ADR IN REG X)
00039          F815  A OUT4HS EQU    $F815    PRINT 4 HEX CHAR & SPACE

00041A 0080                    ORG    $80
00042                     * RAM FOR USER COMMANDS
00043A 0080    0002    A NUMBER RMB    2        HEX NUMBER
00044A 0082    0001    A TEMP   RMB    1        TEMPORARY STORAGE

00046A 6500                    ORG    $6500
00047                     * USE EXPANDED MODE W/MEMORY AT $6500
00048                     * SET THE COMMAND TABLE PNTR
00049                     * JUMP TO MONITOR TO CONTINUE INITIALIZATION
00050                     * BE SURE MONITOR DOES NOT RESET FCTPTR
00051A 6500 CE 650D  A START  LDX    #TABLE   GET ADR
00052A 6503 DF FA    A        STX    FCTPTR
00053A 6505 CE F85B  A        LDX    #CI      ALSO SET IO PNTR
00054A 6508 DF FC    A        STX    IOPTR    NORMALLY DONE BY MONITOR
00055                     * VECTOR TABLE NOT NEEDED IF USING EXTERNAL VECTORS
00056                     *  SO DO NOT NEED TO SET VECPTR
```

```
PAGE  002   USRCMD

00057A 650A 7E F90D  A        JMP    ENTRY

00059                     ***** USER COMMAND TABLE
00060                     * THIS C COMMAND REPLACES THE MONITOR'S C CMD
00061A 650D    04    A TABLE  FCB    4        * BYTES IN ENTRY
00062A 650E    43    A        FCC    /C/      COMMAND NAME - CONVERT DEC
00063A 650F    651E  A        FDB    CONV     ADR OF PROCESSING ROUTINE
00064                     * BEGINNING OF 2ND ENTRY
00065A 6511    0C    A        FCB    12       * BYTES
00066A 6512    43    A        FCC    ./CNVRT$DEC/ANOTHER NAME FOR C CMD
00067A 651B    651E  A        FDB    CONV     ADR
00068A 651D    FF    A        FCB    -1       TERMINATOR - MONITOR COMMA

00070                     ****************************************
00071                     ***** USER COMMAND-CONVERT DECIMAL TO HEX
00072A 651E CE 0080  A CONV   LDX    #NUMBER
00073A 6521 6F 00    A        CLR    0,X      INIT
00074A 6523 6F 01    A        CLR    1,X
00075A 6525 BD F806  A CONV1  JSR    INCH     READ A CHAR
00076A 6528 81 3D    A        CMPA   #'=      = TERMINATES NUMBER
00077A 652A 27 24 6550        BEQ    PRINT    GO PRINT HEX NUMBER
00078A 652C 81 30    A        CMPA   #'0      DECIMAL CHECK
00079A 652E 2B 04 6534        BMI    ERROR    NOT DEC
00080A 6530 81 39    A        CMPA   #'9
00081A 6532 2F 03 6537        BLE    CONV2    GOOD DEC

00083                     * ERROR - SET NEG CONDITION CODE, RETURN
00084                     *         MONITOR WILL PRINT ?.
00085A 6534 86 FF    A ERROR  LDAA   #$FF
00086A 6536 39                RTS

00088A 6537 84 0F    A CONV2  ANDA   #$F      CONVERT ASCII TO DEC
00089A 6539 97 82    A        STAA   TEMP
00090A 653B EC 00    A        LDD    0,X      GET PREVIOUS NUMBERS
00091A 653D 04                ASLD            DOUBLE LEFT SHIFT
00092A 653E 37                PSHB            SAVE DIV BY 2
00093A 653F 36                PSHA
00094A 6540 04                ASLD
00095A 6541 04                ASLD
00096A 6542 ED 00    A        STD    0,X      SAVE DIV BY 8
00097A 6544 32                PULA
00098A 6545 33                PULB
00099A 6546 E3 00    A        ADDD   0,X      CONVERSION
00100A 6548 DB 82    A        ADDB   TEMP     ADD IN LATEST NUM
00101A 654A 89 00    A        ADCA   #0
00102A 654C ED 00    A        STD    0,X
00103A 654E 20 D5 6525        BRA    CONV1    GET NXT NUM

00105                     * PRINT 4 HEX NUMBERS & SPACE
00106                     * REG X POINT TO FIRST BYTE TO PR
00107A 6550 CE 0080  A PRINT  LDX    #NUMBER
00108A 6553 BD F815  A        JSR    OUT4HS
00109A 6556 4F                CLRA            CLR CCR - PREVENT ERR RETU
00110A 6557 39                RTS
00111                     ****************************************


00113                     * SET EXTERNAL VECTORS
```

```
PAGE  003   USRCMD

00114                    * SET SWI AND NMI FOR TRACE, BRKPNT
00115                    * USE ON-CHIP RESOURCES
00116                    * SET RESTART TO COME HERE TO SET TABLE PNTR
00117A FFFA                        ORG     $FFFA
00118A FFFA    FD53  A             FDB     $FD53     SWI VECTOR
00119A FFFC    F803  A             FDB     $F803     NMI VECTOR
00120A FFFE    6500  A             FDB     START
00121          6500  A             END     START
TOTAL ERRORS 00000


   F85B CI      00029*00053
   651E CONV    00063 00067 00072*
   6525 CONV1   00075*00103
   6537 CONV2   00081 00088*
   F90D ENTRY   00027*00057
   6534 ERROR   00079 00085*
   00FA FCTPTR  00032*00052
   F809 INCH    00036*00075
   00FC IOPTR   00033*00054
   0080 NUMBER  00043*00072 00107
   F815 OUT4HS  00035*00105
   F803 OUTCH   00037*
   F80C PDATA1  00038*
   6550 PRINT   00077 00107*
   6500 START   00051*00120 00121
   5E0D TABLE   00051 00061*
   0082 TEMP    00044*00089 00100
```



```
PAGE  001   USEPTM

00001                              OPT     Z01
00002                              NAM     USEPTM
00003                    * USE PTM INSTEAD OF ON-CHIP COUNTER
00004                    *     FOR HARDWARE TRACE
00005                    * THE MONITOR WILL INITIALIZE AND USE
00006                    * RESOURCES FOR TRACE ACCORDING TO THE
00007                    * ADDRESS IN THE NMI VECTOR
00008                    *     $F800 - PTM ADDRESS
00009                    *     $F803 - ADDRESS FOR ON-CHIP CLK/CNTR
00010                    * IF EXTERNAL VECTORS ARE USED, THE MONITOR
00011                    *     GETS THE NMI VECTOR DIRECTLY.
00012                    * IF INTERNAL VECTORS ARE USED, THE NMI
00013                    *     VECTOR IS RETRIEVED INDIRECTLY FROM
00014                    *     THE TABLE POINTED TO BY 'VECPTR'.

00016                    * MONITOR EQUATES
00017          F821  A SWI     EQU     $F821     SWI ENTRY
00018          F90D  A MONITR  EQU     $F90D     INIT MONITOR
00019          F800  A EXNMI   EQU     $F800     MONITOR ENTRY FOR PTM NMI
00020          CF00  A PTMADR  EQU     $CF00     ADR FOR PTM
00021          F85B  A CI      EQU     $F85B     IO TABLE
00022          F824  A FCTABL  EQU     $F824

00024                    * RAM USED BY MONITOR
00025A 00F8                        ORG     $0F8
00026A 00F8    0002  A PTM     RMB     2         PNTR TO PTM ADR
00027A 00FA    0002  A FCTPTR  RMB     2         PNTR TO CMD TABLE
00028A 00FC    0002  A IOPTR   RMB     2         PNTR TO IO TABL

00030                    ****************************************
00031                    * INITIALIZE TABLES USED BY MONITOR
00032                    * THEN JUMP TO MONITOR
00033                    *       SUCH THAT MONITOR DOES NOT
00034                    *       RESET PTM ADDRESS
00035A 0080                        ORG     $80
00036A 0080 CE CF00  A START  LDX     #PTMADR   RESET PTM ADR
00037A 0083 DF F8    A        STX     PTM       CHANGE MONITOR DEFAULT
00038A 0085 CE F85B  A        LDX     #CI       SET IO TABLE
00039A 0088 DF FC    A        STX     IOPTR
00040A 008A CE F824  A        LDX     #FCTABL   SET COMMAND TABL
00041A 008D DF FA    A        STX     FCTPTR
00042A 008F 7E F90D  A        JMP     MONITR

00044                    ****************************************
00045                    * EXTERNAL INTERRUPT VECTORS
00046A FFFA                        ORG     $FFFA
00047A FFFA    F821  A             FDB     SWI       BRKPNT ENTRY INTO MONITOR
00048A FFFC    F800  A             FDB     EXNMI     NMI ENTRY FOR PTM
00049A FFFE    0080  A             FDB     START     INIT FROM USER PROG
00050          0080  A             END     START
TOTAL ERRORS 00000


CI      F85B  EXNMI   F800  FCTABL F824  FCTPTR 00FA  IOPTR  00FC
MONITR F90D  PTM     00F8  PTMADR CF00  START  0080  SWI    F821
```

The following programs test various features of the M6801:

Program Description

MODE 4 Test the single chip mode 4 in low speed — 300 baud. Serial I/O rutine is defined. Interrupt by IS3 is enabled. Port 1 is tied to Port 3. Write Port 1 causes IS3 and IRQ. The interrupt routine writes one character of a message. The interrupt vectors use $F0-$FF.

PORTS Cycle through Ports 1, 3, and 4. Read port and print hexadecimal value. Use LILbug print routine.

TIMER Test on-chip timer. Generate square wave and watch on scope. This can be run in single chip mode.

TOGGLE Cycle through all ports, writing to each and watching on scope. This does not use LILbug. The test can be run in single chip mode. LILbug is not used.

TSTIO TSTIO is a basic on-chip serial I/O test that reads the keyboard and echoes to terminal. It sets NRZ and baud rate of 300.

TSTPTM This program tests the PTM which can be used as alternative hardware for LILbug monitor trace. It sets latches to zero and brings the PTM out of RESET. Then latches are set and the program loops waiting for NMI interrupt. NMI is tied to the inverted output of Timer 1. The NMI routine outputs a counter indicating when the NMI occurred. LILbug I/O routines are used.

```
PAGE  001   MODE4

00001                    * TEST FOR MODE 4
00002                          NAM    MODE4
00003                          OPT    Z01

00005                    * SINGLE CHIP TEST MODE
00006                    * WITH INTERNAL ROM MONITOR DISABLED
00007                    * CAN CHANGE TO MODE 5 IN SOFTWARE
00008                    *    BY WRITING 1 INTO PC0 BIT OF IO PORT 2.
00009                    * DEFINE RESET VECTOR
00010          FFFE  A RESET  EQU    $FFFE     USED TO GO TO MONITOR

00012                    * DEFINE PORT 2
00013          0003  A PORT2  EQU    $03

00015                    * DEFINE PORT 3
00016          0006  A PORT3  EQU    $06       PORT 3
00017          0004  A DDR3   EQU    $04       DATA DIRECTION
00018          000F  A CR3    EQU    $0F       CONTROL REG

00020                    * DEFINE PORT 1
00021          0002  A PORT1  EQU    $02
00022          0000  A DDR1   EQU    $00
00023                    * SERIAL IO EQUATES
00024          0010  A RMCR   EQU    $10       RATE & MODE CONTROL
00025          0011  A TRCS   EQU    $11       TRANS/RECEIV CNTRL/STAT
00026          0013  A TX     EQU    $13       TRANS REG


00028                    ********************************
00029A 0080                    ORG    $80
00030                    * SET IS3 TO CAUSE IRQ
00031                    * PORT1 IS TIED TO PORT3
00032                    * WRITE TO PORT1 TO CAUSE INPUT INTERRUPT W/PORT3
00033                    * SET SPEED = 30 CPS, INIT IO
00034A 0080 8E 00F7  A START  LDS    #STACK
00035A 0083 CC 0702  A        LDD    #$0702
00036A 0086 DD 10    A        STD    RMCR      SET BAUD RATE BITS, SET TE
00037                    * INIT PORT 3
00038A 0088 CC 4000  A        LDD    #$4000
00039A 008B 97 0F    A        STAA   CR3       ENABLE INTERRUPT
00040A 008D D7 04    A        STAB   DDR3      SET FOR INPUT
00041A 008F 86 FF    A        LDAA   #$FF
00042A 0091 97 02    A        STAA   PORT1
00043A 0093 97 00    A        STAA   DDR1      SET FOR OUTPUT
00044A 0095 7F 00E1  A        CLR    OFFSET
00045A 0098 0E                 CLI              ALLOW IRQ
00046                    * MAIN LOOP
00047A 0099 86 0A    A LOOP   LDAA   #10       SET LOOP CNTR
00048A 009B 7F 0002  A LOOP2  CLR    PORT1     GO THRU LOOP 10 TIMES
00049A 009E 4A                 DECA
00050A 009F 26 FA 009B         BNE    LOOP2
00051A 00A1 B6 00E2  A        LDAA   MODCHG    TRY TO CHANGE MODES
00052A 00A4 97 03    A        STAA   PORT3
00053A 00A6 FE FFFE  A        LDX    RESET     GET RESTART VECTOR
00054A 00A9 6E 00    A        JMP    0,X       GO TO MONITOR
00055                    ********************************

00057                    * WRITE IRQ MESSAGE WITH SERIAL TRANSMITTER
```

```
PAGE  002   MODE4

00058                    * OUTPUT CHAR - NO PADDING
00059A 00AB 37           OUTCH  PSHB
00060A 00AC D6 11    A OUTCH2 LDAB   TRCS      GET CNTRL WRD
00061A 00AE C5 20    A        BITB   #$20       TDRE SET?
00062A 00B0 27 FA 00AC        BEQ    OUTCH2    WAIT UNTIL IT IS
00063A 00B2 97 13    A        STAA   TX        WRITE WRD
00064A 00B4 33                 PULB
00065A 00B5 39                 RTS

00067                    * IRQ
00068                    * PRINT 1 CHAR OF MESSAGE W/EACH INTERRUPT
00069A 00B6 96 0F    A IRQ    LDAA   CR3       CLEAR INTERRUPT
00070A 00B8 96 06    A        LDAA   PORT3
00071A 00BA 8A FF    A        LDAA   #$FF
00072A 00BC 97 02    A        STAA   PORT1
00073A 00BE F6 00E1  A        LDAB   OFFSET    GET MSG OFFSET
00074A 00C1 CE 00D4  A        LDX    #MSG
00075A 00C4 3A                 ABX              GET NEXT CHAR OF MSG
00076A 00C5 A6 00    A        LDAA   0,X       GET CHAR
00077A 00C7 8D E2 00AB        BSR    OUTCH     PRINT
00078A 00C9 5C                 INCB
00079A 00CA 8C 00E1  A        CPX    #ENDMSG   END OF MSG?
00080A 00CD 26 01 00D0         BNE    IRQ2
00081A 00CF 5F                 CLRB
00082A 00D0 F7 00E1  A IRQ2   STAB   OFFSET
00083A 00D3 3B                 RTI

00085A 00D4    53    A MSG    FCC    /SUMMER WINE/
00086A 00DF    0D    A        FCB    $D,$A
00087          00E1  A ENDMSG EQU    *
00088A 00E1    0001  A OFFSET RMB    1
00089A 00E2    E0    A MODCHG FCB    $E0       TRY MODE CHANGE

00091                    * INTERRUPT VECTORS
00092A 00F8                    ORG    $F8
00093          00F7  A STACK  EQU    *-1       BETTER NOT USE ANY OTHER I
00094A 00F8    00B6  A        FDB    IRQ
00095A 00FA    0004  A        RMB    4         SWI,NMI
00096A 00FE    0080  A        FDB    START
00097                          END
TOTAL ERRORS 00000


CR3     000F  DDR1    0000  DDR3    0004  ENDMSG 00E1  IRQ     00B6
IRQ2    00D0  LOOP    0099  LOOP2   009B  MODCHG 00E2  MSG     00D4
OFFSET  00E1  OUTCH   00AB  OUTCH2  00AC  PORT1  0002  PORT2   0003
PORT3   0006  RESET   FFFE  RMCR    0010  STACK  00F7  START   0080
TRCS    0011  TX      0013
```

```
PAGE  001   PORTS

00001                           NAM    PORTS
00002                           OPT    Z01
00003                     * CYCLE AROUND PORTS AND READ & PRINT
00004                     * DO NOT CHANGE PORT 2 - USED FOR SERIAL IO

00006                     * PORT EQUATES
00007           0000  A DDR1    EQU    $00       DATA DIRECTION
00008           0004  A DDR3    EQU    $04
00009           0002  A P1      EQU    $02       PORT1 DATA
00010           0006  A P3      EQU    $06       PORT3 DATA
00011           0007  A P4      EQU    $07       PORT4 DATA

00013                     * MONITOR EQUATE
00014           F812  A OUT2HS  EQU    $F812     PR 2 HEX & SP
00015           F818  A PCRLF   EQU    $F818     CR/LF

00017A 0080                     ORG    $80

00019                     * INIT PORT 1,3,4
00020A 0080 CC 0000   A INIT    LDD    #0        SET FOR INPUT
00021A 0083 97 00     A         STAA   DDR1
00022A 0085 DD 04     A         STD    DDR3

00024                     * MAIN LOOP
00025A 0087 CE 0002   A LOOP    LDX    #P1       GET PORT 1 ADR
00026A 008A BD F812   A         JSR    OUT2HS    GO PR
00027A 008D CE 0006   A         LDX    #P3       GET PORT 3 ADR
00028A 0090 BD F812   A         JSR    OUT2HS    GO PR
00029A 0093 08                  INX              SET TO PORT 4 ADR
00030A 0094 BD F812   A         JSR    OUT2HS
00031A 0097 BD F818   A         JSR    PCRLF
00032A 009A 20 EB 0087          BRA    LOOP

00034           0080  A         END    INIT
TOTAL ERRORS 00000


   0000 DDR1    00007*00021
   0004 DDR3    00008*00022
   0080 INIT    00020*00034
   0087 LOOP    00025*00032
   F812 OUT2HS  00014*00026 00028 00030
   0002 P1      00009*00025
   0006 P3      00010*00027
   0007 P4      00011*
   F818 PCRLF   00015*00031
```



```
PAGE  001   TIMER

00001                           NAM    TIMER
00002                           OPT    Z01
00003                     * GENERATE WAVE COMPARABLE TO WHAT IS NEEDED
00004                     *      FOR LINE TIED TO NMI FOR LILBUG TRACE
00005                     * WATCH ON SCOPE
00006           0001  A P2DDR   EQU    $01
00007           0009  A CLOCK   EQU    $09
00008           0008  A TCSR    EQU    $08       TIMER CONTROL STATUS REG
00009           000B  A OCREG   EQU    $0B       OUTPUT COMPARE REG

00011                     * LILBUG EQUATE
00012           F815  A OUT4HS  EQU    $F815

00014A 0080                     ORG    $80
00015                     * INIT DDR FOR PORT2
00016A 0080 86 02     A         LDAA   #2
00017A 0082 97 01     A         STAA   P2DDR     SET BIT 1 FOR OUTPUT
00018                     * SET FOR OUTPUT COMPARE PULSE, SET LEVEL
00019A 0084 D6 08     A PULSE   LDAB   TCSR      GET CONTROL/STAT REG
00020A 0086 C4 FE     A         ANDB   #$FE      CLEAR OLVL BIT
00021A 0088 D7 08     A         STAB   TCSR      RESET
00022A 008A 8D 07 0093          BSR    SETCLK    SET CMPR REG, WAIT FOR CMP
00023A 008C 7C 0008   A         INC    TCSR      MAKE LEVEL HI
00024A 008F 8D 02 0093          BSR    SETCLK    ALLOW LEVEL TO GO HIGH
00025A 0091 20 F1 0084          BRA    PULSE     DO IT AGAIN

00027                     ***** SETCLK *****
00028                     * READ CLOCK
00029                     * SET COMPARE REG = CLOCK + 20
00030                     * WAIT FOR PULSE
00031                     * LOOP W/ A INCR WITH EACH LOOP
00032A 0093 C6 20     A SETCLK LDAB   #$20
00033A 0095 DE 09     A         LDX    CLOCK     GET CLOCK TIME
00034A 0097 3A                  ABX              ADD #CYCLES
00035A 0098 DF 0B     A         STX    OCREG     STORE IN COMPARE REG
00036                     * PRINT COUNTER - INDICATE WHEN OCF CLEARED
00037A 009A CE 0000   A         LDX    #0        # TIMES THRU LOOP
00038A 009D D6 08     A NOTSET  LDAB   TCSR      GET STATUS
00039A 009F 58                  ASLB             LOOK AT OCF
00040A 00A0 08                  INX              TIMES THRU LOOP
00041A 00A1 2A FA 009D          BPL    NOTSET    WAIT UNTIL OCF SET
00042A 00A3 FF 00AD   A         STX    TEMP      SAVE # TIMES THRU LOOP
00043A 00A6 CE 00AD   A         LDX    #TEMP     SET ADR - GO PR
00044A 00A9 BD F815   A         JSR    OUT4HS
00045A 00AC 39                  RTS

00047A 00AD    0002   A TEMP    RMB    2
00048                           END
TOTAL ERRORS 00000


CLOCK  0009  NOTSET 009D  OCREG  000B  OUT4HS F815  P2DDR  0001
PULSE  0084  SETCLK 0093  TCSR   0008  TEMP   00AD
```

```
PAGE  001   TOGGLE

00001                           NAM    TOGGLE
00002                           OPT    Z01
00003                     * CYCLE AROUND PORTS AND WRITE TO THEM

00005                     * PORT EQUATES
00006           0000  A DDR1    EQU    $00       DATA DIRECTION
00007           0004  A DDR3    EQU    $04
00008           0002  A P1      EQU    $02       PORT1 DATA
00009           0003  A P2      EQU    $03       PORT 2 DATA
00010           0006  A P3      EQU    $06       PORT3 DATA
00011           0007  A P4      EQU    $07       PORT4 DATA

0013A 0080                      ORG    $80
00014                     * INIT PORTS 1,3,4 FOR OUTPUT
0015A 0080 8E 00AD  A INIT    LDS    #ENDPGM+10
0016A 0083 CC FFFF  A         LDD    ##FFFF    SET FOR OUTPUT
0017A 0086 DD 00 *  A         STD    DDR1      PORTS 1, 2
0018A 0088 DD 04    A         STD    DDR3      PORTS 3 & 4

00020                     * MAIN LOOP
00021                     * OUTPUT $FF TO $00 TO PORTS 1,2,3 & 4
00022                     * WATCH ON SCOPE
0023A 008A 4A             LOOP   DECA              HI TO LOW
0024A 008B 97 02    A             STAA   P1        WRITE P1
0025A 008D 8D 0E 009D             BSR    DELAY     PAUSE
0026A 008F 97 03    A             STAA   P2        WRITE PORT 2
0027A 0091 8D 0A 009D             BSR    DELAY
0028A 0093 97 06    A             STAA   P3        WRITE PORT 3
0029A 0095 8D 06 009D             BSR    DELAY
0030A 0097 97 07    A             STAA   P4        WRITE PORT4
0031A 0099 8D 02 009D             BSR    DELAY
0032A 009B 20 ED 008A             BRA    LOOP

00034                     * DELAY ROUTINE
0035A 009D C6 80    A DELAY   LDAB   #$80
0036A 009F 5A             DEL2   DECB
0037A 00A0 26 FD 009F             BNE    DEL2
0038A 00A2 39                     RTS

00040           00A3  A ENDPGM EQU    *
00041           0080  A        END    INIT
TOTAL ERRORS 00000


DDR1    0000  DDR3    0004  DEL2    009F  DELAY   009D  ENDPGM 00A3
INIT    0080  LOOP    008A  P1      0002  P2      0003  P3     0006
P4      0007
```

```
PAGE  001   TSTIO

00001                           OPT    Z01
00002                           NAM    TSTIO
00003                     * ON-CHIP IO EQUATES
00004           0010  A RMCR    EQU    $010      RATE & MODE CONTROL
00005           0011  A TRCS    EQU    $011      TRANS/REC CNTRL STAT
00006           0012  A RECEV   EQU    $012      READ REG
00007           0013  A TRANS   EQU    $013      TRANSMIT REG
00008                     ******
00009           00A0  A STACK   EQU    $0A0
00010                     *
0011A 00C0                      ORG    $0C0
0012A 00C0 8E 00A0  A START   LDS    #STACK
00013                     * INIT SERIAL IO - SET RMCR, TRCS
0014A 00C3 CC 070A  A         LDD    #$070A    NRZ, SERIAL
0015A 00C6 DD 10    A         STD    RMCR      BAUD = 30
00016                     *
00017                     * TEST - LOOP ON READING INPUT FROM
00018                     *        KEYBOARD AND PRINTING
0019A 00C8 8D 02 00CC LOOP    BSR    INCH
0020A 00CA 20 FC 00C8         BRA    LOOP

00022                     * INPUT ROUTINE
0023A 00CC 96 11    A INCH    LDAA   TRCS      GET CNTRL WRD
0024A 00CE 48             ASLA                CHK RDRF SET
0025A 00CF 24 FB 00CC             BCC    INCH      LOOP UNTIL IT IS
0026A 00D1 96 12    A             LDAA   RECEV     READ
0027A 00D3 84 7F    A             ANDA   #$7F      CLEAR PARITY
0028A 00D5 81 7F    A             CMPA   #$7F      RUBOUT?
0029A 00D7 27 F3 00CC             BEQ    INCH
0030A 00D9 8D 01 00DC             BSR    OUTCH     ECHO PRINT
0031A 00DB 39                     RTS

00033                     * OUTPUT ROUTINE
0034A 00DC D6 11    A OUTCH   LDAB   TRCS      GET CNTRL WRD
0035A 00DE C5 20    A         BITB   #$20      TDRE SET?
0036A 00E0 27 FA 00DC             BEQ    OUTCH
0037A 00E2 97 13    A             STAA   TRANS
0038A 00E4 39                     RTS
00039           00C0  A         END    START
TOTAL ERRORS 00000


INCH    00CC  LOOP    00C8  OUTCH   00DC  RECEV   0012  RMCR   0010
STACK   00A0  START   00C0  TRANS   0013  TRCS    0011
```

```
PAGE  001   TSTPTM

00001                           NAM    TSTPTM
00002                           OPT    Z01

00004                         * TEST PTM BOARD
00005                         * NMI IS TIED TO INVERTED OUTPUT OF TIMER 1
00006                         * SET LATCHES TO 0 AND BRING OUT OF RESET
00007                         *   TO PREVENT CAUSING NMI AT INITIALIZATION
00008                         * THEN SET REQUIRED PTM MODE

00010                         * LILBUG EQUATES
00011           F812  A OUT2HS EQU    $F812
00012           F815  A OUT4HS EQU    $F815
00013           F818  A PCRLF  EQU    $F818

00015                         * ADDRESS FOR PTM
00016           CF00  A PTM    EQU    $CF00

00018                         * TEMPORARY STORAGE
00019A 2000                           ORG    $2000
00020A 2000     0001  A REGA   RMB    1          SAVE REG A
00021A 2001     0002  A TEMP   RMB    2          SAVE REG X & PRT
00022A 2003     04    A        FCB    4
00023A 2004 8E 2100   A        LDS    #$2100     SET STK PNTR
00024                         * SET NMI VECTOR
00025A 2007 CE 2043   A        LDX    #NMIRT
00026A 200A FF FFFC   A        STX    $FFFC
00027                         * INIT PTM
00028A 200D CE CF00   A        LDX    #PTM
00029A 2010 6F 02     A        CLR    2,X        SET LATCHES = 0
00030A 2012 6F 03     A        CLR    3,X
00031A 2014 CC 0122   A        LDD    #$122      GET OUT OF RESET
00032A 2017 A7 01     A        STAA   1,X        SET TO WRITE REG 1
00033A 2019 E7 00     A        STAB   0,X        WRITE REG 1
00034A 201B 86 A6     A        LDAA   #$A6       ENABLE TIMER OUTPUT
00035A 201D A7 00     A        STAA   0,X
00036A 201F 6F 01     A        CLR    1,X        NO MORE WRITE TO CR 1
00037A 2021 4F                 CLRA              FLAG = 0, PRINTED BY NMI R
00038                         *   NMI MIGHT OCCUR AFTER INIT, BUT SHOULDN'T
00039A 2022 CE 2000   A        LDX    #$2000     WAIT LOOP
00040A 2025 09          DLY    DEX
00041A 2026 26 FD 2025         BNE    DLY
00042                         * MAIN LOOP
00043A 2028 CE 2028   A MAIN   LDX    #MAIN      SIGNAL TOP OF LOOP
00044A 202B BD F812   A        JSR    OUT2HS
00045A 202E BD F818   A        JSR    PCRLF      PR CE
00046                         * SET CNTR TO CAUSE NMI
00047A 2031 CE CF00   A        LDX    #PTM
00048A 2034 CC 0601   A        LDD    #$601      SET L,M
00049A 2037 ED 02     A        STD    2,X
00050A 2039 86 FF     A        LDAA   #$FF       RESET FLAG AFTER LATCHES S
00051A 203B CE 2000   A        LDX    #$2000     DELAY - NMI SHOULD OCCUR
00052A 203E 09          DLY2   DEX
00053A 203F 26 FD 203E         BNE    DLY2
00054A 2041 20 E5 2028         BRA    MAIN

00056                         * NMI ROUTINE
00057A 2043 B7 2000   A NMIRT  STAA   REGA       SAVE FLAG-SHOW WHEN NMI OC
00058A 2046 FF 2001   A        STX    TEMP       SAVE LOOP COUNTER
```

```
PAGE  002   TSTPTM

00059A 2049 CE 2000   A        LDX    #REGA
00060                         * PRINT REG A - 0 IF NMI AFTER INIT
00061                         *               FF IF AFTER SET LATCHES
00062                         * PRINT REG X FROM DELAY LOOP
00063A 204C BD F812   A        JSR    OUT2HS
00064A 204F BD F815   A        JSR    OUT4HS
00065A 2052 BD F818   A        JSR    PCRLF
00066A 2055 3B                 RTI
00067                          END
TOTAL ERRORS 00000


DLY     2025  DLY2   203E  MAIN   2028  NMIRT  2043  OUT2HS F812
OUT4HS F815  PCRLF  F818  PTM    CF00  REGA   2000  TEMP   2001
```

APPENDIX D
LILbug PROGRAM LISTING

```
PAGE  001   LILBUG 6801 DEBUG MONITOR *** VER 1.0 ***

00001                           OPT     Z01
00002                           TTL     6801 DEBUG MONITOR *** VER 1.0 ***
00003                           NAM     LILBUG




00005                   * * * * * * * * * * * * * * * * * * * * * * *
00006                   *                                           *
00007                   * * * * *         L I L B U G       * * * * *
00008                   *                                           *
00009                   *                 M O T O R O L A           *
00010                   *           A U S T I N ,  T E X A S        *
00011                   *                                           *
00012                   * * * * * * * * * * * * * * * * * * * * * * *




00014                   *********************************************
00015                   * ALTHOUGH THE INFORMATION CONTAINED HEREIN, AS
00016                   * WELL AS ANY INFORMATION PROVIDED RELATIVE
00017                   * THERETO, HAS BEEN CAREFULLY REVIEWED AND IS
00018                   * BELIEVED ACCURATE, MOTOROLA ASSUMES NO LIABILITY
00019                   * ARISING OUT OF ITS APPLICATION OR USE, NEITHER
00020                   * DOES IT CONVEY ANY LICENSE UNDER ITS PATENT
00021                   * RIGHTS NOR THE RIGHTS OF OTHERS.
00022                   *********************************************


00024                   * COPYRIGHT (C) MOTOROLA, INC., 1978
```

```
00018                    ***************** COMMANDS *****************
00019                    * L   LOAD A PROGRAM FROM TAPE
00020                    * L (OFFSET)  LOAD FROM TAPE WITH AN OFFSET
00021                    * V   VERIFY THAT A PROGRAM WAS PROPERLY LOADED
00022                    * V (OFFSET)  VERIFY PROGRAM LOADED WITH AN OFFSET
00023                    * D X,Y DISPLAY MEMORY FROM X TO Y
00024                    * P X,Y PUNCH CONTENTS OF MEMORY FROM X TO Y
00025                    * M X    MEMORY EXAMINE/MODIFY
00026                    *       (DATA) CHANGE 1 BYTE IN MEMORY TO (DATA)
00027                    *       LF  INCR POINTER, PR ADR AND VALUE OF NEW PNTR
00028                    *       SP  INCR PNTR, PR NEW VALUE ON SAME LINE
00029                    *       ,   INCR PNTR, NO PR OF ADR OR VALUE
00030                    *       UA  DECR PNTR, PR ADR AND VALUE AT PNTR
00031                    *       /   PR ADR AND VALUE OF CURRENT PNTR
00032                    *       CR  END MEMORY EXAMINE COMMAND
00033                    * X/  SAME AS M X, X MUST START W/ 0-9, MAY NEED LEADING
00034                    * /   PR ADR AND VALUE OF LOC LAST REF WITH MEM/EXAM
00035                    * O X Y CALCULATE RELATIVE OFFSET FOR BRANCH INSTR
00036                    * B   DISPLAY ALL BREAKPOINTS
00037                    * B - DELETE ALL BREAKPOINTS
00038                    * B X ENTER BREAKPOINT AT ADR X
00039                    * B -X DELETE BREAKPOINT AT ADR X
00040                    * G X EXECUTE USER PROG STARTING AT ADR X
00041                    * G   EXECUTE USER PROG STARTING AT CURRENT PC
00042                    * R  DISPLAY/CHANGE USER'S PROGRAM REGS
00043                    * .  TRACE 1 INSTRUCTION
00044                    * T X  TRACE X INSTRUCTIONS
00045                    * C EXECUTE USER'S CODE AS SUBR, RTS TO MONITOR
00046                    * C X XQT USER'S CODE AS SUBR START AT ADR X
00047                    * HI SET HIGH SPEED - 120 CPS FOR ON-CHIP IO
00048                    * HY SET HIGHER YET SPEED, FOR CRT - 9600 BD
00049                    * CONTROL X - TERMINATE D OR T PRINT
00050                    * CONTROL W - WAIT DURING D OR T PRT, ANY CHAR
00051                    *    CAUSES CONTINUATION OF PRINT
```



```
00053                            * CONTROL CHARACTERS RECOGNIZED DURING PRINT
00054          0017  A CNTLW  EQU    $17       WAIT CHARACTER
00055          0018  A CNTLX  EQU    $18       ABORT CHARACTER
00056                            * ON-CHIP CLOCK EQUATES
00057          0001  A P2DDR  EQU    $01       PORT 2 DATA DIRECTION REG
00058          0009  A CLOCK  EQU    $09       TIMER 1
00059          0008  A TCSR   EQU    $08       TIMER CONTROL STATUS REG
00060          000B  A OCREG  EQU    $0B       OUTPUT COMPARE REG
00061                            * ON-CHIP IO EQUATES
00062          0010  A RMCR   EQU    $010      RATE & MODE CONTROL
00063          0011  A TRCS   EQU    $011      TRANSMIT/RECEIVE CNTRL STAT REG
00064          0012  A RECEV  EQU    $012      READ REG
00065          0013  A TRANS  EQU    $013      TRANSMIT REG
00066                            * MODE SELECT WORD
00067          0003  A MODE   EQU    $03       UPPER 3 BITS = MODE
00068                            * DEFAULT ADDRESS FOR PTM
00069          E000  A PTMADR EQU    $E000

00071                            *************** RAM STORAGE ***************
00072          0080  A LOWRAM EQU    $80       USED FOR STK OVFLOW CHK
00073A 00CF                      ORG    $CF
00074A 00CF    0001  A STACK  RMB    1         STK PNTR WILL RUN UP TOWARD USER C
00076A 00D0    0001  A CT     RMB    1         INPUT CHAR CT
00077          00D0  A CKSUM  EQU    CT        USED BY LOAD/VERF
00078A 00D1    0002  A STRTX  RMB    2         INPUT CHAR PTR (ON SATCK)
00079A 00D3    0002  A NEXTX  RMB    2         NEXT TABLE PTR
00080                            * CHRNL - UPPER 6 BITS-# NULLS AFTER CR
00081                            *         LOW 2 BITS-# NULLS AFTER CHAR
00082A 00D5    0001  A CHRNL  RMB    1         NUM NULLS AFTER CHAR
00083A 00D6    0001  A BBLK   RMB    1         BULK STORE BLK + NXT 4 LOC
00084A 00D7    0002  A PNTR   RMB    2         OPEN ADR
00085A 00D9    0002  A TEMPA  RMB    2
00086A 00DB    0001  A TEMP   RMB    1
00087                            * TEMP AND OVFL MUST FOLLOW TEMPA
00088A 00DC    0001  A OVFL   RMB    1         OVERFLOW FLAG
00089A 00DD    0002  A SAVSTK RMB    2         PC
00090A 00DF    0002  A        RMB    2         X
00091A 00E1    0001  A        RMB    1         A
00092A 00E2    0001  A        RMB    1         B
00093A 00E3    0001  A        RMB    1         CC
00094A 00E4    0002  A SPSAVE RMB    2         STK
00095          0004  A NUMBP  EQU    4         NUMBER OF BREAKPOINTS
00096A 00E6    0008  A BKADR  RMB    NUMBP*2   BRKPNT ADDRESS
00097A 00EE    0004  A OPCODE RMB    NUMBP
00098A 00F2    0001  A BRKFLG RMB    1         BRKPNT IN
00099A 00F3    0002  A NTRACE RMB    2         TRACE N INSTR
00100A 00F5    0001  A EXONE  RMB    1         XQT 1 INSTR FLAG
00101A 00F6    0001  A OUTSW  RMB    1         ECHO FLAG
00102A 00F7    0001  A CALLF  RMB    1         FLAG FOR C CMD
00103A 00F8    0002  A PTM    RMB    2         PTM ADDRESS
00104A 00FA    0002  A FCTPTR RMB    2         POINTER TO FUNCTION TABLE
00105A 00FC    0002  A IOPTR  RMB    2         IO TABLE POINTER
00106A 00FE    0002  A VECPTR RMB    2         VECTOR TABLE POINTER
```

```
00108                    ***************** BEGINNING OF ROM ****************
00109A F800                      ORG     $F800

00111                    * JUMP TABLE TO SUBROUTINES
00112A F800 7E FC7E   A EX.NMI   JMP     M.NMI       NMI VECTOR FOR PTM
00113A F803 7E FC79   A IN.NMI   JMP     C.NMI       NMI VECTOR FOR ON-CHIP TIMER
00114A F806 7E F873   A INCHNP   JMP     INCH1       INPUT 1 CHAR W/ NO PARITY
00115A F809 7E F88A   A OUTCH    JMP     OUTCH1      OUTPUT 1 CHAR W/PADDING
00116A F80C 7E FB07   A          JMP     PDATA1      PRINT DATA STRING
00117A F80F 7E FB0E   A          JMP     PDATA       PR CR/LF, DATA STRING
00118A F812 7E FADB   A          JMP     OUT2HS      PR 2 HEX + SP (X)
00119A F815 7E FAD8   A          JMP     OUT4HS      PR 4 HEX + SP (X)
00120A F818 7E FB12   A          JMP     PCRLF       PRINT CR/LF
00121A F81B 7E FADD   A          JMP     SPACE       PRINT A SPACE
00122A F81E 7E F8F6   A STRT     JMP     START       RESTART ADDRESS
00123A F821 7E FD5F   A IN.SWI   JMP     M.SWI       SWI VECTOR

00125                    *********** COMMAND TABLE ***********
00126                    * THERE MAY BE AN EXTERNAL TABLE OF THE SAME
00127                    *    FORMAT. 'FCTPTR' POINTS TO THE TABLE TO
00128                    *    BE SEARCHED FIRST. THE USER CAN DEFINE
00129                    *    HIS OWN TABLE AND SET FCTPTR.
00130                    *
00131                    * EACH ENTRY IN THE TABLE IS AS FOLLOWS:
00132                    *     FCB   XXX   XXX=TOTAL SIZE OF ENTRY
00133                    *     FCC   /STRING/ STRING IS THE INPUT STRING
00134                    *     FDB   ADDR     ADDR IS THE ROUTINE ADDRESS
00135                    *
00136                    * THE LAST ENTRY IS:
00137                    *     -1=END EXTERNAL TABLE,SEARCH INTERNAL TABLE
00138                    *     -2=END OF TABLE(S)
00139                    *
00140                    * NOTE: AN EXTERNAL FUNCTION TABLE TERMINATED BY
00141                    *   -1, THE INTERNAL TABLE WILL ALSO BE SEARCHED.
00142                    *   IF TERMINATED BY -2, INTERNAL TABLE NOT CHECKED.
00143                    *
00144          F824   A FCTABL  EQU     *
00145A F824    04    A          FCB     4           *
00146A F825    42    A          FCC     /B/         *
00147A F826    FB98  A          FDB     BRKPNT
00148A F828    04    A          FCB     4           *
00149A F829    43    A          FCC     /C/
00150A F82A    FC0F  A          FDB     CALL
00151A F82C    04    A          FCB     4
00152A F82D    44    A          FCC     /D/
00153A F82E    FD86  A          FDB     DISPLY
00154A F830    04    A          FCB     4           *
00155A F831    47    A          FCC     /G/
00156A F832    FC11  A          FDB     GOXQT
00157A F834    04    A          FCB     4           *
00158A F835    4C    A          FCC     /L/
00159A F836    FE9A  A          FDB     LOAD
00160A F838    04    A          FCB     4           *
00161A F839    4D    A          FCC     /M/         *
00162A F83A    FA5B  A          FDB     MEMORY
00163A F83C    04    A          FCB     4           *
00164A F83D    4F    A          FCC     /O/
```



```
00165A F83E    FAB4  A          FDB     OFFSET
00166A F840    04    A          FCB     4           *
00167A F841    50    A          FCC     /P/
00168A F842    FE72  A          FDB     PUNCH
00169A F844    04    A          FCB     4           *
00170A F845    52    A          FCC     /R/
00171A F846    FB1E  A          FDB     REGSTR
00172A F848    05    A          FCB     5
00173A F849    48    A          FCC     /HI/
     A F84A    49    A
00174A F84B    F8E9  A          FDB     S120
00175A F84D    05    A          FCB     5
00176A F84E    48    A          FCC     /HY/
     A F84F    59    A
00177A F850    F8F1  A          FDB     HY
00178A F852    04    A          FCB     4           *
00179A F853    54    A          FCC     /T/
00180A F854    FC46  A          FDB     TRACE
00181A F856    04    A          FCB     4           *
00182A F857    56    A          FCC     /V/
00183A F858    FEAB  A          FDB     VERF
00184A F85A    FE    A          FCB     -2          *END OF TABLE

00186                    ***************** IO TABLE ***************
00187                    * ROUTINE IO IS CALLED WITH
00188                    * INDEX INTO IO TABLE CI OR INTO USER IO TABLE
00189                    * IOPTR POINTS TO THE IO TABLE TO BE USED
00190                    * THE INDEX TABLE DEFINES ORDER OF IO ROUTINES IN IO TABL
00191A F85B    F8C8  A CI       FDB     CION,CIDTA,CIOFF
     A F85D    F891  A
     A F85F    F8D1  A
00192A F861    F8D2  A          FDB     COON,CODTA,COOFF
     A F863    F8A8  A
     A F865    F8D1  A
00193A F867    F8D1  A          FDB     HSON,HSDTA,HSOFF
     A F869    FDEC  A
     A F86B    F8D1  A
00194A F86D    FEAF  A          FDB     BSON,BSDTA,BSOFF
     A F86F    FEC3  A
     A F871    FEBA  A

00196                    * THE FOLLOWING ARE INDICES INTO IO TABLE
00197          0000  A CI.ON   EQU     0           INIT INPUT DEVICE
00198          0002  A CI.DTA  EQU     2           INPUT A CHAR W/NO WAIT
00199          0004  A CI.OFF  EQU     4           DISABLE INPUT DEVICE
00200          0006  A CO.ON   EQU     6           INIT OUTPUT DEVICE
00201          0008  A CO.DTA  EQU     8           OUTPUT A CHAR W/PADDING
00202          000A  A CO.OFF  EQU     $A          DISABLE OUTPUT DEVICE
00203          000C  A HS.ON   EQU     $C          INIT HIGH SPEED OUTPUT DEVICE
00204          000E  A HS.DTA  EQU     $E          OUTPUT BLOCK OF DATA
00205          0010  A HS.OFF  EQU     $10         DISABLE HIGH SPEED DEVICE
00206          0012  A BS.ON   EQU     $12         INIT PUNCH/LOAD
00207          0014  A BS.DTA  EQU     $14         WRITE DATA BLK TO PNCH/LOAD
00208          0016  A BS.OFF  EQU     $16         DISABLE PUNCH/LOAD
00209                    *
00210                    ************** INCH **************
00211                    * CALL IO ROUTINE W/ INDEX TO INPUT DATA
00212                    * CLEARS PARITY
```

```
00213                    * IGNORES RUBOUT CHAR
00214                    * ECHOES OUTPUT IF FLAG CLEAR
00215                    * SAVE, RESTORE REG B
00216A F873 37           INCH1  PSHB
00217A F874 C6 02    A   INCH15 LDAB   #CI.DTA  OFFSET TO CIDTA
00218A F876 8D 67 F8DF   INCH2  BSR    IO        SCAN IO DEVICE
00219A F878 24 FA F874           BCC    INCH15    LOOP ON NO WAIT INPUT
00220A F87A 84 7F    A           ANDA   #$7F      CLEAR PARITY
00221A F87C 27 F6 F874           BEQ    INCH15    IGNORE NULLS
00222A F87E 81 7F    A           CMPA   #$7F      RUBOUT?
00223A F880 27 F2 F874           BEQ    INCH15
00224A F882 D6 F6    A           LDAB   OUTSW     CHK IF ECHO
00225A F884 26 02 F888           BNE    INCH4
00226A F886 8D 02 F88A           BSR    OUTCH1    ECHO INPUT
00227A F888 33           INCH4  PULB
00228A F889 39                  RTS

00230                    ********** OUTCH **********
00231                    * CALL IO ROUTINE W/ INDEX TO OUTPUT DATA
00232                    * SAVES, RESTORES REG B
00233A F88A 37           OUTCH1 PSHB
00234A F88B C6 08    A           LDAB   #CO.DTA   PNTR TO OUTPUT A CHAR W/PADDING
00235A F88D 8D 50 F8DF           BSR    IO
00236A F88F 33                  PULB
00237A F890 39                  RTS

00239                    ********** CIDTA **********
00240                    * READ 1 CHAR FROM INPUT W/ NO WAIT
00241                    * RETURN W/ C CLEAR IF NO READ
00242                    *      ELSE REG A = INPUT & C IS SET
00243A F891 96 11    A CIDTA  LDAA   TRCS      GET CONTROL WORD
00244A F893 48                  ASLA           CHK THAT RDRF IS SET
00245A F894 25 03 F899           BCS    CIDTA1    READ DATA IF SET
00246A F896 48                  ASLA           LOOK AT ERR BIT
00247A F897 24 03 F89C           BCC    CIDTA2    RTN W/C CLR IF NO READ
00248                    * IF FRAMING ERR OR OVER RUN-READ
00249A F899 96 12    A CIDTA1 LDAA   RECEV     READ
00250                    * RETURN W/CARRY SET & LDAA BITS SET
00251A F89B 0D                  SEC            FLAG READ-NO WAIT ACOMPLISHD
00252A F89C 39           CIDTA2 RTS

00254                    ********** CODTA **********
00255                    * OUTPUT CHAR FROM REG A
00256                    * OUTC - SUBR CALLED BY CODTA
00257                    * EXPECT 30 OR 120 CPS
00258                    * DEFAULT SPEED = 30 CPS
00259                    * PADS CR AND CHAR FOR 120
00260                    * PAD 4 NULLS IF PUNCH CR
00261A F89D 37           OUTC   PSHB
00262A F89E D6 11    A OUTC1  LDAB   TRCS      GET CONTRL WRD
00263A F8A0 C5 20    A           BITB   #$20      TDRE SET?
00264A F8A2 27 FA F89E           BEQ    OUTC1     WAIT UNTIL IT IS
00265A F8A4 97 13    A           STAA   TRANS
00266A F8A6 33                  PULB
00267A F8A7 39           CRTN   RTS

00269A F8A8 8D F3 F89D CODTA  BSR    OUTC      OUTPUT CHAR
00270A F8AA D6 F6    A        LDAB   OUTSW     GET TAPE FLAG
```



```
00271A F8AC 26 02 F8B0           BNE    N1
00272A F8AE D6 D5    A           LDAB   CHRNL     NOT TAPE
00273A F8B0 81 0D    A N1       CMPA   #$D       CR
00274A F8B2 27 08 F8BC           BEQ    N3
00275A F8B4 81 10    A           CMPA   #$10      NO PADDING IF DLE
00276A F8B6 27 EF F8A7           BEQ    CRTN
00277A F8B8 C4 03    A           ANDB   #$3       MASK OUT HIGH 6-BIT CNTR
00278A F8BA 20 02 F8BE           BRA    N4
00279A F8BC 54           N3     LSRB           REMOVE LOW 2-BIT CNTR
00280A F8BD 54                  LSRB
00281A F8BE 5A           N4     DECB           DECR NULL CNTR
00282A F8BF 2B E6 F8A7           BMI    CRTN      EXIT IF ENOUGH NULLS
00283A F8C1 36                  PSHA
00284A F8C2 4F                  CLRA
00285A F8C3 8D D8 F89D           BSR    OUTC      OUTPUT NULL
00286A F8C5 32                  PULA
00287A F8C6 20 F6 F8BE           BRA    N4        PR NXT NULL

00289                    ********** CION **********
00290                    * INITIALIZE ON-CHIP SERIAL IO
00291A F8C8 CC 1007  A CION   LDD    #$1007    SET PADDING FOR 300
00292A F8CB 8D 1F F8EC           BSR    S1205     SET RMCR
00293A F8CD 86 0A    A           LDAA   #$0A      SET TRCS FOR ON-CHP IO
00294A F8CF 97 11    A           STAA   TRCS
00295                    * NO ACTION NEEDED BY THESE DEVICES
00296          F8D1  A CIOFF  EQU    *         TURN CONSOLE IN OFF
00297          F8D1  A HSON   EQU    *         TURN ON HIGH SPEED
00298          F8D1  A HSOFF  EQU    *         TURN OFF HIGH SPEED
00299A F8D1 39           COOFF  RTS
00300
00301                    ********** COON **********
00302                    * INITIALIZE OUTPUT DEVICE-SILENT 700 PRT
00303                    * TURN ON TI PRINTER
00304A F8D2 CE FF86  A COON   LDX    #PRTON    ACTIVATE ACD
00305A F8D5 BD FB07  A COON2  JSR    PDATA1
00306                    * ENTRY FROM BSOFF FOR DELAY AFTER TURN OFF PUNCH
00307A F8D8 CE 411B  A DELAY  LDX    #$411B    100 MS DELAY
00308A F8DB 09           DLY    DEX
00309A F8DC 26 FD F8DB           BNE    DLY
00310A F8DE 39                  RTS
00311                    *
00312                    ********** IO ROUTINE **********
00313                    * THIS ROUTINE USES INDEX TO RETRIEVE IO
00314                    * ROUTINE ADR FROM IO TABLE, THEN CALL AS SUBR
00315                    * REG B IS INDEX INTO IO TABLE
00316                    * TO DO IO, REG B IS SET, IO ROUTINE IS CALLED
00317                    * SAVES REG X
00318A F8DF 3C           IO     PSHX
00319A F8E0 DE FC    A          LDX    IOPTR     ADR OF IO TABLE
00320A F8E2 3A                  ABX            ADD OFFSET
00321A F8E3 EE 00    A          LDX    0,X       GET IO ROUTINE ADR
00322A F8E5 AD 00    A          JSR    0,X       DO IO
00323A F8E7 38                  PULX
00324A F8E8 39                  RTS

00326                    ********** HY / HI **********
00327                    * HY & HI SET CHRNL FLAG FOR PADDING
00328                    * AND SET ON-CHIP SERIAL IO RMCR
```

```
00329                    * LOW 2 BITS = NUM NULLS AFTER CHAR
00330                    * HIGH 6 BITS = NUM NULLS AFTER CR

00332                    ************** HI **************
00333                    * SET SPEED FOR 120 CPS
00334                    * SET # NULLS TO PAD CHAR
00335                    * SET BITS FOR 1200 BAUD IN RMCR
00336A F8E9 CC 4F06  A S120   LDD    #$4F06
00337A F8EC 97 D5    A S1205  STAA   CHRNL
00338A F8EE D7 10    A        STAB   RMCR      SET BAUD RATE
00339A F8F0 39                RTS

00341                    *************** HY ***************
00342                    * HIGHER YET - 9600 BAUD ON CRT
00343                    * SET PADDING TO ZERO
00344A F8F1 CC 0005  A HY     LDD    #$0005    ALSO SET RMCR
00345A F8F4 20 F6 F8EC        BRA    S1205
```



```
00347                    ********** RESET **********
00348                    * COME HERE FOR MONITOR RESTART
00349                    * INIT IO & FCN TABLE POINTERS
00350                    * TURN ON CONSOLE
00351                    * PRINT MONITOR NAME
00352                    * INIT RAM USED BY MONITOR
00353                    * MASK I BIT IN USER CC
00354                    * SET INITIAL SPEED
00355                    * INIT HARDWARE TRACE DEVICE
00356A F8F6 8E 00CF  A START  LDS    #STACK    INIT STK PNTR
00357A F8F9 CE F85B  A        LDX    #CI       INIT I/O PNTR TABLE
00358A F8FC DF FC    A        STX    IOPTR
00359A F8FE CE FFC8  A        LDX    #SERIAL   INIT VECTOR TABLE POINTER
00360A F901 DF FE    A        STX    VECPTR
00361A F903 CE F824  A        LDX    #FCTABL   INIT FUNCTION TABLE PTR
00362A F906 DF FA    A        STX    FCTPTR
00363A F908 CE E000  A        LDX    #PTMADR   SET ADR FOR PTM
00364A F90B DF F8    A        STX    PTM
00365A F90D 8E 00BB  A        LDS    #STACK-20 RESET INCASE USER DIDN'T
00366A F910 9F E4    A        STS    SPSAVE    INIT USER STACK
00367A F912 8E 00CF  A        LDS    #STACK    RESET MONITOR STK
00368A F915 CE 00E6  A        LDX    #BKADR    ZERO BKADR TO OVFL
00369A F918 6F 00    A CLRAM  CLR    0,X
00370A F91A 08                INX
00371A F91B 8C 00F8  A        CPX    #CALLF+1
00372A F91E 26 F8 F918        BNE    CLRAM
00373A F920 5F                CLRB             OFFSET FOR CION
00374A F921 8D BC F8DF        BSR    IO        TURN ON CONSOLE IN
00375A F923 C6 06    A        LDAB   #CO.ON    OFFSET FOR COON
00376A F925 8D B8 F8DF        BSR    IO        TURN ON CONSOLE OUTPUT
00377A F927 CE FF90  A        LDX    #LIL      PR LILBUG
00378A F92A BD FB0E  A        JSR    PDATA     WITH CR/LF
00379A F92D 86 D0    A        LDAA   #$D0      MASK I IN CC
00380A F92F 97 E3    A        STAA   SAVSTK+6


00383                    * INIT FOR HARDWARE TRACE -
00384                    *    CLOCK OR PTM
00385A F931 BD FCEA  A        JSR    IFPTM
00386A F934 27 0A F940        BEQ    INPTM     GO INIT PTM
00387                    * INIT ON-CHIP CLOCK
00388A F936 8C F803  A        CPX    #IN.NMI   MAY NOT WANT ANY TRACE
00389A F939 26 17 F952        BNE    MAIN      IF NMI NOT SET-NO TRACE
00390A F93B 7C 0008  A        INC    TCSR      SET OLVL BIT HI
00391A F93E 20 12 F952        BRA    MAIN
00392                    * INIT PTM - SINGLE SHOT, 8 BIT
00393                    * USER MUST SET NMI VCTR FOR PTM TRACE
00394                    * MONITOR CHK IF VCTR SET
00395A F940 DE F8    A INPTM  LDX    PTM       GET PTM ADDRESS
00396A F942 6F 02    A        CLR    2,X       SET LATCH FOR BRING OUT
00397A F944 6F 03    A        CLR    3,X       OF RESET, MAKE G HI
00398A F946 CC 0122  A        LDD    #$0122
00399A F949 A7 01    A        STAA   1,X       SET TO WRITE TO CR1
00400A F94B E7 00    A        STAB   0,X       BRING OUT OF RESET
00401A F94D CC A600  A        LDD    #$A600    SET SINGLE SHOT MODE
00402A F950 ED 00    A        STD    0,X       ALSO SET NO WRITE TO CR1
00403                    *
```

```
00404                    ***************** MAIN ****************
00405                    * PROMPT USER
00406                    * READ NEXT COMMAND
00407                    *
00408A F952 8E 00CF  A MAIN   LDS    #STACK
00409A F955 7F 00F6  A        CLR    OUTSW    MAKE SURE INPUT IS ECHOED
00410A F958 BD FB12  A        JSR    PCRLF    PRINT CR/LF
00411A F95B 86 21    A        LDAA   #'!
00412A F95D BD F809  A        JSR    OUTCH
00413A F960 BD FA09  A        JSR    INPUTA   A-F ALPHA
00414A F963 2B ED F952        BMI    MAIN     ABORT
00415A F965 27 0A F971        BEQ    MAIN01
00416                    * HEX VALIDITY CHK
00417A F967 BD F9FC  A        JSR    VALIN
00418A F96A 2B E6 F952        BMI    MAIN     <ADR>/ VALID?
00419A F96C CE FA5F  A        LDX    #MEM01   ENTER MEMORY ROUTINE
00420A F96F 20 74 F9E5        BRA    MAIN08   SET UP FOR RTN
00421                    * A CONTAINS FIRST INPUT CHARACTER
00422A F971 CE FC41  A MAIN01 LDX    #NEXT    CHK FOR TRACE 1
00423A F974 81 2E    A        CMPA   #'.      QUICK TRACE
00424A F976 27 6D F9E5        BEQ    MAIN08
00425A F978 CE FAA8  A        LDX    #MEMSL   CHK FOR /
00426A F97B 81 2F    A        CMPA   #'/      QUICK MEM EXAMINE
00427A F97D 27 66 F9E5        BEQ    MAIN08
00428                    *
00429                    * READ IN STRING. PUSH STRING UNTO THE
00430                    *   STACK. MARK TOP OF STRING IN 'STRTX'
00431                    *
00432A F97F 9F D1    A        STS    STRTX    SAVE PTR TO INPUT STRING
00433A F981 7F 00D0  A        CLR    CT       INPUT CHAR CT
00434A F984 8D 65 F9EB MAIN03 BSR    TERM     CHECK FOR TERMINATORS
00435A F986 27 13 F99B        BEQ    SRCH     GOT ONE,GO DO COMPARES
00436A F988 7C 00D0  A        INC    CT       CT + 1 -> CT
00437A F98B 36               PSHA             SAVE INPUT CHAR ON STACK
00438A F98C 30               TSX              CHECK STACK POINTER
00439A F98D 8C 0080  A        CPX    #LOWRAM
00440A F990 27 2B F9BD        BEQ    MERROR   CHK IF END OF STK
00441A F992 BD FA09  A        JSR    INPUTA   GO GET NEXT CHAR
00442A F995 2B 2C F9C3        BMI    MAIN07   ESCAPE
00443A F997 26 24 F9BD        BNE    MERROR   NBRS ARE NOT ALLOWED
00444A F999 20 E9 F984        BRA    MAIN03   LOOP
00445                    *
00446                    * HERE AFTER STRING WAS INPUT. CHECK IT AGAINST
00447                    *   STRINGS IN THE EXTERNAL AND/OR INTERNAL
00448                    *   FUNCTION TABLES. STRTX POINTS TO THE
00449                    *   INPUT STRING. FCTPTR POINTS TO THE START
00450                    *   OF THE FIRST TABLE TO SEARCH (EXTERNAL OR
00451                    *   INTERNAL).
00452                    *
00453A F99B 97 D6    A SRCH   STAA   BBLK     LOCAL VAR - SAVE DELIMITER
00454A F99D DE FA    A        LDX    FCTPTR   GET PTR TO TABLE
00455A F99F DF D3    A        STX    NEXTX    SAVE IN TEMP
00456A F9A1 DE D3    A SRCH01 LDX    NEXTX    GET NEXT PTR INTO TABLE
00457A F9A3 3C               PSHX             SAVE A COPY ON STACK
00458A F9A4 E6 00    A        LDAB   0,X      GET ENTRY SIZE
00459A F9A6 3A               ABX              CALCULATE ADDR OF NEXT ENTRY
00460A F9A7 DF D3    A        STX    NEXTX    SAVE FOR NEXT SEARCH
00461A F9A9 C0 03    A        SUBB   #3       SUB OFF ADDR SIZE
```



```
00462A F9AB D1 D0    A        CMPB   CT       IS INPUT LENGTH=ENTRY LENGTH?
00463A F9AD 27 16 F9C5        BEQ    SRCH03   YES,A POSSIBLE MATCH
00464                    * NO MATCH ON THIS ENTRY
00465                    * CHECK FOR TABLE TERMINATORS
00466                    * -1 = END OF EXTERNAL TABLE
00467                    * -2 = END OF TABLE(S)
00468                    * IF NOT -1 OR -2, NOT RECOGNIZE END OF TABLE
00469                    * B IS ALLREADY TERM-3
00470A F9AF 38               PULX             CLEAN STACK
00471A F9B0 C1 FC    A        CMPB   #-4       END OF EXTERNAL TABLE?
00472A F9B2 26 05 F9B9        BNE    SRCH02   NO
00473                    * SWITCH FROM EXT TO INT TABLE
00474A F9B4 CE F824  A        LDX    #FCTABL  GET INNER TABLE
00475A F9B7 DF D3    A        STX    NEXTX
00476A F9B9 C1 FB    A SRCH02 CMPB   #-5       END OF TABLE SEARCH?
00477A F9BB 26 E4 F9A1        BNE    SRCH01   NO,KEEP TRUCKIN
00478                    * INPUT STRING NOT FOUND ! GO GRIPE
00479                    * HERE ON ERROR. PRINT ? AND
00480                    *   GO BACK TO MAIN START
00481A F9BD CE FF8E  A MERROR LDX    #QMARK
00482A F9C0 BD FB0E  A        JSR    PDATA
00483A F9C3 20 8D F952 MAIN07 BRA    MAIN
00484                    *
00485                    * INPUT LENGTH=TABLE ENTRY LENGTH. TRY
00486                    *   FOR A MATCH. B=SIZE; (SP) = TABLE PTR
00487                    *
00488A F9C5 DE D1    A SRCH03 LDX    STRTX    INIT PTR TO INPUT STRING
00489A F9C7 DF D9    A        STX    TEMPA
00490A F9C9 38           SRCH04 PULX             RESTORE CURRENT TABLE PTR
00491A F9CA 08               INX
00492A F9CB A6 00    A        LDAA   0,X      GET TABLE CHAR
00493A F9CD 3C               PSHX             SAVE FOR NEXT LOOP
00494A F9CE DE D9    A        LDX    TEMPA    GET INPUT PTR
00495A F9D0 A1 00    A        CMPA   0,X      INPUT CHAR=TABLE CHAR?
00496A F9D2 27 03 F9D7        BEQ    SRCH05   YES
00497A F9D4 38               PULX             NO,CLEAN STAACK
00498A F9D5 20 CA F9A1        BRA    SRCH01   GET NEXT TABLE VALUE
00499                    * HERE WHEN A CHARACTER MATCHED
00500A F9D7 09           SRCH05 DEX              DEC INPUT PTR FOR NEXT TIME
00501A F9D8 DF D9    A        STX    TEMPA
00502A F9DA 5A               DECB             COMPARED ALL CHARS?
00503A F9DB 26 EC F9C9        BNE    SRCH04
00504                    *
00505                    * WE HAVE A MATCH! GO TO THE ROUTINE
00506                    *
00507A F9DD 38               PULX             GET TABLE PTR
00508A F9DE 08               INX              POINT TO ADDRESS IN TABLE
00509A F9DF 9E D1    A        LDS    STRTX    CLEAN STACK
00510A F9E1 EE 00    A        LDX    0,X      GET ROUTINE ADDRESS
00511A F9E3 96 D6    A        LDAA   BBLK     LOAD TERMINATOR
00512A F9E5 AD 00    A MAIN08 JSR    0,X      GO TO ROUTINE
00513A F9E7 2B D4 F9BD        BMI    MERROR   ERROR RETURN
00514A F9E9 20 D8 F9C3        BRA    MAIN07   GO BACK TO MAIN
00515                    ********** TERMINATOR SUB
00516                    *
00517                    * CHECK INPUT CHAR FOR A TERMINATOR
00518                    *   TERMINATORS ARE: , BLANK (CR)
00519                    *   CHAR IN A ON CALL
```

```
00520                    *   Z BIT SET ON EXIT IFF CHAR WAS
00521                    *   TERMINATOR
00522                    ***********
00523A F9EB 81 2C    A TERM   CMPA   #',      COMMA?
00524A F9ED 27 0A F9F9        BEQ    TERM02
00525A F9EF 81 20    A        CMPA   #' '     BLANK?
00526A F9F1 27 06 F9F9        BEQ    TERM02
00527A F9F3 81 0D    A        CMPA   #$D      CR?
00528A F9F5 27 02 F9F9        BEQ    TERM02
00529A F9F7 81 2D    A        CMPA   #'-      ALLOW MINUS
00530A F9F9 39           TERM02 RTS            RETURN WITH Z BIT
00531                    *

00533                    ***************** VALIN ****************
00534                    * VALIDATE INPUT - ENTRY VALINP READS INPUT
00535                    * ALLOW 4 DIGIT INPUT W/LEADING 0'S NOT COUNT
00536                    * SET CC NEG IF ERROR
00537A F9FA 8D 11 FA0D VALINP BSR    INPUT    READ HEX
00538A F9FC 2F 0A FA08 VALIN  BLE    VALRTN
00539A F9FE C1 04    A        CMPB   #4
00540A FA00 2F 3A FA3C        BLE    INPUTC
00541A FA02 7D 00DC  A        TST    OVFL     LEADING ZEROES?
00542A FA05 27 35 FA3C        BEQ    INPUTC
00543A FA07 53                COMB             SET C NEG FOR ERR RTN
00544A FA08 39           VALRTN RTS

00546                    *****INPUT - READ ROUTINE
00547                    * INPUT ENTRY SET B=0, READ A-F AS HEX
00548                    * INPUTA ENTRY SET B#0, READ A-F AS ALPHA
00549                    * X= HEX NUMBER (ALSO IN TEMPA)
00550                    * A=LAST CHAR READ (NON-HEX)
00551                    * B= # HEX CHAR READ (TEMP)
00552                    * OVFL # 0 IF OVERFLOW FROM LEFT SHIFT
00553                    * CC SET FROM LDAB BEFORE RETRN
00554                    * CC SET NEG IF ABORT
00555A FA09 C6 F0    A INPUTA LDAB   #$F0     READ A-F AS ALPHA
00556A FA0B 20 01 FA0E        BRA    INPUT2
00557A FA0D 5F           INPUT  CLRB             READ A-F AS HEX
00558A FA0E CE 0000  A INPUT2 LDX    #0       INIT VAR TO 0
00559A FA11 DF D9    A        STX    TEMPA
00560A FA13 DF DB    A        STX    TEMP     0 TTEMP, OVFL
00561A FA15 CE 00D9  A        LDX    #TEMPA   X PNT TO WH INPUT CHR STORED
00562A FA18 8D 25 FA3F INPUT3 BSR    INHEX    READ A CHAR
00563A FA1A 2B 17 FA33        BMI    INPUT7   JMP IF NOT HEX
00564A FA1C C6 04    A        LDAB   #4
00565A FA1E 68 01    A INPUT5 ASL    1,X
00566A FA20 69 00    A        ROL    0,X
00567A FA22 24 03 FA27        BCC    INPUT6   SET FLAG IF OVERFLOW
00568A FA24 7C 00DC  A        INC    OVFL
00569A FA27 5A           INPUT6 DECB             LEFT SHIFT 4 BITS
00570A FA28 26 F4 FA1E        BNE    INPUT5
00571A FA2A AA 01    A        ORAA   1,X      ADD IN LSB
00572A FA2C A7 01    A        STAA   1,X
00573A FA2E 7C 00DB  A        INC    TEMP
00574A FA31 20 E5 FA18        BRA    INPUT3
00575A FA33 81 18    A INPUT7 CMPA   #CNTLX   CHK IF ABORT
00576A FA35 26 03 FA3A        BNE    INPUT9   SKIP IF NOT ABORT
00577          FA37  A NOTHEX EQU    *        ERROR ENTRY FROM INHEX
```



```
00578A FA37 C6 FF    A        LDAB   #$FF     SET CC NEG
00579A FA39 39                RTS
00580A FA3A DE D9    A INPUT9 LDX    TEMPA    SET REG X=# READ
00581A FA3C D6 DB    A INPUTC LDAB   TEMP     SET REG B=# HEX CHAR READ
00582A FA3E 39                RTS

00584                    ***************** INHEX *****************
00585                    * INPUT 1 HEX CHAR, CONVERT TO HEX
00586                    * RETURN HEX IN REG A
00587                    * REG B = 0 CONVERT A-F TO HEX
00588                    * REG B < 0 LEAVE A-F ALPHA
00589A FA3F BD F806  A INHEX  JSR    INCHNP   (INHEX) MUST BE NEG
00590A FA42 81 30    A        CMPA   #'0
00591A FA44 2B F1 FA37        BMI    NOTHEX   NOT HEX
00592A FA46 81 39    A        CMPA   #'9
00593A FA48 2F 0D FA57        BLE    IN1HG    GOOD
00594A FA4A 5D                TSTB            A-F NUMBERS?
00595A FA4B 2B EA FA37        BMI    NOTHEX   NO
00596A FA4D 81 41    A        CMPA   #'A
00597A FA4F 2B E6 FA37        BMI    NOTHEX   NOT HEX
00598A FA51 81 46    A        CMPA   #'F
00599A FA53 2E E2 FA37        BGT    NOTHEX   NOT HEX
00600A FA55 80 07    A        SUBA   #7
00601A FA57 84 0F    A IN1HG  ANDA   #$F
00602A FA59 5F                CLRB            AFTER FIND 0-9 CLEARR
00603A FA5A 39                RTS             GOOD HEX - RTN

00605                    ***********  MEMORY EXAMINE/CHANGE ***************
00606                    * PRINT VALUE AT (ADR), MAINTAIN PNTR
00607                    * M (ADR)(SPACE)
00608                    * (ADR)/
00609                    * (ADR) IS 1-4 HEX, NOT COUNTING LEADING ZEROES
00610                    * SUBCOMMANDS
00611                    *       (DATA) MODIFY VALUE AT CURRENT LOC
00612                    *       SP     INCR POINTER, PR VALUE AT NEXT ADR
00613                    *       ,      INCR PNTR, NO PRINT
00614                    *       LF     INCR PNTR, PR ADR & VALUE ON NEXT LINE
00615                    *       UA     DECR PNTR, PR ADR & VALUE ON NEXT LINE
00616                    *       /      PR CURRENT ADR AND VALUE
00617                    *       CR     TERMINATE MEM/EXAM COMMAND
00618A FA5B 8D 9D F9FA MEMORY BSR    VALINP
00619A FA5D 2F 52 FAB1        BLE    MERRTN   NOT HEX - ERROR
00620A FA5F DE D9    A MEM01  LDX    TEMPA    RRESET FOR ADR/
00621A FA61 81 2F    A        CMPA   #'/      / DELIMITER?
00622A FA63 27 04 FA69        BEQ    MEM02
00623A FA65 81 20    A        CMPA   #$20     SPACE?
00624A FA67 26 48 FAB1        BNE    MERRTN
00625A FA69 8D 76 FAE1 MEM02  BSR    OUT2H    PRINT VALUE
00626A FA6B DF D7    A MEM25  STX    PNTR
00627A FA6D 3C                PSHX
00628A FA6E 5F                CLRB            A-F NUMBER FLAG
00629A FA6F 8D 9C FA0D        BSR    INPUT    X=ADR
00630A FA71 38                PULX
00631A FA72 2B 3F FAB3        BMI    RETRN    IF NEG - ABORT
00632A FA74 27 07 FA7D        BEQ    MEM03    JUMP IF NOT HEX
00633A FA76 D6 DA    A        LDAB   TEMPA+1  GET LAST BYTE
00634A FA78 BD FAF7  A        JSR    STRCHK   STORE B AND CHK FOR CHG MEM
00635A FA7B 2B 36 FAB3        BMI    RETRN    ERR IN CHG MEMORY
```

```
00636A FA7D 81 0D    A MEM03  CMPA   #$D      CR?
00637A FA7F 27 32 FAB3        BEQ    RETRN    END MEM/EX?
00638                 *** X = ADR OF CURRENT BYTE
00639A FA81 81 2C    A        CMPA   #',      COMMA?
00640A FA83 26 03 FA88        BNE    MEM33
00641A FA85 08                INX             OPEN NEXT LOC, DO NOT PR
00642A FA86 20 E3 FA6B        BRA    MEM25
00643A FA88 81 20    A MEM33  CMPA   #$20     SPACE?
00644A FA8A 26 03 FA8F        BNE    MEM04
00645A FA8C 08                INX             INCR PNTR
00646A FA8D 20 DA FA69        BRA    MEM02    GO PR VALUE
00647A FA8F 81 0A    A MEM04  CMPA   #$A      LF?
00648A FA91 26 06 FA99        BNE    MEM06
00649A FA93 08                INX
00650A FA94 BD FB1B  A        JSR    PCR      OUT CR, NO LF
00651A FA97 20 0D FAA6        BRA    MEM12    PR ADDR,SPACE
00652A FA99 81 5E    A MEM06  CMPA   #$5E     UA?
00653A FA9B 26 03 FAA0        BNE    MEM08
00654A FA9D 09                DEX
00655A FA9E 20 04 FAA4        BRA    MEM10
00656A FAA0 81 2F    A MEM08  CMPA   #'/      SLASH?
00657A FAA2 26 0D FAB1        BNE    MERRTN
00658A FAA4 8D 6C FB12 MEM10  BSR    PCRLF    PR CR/LF
00659A FAA6 DF D7    A MEM12  STX    PNTR     SAVE NEW PNTR ADR
00660          FAA8   A MEMSL  EQU    *        FOUND / AS INSTR
00661A FAA8 CE 00D7  A        LDX    #PNTR    X PNT TO PR OBJECT
00662A FAAB 8D 2B FAD8        BSR    OUT4HS   ADR,SP
00663A FAAD DE D7    A        LDX    PNTR     RESET X TO PNTR
00664A FAAF 20 B8 FA69        BRA    MEM02
00665                 *
00666A FAB1 86 FF    A MERRTN LDAA   #$FF     SET CC NEG FOR RTN
00667A FAB3 39                RETRN  RTS

00669                 ********** OFFSET **********
00670                 *0 (ADR) CALCULATES OFFSET FROM LAST MEMORY REF
00671                 *WHICH SHOULD BE LOC OF REL ADR OF BR INSTR, TO
00672                 *THE (ADR) SPECIFIED
00673                 * IF A=0, B<80 DISTANCE CHK
00674                 * IF A=FF, B>7F
00675                 *
00676A FAB4 BD FE57  A OFFSET JSR    RD2ADR   READ 2 ADDR
00677A FAB7 DC D9    A        LDD    TEMPA
00678A FAB9 83 0001  A        SUBD   #1
00679A FABC 93 D7    A        SUBD   PNTR     OFFSET=TO-(FROM+1)
00680A FABE C1 7F    A        CMPB   #$7F     CHK IF VALID DISTANCE
00681A FAC0 22 05 FAC7        BHI    OFF4
00682A FAC2 4D                TSTA            POSITIVE DISTANCE?
00683A FAC3 27 06 FACB        BEQ    OFF6
00684A FAC5 20 EA FAB1        BRA    MERRTN
00685A FAC7 81 FF    A OFF4   CMPA   #$FF     NEG DISTANCE
00686A FAC9 26 E6 FAB1        BNE    MERRTN
00687A FACB D7 DB    A OFF6   STAB   TEMP     PR OFFSET
00688A FACD 8D 43 FB12        BSR    PCRLF    PR LF AFTER USER CR
00689A FACF CE 00DB  A        LDX    #TEMP
00690A FAD2 8D 07 FADB        BSR    OUT2HS
00691A FAD4 8D 3C FB12        BSR    PCRLF
00692A FAD6 20 D0 FAA8        BRA    MEMSL    GO TO / ROUTINE
```



```
00694                 ********** OUT4HS **********
00695                 * PRINT 2 BYTES AND SPACE
00696                 * REG X - ADR OF 1ST BYTE
00697                 * X WILL BE INCREMENTED BY 1
00698A FAD8 8D 07 FAE1 OUT4HS BSR    OUT2H
00699A FADA 08                INX             GET NEXT BYTE
00700                 * FALL THRU OUT2HS

00702                 ********** OUT2HS **********
00703                 * PRINT 1 BYTE AND SPACE
00704                 * REG X - ADR OF BYTE
00705A FADB 8D 04 FAE1 OUT2HS BSR    OUT2H    1 BYTE
00706A FADD 86 20    A SPACE  LDAA   #$20     PR SPACE
00707A FADF 20 3A FB1B        BRA    XOUTCH   PR 1 CHAR & RTN

00709                 ********** OUT2H **********
00710                 * PRINT 1 BYTE
00711                 * REG X - ADR OF BYTE
00712A FAE1 A6 00    A OUT2H  LDAA   0,X
00713A FAE3 36                PSHA            READ BYTE ONLY ONCE
00714A FAE4 8D 03 FAE9        BSR    OUTHL
00715A FAE6 32                PULA
00716A FAE7 20 04 FAED        BRA    OUTHR    RIGHT

00718                 ********** OUTHL **********
00719                 * CONVERT LEFT 4 BITS OF BYTE TO DISPLAY
00720A FAE9 44                OUTHL  LSRA             OUTPUT LEFT 4 BINARY BITTS
00721A FAEA 44                LSRA
00722A FAEB 44                LSRA
00723A FAEC 44                LSRA

00725                 ********** OUTHR **********
00726                 * CONVERT RIGHT 4 BITS OF BYTE AND PRINT
00727A FAED 84 0F    A OUTHR  ANDA   #$F       OUTPUT RIGHT 4 BITS
00728A FAEF 8B 90    A        ADDA   #$90      CONVERT TO DISPLAY
00729A FAF1 19                DAA
00730A FAF2 89 40    A        ADCA   #$40
00731A FAF4 19                DAA
00732A FAF5 20 24 FB1B        BRA    XOUTCH   PR 1 CHAR & RTN

00734                 ********** STRCHK **********
00735                 * STORE B AT 0,X & VERIFY STORE *****
00736                 * DETECTS NON-EXISTENT MEMORY, ROM, PROTECTED RAM
00737A FAF7 E7 00    A STRCHK STAB   0,X       STORE B
00738A FAF9 E1 00    A        CMPB   0,X       VERIFY MEMORY CHG
00739A FAFB 27 B6 FAB3        BEQ    RETRN     OK
00740A FAFD CE FF9B  A        LDX    #NOCHG    MSG
00741A FB00 8D 0C FB0E        BSR    PDATA
00742A FB02 20 AD FAB1        BRA    MERRTN    SET CC NEG
00743                 *
00744                 ********** PDATA1 **********
00745                 * PRINT DATA STRING
00746                 * REG X POINTS TO PRINT ARRAY
00747                 * X WILL BE INCREMENTED
00748A FB04 8D 15 FB1B PDATA2 BSR    XOUTCH   CALL OUTPUT ROUTINE
00749A FB06 08                INX             X=ADR OF OUTPUT ARRAY
00750A FB07 A6 00    A PDATA1 LDAA   0,X       GET CHAR
00751A FB09 81 04    A        CMPA   #4        EOT?
```

```
00752A FB0B 26 F7 FB04          BNE     PDATA2
00753A FB0D 39                  RTS

00755                   ***************** PDATA ****************
00756                   * CR/LF THEN PRINT DATA STRING
00757A FB0E 8D 02 FB12 PDATA    BSR     PCRLF     CR/LF, DATA STRING
00758A FB10 20 F5 FB07          BRA     PDATA1

00760                   ***************** PCRLF ****************
00761                   * OUTPUT CR/LF
00762                   * SAVE, RESTORE REG X
00763A FB12 86 0A     A PCRLF   LDAA    #$A       OUTPUT LF
00764A FB14 8D 05 FB1B          BSR     XOUTCH    PR & RTN

00766A FB16 86 0D     A PCR     LDAA    #$D       DO CR
00767A FB18 8D 01 FB1B          BSR     XOUTCH    PR & RTN
00768A FB1A 4F                  CLRA
00769A FB1B 7E F809   A XOUTCH  JMP     OUTCH     OUTPUT & RTN

00771                   ************ PRINT REGISTERS ***********
00772                   * PR REGISTERS ACROSS PAGE
00773                   * PR 2ND LINE REG, READING INPUT
00774                   *       SPACE - PR CONTENTS REG, GO TO NEXT REG
00775                   *       HEX,SP - MODIFY REG, GO TO NEXT REG
00776                   *       HEX,CR - MODIFY REG, RTN
00777                   *       CR OR OTHER COMBINATION - NO CHG, RTN
00778A FB1E 8D 66 FB86 REGSTR BSR     PREGS1
00779A FB20 8D F0 FB12          BSR     PCRLF     CR/LF AFTER REG PRINT
00780A FB22 CE 00DD   A REGS1   LDX     #SAVSTK   PSEUDO REGS
00781A FB25 5F                  CLRB              INIT OFFSET
00782A FB26 3C          REGS2   PSHX              SAVE REG PNTR
00783A FB27 CE FFB5   A         LDX     #ARRAY    CONTAINS REG NAMES
00784A FB2A 3A                  ABX               ADD OFFSET
00785A FB2B A6 00     A         LDAA    0,X       GET CURRENT REG
00786A FB2D 8D 62 FB91          BSR     OUTDA     PR REG NAME, DASH
00787A FB2F A6 01     A         LDAA    1,X       #BYTES FLAG
00788A FB31 38                  PULX              REG PNTR
00789A FB32 7D 00D0   A         TST     CT        PRINT OR MOD?
00790A FB35 27 0B FB42          BEQ     REGS3     MODIFY
00791A FB37 4D                  TSTA              CHK # BYTES
00792A FB38 27 03 FB3D          BEQ     REGS4
00793A FB3A 8D A5 FAE1          BSR     OUT2H     PR 2 HEX DIGITS
00794A FB3C 08                  INX
00795A FB3D 8D 9C FADB REGS4    BSR     OUT2HS    PR 2 HEX + SP_
00796A FB3F 08                  INX
00797A FB40 20 04 FB46          BRA     REGS6
00798A FB42 37          REGS3   PSHB              SAVE OFFSET
00799A FB43 8D 08 FB4D          BSR     INDAT     GO READ INPUT
00800A FB45 33                  PULB              RETRIEVE OFFSET
00801A FB46 CB 02     A REGS6   ADDB    #2        UPDATE
00802A FB48 C1 0C     A         CMPB    #12       ALL REG CHKED
00803A FB4A 26 DA FB26          BNE     REGS2     NO - LOOP
00804A FB4C 39                  RTS

00806                   *************** INDAT ***************
00807                   * INPUT FOR REG MODIFICATION
00808A FB4D 36          INDAT   PSHA              SAVE LEN FLG
00809A FB4E 3C                  PSHX              REG PNTR ADR
```



```
00810A FB4F BD FA0D   A         JSR     INPUT
00811A FB52 38                  PULX              RESTORE
00812A FB53 33                  PULB
00813A FB54 2B 1B FB71          BMI     PRERR     ABORT
00814A FB56 27 1D FB75          BEQ     INDAT2    NOT HEX
00815A FB58 BD F9EB   A         JSR     TERM      ACCEPT SP , CR
00816A FB5B 26 14 FB71          BNE     PRERR     RTN TO MAIN
00817A FB5D 5D                  TSTB              CHK LENGTH FLAG
00818A FB5E 27 09 FB69          BEQ     INDAT0
00819A FB60 36                  PSHA              SAVE LAST CHAR READ
00820A FB61 DC D9     A         LDD     TEMPA     GET 2 BYTE READ IN
00821A FB63 ED 00     A         STD     0,X
00822A FB65 32                  PULA              RESTORE LAST CHAR
00823A FB66 08                  INX               INCR REG PNTR
00824A FB67 20 04 FB6D          BRA     INDAT5
00825A FB69 D6 DA     A INDAT0  LDAB    TEMPA+1   1 BYTE CHANGE
00826A FB6B E7 00     A         STAB    0,X
00827A FB6D 81 0D     A INDAT5  CMPA    #$D       CR - RTN
00828A FB6F 26 13 FB84          BNE     INDAT1
00829A FB71 38          PRERR   PULX              POP RTN ADR
00830A FB72 33                  PULB              REMOVE FLAG FROM STK
00831A FB73 4F                  CLRA              NO BELL ON RETURN
00832A FB74 39                  RTS               RTN TO MAIN
00833A FB75 81 20     A INDAT2  CMPA    #$20      NO HEX, SPACE
00834A FB77 26 F8 FB71          BNE     PRERR     RTN TO MAIN
00835A FB79 5D                  TSTB              2 OR 4 CHAR
00836A FB7A 26 05 FB81          BNE     INDAT4
00837A FB7C BD FADB   A         JSR     OUT2HS    PR 2 CHAR,SPACE
00838A FB7F 20 03 FB84          BRA     INDAT1
00839A FB81 BD FAD8   A INDAT4  JSR     OUT4HS    PR 4 CHAR, SPACE
00840A FB84 08          INDAT1  INX               ADJUST REG PNTR
00841A FB85 39                  RTS

00843                   ***************** PREGS ****************
00844                   * PRINT REGS - P,X,A,B,C,S
00845A FB86 8D 8A FB12 PREGS1 BSR     PCRLF
00846A FB88 7C 00D0   A PREGS   INC     CT        SET FLAG-PRT REG
00847A FB8B 8D 95 FB22          BSR     REGS1     GO PRINT
00848A FB8D 7F 00D0   A         CLR     CT        RESET FLAG
00849A FB90 39                  RTS

00851                   *************** OUTDA ***************
00852                   * PRINT REG A, -
00853A FB91 8D 02 FB95 OUTDA    BSR     ZOUTCH    OUTPUT REG A
00854A FB93 86 2D     A         LDAA    #'-       DASH
00855A FB95 7E F809   A ZOUTCH  JMP     OUTCH

00857                   ********** BRKPNT **********
00858                   * COME HERE AFTER RECOGNIZE B<DELIM>
00859                   * B     DISPLAY ALL
00860                   * B -   REMOVE ALL
00861                   * B <ADR> INSERT BRKPNT
00862                   * B -<ADR> REMOVE BRKPNT
00863A FB98 81 0D     A BRKPNT CMPA   #$D       CR?
00864A FB9A 27 2D FBC9          BEQ     PRBRK     PRINT
00865A FB9C 81 2D     A         CMPA    #'-       DELETE?
00866A FB9E 27 4C FBEC          BEQ     DELBRK
00867A FBA0 BD F9FA   A         JSR     VALINP
```

```
00868A FBA3 2B 44 FBE9        BMI    GOX2     ABORT?
00869A FBA5 26 06 FBAD        BNE    BP02     HEX?
00870A FBA7 81 2D    A        CMPA   #'-      DELETE
00871A FBA9 27 41 FBEC        BEQ    DELBRK
00872A FBAB 20 3C FBE9        BRA    GOX2     ERR IF NOT DEL
00873A FBAD 81 0D    A BP02   CMPA   #$D      CR
00874A FBAF 26 38 FBE9        BNE    BERRTN   ERROR RTN
00875A FBB1 8D 26 FBD9        BSR    BRKTAB   IN TABL
00876A FBB3 27 14 FBC9        BEQ    PRBRK    YES - OK RTN
00877A FBB5 CE 00E6  A        LDX    #BKADR
00878A FBB8 EC 00    A BP04   LDD    0,X
00879A FBBA 27 09 FBC5        BEQ    BP06     AVAIL SP?
00880A FBBC 08                INX             CHK NEXT POSN
00881A FBBD 08                INX
00882A FBBE 8C 00EE  A        CPX    #OPCODE  END TABL?
00883A FBC1 26 F5 FBB8        BNE    BP04
00884A FBC3 20 24 FBE9        BRA    BERRTN   NO AVAIL SP
00885A FBC5 DC D9    A BP06   LDD    TEMPA    GET ADR
00886A FBC7 ED 00    A        STD    0,X      STORE IN TABLE
00887                 * FALL THRU AND PR BRKPNTS
00888                 * PRINT BREAKPOINTS
00889A FBC9 BD FB12  A PRBRK   JSR    PCRLF
00890A FBCC CE 00E6  A        LDX    #BKADR
00891A FBCF C6 04    A        LDAB   #4
00892A FBD1 BD FAD8  A PRBRK2  JSR    OUT4HS
00893A FBD4 08                INX             INCR PNTR TO BRKPNTS
00894A FBD5 5A                DECB
00895A FBD6 26 F9 FBD1        BNE    PRBRK2
00896A FBD8 39                RTS

00898                 * SEARCH BREAKPOINT TABLE
00899                 * RETURN -1 IF BRKPNT NOT IN TABL
00900                 * OTHERWISE REG X POINT TO BRKPNT IN TABL
00901A FBD9 CE 00E6  A BRKTAB LDX    #BKADR
00902A FBDC DC D9    A TAB1   LDD    TEMPA    GET PC
00903A FBDE A3 00    A        SUBD   0,X
00904A FBE0 27 09 FBEB        BEQ    BRTN
00905A FBE2 08                INX
00906A FBE3 08                INX
00907A FBE4 8C 00EE  A        CPX    #OPCODE  CMPAR TO END TABLE
00908A FBE7 26 F3 FBDC        BNE    TAB1
00909         FBE9    A GOX2   EQU    *        ERROR RETURN ENTRY FROM G
00910A FBE9 86 FF    A BERRTN LDAA   #$FF
00911A FBEB 39                BRTN   RTS

00913                 * DELETE BRKPNT
00914A FBEC BD F9FA  A DELBRK JSR    VALINP
00915A FBEF 2B F8 FBE9        BMI    BERRTN   ABORT OR ERR?
00916A FBF1 81 0D    A        CMPA   #$D      CR?
00917A FBF3 26 F4 FBE9        BNE    BERRTN
00918A FBF5 5D                TSTB            HEX?
00919A FBF6 26 0D FC05        BNE    DBRK6    JMP IF SO
00920A FBF8 CE 00E5  A        LDX    #BKADR-1
00921A FBFB C6 0C    A        LDAB   #12      0 BRKPNT TABLE
00922A FBFD 08                DBRK2  INX
00923A FBFE 6F 00    A        CLR    0,X
00924A FC00 5A                DECB
00925A FC01 26 FA FBFD        BNE    DBRK2
```



```
00926A FC03 20 C4 FBC9        BRA    PRBRK
00927                 * DELETE 1 BRKPNT
00928A FC05 8D D2 FBD9 DBRK6  BSR    BRKTAB
00929A FC07 26 E0 FBE9        BNE    BERRTN
00930A FC09 ED 00    A        STD    0,X      D=0 FROM BRKTAB
00931A FC0B 6F 08    A        CLR    8,X      CLR OP CODE
00932A FC0D 20 BA FBC9        BRA    PRBRK

00934                 ********** CALL **********
00935                 * CALL USER ROUTINE AS SUBR
00936                 * USER RTS RETURNS TO MONITOR
00937                 * STK PNTR NOT GOOD ON RETURN
00938                 * C (ADR) (CR) OR C (CR)
00939A FC0F 97 F7    A CALL   STAA   CALLF    SET FLAG * 0

00941                 ********** G **********
00942                 * GO EXECUTE USER CODE
00943                 * G(CR) OR G (ADR)
00944A FC11 81 0D    A GOXQT  CMPA   #$D      CR
00945A FC13 27 0E FC23        BEQ    GOX6     XQT FROM CURRENT PC
00946A FC15 BD F9FA  A        JSR    VALINP
00947A FC18 2F CF FBE9        BLE    GOX2
00948A FC1A 81 0D    A        CMPA   #$D      CR?
00949A FC1C 26 CB FBE9        BNE    GOX2     ERR
00950A FC1E 7F 00F5  A        CLR    EXONE    SEE BRKPNT, IF ANY
00951A FC21 DF DD    A        STX    SAVSTK   SET USER PC
00952A FC23 BD FB12  A GOX6   JSR    PCRLF
00953A FC26 96 F7    A        LDAA   CALLF    CALL CMD?
00954A FC28 27 0E FC38        BEQ    GOX7     NO
00955A FC2A 7F 00F7  A        CLR    CALLF
00956A FC2D DE E4    A        LDX    SPSAVE   GET USER STK
00957A FC2F CC FC58  A        LDD    #CRTS    RTN TO MONITOR ADR
00958A FC32 09                DEX
00959A FC33 ED 00    A        STD    0,X      STOR ON USER STK
00960A FC35 09                DEX             ADJUST USER STK
00961A FC36 DF E4    A        STX    SPSAVE   RESAVE STK
00962                 * NOW GO XQT USER SUBR
00963A FC38 96 F5    A GOX7   LDAA   EXONE    STOPPED ON BRKPNT
00964A FC3A 26 03 FC3F        BNE    GOX8
00965A FC3C BD FD36  A        JSR    SETB
00966A FC3F 20 15 FC56 GOX8   BRA    BARMS

00968                 **********  (PERIOD) **********
00969                 * TRACE 1 INSTRUCTION
00970A FC41 CE 0001  A NEXT   LDX    #1
00971A FC44 20 09 FC4F        BRA    TRACE2

00973                 ********** T **********
00974                 * T (HEX) - TRACE (HEX) INSTTR
00975A FC46 81 0D    A TRACE  CMPA   #$D      T(CR) ? - TRACE 1
00976A FC48 27 F7 FC41        BEQ    NEXT
00977A FC4A BD FA0D  A        JSR    INPUT    GET (HEX)
00978A FC4D 2F 9A FBE9        BLE    GOX2     RTN IF ABORT OR NOT HEX
00979A FC4F DF F3    A TRACE2 STX    NTRACE   STORE (HEX)
00980A FC51 27 96 FBE9        BEQ    GOX2     RTN IF TRACE = 0
00981A FC53 7C 00F5  A        INC    EXONE    XQT 1 INSTR
00982A FC56 20 62 FCBA BARMS  BRA    ARMSTK
```

```
00984                    ********** CALL SUBR **********
00985                    * ENTRY AFTER C COMMAND, AFTER XQT USER RTS
00986                    * SAVE USER REGISTERS
00987                    * PRINT REGISTERS
00988                    * RETURN TO ROUTINE CALLING C COMMAND ROUTINE
00989A FC58 36           CRTS   PSHA            SAVE TO GET CC
00990A FC59 07                  TPA
00991A FC5A 97 E3     A         STAA   SAVSTK+6 CC
00992A FC5C 32                  PULA
00993A FC5D 9F E4     A         STS    SPSAVE   STK PNTR
00994A FC5F 8E 00CF   A         LDS    #STACK
00995A FC62 DD E1     A         STD    SAVSTK+4 A,B
00996A FC64 DF DF     A         STX    SAVSTK+2 X
00997A FC66 CE FC58   A         LDX    #CRTS    PC PNT TO MONITOR
00998A FC69 DF DD     A         STX    SAVSTK
00999A FC6B BD FD15   A         JSR    RBRK     REMOVE BRKPNTS
01000A FC6E 7E FD7A   A         JMP    ENDCAL   GO PR REGS, 0 EXONE

01002                    * SETCLK - USED BY ON-CHIP CLOCK
01003                    * FOR HARDWARE TRACE
01004                    * SET TIMER TO COMPARE AFTER 1 CYCLE OF USER INSTR
01005A FC71 C6 18     A SETCLK LDAB   #$18     SET #CYCLES
01006A FC73 DE 09     A         LDX    CLOCK    GET CLOCK TIME
01007A FC75 3A                  ABX             ADD # CYCLES
01008A FC76 DF 0B     A         STX    OCREG    STORE IN COMPARE REG
01009A FC78 39                  RTS

01011                    ********** NMI ENTRY **********
01012                    * ENTER FROM XQT 1 INSTR - TRACE OR XQT OVER BRKPNT
01013                    * MOVE REGS FROM USER STK TO MONITOR STORAGE
01014                    * REPLACE BRKPNTS WITH USER CODE
01015                    * IF NOT TRACING, REPLACE CODE WITH BRKPNTS (3F)
01016                    * IF TRACING, PRINT REGISTERS
01017                    *              EXECUTE NEXT USER INSTR
01018                    * ENTRY FOR ONCHIP CLOCK TRACE
01019A FC79 7C 0008   A C.NMI  INC    TCSR     BRING LEVEL HIGH
01020A FC7C 8D F3 FC71         BSR    SETCLK   NO NMI, BUT LEVEL CHG

01022                    * ENTRY FOR PTM HARDWARE TRACE
01023A FC7E 30           M.NMI  TSX             TRANSFER STK PNTR
01024A FC7F 8E 00CF   A         LDS    #STACK
01025A FC82 8D 79 FCFD          BSR    MOVSTK   SAVE USER REGS
01026A FC84 BD FD15   A         JSR    RBRK     REMOVE BRKPNT
01027A FC87 DE F3     A         LDX    NTRACE   TRACE?
01028A FC89 26 0A FC95          BNE    NMI01
01029A FC8B 7F 00F5   A         CLR    EXONE
01030A FC8E BD FD36   A         JSR    SETB
01031A FC91 2B 24 FCB7          BMI    NMI03
01032A FC93 20 25 FCBA          BRA    ARMSTK
01033A FC95 09           NMI01  DEX
01034A FC96 DF F3     A NMI015 STX    NTRACE
01035A FC98 26 03 FC9D          BNE    NMI02
01036A FC9A 7F 00F5   A         CLR    EXONE
01037                    * PRINT TRACE LINE:
01038                    * OP-XX P-XXXX X-XXXX A-XX B-XX C-XX S-XXXX
01039                    * CHECK IF USER HIT CONTROL X TO TERMINATE TRACE
01040A FC9D CE 0000   A NMI02  LDX    #0       CLR TRACE & EXONE IF TERMINATE
01041A FCA0 BD FDD7   A         JSR    CHKABT
```



```
01042A FCA3 27 F1 FC96          BEQ    NMI015   TERMINT IF = CNTL X
01043A FCA5 CE FFB1   A         LDX    #PRTOP   GET ADR OF OP-
01044A FCA8 BD FB0E   A         JSR    PDATA
01045A FCAB DE D9     A         LDX    TEMPA    GET OLD PC
01046A FCAD BD FADB   A         JSR    OUT2HS   PR OPCODE
01047A FCB0 BD FB88   A         JSR    PREGS    PR TRACE LINE
01048A FCB3 96 F5     A         LDAA   EXONE
01049A FCB5 26 03 FCBA          BNE    ARMSTK
01050A FCB7 7E F952   A NMI03  JMP    MAIN
01051                    * STACK USER REGISTERS
01052                    * MOVE FROM MONITOR STORAGE TO USER STACK
01053                    * IF TRACE - SET HARDWARE
01054A FCBA 9E E4     A ARMSTK LDS    SPSAVE   SET STK FOR RTI
01055A FCBC DE DD     A         LDX    SAVSTK   PC
01056A FCBE 3C                  PSHX
01057A FCBF DE DF     A         LDX    SAVSTK+2 X
01058A FCC1 3C                  PSHX
01059A FCC2 DC E1     A         LDD    SAVSTK+4 GET A, B
01060A FCC4 36                  PSHA            MOVE TO STK
01061A FCC5 37                  PSHB
01062A FCC6 96 E3     A         LDAA   SAVSTK+6 GET CC
01063A FCC8 36                  PSHA
01064A FCC9 96 F5     A         LDAA   EXONE
01065A FCCB 27 1C FCE9          BEQ    ARMS04
01066A FCCD DE DD     A         LDX    SAVSTK   SAVE PC PNTR FOR NXT TRACE PRT
01067A FCCF DF D9     A         STX    TEMPA
01068                    * CHECK IF USE PTM OR ON-CHIP CLOCK
01069A FCD1 8D 17 FCEA          BSR    IFPTM
01070A FCD3 27 0D FCE2          BEQ    SETPTM   GO USE PTM
01071                    * IF USER ISSUE TRACE COMMAND AND
01072                    * NOT USING PTM - ASSUME ON-CHIP
01073A FCD5 86 02     A         LDAA   #2       SET DDR FOR OUTPUT
01074A FCD7 97 01     A         STAA   P2DDR    PORT 2
01075A FCD9 D6 08     A         LDAB   TCSR     SET UP FOR ON-CHIP CLOCK
01076A FCDB C4 FE     A         ANDB   #$FE     CLEAR OLVL BIT
01077A FCDD D7 08     A         STAB   TCSR
01078A FCDF 8D 90 FC71          BSR    SETCLK   SET CMPR REG, WAIT FOR CMPR
01079                    FCE1 A DUMMY  EQU    *        INTERRUPT VECTORS USE THIS
01080A FCE1 3B                  RTI

01082                    * SET HARDWARE FOR PTM
01083                    * INITIATE COUNTER
01084A FCE2 CC 0501   A SETPTM LDD    #$0501   M=5,L=1 TURN ON TRACE
01085A FCE5 DE F8     A         LDX    PTM      GET ADR OF PTM
01086A FCE7 ED 02     A         STD    2,X      STORE AT PTM ADR +2
01087A FCE9 3B           ARMS04 RTI

01089                    * CHECK NMI VECTOR
01090                    * DETERMINE IF USE ON-CHIP CLOCK OR PTM
01091                    *    FOR HARDWARE TRACE
01092A FCEA CE FFF0   A IFPTM  LDX    #VECTR   GET ADR OF VECTORS
01093A FCED 96 03     A         LDAA   MODE     EXTERNAL VECTRS?
01094A FCEF 84 E0     A         ANDA   #$E0     CHK 3 MSB
01095A FCF1 81 20     A         CMPA   #$20     MODE 1?
01096A FCF3 27 02 FCF7          BEQ    IFPTM2
01097A FCF5 DE FE     A         LDX    VECPTR   GET VECTOR TABLE
01098A FCF7 EE 0C     A IFPTM2 LDX    $C,X     GET NMI ADDRESS
01099A FCF9 8C F800   A         CPX    #EX.NMI  PTM ENTRY?
```

```
01100A FCFC 39                RTS             RETURN WITH CC SET

01102                  ********** MOVSTK **********
01103                  * MOVE USER REGS FROM USER STACK TO MONITOR STORAGE
01104                  * RESET USER STACK POINTER
01105A FCFD A6 00    A MOVSTK LDAA   0,X      MOVE C,B,A,X,PC
01106A FCFF 97 E3    A        STAA   SAVSTK+6 TO PC,X,A,B,C
01107A FD01 EC 01    A        LDD    1,X
01108A FD03 97 E2    A        STAA   SAVSTK+5
01109A FD05 D7 E1    A        STAB   SAVSTK+4
01110A FD07 EC 03    A        LDD    3,X
01111A FD09 DD DF    A        STD    SAVSTK+2
01112A FD0B EC 05    A        LDD    5,X
01113A FD0D DD DD    A        STD    SAVSTK
01114A FD0F C6 06    A        LDAB   #6
01115A FD11 3A                ABX
01116A FD12 DF E4    A        STX    SPSAVE
01117A FD14 39                RTS

01119                  ********** RBRK **********
01120                  * REPLACE BRKPNTS (SWI) WITH USER CODE
01121                  * BKADR - TABLE OF 4 BRKPNT ADR
01122                  * OPCODE - TABLE OF OPCODES, CORRESPOND TO ADR
01123A FD15 96 F2    A RBRK   LDAA   BRKFLG   IGNORE IF BRKPNTS NOT IN
01124A FD17 27 1C FD35        BEQ    RBRK6
01125A FD19 CE 00E6  A        LDX    #BKADR   GET TABLE OF ADR
01126A FD1C C6 08    A        LDAB   #NUMBP*2 INDEX INTO OPCODE TABLE
01127A FD1E 3C        RBRK2   PSHX            SAVE TABLE ADR
01128A FD1F 3C                PSHX
01129A FD20 3A                ABX
01130A FD21 A6 00    A        LDAA   0,X      GET OPCODE
01131A FD23 38                PULX
01132A FD24 EE 00    A        LDX    0,X      GET USER BRKPNT ADR
01133A FD26 27 02 FD2A        BEQ    RBRK3    NO ADR
01134A FD28 A7 00    A        STAA   0,X      RESTORE OPCODE
01135A FD2A 38        RBRK3   PULX            GET NXTT ADR FROM TABL
01136A FD2B 08                INX
01137A FD2C 08                INX
01138A FD2D 5A                DECB            ADJUST OPCODE INDEX
01139A FD2E C1 04    A        CMPB   #NUMBP   END TABLE?
01140A FD30 26 EC FD1E        BNE    RBRK2
01141A FD32 7F 00F2  A        CLR    BRKFLG   CLR BRKPNT FLAG
01142A FD35 39        RBRK6   RTS

01144                  ********** SETB **********
01145                  * REPLACE USER CODE WITH 3F AT BRKPNT ADDRESSES
01146                  * IGNORE IF BREAKPOINTS ALREADY IN
01147A FD36 96 F2    A SETB   LDAA   BRKFLG   ALREADY IN?
01148A FD38 26 70 FDAA        BNE    SHERR    SET NEG RETURN
01149A FD3A CE 00E6  A        LDX    #BKADR
01150A FD3D C6 08    A        LDAB   #NUMBP*2 SET INDEX INTO OPCODES
01151A FD3F 3C        SETB2   PSHX            SAVE ADR PNTR
01152A FD40 EE 00    A        LDX    0,X      GET USER BRKPNT ADR
01153A FD42 27 10 FD54        BEQ    SETB4    SKIP IF NO ADR
01154A FD44 A6 00    A        LDAA   0,X      GET OPCODE
01155A FD46 37                PSHB            SAVE OPCODE INDEX
01156A FD47 C6 3F    A        LDAB   #$3F     SET SWI
01157A FD49 BD FAF7  A        JSR    STRCHK   STORE & CHK CHG
```



```
01158A FD4C 33                PULB            INDEX
01159A FD4D 38                PULX            ADR TABLE PNTR
01160A FD4E 2B 0E FD5E        BMI    SETB6    3F STORED GOOD?
01161A FD50 3C                PSHX            RESAVE TABLE PNTR
01162A FD51 3A                ABX             CALCLATE OP POS IN TABLE
01163A FD52 A7 00    A        STAA   0,X      SAVE OPCODE
01164A FD54 38        SETB4   PULX            GET TABLE ADR
01165A FD55 08                INX
01166A FD56 08                INX             GET NXT ADT
01167A FD57 5A                DECB            ADJUST OPCODE INDEX
01168A FD58 C1 04    A        CMPB   #NUMBP   END TABLE?
01169A FD5A 26 E3 FD3F        BNE    SETB2    LOOP IF NOT
01170A FD5C D7 F2    A        STAB   BRKFLG   SET BRKPNT FLAG
01171A FD5E 39        SETB6   RTS


01174                  ********** SWI ENTRY **********
01175                  * ENTER WITH BRKPOINT SETTING
01176                  * SAVE USER REGISTERS
01177                  * DECR PC TO POINT AT SWI
01178                  * REPLACE SWI'S WITH USER CODE
01179                  * PRINT REGISTERS
01180                  * GO TO MAIN CONTROL LOOP
01181A FD5F 30        M.SWI   TSX             GET USER STK
01182A FD60 8E 00CF  A        LDS    #STACK   SET TO INTERNAL STK
01183A FD63 8D 98 FCFD        BSR    MOVSTK   SAVE USER REGS
01184A FD65 DE DD    A        LDX    SAVSTK   DECR USER PC
01185A FD67 09                DEX
01186A FD68 DF DD    A        STX    SAVSTK
01187A FD6A DF D9    A        STX    TEMPA    SAVE FOR BRKTAB CHK
01188A FD6C 96 F2    A        LDAA   BRKFLG   ERR IF NOT BRKPOINT
01189A FD6E 27 0A FD7A        BEQ    BKPERR
01190A FD70 8D A3 FD15        BSR    RBRK     REMOVE BRKPNT FROM CODE
01191A FD72 BD FBD9  A        JSR    BRKTAB   BRKPNT IN TABLE?
01192A FD75 26 03 FD7A        BNE    BKPERR
01193                  * REG A = 0 IF BRKTAB FIND BRKPNT
01194A FD77 4C                INCA
01195A FD78 20 04 FD7E        BRA    SWI3
01196                  * ENTRY FROM CRTS - PR REGS, RTN TO MAIN
01197          FD7A  A ENDCAL EQU    *
01198A FD7A 4F        BKPERR  CLRA
01199A FD7B 5F                CLRB
01200A FD7C DD F3    A        STD    NTRACE   RESET NUM INSTR TO TRACE
01201A FD7E 97 F5    A SWI3   STAA   EXONE    CLEAR XQT 1 INSTR
01202A FD80 BD FB86  A        JSR    PREGS1
01203A FD83 7E F952  A        JMP    MAIN     GO TO MAIN LOOP

01205                  ********** DISPLAY **********
01206                  * D   OR D <ADR>  OR D <ADR> <ADR>
01207                  * DISPLAY MEMORY - BLK OF MEMORY AROUND LAST
01208                  *   REFERENCED BYTE FROM MEM/EX
01209                  * DISPLAY 16 BYTES AROUND <ADR> SPECIFIED
01210                  * OR DISPLAY FROM <ADR> TO <ADR> MOD 16
01211                  * ASCII CHAR WILL BE PRINTED ON THE RIGHT
01212                  * MEM/EX PNTR WILL PNT TO LAST ADR REFERENCED
01213                  * AT END OF DISPLAY COMMAND
01214                  *
01215A FD86 DE D7    A DISPLY LDX    PNTR     SAVE MEMORY/EX PNTR
```

```
01216A FD88 3C                  PSHX
01217A FD89 81 0D    A          CMPA    #$D       CR?
01218A FD8B 27 0A FD97          BEQ     SHOW35    NO ARG
01219A FD8D 8D 5A FDE9          BSR     PVALIN
01220A FD8F 2F 16 FDA7          BLE     SHERR2    ERR IF NOT HEX, OR ABORT
01221A FD91 DF D7    A          STX     PNTR
01222A FD93 81 0D    A          CMPA    #$D       CR?
01223A FD95 26 16 FDAD          BNE     SHOW4
01224A FD97 DC D7    A SHOW35   LDD     PNTR      DEFINE BLK TO DMP
01225A FD99 C4 F0    A          ANDB    #$F0      MASK OUT LOW DIGIT
01226A FD9B 83 0010  A          SUBD    #$10
01227A FD9E DD D7    A          STD     PNTR
01228A FDA0 C3 0020  A          ADDD    #$20
01229A FDA3 DD D9    A          STD     TEMPA     TO ADR
01230A FDA5 20 1A FDC1          BRA     SHOW8
01231A FDA7 38               SHERR2 PULX          RESET MEM/EX PNTR
01232A FDA8 DF D7    A          STX     PNTR
01233A FDAA 86 FF    A SHERR    LDAA    #$FF
01234A FDAC 39                  RTS
01235A FDAD 8D 3A FDE9 SHOW4    BSR     PVALIN    READ HEX #
01236A FDAF 2F F6 FDA7          BLE     SHERR2    JMP IF ERR
01237A FDB1 DC D7    A          LDD     PNTR      FROM ADR < TO ADR?
01238A FDB3 C4 F0    A          ANDB    #$F0      MASK OUT LOW ORDER DIGIT
01239A FDB5 D7 D8    A          STAB    PNTR+1
01240A FDB7 93 D9    A          SUBD    TEMPA
01241A FDB9 22 EC FDA7          BHI     SHERR2
01242A FDBB 96 DA    A          LDAA    TEMPA+1   MASK TO FULL LINE
01243A FDBD 84 F0    A          ANDA    #$F0
01244A FDBF 97 DA    A          STAA    TEMPA+1   CHANGES LAST REF ADR
01245                       * TURN ON HIGH SPEED DEVICE
01246                       * CALL HIGH SPEED DATA ROUTINE TO OUTPUT
01247                       *    DATA FROM ADR IN PNTR TO ADR IN TEMPA
01248A FDC1 C6 0C    A SHOW8    LDAB    #HS.ON
01249A FDC3 BD FE54  A          JSR     IO2
01250A FDC6 CE 00D7  A          LDX     #BBLK+1   GET TRANSFER PACKET
01251A FDC9 C6 0E    A          LDAB    #HS.DTA
01252A FDCB BD F8DF  A          JSR     IO
01253A FDCE 38                  PULX              RETRIEVE MEM/EX PNTR
01254A FDCF DF D7    A          STX     PNTR
01255A FDD1 C6 10    A          LDAB    #HS.OFF
01256A FDD3 8D 7F FE54          BSR     IO2
01257A FDD5 4F                  CLRA              CLEAR CC FOR RETURN
01258A FDD6 39                  RTS

01260                       ***************** CHKABT ***************
01261                       * READ WITH NO WAIT
01262                       * CHK FOR CONTROL X - ESCAPE FROM PRINT
01263                       * CHK FOR CONTROL W - WAIT DURING T OR D PRINT
01264                       *    ANY CHARACTER CONTINUES PRINT
01265                       * ANY OTHER CHARACTER - READ & IGNORE
01266A FDD7 36              CHKABT PSHA
01267A FDD8 C6 02    A          LDAB    #CI.DTA   READ A CHAR
01268A FDDA 8D 78 FE54          BSR     IO2
01269A FDDC 84 7F    A          ANDA    #$7F      CLEAR PARITY
01270A FDDE 81 17    A          CMPA    #CNTLW    CONTROL W?
01271A FDE0 26 03 FDE5          BNE     CHK2      IF SO WAIT FOR INPUT
01272A FDE2 BD F806  A          JSR     INCHNP    TO CONTINUE PRINT
01273A FDE5 81 18    A CHK2     CMPA    #CNTLX    CONTROL X?
```



```
01274                       * RETURN WITH CC SET
01275A FDE7 32                  PULA
01276A FDE8 39              SHOW19 RTS

01278A FDE9 7E F9FA  A PVALIN JMP    VALINP    SAVE BYTES

01280                       *************** HSDTA ***************
01281                       * FROM ADR, TO ADR IN TRANSFER BLOCK
01282                       * ADR ARE DIVISIBLE BY 16
01283                       * ADR OF BLOCK WAS IN REG X
01284                       * X SAVED ON STK BY IO
01285A FDEC 30              HSDTA  TSX               GET TRANSFER PACKET
01286A FDED EE 02    A             LDX     2,X
01287A FDEF EC 00    A             LDD     0,X       GET FROM ADR
01288A FDF1 DD D7    A             STD     PNTR      SAVE FOR DUMP
01289A FDF3 EC 02    A             LDD     2,X       GET TO ADR
01290A FDF5 DD D9    A             STD     TEMPA
01291A FDF7 BD FB12  A SHOW8  JSR     PCRLF     LINE FEED
01292                       * PRINT BLOCK HEADING
01293A FDFA CE FFC1  A          LDX     #SPACE6   PR LEADING BLANKS
01294A FDFD BD FB0E  A          JSR     PDATA
01295A FE00 4F                  CLRA
01296A FE01 36              PRTTL  PSHA
01297A FE02 BD FAED  A          JSR     OUTHR     CONVERT TO DISPLAY
01298A FE05 BD FADD  A          JSR     SPACE
01299A FE08 BD FADD  A          JSR     SPACE     PR 2 SPACES
01300A FE0B 32                  PULA              GET CNTR
01301A FE0C 4C                  INCA
01302A FE0D 81 10    A          CMPA    #$10      PR 0-F
01303A FE0F 26 F0 FE01          BNE     PRTTL     FINISHED?
01304                       * CHECK IF USER WANT TO TERMINT DISPLAY CMD
01305A FE11 8D C4 FDD7 SHOW10 BSR     CHKABT
01306A FE13 27 D3 FDE8          BEQ     SHOW19    RETURN IF CONTROL X
01307A FE15 BD FB12  A          JSR     PCRLF
01308A FE18 CE 00D7  A          LDX     #PNTR     GET ADR OF LINE
01309A FE1B BD FAD8  A          JSR     OUT4HS    PRINT ADR
01310A FE1E DE D7    A          LDX     PNTR      GET CONTENTS OF MEMORY
01311A FE20 C6 10    A          LDAB    #16       CNTR FOR LINE
01312A FE22 BD FADB  A SHOW12 JSR     OUT2HS    PR DATA
01313A FE25 08                  INX               INCR ADR PNTR
01314A FE26 5A                  DECB
01315A FE27 26 F9 FE22          BNE     SHOW12    LOOP
01316A FE29 BD FADD  A          JSR     SPACE     PRINT ASCII DUMP
01317A FE2C C6 10    A          LDAB    #16       NUM CHAR/LINE
01318A FE2E DE D7    A          LDX     PNTR
01319A FE30 A6 00    A SHOW14 LDAA    0,X
01320A FE32 84 7F    A          ANDA    #$7F      CHK PRINTABLE
01321A FE34 81 20    A          CMPA    #$20
01322A FE36 2D 04 FE3C         BLT     SHOW16    NON-CHAR
01323A FE38 81 61    A          CMPA    #$61
01324A FE3A 2D 02 FE3E         BLT     SHOW18
01325A FE3C 86 2E    A SHOW16 LDAA    #'.       PR . FOR NON-CHAR
01326A FE3E BD F809  A SHOW18 JSR     OUTCH
01327A FE41 08                  INX
01328A FE42 5A                  DECB
01329A FE43 26 EB FE30          BNE     SHOW14    LOOP
01330A FE45 DC D9    A          LDD     TEMPA
01331A FE47 93 D7    A          SUBD    PNTR
```

```
01332A FE49 27 9D FDE8        BEQ     SHOW19    RETURN
01333A FE4B DF D7     A       STX     PNTR      SAVE  FROM ADR
01334A FE4D 7D 00D8   A       TST     PNTR+1
01335A FE50 26 BF FE11        BNE     SHOW10    END OF LINE
01336A FE52 20 A3 FDF7        BRA     SHOW9     END OF BLOCK

01338                 * IO CALL - TO SAVE A FEW BYTES
01339A FE54 7E F8DF   A IO2   JMP     IO

01341                 ********** RD2ADR **********
01342                 * READ (DELIM) (ADR1) (ADR2)
01343A FE57 81 0D     A RD2ADR CMPA    #$0D      CR?
01344A FE59 27 13 FE6E        BEQ     PNCHER
01345A FE5B 8D 8C FDE9        BSR     PVALIN    CALL INPUT ROUTINE
01346A FE5D 2F 0F FE6E        BLE     PNCHER    CHK IF NUMBER
01347A FE5F DF D7     A       STX     BBLK+1    SAVE 1ST ADR (PNTR)
01348                 * INPUT CHECKS FOR DELIMITER
01349A FE61 81 0D     A       CMPA    #$D       CR?
01350A FE63 27 09 FE6E        BEQ     PNCHER    DO NOT ALLOW CR
01351A FE65 BD FDE9   A PNCH3 JSR     PVALIN    READ NEXT ADR
01352A FE68 2F 04 FE6E        BLE     PNCHER    VALID ADR?
01353A FE6A 81 0D     A       CMPA    #$D       REQUIRE CR AFTER ADR
01354A FE6C 27 03 FE71        BEQ     PNCRTN
01355A FE6E 86 FF     A PNCHER LDAA    #$FF      ERR RTN
01356A FE70 38                PULX              REMOVE SUBR CALL ADR
01357A FE71 39        PNCRTN  RTS

01359                 ********** PUNCH **********
01360                 * P (ADR1) (ADR2)
01361                 * PUNCH FROM (ADR1) TO (ADR2)
01362                 * ERROR IF (ADR2) LT (ADR1)
01363                 * SET UP TRANSFER PACKET
01364                 * 1ST WRD - FCN FOR PUNCH = 0
01365                 * 2ND, 3RD WRDS = (ADR1)
01366                 * 4TH, 5TH WRDS = (ADR2)
01367                 * LDX W/ ADR OF TRANSFER PACKET
01368                 * JMP THRU IO VECTOR TO BSDTA
01369A FE72 7F 00D6   A PUNCH  CLR     BBLK      SET BULK STR FCN
01370A FE75 8D E0 FE57        BSR     RD2ADR    READ 2 ADDRESSES
01371                 * HEX STILL IN TEMPA (BBLK+3) - END ADR
01372A FE77 BD FB12   A PNCH4  JSR     PCRLF
01373                 * SET NO ECHO FLAG/ TAPE FLAG
01374A FE7A 86 10     A       LDAA    #$10      * NULLS W/TAPE CR
01375A FE7C 97 F6     A       STAA    OUTSW
01376A FE7E C6 12     A       LDAB    #BS.ON    TURN PUNCH ON
01377A FE80 8D D2 FE54        BSR     IO2
01378A FE82 CE 00D6   A       LDX     #BBLK     ADR OF BULK STORE BLK
01379A FE85 C6 14     A       LDAB    #BS.DTA   OFFSET TO BULK ROUTINE
01380A FE87 8D CB FE54        BSR     IO2
01381A FE89 36                PSHA              SAVE FOR RETURN CC
01382A FE8A C6 16     A       LDAB    #BS.OFF   TURN OFF TAPE
01383A FE8C 8D C6 FE54        BSR     IO2
01384A FE8E BD FDD7   A       JSR     CHKABT    CLEAR IO BUF
01385A FE91 BD FDD7   A       JSR     CHKABT    DOUBLE BUF
01386A FE94 7F 00F6   A       CLR     OUTSW     TURN PR ON
01387A FE97 32                PULA
01388A FE98 4D                TSTA              SET RETURN PR
01389A FE99 39                RTS
```



```
01391                 ********** LOAD **********
01392                 * L  LOAD A TAPE FILE
01393                 * L (OFFSET)  LOAD WITH AN OFFSET
01394                 * SET FUNCTION IN BULK STORE PACKET
01395                 * IF OFFSET - 3RD, 4TH WRDS OF PACKET = OFFSET
01396                 * LDX W/ ADR OF TRANSFER PACKET
01397                 * JMP THRU IO VECTOR TO BSDTA
01398A FE9A C6 01     A LOAD   LDAB    #1        SET LOAD FCN = 1
01399A FE9C D7 D6     A LOAD2  STAB    BBLK
01400A FE9E CE 0000   A        LDX     #0        INIT OFFSET=0
01401A FEA1 DF D9     A        STX     BBLK+3
01402A FEA3 81 0D     A        CMPA    #$D       CR?
01403A FEA5 27 D0 FE77         BEQ     PNCH4     YES
01404A FEA7 8D BC FE65         BSR     PNCH3
01405A FEA9 20 CC FE77         BRA     PNCH4

01407                 ********** VERIFY **********
01408                 * V  VERIFY THAT TAPE LOADED CORRECTLY
01409                 * V (OFFSET) CHECK PROG LOADED WITH OFFSET CORRECTLY
01410                 * SET FCN IN BULK STORE PACKET
01411                 * IF OFFSET - 3RD, 4TH WRDS = OFFSET
01412                 * LDX W/ ADR OF PACKET
01413                 * JMP THRU IO VECTOR TO BSDTA
01414A FEAB C6 FF     A VERF   LDAB    #$FF
01415A FEAD 20 ED FE9C         BRA     LOAD2

01417                 ********** BSON **********
01418                 * TURN PUNCH ON FOR READ OR WRITE
01419                 * BBLK MUST BE SET - BBLK=0 WRITE
01420                 *                 BBLK#0 ON FOR READ
01421A FEAF 86 11     A BSON   LDAA    #$11      SET FOR READ
01422A FEB1 7D 00D6   A        TST     BBLK
01423A FEB4 26 01 FEB7         BNE     BSON2     JUMP IF VERF/LOAD
01424A FEB6 4C                 INCA              SET REG A=$12 FOR WRT TAPE
01425A FEB7 7E F809   A BSON2  JMP     OUTCH

01427                 ********** BSOFF **********
01428A FEBA CE FF8B   A BSOFF  LDX     #PUNOFF   TURN PUNCH OFF
01429A FEBD BD FB07   A        JSR     PDATA1    OUTPUT STRG & RTN
01430A FEC0 7E F8D8   A        JMP     DELAY     WAIT FOR PRT SYNC

01432                 ********** BSDTA **********
01433A FEC3 30        BSDTA    TSX               BULK STORE DATA
01434A FEC4 EE 02     A        LDX     2,X       GET IO PACK VECTOR
01435A FEC6 A6 00     A        LDAA    0,X       GET FCN
01436A FEC8 97 D6     A        STAA    BBLK      USED BY VERF/LOAD
01437A FECA 27 5S FF25         BEQ     BSPUN     JUMP TO PUNCH, FCN=0
01438                 * FALL THRU TO VERF-BBLK=-1, LOAD-BBLK=1

01440                 * VERIFY, LOAD
01441                 * GET OFFSET FROM IO PACKET
01442                 * FIND S1 REC - DATA
01443                 * READ BYTE CNT (TEMP)
01444                 * READ ADDRESS - SET REG X
01445                 * READ & STORE DATA, COMPUTE CHK SUM
01446                 * COMPARE TAPE TO COMPUTED CKSUM
01447A FECC EC 03     A        LDD     3,X       GET OFFSET
```

```
01448A FECE DD D7    A         STD    PNTR
01449A FED0 BD F806  A LOAD3   JSR    INCHNP    READ
01450A FED3 81 53    A LOAD4   CMPA   #'S       GET 1ST GOOD REC
01451A FED5 26 F9 FED0         BNE    LOAD3
01452A FED7 BD F806  A         JSR    INCHNP
01453A FEDA 81 39    A         CMPA   #'9
01454A FEDC 27 32 FF10         BEQ    LOAD20    FINI AFTER S9
01455A FEDE 81 31    A         CMPA   #'1       DATA REC
01456A FEE0 26 F1 FED3         BNE    LOAD4     NO
01457A FEE2 7F 00D0  A         CLR    CKSUM     INIT CHECK SUM
01458A FEE5 8D 2A FF11         BSR    BYTE      GET BYTE CNT
01459A FEE7 C0 02    A         SUBB   #2        DECR BYTE CNT FROM IT
01460A FEE9 D7 DB    A         STAB   TEMP      STORAGE FOR BYTE CNT
01461                        * READ 4 HEX DIGITS FROM INPUT
01462                        * FORM ADDRESS AND STORE IN REG X
01463A FEEB 8D 24 FF11         BSR    BYTE      1 BYTE
01464A FEED 37                 PSHB             SAVE 1ST BYTE
01465A FEEE 8D 21 FF11         BSR    BYTE      2ND BYTE
01466A FEF0 32                 PULA             GET 1ST BYTE
01467A FEF1 D3 D7    A         ADDD   PNTR      ADD OFFSET
01468A FEF3 37                 PSHB             MOVE A:B TO X
01469A FEF4 36                 PSHA
01470A FEF5 38                 PULX             SET REG X = ADR
01471                        * STORE DATA
01472A FEF6 8D 19 FF11 LOAD11  BSR    BYTE      GET BYTE IN REG B
01473A FEF8 7A 00DB  A         DEC    TEMP      DEC BYTE CNT
01474A FEFB 27 0E FF0B         BEQ    LOAD15    END REC?
01475A FEFD 7D 00D6  A         TST    BBLK      SKIP STORE IF VERF
01476A FF00 2B 02 FF04         BMI    LOAD12    JUST COMPARE
01477A FF02 E7 00    A         STAB   0,X
01478A FF04 E1 00    A LOAD12  CMPB   0,X
01479A FF06 26 06 FF0E         BNE    LOAD19    ERROR
01480A FF08 08                 INX
01481A FF09 20 EB FEF6         BRA    LOAD11
01482                        * CHECKSUMS GOOD?
01483                        * CKSUM IS ONE'S COMPLE
01484A FF0B 4C               LOAD15 INCA             CHKSUM ADDED INTO B
01485A FF0C 27 C2 FED0         BEQ    LOAD3     GET NEXT REC
01486                        * CHECKSUM ERROR, VERF FAILURE, LOAD FAIL ERR
01487A FF0E 86 FF    A LOAD19  LDAA   #$FF      SET NEG FOR ER RTN
01488A FF10 39               LOAD20 RTS

01490                        ********** BYTE **********
01491                        * FORM A HEX BYTE FROM 2 DISPLAY BYTES
01492                        * CALL INHEX TO READ 1 HEX DIGIT FROM INPUT
01493A FF11 5F               BYTE   CLRB             READ A-F AS HEX
01494A FF12 BD FA3F  A         JSR    INHEX
01495A FF15 C6 10    A         LDAB   #16
01496A FF17 3D                 MUL              LSB IN REG B
01497A FF18 37                 PSHB             SAVE
01498A FF19 5F                 CLRB             FLAG FOR INHEX
01499A FF1A BD FA3F  A         JSR    INHEX
01500A FF1D 33                 PULB
01501A FF1E 1B                 ABA              GET 1 BYTE
01502A FF1F 16                 TAB              SAVE IN B
01503A FF20 9B D0    A         ADDA   CKSUM
01504A FF22 97 D0    A         STAA   CKSUM
01505A FF24 39                 RTS
```



```
01507                        ********** BSDTA - PUNCH **********
01508                        * MOVE FROM & TO ADDRESSES TO STORAGE
01509                        * PNTR - FROM ADR   TEMPA - TO ADR
01510                        * BBLK - REUSED FOR FRAME CNT
01511                        * TEMP - REUSED FOR BYTE CNT
01512                        * PUNCH NULLS AS LEADER ON TAPE
01513                        * PUNCH CR/LF, NULL, S1(RECORD TYPE),
01514                        *       FRAME COUNT, ADDRESS, DATA, CHECKSUM
01515                        * EOF RECORD - S9030000FC
01516A FF25 EC 01    A BSPUN   LDD    1,X       GET FROM ADR
01517A FF27 DD D7    A         STD    PNTR
01518A FF29 EC 03    A         LDD    3,X       GET TO ADR
01519A FF2B DD D9    A         STD    TEMPA
01520                        * PUNCH LEADER
01521A FF2D C6 19    A         LDAB   #25
01522A FF2F 4F               PNULL  CLRA             OUTPUT NULL CHAR
01523A FF30 BD F809  A         JSR    OUTCH
01524A FF33 5A                 DECB
01525A FF34 26 F9 FF2F         BNE    PNULL     LOOP IF NOT FINI
01526A FF36 DC D9    A PUN11   LDD    TEMPA
01527A FF38 D0 D8    A         SUBB   PNTR+1
01528A FF3A 92 D7    A         SBCA   PNTR      FROM ADR < TO ADR?
01529A FF3C 26 04 FF42         BNE    PUN22
01530A FF3E C1 18    A         CMPB   #24
01531A FF40 25 02 FF44         BCS    PUN23
01532A FF42 C6 17    A PUN22   LDAB   #23       SET FRAME CNT
01533A FF44 CB 04    A PUN23   ADDB   #4
01534A FF46 D7 D6    A         STAB   BBLK
01535A FF48 C0 03    A         SUBB   #3
01536A FF4A D7 DB    A         STAB   TEMP      BYTE CNT THIS REC
01537                        * PUNCH CR/LF, NULLS,S,1
01538A FF4C CE FFA2  A         LDX    #MTAPE
01539A FF4F BD FB0E  A         JSR    PDATA
01540A FF52 5F                 CLRB             ZERO CHKSUM
01541                        * PUNCH FRAME CNT
01542A FF53 CE 00D6  A         LDX    #BBLK
01543A FF56 8D 29 FF81         BSR    PUNT2     PUNCH 2 HEX CHAR
01544                        * PUNCH ADDRESS
01545A FF58 CE 00D7  A         LDX    #PNTR
01546A FF5B 8D 24 FF81         BSR    PUNT2
01547A FF5D 08                 INX
01548A FF5E 8D 21 FF81         BSR    PUNT2
01549                        * PUNCH DATA
01550A FF60 DE D7    A         LDX    PNTR
01551A FF62 8D 1D FF81 PUN32   BSR    PUNT2     PUNCH 1BYTE (2 FRAMES)
01552A FF64 08                 INX              INCR X PNTR
01553A FF65 7A 00DB  A         DEC    TEMP      DECR BYTE CNT
01554A FF68 26 F8 FF62         BNE    PUN32
01555A FF6A DF D7    A         STX    PNTR
01556A FF6C 53                 COMB
01557A FF6D 37                 PSHB
01558A FF6E 30                 TSX
01559A FF6F 8D 10 FF81         BSR    PUNT2     PUNCH CHKSUM
01560A FF71 33                 PULB             RESTORE
01561A FF72 DE D7    A         LDX    PNTR
01562A FF74 09                 DEX
01563A FF75 9C D9    A         CPX    TEMPA
```

```
01564A FF77 26 BD FF36          BNE     PUN11
01565A FF79 CE FFA5  A          LDX     #MEOF     PUNCH EOF
01566A FF7C BD FB0E  A          JSR     PDATA
01567A FF7F 4F                  CLRA              CLEAR CC FOR RETURN
01568A FF80 39                  RTS
01569                        * PUNCH 2 HEX CHAR, UPDATE CHKSUM
01570A FF81 EB 00    A  PUNT2   ADDB    0,X
01571A FF83 7E FAE1  A          JMP     OUT2H     OUTPUT 2 HEX & RTN

01573                        ********** ROM DATA **********
01574A FF86    10    A  PRTON   FCB     $10,$3A,$10,$39,4 TURN ON PRT
     A FF87    3A    A
     A FF88    10    A
     A FF89    39    A
     A FF8A    04    A
01575A FF8B    14    A  PUNOFF  FCB     $14,$13   TAPE CONTROL
     A FF8C    13    A
01576A FF8D    04    A          FCB     4         EOF
01577A FF8E    3F    A  QMARK   FCB     $3F,4     PR ?
     A FF8F    04    A
01578A FF90    4C    A  LIL     FCC     /LILBUG 1.0/
     A FF91    49    A
     A FF92    4C    A
     A FF93    42    A
     A FF94    55    A
     A FF95    47    A
     A FF96    20    A
     A FF97    31    A
     A FF98    2E    A
     A FF99    30    A
01579A FF9A    04    A          FCB     4
01580A FF9B    4E    A  NOCHG   FCC     /NO CHG/
     A FF9C    4F    A
     A FF9D    20    A
     A FF9E    43    A
     A FF9F    48    A
     A FFA0    47    A
01581A FFA1    04    A          FCB     4         EOF
01582A FFA2    53    A  MTAPE   FCB     'S,'1,4
     A FFA3    31    A
     A FFA4    04    A
01583A FFA5    53    A  MEOF    FCC     /S9030000FC/
     A FFA6    39    A
     A FFA7    30    A
     A FFA8    33    A
     A FFA9    30    A
     A FFAA    30    A
     A FFAB    30    A
     A FFAC    30    A
     A FFAD    46    A
     A FFAE    43    A
01584A FFAF    0D    A          FCB     $D,4
     A FFB0    04    A
01585A FFB1    4F    A  PRTOP   FCC     /OP-/     PRT FOR TRACE LINE
     A FFB2    50    A
     A FFB3    2D    A
01586A FFB4    04    A          FCB     4
01587A FFB5    50    A  ARRAY   FCB     'P,1      ARRAY OF REG AND WRD LEN
```



```
     A FFB6    01    A
01588A FFB7    58    A          FCB     'X,1
     A FFB8    01    A
01589A FFB9    41    A          FCB     'A,0
     A FFBA    00    A
01590A FFBB    42    A          FCB     'B,0
     A FFBC    00    A
01591A FFBD    43    A          FCB     'C,0
     A FFBE    00    A
01592A FFBF    53    A          FCB     'S,1
     A FFC0    01    A
01593A FFC1    20    A  SPACE6  FCC     /      / 6 SPACES FOR SHOW HEADER
     A FFC2    20    A
     A FFC3    20    A
     A FFC4    20    A
     A FFC5    20    A
     A FFC6    20    A
01594A FFC7    04    A          FCB     4

01596                        ********** VECTORS **********
01597                        * VECTOR INDEPENDENCE
01598                        * ALSO SAVE ON RAM USAGE
01599                        * VECPTR - RAM PNTR TO VECTOR TABLE
01600                        * VECTOR TABLE - ADR OF INTERRUPT VECTORS
01601                        * MAY BE REDEFINED BY USER TABLE IN SAME FORM
01602A FFC8    FCE1  A  SERIAL FDB     DUMMY     NOT USED BY MONITOR
01603A FFCA    FCE1  A  TIMOVF FDB     DUMMY
01604A FFCC    FCE1  A  TIMOUT FDB     DUMMY
01605A FFCE    FCE1  A  TIMIN  FDB     DUMMY
01606A FFD0    FCE1  A  IRQ1   FDB     DUMMY
01607A FFD2    F821  A  SWI    FDB     IN.SWI
01608A FFD4    F803  A  NMI    FDB     IN.NMI
01609                        * DUMMY IS AN RTI

01611A FFD6                     ORG     $FFD6
01612                        * USE ADR ON STK TO OBTAIN INDEX
01613                        * USE INDEX TO GET CORRECT VECTOR
01614                        *    ROUTINE ADR FROM VECTOR TABLE.
01615A FFD6 32               VECTOR PULA             THROW AWAY MSB OF ADR
01616A FFD7 33                  PULB              GET LSB
01617A FFD8 83 FFE4  A         SUBD    #I.SER+2
01618A FFDB DE FE    A         LDX     VECPTR    ADR OF VECTOR TABLE
01619A FFDD 3A                  ABX               ADD OFFSET
01620A FFDE EE 00    A         LDX     0,X       GET VECTOR ADR
01621A FFE0 6E 00    A         JMP     0,X       GO THRU VECTOR

01623                        * INTERRUPTS GO THRU VECTORS, THEN HERE
01624                        * BSR STORES ADR ON STACK
01625                        * ADR USED TO OBTAIN INDEX INTO VECTOR TABL
01626A FFE2 8D F2 FFD6 I.SER  BSR     VECTOR
01627A FFE4 8D F0 FFD6 I.TOVF BSR     VECTOR
01628A FFE6 8D EE FFD6 I.TOVT BSR     VECTOR
01629A FFE8 8D EC FFD6 I.TIN  BSR     VECTOR
01630A FFEA 8D EA FFD6 I.IRQ1 BSR     VECTOR
01631A FFEC 8D E8 FFD6 I.SWI  BSR     VECTOR
01632A FFEE 8D E6 FFD6 I.NMI  BSR     VECTOR

01634                        * INTERRUPT VECTORS
```

```
01635A FFF0    FFE2  A VECTR  FDB    I.SER
01636A FFF2    FFE4  A        FDB    I.TOVF
01637A FFF4    FFE6  A        FDB    I.TOVT
01638A FFF6    FFE8  A        FDB    I.TIN
01639A FFF8    FFEA  A        FDB    I.IRQ1
01640A FFFA    FFEC  A        FDB    I.SWI
01641A FFFC    FFEE  A        FDB    I.NMI
01642A FFFE    F81E  A        FDB    STRT
01643          F8F6  A        END    START
TOTAL ERRORS 00000
```

```
   FCE9 ARMS04 01065 01087*
   FCBA ARMSTK 00982 01032 01049 01054*
   FFB5 ARRAY  00783 01587*
   FC56 BARMS  00966 00982*
   00D6 BBLK   00083*00453 00511 01250 01347 01369 01378 01399 01401 01422
               01436 01475 01534 01542
   FBE9 BERRTN 00874 00884 00910*00915 00917 00929
   00E6 BKADR  00096*00368 00877 00890 00901 00920 01125 01149
   FD7A BKPERR 01183 01192 01198*
   FBAD BP02   00869 00873*
   FBB8 BP04   00878*00883
   FBC5 BP06   00879 00885*
   00F2 BRKFLG 00098*01123 01141 01147 01170 01188
   FB98 BRKPNT 00147 00863*
   FBD9 BRKTAB 00875 00901*00928 01191
   FBEB BRTN   00904 00911*
   0014 BS.DTA 00207*01379
   001E BS.OFF 00208*01382
   001Z BS.ON  00206*01376
   FEC3 BSDTA  00154 01433*
   FEBA BSOFF  00154 01428*
   FEAF BSON   00194 01421*
   FEB7 BSON2  01423 01425*
   FF25 BSPUN  01437 01516*
   FF11 BYTE   01458 01463 01465 01472 01493*
   FC79 C.NMI  00113 01019*
   FC0F CALL   00150 00339*
   00F7 CALLF  00102*00371 00939 00953 00955
   FDE5 CHK2   01271 01273*
   FDD7 CHKABT 01041 01266*01305 01384 01385
   00D5 CHRNL  00082*00272 00337
   F85B CI     00191*00357
   0002 CI.DTA 00198*00217 01267
   0004 CI.OFF 00199*
   0000 CI.ON  00197*
   F891 CIDTA  00191 00243*
   F899 CIDTA1 00245 00249*
   F89C CIDTA2 00247 00252*
   F8D1 CIOFF  00191 00296*
   F8C8 CION   00191 00291*
   00D0 CKSUM  00077*01457 01503 01504
   0009 CLOCK  00058*01006
   F918 CLRAM  00369*00372
   0017 CNTLN  00054*01270
   0018 CNTLX  00055*00575 01273
PAGE  033   LILBUG 6801 DEBUG MONITOR *** VER 1.0 ***
```

```
   0008 CO.DTA 00201*00234
   000A CO.OFF 00202*
   0006 CO.ON  00200*00375
   F8A8 CODTA  00192 00269*
   F8D1 COOFF  00192 00299*
   F8D2 COON   00192 00304*
   F8D5 COON2  00305*
   F8A7 CRTN   00267*00276 00282
   FC58 CRTS   00957 00989*00997
   00D0 CT     00076*00077 00433 00436 00462 00789 00846 00848
   FBFD DBRK2  00922*00925
   FC05 DBRK6  00919 00928*
   F8D8 DELAY  00307*01430
   FBEC DELBRK 00866 00871 00914*
   FD86 DISPLY 00153 01215*
   F8DB DLY    00308*00309
   FCE1 DUMMY  01079*01602 01603 01604 01605 01606
   FD7A ENDCAL 01000 01197*
   F800 EX.NMI 00112*01099
   00F5 EXONE  00100*00950 00963 00981 01025 01036 01048 01064 01201
   F824 FCTABL 00144*00361 00474
   00FA FCTPTR 00104*00362 00454
   FBE9 GOX2   00868 00872 00909*00947 00949 00978 00980
   FC23 GOX6   00945 00952*
   FC38 GOX7   00954 00963*
   FC3F GOX8   00964 00966*
   FC11 GOXQT  00156 00944*
   000E HS.DTA 00204*01251
   0010 HS.OFF 00205*01255
   000C HS.ON  00203*01248
   FDEC HSDTA  00193 01285*
   F8D1 HSOFF  00193 00298*
   F8D1 HSON   00193 00297*
   F8F1 HY     00177 00344*
   FFEA I.IRQ1 01630*01639
   FFEE I.NMI  01632*01641
   FFE2 I.SER  01617 01626*01635
   FFEC I.SWI  01631*01640
   FFE8 I.TIN  01629*01638
   FFE4 I.TOVF 01627*01636
   FFE6 I.TOVT 01628*01637
   FCEA IFPTM  00385 01069 01092*
   FCF7 IFPTM2 01096 01098*
   F803 IN.NMI 00113*00388 01608
   F821 IN.SWI 00123*01607
   FA57 IN1HG  00593 00601*
   F873 INCH1  00114 00216*
   F874 INCH15 00217*00219 00221 00223
   F876 INCH2  00218*
   F888 INCH4  00225 00227*
   F806 INCHNP 00114*00589 01272 01449 01452
   FB4D INDAT  00799 00808*
   FB69 INDAT0 00818 00825*
   FB84 INDAT1 00828 00838 00840*
   FB75 INDAT2 00814 00833*
   FB81 INDAT4 00836 00839*
   FB6D INDAT5 00824 00827*
   FA3F INHEX  00562 00589*01494 01499
PAGE  034   LILBUG 6801 DEBUG MONITOR *** VER 1.0 ***
```

```
F940 INPTM   00386 00395*
FA0D INPUT   00537 00557*00629 00810 00977
FA0E INPUT2  00556 00558*
FA18 INPUT3  00562*00574
FA1E INPUT5  00565*00570
FA27 INPUT6  00567 00569*
FA33 INPUT7  00563 00575*
FA3A INPUT9  00576 00580*
FA09 INPUTA  00413 00441 00555*
FA3C INPUTC  00540 00542 00581*
F8DF IO      00218 00235 00318*00374 00376 01252 01339
FE54 IO2     01249 01256 01268 01339*01377 01380 01383
00FC IOPTR   00105*00319 00358
FFD0 IRQ1    01606*
FF90 LIL     00377 01578*
FE9A LOAD    00159 01398*
FEF6 LOAD11  01472*01481
FF04 LOAD12  01476 01478*
FF0B LOAD15  01474 01484*
FF0E LOAD19  01479 01487*
FE9C LOAD2   01399*01415
FF10 LOAD20  01454 01488*
FED0 LOAD3   01449*01451 01485
FED3 LOAD4   01450*01456
0080 LOWRAM  00072*00439
FC7E M.NMI   00112 01023*
FD5F M.SWI   00123 01181*
F952 MAIN    00389 00391 00408*00414 00418 00483 01050 01203
F971 MAIN01  00415 00422*
F984 MAIN03  00434*00444
F9C3 MAIN07  00442 00483*00514
F9E5 MAIN08  00420 00424 00427 00512*
FA5F MEM01   00419 00620*
FA69 MEM02   00622 00625*00646 00664
FA7D MEM03   00632 00636*
FA8F MEM04   00644 00647*
FA99 MEM06   00648 00652*
FAA0 MEM08   00653 00656*
FAA4 MEM10   00655 00658*
FAA6 MEM12   00651 00659*
FA6B MEM25   00626*00642
FA88 MEM33   00640 00643*
FA5B MEMORY  00162 00618*
FAA8 MEMSL   00425 00660*00692
FFA5 MEOF    01565 01583*
F9BD MERROR  00440 00443 00481*00513
FAB1 MERRTN  00619 00624 00657 00666*00684 00686 00742
0003 MODE    00067*01093
FCFD MOVSTK  01025 01105*01183
FFA2 MTAPE   01538 01582*
F8B0 N1      00271 00273*
F8BC N3      00274 00279*
F8BE N4      00278 00281*00287
FC41 NEXT    00422 00970*00976
00D3 NEXTX   00079*00455 00456 00460 00475
FFD4 NMI     01608*
FC95 NMI01   01028 01033*
FC96 NMI015  01034*01042
```



```
FC9D NMI02   01035 01040*
FCB7 NMI03   01031 01050*
FF9B NOCHG   00740 01580*
FA37 NOTHEX  00577*00591 00595 00597 00599
00F3 NTRACE  00099*00979 01027 01034 01200
0004 NUMBP   00095*00096 00097 01126 01139 01150 01168
000B OCREG   00060*01008
FAC7 OFF4    00681 00685*
FACB OFF6    00683 00687*
FAB4 OFFSET  00165 00676*
00EE OPCODE  00097*00882 00907
FAE1 OUT2H   00625 00698 00705 00712*00793 01571
FADB OUT2HS  00118 00690 00705*00795 00837 01046 01312
FAD8 OUT4HS  00119 00662 00698*00839 00892 01309
F89D OUTC    00261*00269 00285
F89E OUTC1   00262*00264
F809 OUTCH   00115*00412 00769 00855 01326 01425 01523
F88A OUTCH1  00115 00226 00233*
FB91 OUTDA   00786 00853*
FAE9 OUTHL   00714 00720*
FAED OUTHR   00716 00727*01297
00F6 OUTSW   00101*00224 00270 00409 01375 01386
00DC OVFL    00088*00541 00568
0001 P2DDR   00057*01074
FB16 PCR     00650 00766*
FB12 PCRLF   00120 00410 00658 00688 00691 00757 00763*00779 00845 00889
             00952 01291 01307 01372
FB0E PDATA   00117 00378 00482 00741 00757*01044 01294 01539 01566
FB07 PDATA1  00116 00305 00750*00758 01429
FB04 PDATA2  00748*00752
FE65 PNCH3   01351*01404
FE77 PNCH4   01372*01403 01405
FE6E PNCHER  01344 01346 01350 01352 01355*
FE71 PNCRTN  01354 01357*
00D7 PNTR    00084*00626 00659 00661 00663 00679 01215 01221 01224 01227
             01232 01237 01239 01254 01288 01308 01310 01318 01331 01333
             01334 01448 01467 01517 01527 01528 01545 01550 01555 01561
FF2F PNULL   01522*01525
FBC9 PRBRK   00864 00876 00889*00926 00932
FBD1 PRBRK2  00892*00895
FB88 PREGS   00846*01047
FBA6 PREGS1  00778 00845*01202
FB71 PRERR   00813 00816 00829*00834
FF86 PRTON   00304 01574*
FFB1 PRTOP   01043 01585*
FE01 PRTTL   01298*01303
00F8 PTM     00103*00364 00395 01085
E000 PTMADR  00069*00363
FF36 PUN11   01526*01564
FF42 PUN22   01529 01532*
FF44 PUN23   01531 01533*
FF62 PUN32   01551*01554
FE72 PUNCH   00168 01369*
FF8B PUNOFF  01428 01575*
FF81 PUNT2   01543 01546 01548 01551 01559 01570*
FDE9 PVALIN  01219 01235 01278*01345 01351
FF8E QMARK   00481 01577*
FD15 RBRK    00999 01026 01123*01190
```

```
FD1E RBRK2   01127*01140
FD2A RBRK3   01133 01135*
FD35 RBRK6   01124 01142*
FE57 RD2ADR  00676 01343*01370
0012 RECEV   00064*00249
FB22 REGS1   00780*00847
FB26 REGS2   00782*00803
FB42 REGS3   00790 00798*
FB3D REGS4   00792 00795*
FB46 REGS6   00797 00801*
FB1E REGSTR  00171 00778*
FAB3 RETRN   00631 00635 00637 00667*00739
0010 RMCR    00062*00338
F8E9 S120    00174 00336*
F8EC S1205   00292 00337*00345
00DD SAVSTK  00089*00380 00780 00951 00991 00995 00996 00998 01055 01057
             01059 01062 01066 01106 01108 01109 01111 01113 01184 01186
FFC8 SERIAL  00359 01602*
FD36 SETB    00965 01030 01147*
FD3F SETB2   01151*01169
FD54 SETB4   01153 01164*
FD5E SETB6   01160 01171*
FC71 SETCLK  01005*01020 01078
FCE2 SETPTM  01070 01084*
FDAA SHERR   01148 01233*
FDA7 SHERR2  01220 01231*01236 01241
FE11 SHOW10  01305*01335
FE22 SHOW12  01312*01315
FE30 SHOW14  01319*01329
FE3C SHOW16  01322 01325*
FE3E SHOW18  01324 01326*
FDE8 SHOW19  01276*01306 01332
FD97 SHOW35  01218 01224*
FDAD SHOW4   01223 01235*
FDC1 SHOW8   01230 01248*
FDF7 SHOW9   01291*01336
FADD SPACE   00121 00706*01298 01299 01316
FFC1 SPACE6  01293 01593*
00E4 SPSAVE  00094*00366 00956 00961 00993 01054 01116
F99B SRCH    00435 00453*
F9A1 SRCH01  00456*00477 00498
F9B9 SRCH02  00472 00476*
F9C5 SRCH03  00463 00488*
F9C9 SRCH04  00490*00503
F9D7 SRCH05  00496 00500*
00CF STACK   00074*00356 00365 00367 00408 00994 01024 01182
F8F6 START   00122 00356*01643
FAF7 STRCHK  00634 00737*01157
F81E STRT    00122*01642
00D1 STRTX   00078*00432 00488 00509
FFD2 SWI     01607*
FD7E SWI3    01195 01201*
FBDC TAB1    00902*00908
0008 TCSR    00059*00390 01019 01075 01077
00DB TEMP    00086*00560 00573 00581 00687 00689 01460 01473 01536 01553
00D9 TEMPA   00085*00489 00494 00501 00559 00561 00580 00620 00633 00677
             00820 00825 00885 00902 01045 01067 01187 01229 01240 01242
             01244 01290 01330 01519 01526 01563
F9EB TERM    00434 00523*00815
F9F9 TERM02  00524 00526 00528 00530*
FFCE TIMIN   01605*
FFCC TIMOUT  01604*
FFCA TIMOVF  01603*
FC46 TRACE   00180 00975*
FC4F TRACE2  00971 00979*
0013 TRANS   00065*00265
0011 TRCS    00063*00243 00262 00294
F9FC VALIN   00417 00538*
F9FA VALINP  00537*00618 00867 00914 00946 01278
FA08 VALRTN  00538 00544*
00FE VECPTR  00106*00360 01097 01618
FFD6 VECTOR  01615*01626 01627 01628 01629 01630 01631 01632
FFF0 VECTR   01052 01635*
FEAB VERF    00183 01414*
FB1B XOUTCH  00707 00732 00748 00764 00767 00769*
FB95 ZOUTCH  00853 00855*
```

MOTOROLA ***Semiconductor Products Inc.***

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

12007-1 PRINTED IN USA 8/80 IMPERIAL LITHO B88625