4.0 SOFTWARE DESCRIPTION

4.1 Overview

The operating system for the Technico computer consists of several EPROM based subsystems:

- o Monitor - provides basic debug via the terminal
- o CVM Monitor - extension to the monitor for CVM
- o Disk Handler - extension to the monitor for disk operation
- o Instant Input Assembler - resident line by line miniassembler

During system restart, the monitor will gain control and then wait for the user to type the letter "X". When the letter "X" is typed, the monitor can calculate the terminal baud rate and begin operation. If the disk handler EPROM is in the system, then operation will begin there and you can enter disk related commands. To execute the monitor when in the disk handler, just type MON. If the disk handler EPROM is not in the system, then operation will begin with the monitor. When the CVM monitor extension is present, and a terminal is not connected and active then all terminal commands for the disk handler or monitor will be routed to the color video module. As you can see, the Technico system will automatically determine the required configuration and will begin execution accordingly.

Note: The monitor prompt character is "?" and the disk handler prompt is ">". This will serve to indicate what subsystem is active. To enter the monitor from the disk handler, type MON. To return to the disk handler from the monitor type the command "R".

4.1.1. Memory Map

The Monitor and other system routines use various RAM memory locations. These locations are defined below.

Address (hex)	Contents
0-3	Interrupt vector - level 0
4-7	Interrupt vector - level 1
8-B	Interrupt vector - level 2
C-F	Interrupt vector - level 3
10-13	Interrupt vector - level 4
14-17	Interrupt vector - level 5
18-1B	Interrupt vector - level 6
1C-1F	Interrupt vector - level 7
20	Carriage return delay
22	Echo flag (0=half duplex, <0=full)
24	Terminal speed
26	No. of words for a break
28	User instruction - one
2A	User intruction - two
2C	User instruction - three
2E	Return branch (two words)
32	Next stop
34	Stop increment
36	Maximum number of stops
38	Register bounds - first
3A	Register bounds - last
3C	Memory bounds - first
3E	Memory bounds - last
40-43	XOP vector - XOP 0
44-47	XOP vector - XOP 1
48-4B	XOP vector - XOP 2
4C-4F	XOP vector - XOP 3
50-53	XOP vector - XOP 4
54-57	XOP vector - XOP 5
58-5B	XOP vector - XOP 6
5C-5F	XOP vector - XOP 7
60-63	XOP vector - XOP 8
64-67	XOP vector - XOP 9
68-6B	XOP vector - XOP 10
6C-6F	XOP vector - XOP 11
70-73	XOP vector - XOP 12
74-77	XOP vector - XOP 13
78-7B	XOP vector - XOP 14
7C-7F	XOP vector - XOP 15
80-9F	Monitor Workspace
AO-AF	XOP Workspace (only 8 registers)
BO-CF	System Software Workspace
DO	User - RO
D2	User - R1

D4	User - R2
D6	User - R3
D8	User - R4
DA	User - R5
DC	User - R6
DE	User - R7
E0	User - R8
E2	User - R9
E4	User - R10 (RA)
E6	User - R11 (RB)
E8	User - R12 (RC)
EA	User - R13 (RD)
EC	User - R14 (RE)
EE	User - R15 (RF)

4.2 Mighty Monitor

The Mighty Monitor provides the following comprehensive set of commands:

A	Alter the contents of RAM
B	Breakpoint set/restore
C	Copy memory to memory
D	Dump memory to display or terminal
G	Go to program in memory
H	Hexadecimal Arithmetic
I	Inspect CRU bit
L	Load program from terminal
M	Modify CRU bit
P	Program EPROM
R	Return
S	Snap definition
W	Workspace dump

The Mighty Monitor accepts input from and produces output for a serial asynchronous ASCII terminal or teletypewriter. To insure maximum flexibility in the choice of a terminal, the monitor always generates two stop bits after each character and a user controlled delay after each carriage return. The Baud rate of the terminal is determined automatically during the start up of the monitor. After a reset (power-on or manual) the monitor will wait for the user to enter the letter 'X'. When the letter 'X' is entered, the monitor automatically calculates the Baud rate (110 to 9600) and begins normal operation. For a disk system, the monitor will immediately transfer control to the disk handler. You can return to the monitor and use the monitor's debug commands by entering the disk command "MON".

During normal operation, the monitor prompts the user to enter a command by typing a question mark at the beginning of a new line. The first entry by the user must be one of the allowable command codes (A, B, C, D, G, H, I, L, M, P, R, S, W), and is followed by the arguments in hexadecimal notation. Multiple arguments are seperated from one another by an arbitrary sequence of symbols or characters, except for hexadecimal digits (0-9,A-F) or carriage return. The command is terminated by any non-hexadecimal digit (including carriage return) after the last argument.

If an argument is typed with more than the required number of digits (usually four), the monitor will use only the right-most digits. This feature can be used to correct input errors. If any argument is shorter than required, the left-most digits will automatically be filled with zeros.

The monitor uses certain locations in memory to store breakpoint information, etc. The entire system memory map is shown in paragraph 4.1.1. To operate in half-duplex mode (no character echo) change the most significant bit of the echo flag to zero using monitor's Alter command. To insert a delay after carriage return, enter the required delay in the delay word (again using the Alter). The total carriage return delay is Delay*6 microseconds. If you do not know the delay required for your terminal, it can be determined experimentally by increasing the delay until no characters are lost after a carriage return. If you modify any of the other locations used by the monitor, it may not function properly.

A detailed description of each command is provided in the following paragraphs, along with an example of its usage.

4.2.1 Alter - The contents of memory may be examined and modified.

Format: A aaaa

Procedure:

1. Type "A".

2. Type the byte address (aaaa) of the memory location to be examined (in hexadecimal) followed by a space.

3. The monitor will display the contents of the specified location in hexadecimal format followed by a hyphen.

4. If you wish to change the contents of this location, simply enter the desired hexadecimal value followed by a space or carriage return. If not, just type space or carriage return and the monitor will display the contents of the next sequential address. If a space terminated the entry the next value will be displayed on the current line. If a carriage return terminated the entry, the next byte′s address and contents are displayed on the next line.

5. Repeat steps 3-4 until all desired locations have been examined or modified. To exit this routine depress the BREAK key on the terminal.

Note:

1. If the monitor was entered from a user BREAKPOINT, ALTER can be used to examine or modify the working registers. Refer to the memory map for a definition of the addresses used.

2. ALTER can also be used to examine EPROM, memory, but it can not be used to modify it.

Example: The following sequence will alter locations #400 and #404 with #FF. Locations #401-403 are unchanged (user′s entries are underlined).

```
?A400 00-FF 11-_22-_33-_44-FF 55-
```

4.2.2 BREAKPOINT - A breakpoint or trap may be set in any user program that is stored in RAM. Whenever the processor encounters the substituted trap instruction, the state of the machine is saved and control is transferred back to the monitor for user action.

Format: B aaaa,n

Procedure:

1. Type "B".

2. Type the hexadecimal address (aaaa) of the location to be trapped, followed by a delimiter. (aaaa) must be a word address (even number).

3. Type the number of words (n) to be removed for the trap. This should be the number of words (1, 2, or 3) currently occupied by the trapped instruction.

4. Type space or carriage return. The monitor will remove any prior trap and then install the new trap.

Note:

1. If the existing trap is to be removed without setting a new one, the address and word count are omitted and the command terminated by carriage return.

2. After entering the monitor from a trap, the GO command can be used to resume execution (see GO command, discussed later).

3. The contents of the user's workspace registers are saved whenever a breakpoint is encountered. The contents of the registers can be examined or modified using the ALTER command. The System Memory Map shows where the active registers are saved. Note: If the workspace pointer is changed by the user program, the registers will belocated at the address in the workspace pointer.

4. Relative jump instructions should not be breakpointed if a return GO is to be used or if a SNAP is established.

5. An asyncronous breakpoint can be triggered by placing the LOAD/RESET switch in the RESET position and then pressing RESET. This feature can be used to trap a user program that is "hung".

6. A user program can take control of the breakpoints by setting location #40 to #43 with appropriate workspace pointer and program counter. This should be done with care since it eliminates the monitors breakpoints.

Example: Place a breakpoint at location >1276 which contains a two word instruction (e.g. LI R1,1)

```
?B1276,2
```

4.2.3 COPY - The contents of a block of memory may be copied into another area of memory.

Format: C ssss, eeee, dddd

Procedure: 1. Type "C".

2. Type the hexadecimal starting address (ssss) followed by a delimeter, and the hexadecimal ending address (eee) followed by a delimeter of the block of memory you wish to be copied.

3. Type the hexadecimal destination address (dddd) followed by a space or carriage return. For a normal copy operation, the destination address should not be within the bounds of the block of memory that you are copying.

Note: 1. The copy command can be used to set a block of memory to a specified constant. This is done in two steps. First , place the desired constant in the start location (ussing the ALTER command). Then perform a "C ssss, eeee-1, ssss+1", where (ssss) is the start address and (eeee) is the end address of the block.

Example: 1. The following command will copy #410-420 into #430-440.

```
?C410,420,430
```

2. To set all locations #410-41F to zero, the following commands are used.

```
?A410 11-00 22    (sets 410 = 00)
?C410,41E,411     (clears 410-41F)
```

Note that #41E is one less than the ending address #41F and #411 is one greater than the starting address #410.

4.2.4 DUMP - The contents of a block of memory may be listed on the terminal.

Format: D ssss, eeee

Procedure:

1. Type "D".

2. Type the hexadecimal starting address (ssss) followed by a delimiter and then the hexadecimal ending address (eeee) of the memory to be listed.

3. Terminate the command by typing a space or a crriage return. The monitor will now list the requested block of memory, sixteen bytes per line.

Note:

1. The ending address may be omitted (and the command terminated by a carriage return), in which case the monitor assumes that the ending address is the end of memory (65535, or #FFFF).

2. The dump may be stopped at any time by depressing the BREAK key on the terminal.

3. The LOAD command can reload the program if the dump is recorded, on paper tape or other media.

Example: Both of the following examples will dump the entire memory:

```
?DO,FFFF
```

```
?DO
```

4.2.4 DUMP - The contents of a block of memory may be listed on the terminal.

Format: D ssss, eeee

Procedure:

1. Type "D".

2. Type the hexadecimal starting address (ssss) followed by a delimiter and then the hexadecimal ending address (eeee) of the memory to be listed.

3. Terminate the command by typing a space or a crriage return. The monitor will now list the requested block of memory, sixteen bytes per line.

Note:

1. The ending address may be omitted (and the command terminated by a carriage return), in which case the monitor assumes that the ending address is the end of memory (65535, or #FFFF).

2. The dump may be stopped at any time by depressing the BREAK key on the terminal.

3. The LOAD command can reload the program if the dump is recorded, on paper tape or other media.

Example: Both of the following examples will dump the entire memory:

```
?DO,FFFF


?DO
```

4.2.5 GO - Control can be transferred to a specified word in memory. Execution can also be resumed after a breakpoint trap.

Format: G aaaa

Procedure:

1. Type "G".

2. Type the hexadecimal address (aaaa) where control is to be transferred. (aaaa) must be a word address (even).

3. Terminate the command by typing a space or carriage return. The monitor will now transfer control to address (aaaa).

Note:

1. The address (aaaa) may be omitted (and the command terminated ay a carriage return) in which case the monitor will assume that a trap was reached, restore the state of the machine, execute the instruction removed for the trap, and return to the point following the trap. This feature should be used only if the monitor was entered by a trap and the location being trapped does not contain a relative jump.

2. Do not set a new breakpoint, then issue a GO without an address, as this will transfer control to the wrong location.

Example: The following will transfer control to location #106.

```
?G106
```

4.2.6 HEXADECIMAL ARITHMETIC - Calculate the hexadecimal sum and difference of two numbers.

Format: H aaaa, bbbb

Procedure: 1. Type "H".

2. Type the two hexadecimal operands (aaaa) and (bbbb) seperated by a delimiter and followed by a space or carriage return.

3. The monitor will now calculate and display (xxxx)=(aaaa)+(bbbb) and (yyyy)=(aaaa)-(bbbb) as follows:

```
H+=(xxxx)         H=(yyyy)
```

Example: This command is useful for calulating the destination address for a jump. If the jump instruction #1047 is at, say, location #1234 then the destination address is (#1234+2)+ 2*47. This sum is calculated in two steps as follows:

```
Step 1)  ?H1236,47
         H+=127D   H--=11EF


Step 2)  ?H127D,47
         H+=12C4    H--=1236
```

Note that the jump displacement is relative to the next sequential instruction (#1236) not the jump itself.

4.2.7 INSPECT - A CRU bit may be displayed on the terminal.

Format: I bb

Procedure: 1. Type "I".

2. Type the CRU bit (bb) to be tested, followed by a space.

3. The monitor will display the selected CRU bit.

Note: 1. The CRU bit number is used for this command, not the CRU base address. For example, bit number 3 is CRU base address 6. Therefore, when bit 3 is used, R12 is loaded with 6.

Example: Display CRU bit 5 (assume it is set).

```
?I5 1
```

4.2.8 LOAD - A program file may be loaded into memory from paper tape or any other terminal storage media.

Format: L

Procedure: 1. Type "L".

2. Initiate the input (e.g. the paper tape).

Note: 1. The LOAD command will reload programs produced by the DUMP command. The dumped program will be reloaded into the same area of memory that it was dumped from.

2. If you do not wish the input to be listed as it is loaded, simply turn off the monitor echo flag (most significant bit of #22). This will suppress the monitor's echo

3. The carriage return delay must be set to zero (i.e. memory locations #20, #21) prior to loading.

4.2.9 MODIFY - A CRU bit may be set or cleared.

Format: M bb,v

Procedure:

1. Type "M".

2. Type the desired CRU bit (bb) followed by a delimeter.

3. Type the bit value that you desire (0=clear, 1=set).

Note: 1. The CRU bit number is used for this command, not the CRU base address. For example, bit number 7 is CRU base address #E. Therefore, when bit 7 is specified, the monitor will load R12 with #E.

Example: Set bit 12 to 0

```
?M12,0
```

4.2.10 PROGRAM - Program a 2708 EPROM.

Format: P aaaa, bbbb, cccc

Procedure:

1. Type "P".

2. Type the hexadecimal starting address (aaaa) of the area to be placed in EPROM followed by a delimiter, and then the hexadecimal ending address (bbbb) followed by a delimiter.

3. Type the hexadecimal starting address of the EPROM area to be programmed followed by a space or carriage return. (0000 is the first EPROM location)

4. The monitor will now program the EPROM's.

Note:

1. The starting address of the EPROM's, for programming purposes only, is zero.

2. The monitor always programs both EPROM's. Even bytes in one and odd bytes in another.

3. To program only selected locations, place #FF in any location not to be programmed. Since the erased EPROM has #FF in all locations, this will not change the EPROM.

4. The ending address (bbbb) MUST BE EVEN.

Example: Program a copy of the monitor/IIA PROM.

```
?PF800, FFFE,0
```

4.2.11 RETURN - Return to System

Format: R

Procedure: 1. Type "R"

2. Type a carriage return. The monitor will now return to the system (e.g. BASIC or Disk handler).

Note: 1. If the monitor was not called by a system routine, R will merely restart the monitor

4.2.12 SNAP - Snap parameters can be established.

Format:

```
S ffff, iiii, nnnn
R? r1, r2
M? m1, m2
```

Procedure:

1. Type "S".

2. Type the first time a snap is desired (ffff) followed by a delimiter, then the increment between snaps (nnnn) followed by a delimiter, and finally the total number of snaps (nnnn) followed by a delimiter.

3. When the monitor types "R?", enter the workspace registers to be snapped, then type a carriage return.

4. When the monitor types "M?", enter the area of memory to be snapped. If no memory is to be snapped, then type a carriage return.

Note:

1. Prior to establishing a snap a breakpoint must be set.

2. All snap parameters are in hexadecimal.

Example:

The following sequence will snap registers R1-R3 and memory area #100-105 after the fourth execution of the instruction at #130. After the initial snap, it will snap every third time until a total of six snaps.

```
?B130,1         (first set trap)
?S4,3,6         (then set the snap)
R?1,3           registers snapped
M?100,105       memory snapped
```

The sample output given on the next page illustrates the use of A, B, and S commands. The A command is used to enter a program into memory. This program will decrement R1, R2, and R3. The B command is used to set a Breakpoint trap at #130 (which contains a one word instruction). The S command specifies a snap of R1 through R3 and memory locations #100 through #105 to be taken just prior to the 4th, 7th, 10th, 13th, 16th, and 19th times that the instruction at location #130 is executed.

Snap Example Output

```
                ?A130 2C-06 00-
?A130 00-06 00-01 00-06 00-02 00-06 00-03 00-10 00-FC 00
?B130,1
?S4,3,6
R?1,3
M?100,105
?G130


PC=0132 WP=00D0 ST=D000
R1=00B0 R2=2B36 R3=0D38
0100: 02 03 00 01 C0 C1


PC=0132 WP=00D0 ST=D000
R1=00AD R2=2B33 R3=0D35
0100: 02 03 00 01 C0 C1


PC=0132 WP=00D0 ST=D000
R1=00AA R2=2B30 R3=0D32
0100: 02 03 00 01 C0 C1


PC=0132 WP=00D0 ST=D000
R1=00A7 R2=2B2D R3=0D2F
0100: 02 03 00 01 C0 C1


PC=0132 WP=00D0 ST=D000
R1=00A4 R2=2B2A R3=0D2C
0100: 02 03 00 01 C0 C1


PC=0132 WP=00D0 ST=D000
R1=00A1 R2=2B27 R3=0D29
0100: 02 03 00 01 C0 C1


?
```

4.2.13 WORKSPACE - Examine user workspace registers

Format: W aa,bb

Procedure:

1. Type "W".

2. Type the first register to be dumped in hexadecimal followed by a delimiter and then the last register to be dumped followed by a delimiter.

3. The monitor will now dump the specified registers

Note:

1. Since the registers are in memory, the ALTER command can be used to modify them. Refer to the system memory map for the addresses required.

Example: Dump all of the workspace registers.

```
?W0,F
```

4.2.14 Listing

The source listing of the mighty monitor is included in this section. A review of the monitor listing will help you to understand how the 9900 instructions can be used. The monitor listing is relative addressed. That is, the loader modifies the code to operate where loaded. In the T99SS CPU module the monitor is loaded at #FC00. Therefore, you must add #FC00 to the address shown in the listing to obtain the EPROM address of that data. For example, STRT10 is the label of the instruction at relative #16. The actual EPROM address of that word is #FC00 + 16 = FC16.

In addition to the terminal commands, the monitor provides other useful features for the programmer. During power-on the monitor establishes two XOP's (Extended Operations) to be used for terminal input/output. These XOP's can be exploited by a user program to perform input/output to the user terminal. XOP 1 is used for input, and XOP 2 is used for output. The program below, which can be entered by the Instant Input Assembler, uses these XOP's to print the message "pick a number from 1 to 5" and then collect the user response. Notice that the Instant Input Assembler recognizes the XOP's by the mnemonics IN and OUT.

```
LI R1,>110
OUT *R1
MOVB *R1+,R0
JNE -3
IN R1
JMP -7
/110
+>0D0A
$PICK A NUMBER FROM 1 TO 5
+0
```

Other routines in the monitor are also useful. Some of them are:

TYPEN - Proceed to a new line on the terminal. Uses register R4 as scratch. Called by BL @TYPEN.

DMEMN - Display the contents of register R1 as four hex digits. The value is displayed on a new line and is followed by a ":". Input in register R1. Registers R0, R4, R5, and R7 are used as scratch. Called by BL @DMEMN.

DISRG - Display contents of R5 as four hex digits. The format is "XY = dddd" where "XY" are any two characters following the call. Registers R0, R4, R5, and R7 used as scratch. Called as follows:

```
          BL @DISRG
          DATA 'XY'
```

TYPEWD - Display the contents of R5 as four hex digits. Input in R5. Regirters R0, R4, R5 used as scratch. Called by BL @TYPEWD.

RDNUM - This is a powerful routine for accepting hex parameters from the operator. It will read one, two, or three parameters and put them in R1, R2, R3. Refer to the source listing for further details of RDNUM.

DUMP - Dump memory from address in R1 to the address in R2. Registers R0, R1, R5 used as scratch . The following will dump #107 to #311 and then return to the user.

```
          LI R1,>107
          LI R2,>311
          BL @DUMP
```

BDISPS - Display the leftmost byte of R5 as two hex digits preceeded by a space . Input is in R5. Registers R0, R4, R5 used as scratch. Called by BL @BDISPS.

```
                        TITL '9900 MIGHTY MONITOR (VER 4 - 8/78)'
 0000           IDTMM   IDT
 0000                   DREG
                *
                * NOTICE: WHEN THE MONITOR IS ENTERED IT WILL
                * AWAIT USER INPUT TO DETERMINE THE BAUD RATE
                * OF THE TERMINAL DEVICE.  THE USER SHOULD
                * TYPE AN 'X' TO SET THE BAUD RATE.
                *
                * THE BASIC 9900 DEBUG MONITOR OFFERS THE
                * FOLLOWING SET OF COMMANDS(PARAMETERS IN []
                * ARE OPTIONAL):
                *
                * A <ADDRESS>                           ALTER
                * B [<ADDRESS>] [<WORD COUNT>]          BREAKPOINT
                * C <START> <END> <TARGET>              COPY
                * D <START>[<END>]                      DUMP
                * G [<ADDRESS>]                         GO
                * H <NUMBER-1> <NUMBER-2>               HEX ARITH
                * I <BIT>                               INSPECT BIT
                * L [<ADDRESS>]                         LOAD
                * M <BIT> <VALUE>                       MODIFY BIT
                * P <START> <END> <TARGET>              PROGRAM
                * R                                     RETURN
                * S <1ST> <INC> <TOTAL>                 SNAP
                *   ?R [<REG-1> <REG-2>]
                *   ?M [<START> <END>]
                * W <REG>[<REG>]                        WORKSPACE DUMP
                *
                * EXTERNAL DEFINITIONS
                *
                        DEF   TYPE,TYPEN,TYPEH
                        DEF   DMEMN,TYPEWD,RDNUM
                *
                * SYSTEM EQUIVALENCES
                *
 0001           PRG     EQU   1                ; PROGRAM MODE
 0000           TTYI    EQU   0                ; TTY INPUT
 0000           TTYO    EQU   0                ; TTY OUTPUT
 2C00           XOP0    EQU   >2C00            ; XOP-0
 1000           NOOP    EQU   >1000            ; NO-OP
 0080           MTRWP   EQU   >80              ; MONITOR WORKSPACE
 00D0           USRWP   EQU   >D0              ; USER WORKSPACE
 0090           XOPWS   EQU   >90              ; XOP WORKSPACE(8 REG.)
 0020           DELAY   EQU   >20              ; DELAY WORD
 0026           BREAK   EQU   >26              ; BREAKPOINT AREA
 0028           BKRTN   EQU   BREAK+2          ; BREAK RETURN
 000D           CR      EQU   >0D              ; CARRIAGE RETURN
 0A0D           CRLF    EQU   >0A0D            ; CAR. RET., LINE FEED
 001C           MAX     EQU   28               ; (NO. OF COMM. + 1)*2
                *
                * THE FOLLOWING AREA OF RAM IS USED
                * BY THE MONITOR
                *
                * 20  CR DELAY TIME
                * 22  ECHO FLAG
                * 24  TERMINAL SPEED
                * 26  (BREAK) NO. OF WORDS FOR TRAP
                * 28   USER INST. ONE
                * 2A              TWO
                * 2C              THREE
                * 2E   RETURN BRANCH (TWO WORDS)
                * 32   NEXT STOP
                * 34   STOP INCREMENT
                * 36   MAX NO. OF STOPS
                * 38   SNAP REG - FIRST
                * 3A              LAST
                * 3C   SNAP MEM - FIRST
                * 3E              LAST
                * 40-43  XOP-0 BREAKPOINTS
                * 44-47  XOP-1 INPUT
                * 48-4B  XOP-2 OUTPUT
                * 4C-4F  XOP-3
                * 50-53  XOP-4
                * 54-57  XOP-5
                * 58-5B  XOP-6
                * 5C-5F  XOP-7
                * 60-63  XOP-8
                * 64-67  XOP-9
                * 68-6B  XOP-10
                * 6C-6F  XOP-11
                * 70-73  XOP-12
                * 74-77  XOP-13
                * 78-7B  XOP-14
                * 7C-7F  XOP-15
                * 80-9F  MONITOR WORKSPACE
                * A0-AF  XOP WORKSPACE (ONLY 8 REGISTERS)
                * B0-CF DISK/TAPE WORKSPACE
                * D0   USER R0
                * D2        R1
                * D4        R2
                * D6        R3
                * D8        R4
                * DA        R5
                * DC        R6
                * DE        R7
                * E0        R8
                * E2        R9
                * E4        RA (R10)
                * E6        RB (R11)
                * E8        RC (R12-USER ERU BASE)
                * EA        RD (R13)
                * EC        RE (R14)
                * CE        RF (R15)
                *
                *
                * THE FOLLOWING IS MONITOR POWER UP
                * SEQUENCE
                *
0000 02E0 0080  START  LWPI MTRWP
0004 04CC              CLR  R12                ; SET CRU BASE
0006 1D01              SBO  PRG                ; CLEAR PROG. MODE
0008 020D 00D0         LI   R13,USRWP          ; SET USER WP
000C 0201 0040         LI   R1,>40             ; SETUP XOP VECTORS
0010 04C2              CLR  R2
0012 C461 032C  STRT10 MOV  @XOPTB->40(R1),*R1
```

```
0016 CCB1                  MOV   *R1+,*R2+
0018 16FC                  JNE   STRT10
001A 0200 FFEC             LI    R0,-20              ; R0=TERM. TIMER
001E 0201 0020             LI    R1,DELAY
0022 CC41                  MOV   R1,*R1+             ; CLEAR DELAY
0024 0731                  SETO  *R1+                ; SET ECHO
0026 1D00                  SBO   TTYO                ; TTY=HIGH
0028 1F00                  TB    TTYI                ; CRT?
002A 1302                  JEQ   STRT20              ; NO
002C 06A0 E800             BL    @>E800              ; SETUP FOR CRT
0030 1F00          STRT20  TB    TTYI                ; WAIT FOR START
0032 13FE                  JEQ   STRT20 (i.e one)
0034 0580          STRT30  INC   R0                  ; MEASURE A BIT
0036 1F00                  TB    TTYI
0038 16FD                  JNE   STRT30
003A 0920                  SRL   R0,2                ; REDUCE TO BIT COUNT
003C CC40                  MOV   R0,*R1+             ; SAVE SPEED
003E 06A0 00B8             BL    @TYPEN              ; GOTO NEW LINE
0042 020B F000             LI    R11,>F000           ; DISK SYSTEM?
0046 881B 0332             C     *R11,@BRANCH
004A 1328                  JEQ   TYPEX               ; YES-GO THERE
                   *
                   * REMOVE ANY BREAKPOINTS AND THEN
                   * ENTER MONITOR
                   *
004C 05C1                  INCT R1                   ; ADVANCE TO BREAK RET.
004E 0702                  SETO R2
                   *
                   * ROUTINE: BREAK
                   * ESTABLISH A BREAKPOINT OR SNAP AT (R1),
                   * REMOVING R2 INSTRUCTIONS AND SETTING
                   * NEXT=-1, ANY PRIOR BREAK IS REMOVED.
                   * IF OLD BREAK DOES NOT CONTAIN (XOP) IT IS
                   * NOT DISTURBED.  SINCE R1 IS  PRESET TO BKRTN, IT
                   * CAN ACT AS A BREAKPOINT REMOVAL.
                   *
0050 0203 0026     BRK     LI    R3,BREAK            ; R3=BREAK POINTER
0054 C033                  MOV   *R3+,R0             ; GET NO. OF WORDS
0056 C123 0008             MOV   @8(R3),R4           ; GET RETURN
005A 6100                  S     R0,R4               ; READJUST IT TO START
005C 6100                  S     R0,R4
005E C154                  MOV   *R4,R5              ; CHECK FOR XOP
0062               BRKXOP  EQU   $+2
0060 0285 2C00             CI    R5,XOP0
0064 1601                  JNE   BRK1                ; IF NOT XOP, SKIP RESTORE
0066 C513                  MOV   *R3,*R4             ; RESTORE CODE
0068 0643          BRK1    DECT  R3                  ; RESET R3
006A 0742                  ABS   R2                  ; IF R2=-1, R2=1
006C CCC2                  MOV   R2,*R3+             ; STORE NO. OF WORDS
006E C111                  MOV   *R1,R4              ; GET INST
0070 0602                  DEC   R2
0072 CC60 0062             MOV   @BRKXOP,*R1+        ; PUT IN XOP
0076 CCC4          BRK2    MOV   R4,*R3+             ; SAVE INST
0078 0204 1000             LI    R4,NOOP             ; PRESET R=NOOP
007C C082                  MOV   R2,R2               ; IF R2 NOT C, GET INST.
007E 1302                  JEQ   BRK3
0080 0602                  DEC   R2
0082 C131                  MOV   *R1+,R4
```

```
0084 0283 002E  BRK3   CI   R3,BREAK+8      ; CONT TILL THREE WORDS SET
0088 16F6              JNE  BRK2
008A CCE0 0332         MOV  @BRANCH,*R3+    ; SET RETURN BRANCH
008E CCC1              MOV  R1,*R3+         ; PUT IN RETURN ADDRESS
0090 0713              SETO *R3             ; R2=-1
0092 102F              JMP  MTR             ; GOTO MONITOR
                *
                * ROUTINE: TYPE
                * TYPE THE RIGHT BYTE OF R4.  AFTER THAT,
                * TYPE THE LEFT BYTE IF IT IS NOT ZERO.
                *
0094 06C4       TYPE   SWPB R4              ; PUT IN LEFT BYTE
0096 2C84       TYPE1  OUT  R4              ; OUTPUT R4
0098 0A84              SLA  R4,8            ; ANOTHER CHAR?
009A 16FD              JNE  TYPE1           ; YES-TYPE IT
009C 045B       TYPEX  B    *R11            ; RETURN
                *
                * ROUTINE: GET
                * PROMPT THE OPERATOR USING (R11), THEN
                * GET AN UPDATED VALUE.  DEFAULT FOR THE
                * TWO ENTRIES IS -1.
                *
009E C13B       GET    MOV  *R11+,R4        ; GET PROMPT
00A0 C68B              MOV  R11,*R10        ; SAVE RETURN
00A2 06A0 0094         BL   @TYPE           ; PROMPT THE OPERATOR
00A6 0701              SETO R1              ; SET DEFAULTS
00A8 C081              MOV  R1,R2
00AA 0205 0007         LI   R5,7            ; GET USER INPUT
00AE 06A0 0222         BL   @RDNUMB
00B2 C2DA              MOV  *R10,R11        ; RESET RETURN
00B4 CE81              MOV  R1,*R10+
00B6 CE82              MOV  R2,*R10+
                *
                * ROUTINE: TYPEN
                * PROCEED TO A NEW LINE ON THE TERMINAL
                * PRINT CR,LF, THEN WAIT
                *
00B8 0204 0A0D  TYPEN  LI   R4,CRLF         ; PRINT THE CRLF
00BB            CRET   EQU  $-1
00BC 10EB              JMP  TYPE
                *
                * ROUTINE: DMEMN
                * DISPLAY R1 ON A NEW LINE IN FORMAT:
                * 'XXXX:'
                *
00BE 2D         DASH   BYTE '-'
00BF 3D         EQUAL  BYTE '='
00C0 C1CB       DMEMN  MOV  R11,R7          ; SAVE EXIT
00C2 06A0 00B8         BL   @TYPEN          ; GOTO NEW LINE
00C6 C141              MOV  R1,R5           ; DISPLAY R1
00C8 06A0 01BE         BL   @TYPEWD         ; DISPLAY
00CC 2CA0 02D5         OUT  @COLON          ; OUTPUT ':'
00D0 0457              B    *R7             ; EXIT
                *
                * ROUTINE: DISRG
                * DISPLAY REGISTER R5 ON CURRENT LINE.
                * TITLE OF THE DISPLAY IS IN (R11).
                *
```

```
 00D2 C13B           DISRG  MOV  *R11+,R4           ; GET TITLE
 00D4 C1CB           DISRA  MOV  R11,R7             ; SAVE EXIT ADDRESS
 00D6 2C84                  OUT  R4                 ; TYPE LEFT BYTE
 00D8 06C4                  SWPB R4
 00DA 2C84                  OUT  R4                 ; TYPE RIGHT BYTE
 00DC 2CA0 00BF             OUT  @EQUAL             ; OUTPUT '='
 00E0 06A0 01BE             BL   @TYPEWD            ; OUTPUT VALUE
 00E4 2CA0 028B             OUT  @SPACE             ; SPACE AND EXIT
 00E8 0457                  B    *R7
                     *
                     * BASIC MONITOR LOOP.  QUERY OPERATOR FOR
                     * DESIRED FUNCTION; GATHER PARAMETERS; AND
                     * TRANSFER CONTROL TO APPROPRIATE ROUTINE.
                     *
 00EA 02E0 0080      MTRN   LWPI MTRWP
 00EE 06A0 00B8             BL   @TYPEN             ; NEW LINE
 00F2 2CA0 0142      MTR    OUT  @QUEST             ; ISSUE PROMPT
 00F6 2C44                  IN   R4                 ; GET REPLY
 00F8 0201 0336             LI   R1,TABC            ; SEARCH TABLE OF COMMANDS
 00FC C281           FINDC  MOV  R1,R10             ; SAVE TABLE POINTER
 00FE C171                  MOV  *R1+,R5            ; GET NEXT TABLE ENTRY
 0100 13F4                  JEQ  MTRN               ; IF ZERO-TABLE EXHAUSTED
 0102 9144                  CB   R4,R5              ; COMPARE TO USER ENTRY
 0104 16FB                  JNE  FINDC              ; IF NO MATCH - CONT. SEARCH
 0106 06A0 021C             BL   @RDNUM             ; GET PARAMETERS
 010A 0284 000D             CI   R4,CR              ; FORCE NEW LINE IF
 010E 1303                  JEQ  NEWL               ;  TERMINATED BY CR OR IF
 0110 C01A                  MOV  *R10,R0            ;  INDICATED BY P. D.
 0112 0A90                  SLA  R0,9
 0114 1702                  JNC  CONT
 0116 06A0 00B8      NEWL   BL   @TYPEN
 011A C005           CONT   MOV  R5,R0              ; R5=ODD NO. IF PARAM. O.K.
 011C 0810                  SRA  R0,1
 011E 17E5                  JNC  MTRN               ; ILLEGAL ENTRY
                     *
                     * WHEN BRANCHING TO THE INDIVIDUAL COMMAND
                     * PROCESOR, THE FOLLOWING INFO IS PROVIDED:
                     *    R5=PARAM DEC, SHIFTED BY NO. OF
                     *       PARAMS INPUT
                     *    R1=PARAMETER ONE (DEFAULT BKRTN)
                     *    R2=PAREMETER TWO (DEFAULT >FFFF)
                     *    R3=PARAMETER THREE (NO SPECIFIC DEFAULT)
                     *
 0120 04CC                  CLR  R12                ; SET CRU BASE
 0122 C2AA 001C             MOV  @MAX(R10),R10      ; BRANCH TO COMMAND
 0126 069A                  BL   *R10               ; PROCESSING ROUTINE
 0128 10E0                  JMP  MTRN               ; RETURN TO LOOP
                     *
                     * ROUTINE:  COPY
                     * COPY MEMORY FROM (R1) TO (R2) INTO (R3)
                     * ANY NUMBER OF BYTES MAY BE MOVED
                     *
 012A 0582           COPY   INC  R2
 012C DCF1           COPY10 MOVB *R1+,*R3+          ; MOVE ONE BYTE
 012E 8081                  C    R1,R2              ; TEST END
 0130 16FD                  JNE  COPY10             ; CONTINUE TILL DONE
 0132 10DF                  JMP  MTR
                     *
```

```
                    * ROUTINE:  SNAP
                    * ESTABLISH PRIOR BREAKPOINT AS A SNAP.
                    * IF NO PARAMETERS ENTERED, USE EXISTING
                    * DATA; OTHERWISE R1= FIRST SNAP, R2= SNAP
                    * INCREMENT, R3= MAXIMUM NO. OF SNAPS.
                    * IF NEW PARAMETERS ENTERED, QUERY OPERATOR
                    * TO GET REGISTERS AND MEMORY TO BE DUMPED
                    *
0134 020A 0032      SNAP    LI   R10,BREAK+12        ; R10=BREAK POINT
0138 CE81                   MOV  R1,*R10+            ; NEXT=R1
013A CE82                   MOV  R2,*R10+            ; INC-R2
013C CE83                   MOV  R3,*R10+            ; SET MAX.=R3
013E 06A0 009E              BL   @GET
0142                QUEST   EQU  $
0142 3F52                   TEXT '?R'
0144 06A0 009E              BL   @GET
0148 3F4D                   TEXT '?M'
014A 10D3                   JMP  MTR                 ; BACK TO MONITOR
                    *
                    * ROUTINE:  BKIN
                    * THIS ROUTINE IS ENTERED VIA A USER BREAK.
                    * IT PRINTS WP, PC, ST. IF A SNAP ENTRY IT ALSO
                    * PRINTS REGISTERS AND MEMORY.
                    *
014C 0201 0028      BKIN    LI   R1,BKRTN            ; R1=BREAK PTR(BREAK+2)
0150 0621 000A              DEC  @10(R1)             ; NEXT=NEXT-1
0154 1303                   JEQ  BKDSP               ; IF ZERO-DISPLAY
0156 11C9                   JLT  MTRN                ; IF LESS-GOTO MONITOR
                    *
                    * ROUTINE: GO
                    * BRANCH TO (R1).  BRANCH VIA A RETURN WITH
                    * WORKSPACE.  GO ASSUMES R1 IS PRESET TO
                    * BKRTN.  R13(WP) MUST BE PRESET DURING POWER-UP
                    *
0158 C381           GO      MOV  R1,R14              ; PC=R1
015A 0380                   RTWP                     ; BRANCH
                    *
                    * AT THIS POINT, A SNAP HAS BEEN ENCOUNTERED.
                    * DISPLAY THE SELECTED REGISTERS AND MEMORY
                    *
015C C14E           BKDSP   MOV  R14,R5              ; PRINT PC
015E 06A0 00B8              BL   @TYPEN              ; ON A NEW LINE
0162 06A0 00D2              BL   @DISRG
0166 5043                   TEXT 'PC'
0168 C14D                   MOV  R13,R5              ; PRINT WP
016A 06A0 00D2              BL   @DISRG
016E 5750                   TEXT 'WP'
0170 C14F                   MOV  R15,R5              ; PRINT ST
0172 06A0 00D2              BL   @DISRG
0176 5354                   TEXT 'ST'
0178 C0A1 0012              MOV  @18(R1),R2          ; GET RD1,RD2
017C C061 0010              MOV  @16(R1),R1
0180 1104                   JLT  BKDSP2              ; IF RD1=-1, NO REG DISP
0182 06A0 00B8              BL   @TYPEN              ; DISPLAY REGISTERS
0186 06A0 01E4              BL   @DISPW
018A 0203 003C      BKDSP2 LI    R3,BREAK+22         ; GET MD1,MD2
018E C073                   MOV  *R3+,R1
0190 0281 FFFF              CI   R1,-1               ; IF MD1=-1, NO DISP.
```

```
0194 1305                 JEQ  BKDSP3
0196 C093                 MOV  *R3,R2              ;  SET THE END
0198 06A0 02AE            BL   @DUMP               ; DUMP
019C 06A0 00B8            BL   @TYPEN
01A0 0201 0028  BKDSP3 LI   R1,BKRTN            ; DEC. MAX
01A4 0621 000E            DEC  @14(R1)
01A8 13A0                 JEQ  MTRN                ; IF ZERO, GOTO MON.
01AA C861 000C            MOV  @12(R1),@10(R1)     ; SET NEXT=INC
01AE 000A
01B0 10D3                 JMP  GO                  ; RET. TO USER
                *
                * ROUTINE: BDISP
                * DISPLAY THE LEFTMOST BYTE OF R5,
                * PRECEEDED BY A SPACE
                *
01B2 2CA0 028B  BDISPS OUT  @SPACE              ; TYPE SPACE
01B6 06C5       BDISP  SWPB R5                  ; PUT DATA IN LOWER BYTE
01B8 0200 0004         LI   R0,4                ; PRINT AND EXIT
01BC 1002              JMP  TYPEH
                *
                * ROUTINE: TYPEH
                * DISPLAY R5 AS A HEX DIGIT STRING
                * THE SHIFT COUNT IN R0 CONTROLS THE NO.
                * OF DIGITS PRINTED (12=4,4=2)
                *
01BE 0200 000C  TYPEWD LI   R0,12
01C2 C105       TYPEH  MOV  R5,R4               ; EXTRACT ONE NIBBLE
01C4 0B04              SRC  R4,R0
01C6 0244 000F         ANDI R4,>F               ; MASK OFF FOUR BITS
01CA 0224 0030         AI   R4,>30              ; ADJUST FOR ASCII
01CE 0284 003A         CI   R4,>3A              ; TEST 'A'-'F' AND
01D2 1102              JLT  TYPEH2              ;    IF SO-READJUST
01D4 0224 0007         AI   R4,7
01D8 06C4       TYPEH2 SWPB R4                  ; TYPE
01DA 2C84              OUT  R4
01DC 0220 FFFC         AI   R0,-4               ; REDUCE SHIFT COUNT
01E0 18F0              JOC  TYPEH               ; CONT. TILL DONE
01E2 045B              B    *R11                ; EXIT
                *
                * ROUTINE: DISPW
                * DISPLAY WORKSPACE R(R1)-R(R2)
                *
01E4 C0CB       DISPW  MOV  R11,R3              ; SAVE RETURN
01E6 0241 000F         ANDI R1,>F               ; FORCE R1=0-F
01EA 6081              S    R1,R2               ; R2=NO. OF REG.
01EC C101       DISPW1 MOV  R1,R4               ; FORM REG NAME
01EE 0224 5230         AI   R4,'R0'
01F2 0284 523A         CI   R4,'R9'+1
01F6 1102              JLT  DISPW2
01F8 0224 0007         AI   R4,7
01FC C141       DISPW2 MOV  R1,R5               ; GET REGISTER
01FE 0A15              SLA  R5,1                ; FORM A WORD ADDRESS
0200 A14D              A    R13,R5
0202 C155              MOV  *R5,R5
0204 06A0 00D4         BL   @DISRA              ; DISPLAY REGISTER
0208 0602              DEC  R2                  ; TEST FOR END
020A 1107              JLT  DISPW3              ; EXIT IF MINUS
020C 0581              INC  R1                  ; ADVANCE REG. COUNT
```

```
020E 0281 0008        CI    R1,8              ; IF REG. 8, THEN
0212 16EC             JNE   DISPW1            ;    GOTO NEW LINE
0214 06A0 00B8        BL    @TYPEN
0218 10E9             JMP   DISPW1
021A 0453      DISPW3 B     *R3               ; EXIT
               *
               * ROUTINE:RDNUM
               * READ PARAMETERS AND PLACE THEM IN REGISTERS
               * R1,R2,R3.  PARAMETER DESCRIPTION IS IN R5 AND
               * IS SHIFTED RIGHT ONE POSITION FOR EACH PARAM.
               * THAT IS READ.  RDNUMA AVOIDS THE PRESET AND
               * INITIAL READ.  RDNUMB AVOIDS THE PRESET ONLY.
               * R1 IS PRESET TO BKRTN AND R2 IS PRESET TO >FFFF.
               *
021C 0201 0028 RDNUM  LI    R1,BKRTN          ; PRESET R1,R2
0220 0702             SETO R2
0222 04C4      RDNUMB CLR   R4                ; FIRST CHAR PRESET
0224 C20B      RDNUMA MOV   R11,R8            ; SAVE RETURN
0226 02A6             STWP R6                 ; R6=WORKSPACE
0228 04C7             CLR   R7                ; RESET FLAG(R7)
022A C004      STRT   MOV   R4,R0             ; TEST INPUT FOR HEX
022C 0220 FFD0        AI    R0,->30           ; R0=INPUT-'0'
0230 1117             JLT   NOHEX             ; IF MINUS, NOT HEX
0232 0280 000A        CI    R0,10             ; CHECK R0=0-9, IF SO
0236 1108             JLT   ADDIN             ;    GOTO ADDIN
0238 0220 FFF9        AI    R0,-7             ; CHECK R0=A-F, IF SO
023C 0280 000A        CI    R0,10             ;    FALL THRU TO ADDIN
0240 110F             JLT   NOHEX
0242 0280 000F        CI    R0,15
0246 150C             JGT   NOHEX
0248 C1C7      ADDIN  MOV   R7,R7             ; R0=NEXT DIG(BINARY)
024A 1603             JNE   NONEW             ; IF FIRST, PRESET VALUE
024C 0587             INC   R7                ; SET FLAG
024E 05C6             INCT R6                 ; ADVANCE TO NEXT VALUE
0250 04D6             CLR   *R6               ; CLEAR NEXT VALUE
0252 C0D6      NONEW  MOV   *R6,R3            ; GET VALUE
0254 0A43             SLA   R3,4              ; MULT. BY 16
0256 A0C0             A     R0,R3             ; ADD NEW DIGIT
0258 C583             MOV   R3,*R6            ; REPLACE VALUE
025A 2C44      CONT1  IN    R4                ; GET CHAR
025C 0984             SRL   R4,8              ; RIGHT JUST.
025E 10E5             JMP   STRT              ; CONTINUE SCAN
               *
               * AT THIS POINT, WE KNOW THAT THE INPUT
               * IS NOT A HEX DIGIT, SO CHECK FOR END
               * OF ENTRY AND END OF INPUT.
               *
0260 C1C7      NOHEX  MOV   R7,R7             ; IF NON NULL ENTRY, THEN
0262 1306             JEQ   TSTND             ;    REVISE THE P.D.
0264 04C7             CLR   R7                ; RESET FLAG
0266 0915             SRL   R5,1              ; UPDATE P.D.
0268 C005             MOV   R5,R0             ; IF P.D.=XX...XX000X, THEN
026A 0240 000E        ANDI R0,>E              ;    RETURN TO CALLER
026E 1303             JEQ   EXIT
0270 0284 000D TSTND  CI    R4,CR             ; IF CR, THEN RETURN
0274 16F2             JNE   CONT1
0276 0458      EXIT   B     *R8               ; RETURN TO CALLER
               *
```

```
                      * ROUTINE: ALTER
                      * DISPLAY (P1); AWAIT OPERATOR UPDATE, IF ANY;
                      * INCREMENT ADDRESS AND CONTINUE.  IF THE
                      * ENTRY IS TERMINATED BY A CR, DISPLAY CURRENT
                      * ADDRESS ON A NEW LINE, THEN THE DATA BYTE.
                      * IF SPACE ENTERED, SKIP UPDATE OF THIS BYTE.
                      *
0278 C081             ALT    MOV  R1,R2            ; SAVE ADDRESS
027A D152             ALT1   MOVB *R2,R5           ; DISPLAY (R2)
027C 06A0 01B6               BL   @BDISP
0280 2CA0 00BE               OUT  @DASH            ; OUTPUT '-'
0284 2C44                    IN   R4               ; GET REPLY
0286 0984                    SRL  R4,8
0288 0284 0020               CI   R4,' '           ; IF ' ', SKIP UPDATE
028B                  SPACE  EQU  $-1
028C 1308                    JEQ  ALT2
028E 0205 0002               LI   R5,2             ; READ FULL REPLY
0292 D052                    MOVB *R2,R1           ; SET DEFAULT
0294 06C1                    SWPB R1
0296 06A0 0224               BL   @RDNUMA          ; GET REPLY
029A 06C1                    SWPB R1               ; ALTER (R2)
029C D481                    MOVB R1,*R2
029E 0582             ALT2   INC  R2               ; ADV. ADDR POINTER
02A0 0284 000D               CI   R4,CR            ; IF TERMINATED BY CR, THEN
02A4 16EA                    JNE  ALT1             ;    TYPE CURRENT ADDRESS
02A6 C042                    MOV  R2,R1
02A8 06A0 00C0               BL   @DMEMN
02AC 10E6                    JMP  ALT1
                      *
                      * ROUTINE: DUMP
                      * DUMP THE MEMORY FROM (R1) TO (R2).  IF
                      * CALLED FROM MONITOR LOOP, DUMP WILL RETURN
                      * TO MTRN; OTHERWISE IT RETURNS TO CALLING
                      * ROUTINE.
                      *
02AE C0CB             DUMP   MOV  R11,R3           ; SAVE RETURN
02B0 06A0 00C0        DUMP1  BL   @DMEMN           ; DISPLAY ADDRESS
02B4 D171             DUMP2  MOVB *R1+,R5          ; GET NEXT BYTE
02B6 06A0 01B2               BL   @BDISPS          ; DISPLAY IT SPACE FIRST
02BA 8081                    C    R1,R2            ; CHECK END
02BC 1BAE                    JH   DISPW3           ; IF NOT END-CONTINUE
02BE C141                    MOV  R1,R5            ; IF R1=0-EXIT, IF R1 MULT 16
02C0 13AC                    JEQ  DISPW3
02C2 0AC5                    SLA  R5,12            ;    THEN DISP ADDRESS,
02C4 13F5                    JEQ  DUMP1            ;    ELSE CONTINUE
02C6 10F6                    JMP  DUMP2            ; CONTINUE DUMP
                      *
                      * ROUTINE: LOAD
                      * LOAD A MONITOR DUMP BACK TO RAM
                      *
02C8 C081             LOAD   MOV  R1,R2            ; R2=LOAD ADDRESS
02CA 0205 0002        LOAD1  LI   R5,2             ; READ VALUE
02CE 06A0 0222               BL   @RDNUMB
02D2 0284 003A               CI   R4,':'           ; IF TERM. BY ':' RESET R3
02D5                  COLON  EQU  $-1
02D6 13F8                    JEQ  LOAD
02D8 06C1                    SWPB R1               ; DATA IN LEFT BYTE
02DA DC81                    MOVB R1,*R2+          ; STORE ONE BYTE
```

```
02DC 10F6             JMP   LOAD1            ; CONTINUE
                *
                * ROUTINE: INSPECT
                * INSPECT A CRU BIT (R1)
                *
02DE 0A11       INSP  SLA   R1,1             ; ALIGN FOR CRU BASE
02E0 C301             MOV   R1,R12           ; PUT IN CRU BASE
02E2 0204 3031        LI    R4,'01'          ; SET R4=0/1
02E6 1F00             TB    0
02E8 1601             JNE   INSP1
02EA 06C4             SWPB  R4
02EC 2C84       INSP1 OUT   R4               ; DISPLAY THE BIT
02EE 045B             B     *R11             ; BACK TO MONITOR
                *
                * ROUTINE: MODIFY
                * MODIFY A CRU BIT (R1) TO BE (R2)
                *
02F0 0A11       MODIF SLA   R1,1             ; ALIGN FOR CRU BASE
02F2 C301             MOV   R1,R12           ; SET CRU BASE
02F4 06C2             SWPB  R2               ; PUT BIT IN UPPER BYTE
02F6 3042             LDCR  R2,1             ; OUTPUT ONE BIT
02F8 0460 00F2  MODIF1 B    @MTR             ; BACK TO MONITOR
                *
                * ROUTINE: PROG
                * PROGRAM ROM.  SOURCE IS (R1)-(R2).
                * ROM TARGET IS (R3)
                *
02FC 1E01       PROG  SBZ   PRG              ; ENABLE
02FE 05C2             INCT  R2
0300 0242 FFFE        ANDI  R2,>FFFE         ; FORCE EVEN ADDR.
0304 0204 00C8        LI    R4,200           ; REPEAT COUNT
0308 C141       PROG1 MOV   R1,R5            ; SAVE INPUT
030A C183             MOV   R3,R6
030C 0266 F000        ORI   R6,>F000         ; ADJUST FOR ROM
0310 CDB5       PROG2 MOV   *R5+,*R6+         ; PROG ONE WORD
0312 8085             C     R5,R2
0314 16FD             JNE   PROG2            ; CONT. THIS PASS
0316 0604             DEC   R4
0318 16F7             JNE   PROG1            ; NEXT PASS
031A 1D01             SBO   PRG              ; DISABLE PROG.
031C 10ED             JMP   MODIF1           ; BACK TO MONITOR
                *
                * ROUTINE: HEX
                * PRINT R1+R2 AND R1-R2
                *
031E C141       HEX   MOV   R1,R5            ; SUM
0320 A142             A     R2,R5
0322 06A0 00D2        BL    @DISRG
0326 482B             TEXT  'H+'
0328 C141             MOV   R1,R5            ; DIFFERENCE
032A 6142             S     R2,R5
032C 06A0 00D2        BL    @DISRG
0330 482D             TEXT  'H-'
0332 0460 00EA  BRANCH B    @MTRN
                *
                * COMMAND TABLE
                *
0336 4102       TABC  BYTE 'A',>02           ; ALTER
```

```
0338 4287                  BYTE 'B',>87            ; BREAKPOINT
033A 4388                  BYTE 'C',>88            ; COPY
033C 4406                  BYTE 'D',>06            ; DUMP
033E 4783                  BYTE 'G',>83            ; GO
0340 4884                  BYTE 'H',>84            ; HEX ARITH.
0342 4902                  BYTE 'I',>02            ; INSPECT
0344 4C83                  BYTE 'L',>83            ; LOAD
0346 4D84                  BYTE 'M',>84            ; MODIFY
0348 5088                  BYTE 'P',>88            ; PROGRAM
034A 5203                  BYTE 'R',>03            ; RETURN
034C 5388                  BYTE 'S',>88            ; SNAP
034E 5786                  BYTE 'W',>86            ; WORKSPACE DUMP
0350 0000                  BYTE 0,0                ; END OF TABLE
0352 0278 0050             DATA ALT,BRK,COPY,DUMP
0356 012A 02AE
035A 0158 031E             DATA GO,HEX,INSP,LOAD
035E 02DE 02C8
0362 02F0 02FC             DATA MODIF,PROG,RETN,SNAP
0366 03E8 0134
036A 01E4                  DATA DISPW
036C 0080 014C   XOPTB     DATA MTRWP,BKIN
0370 0090 039C             DATA XOPWS,ROUT2
0374 0090 037A             DATA XOPWS,ROUT1
0378 0000                  DATA 0
                 *
                 * XOP ROUTINES
                 * XOP-1 = INPUT
                 * XOP-2 = OUTPUT
                 *
                 *
                 * ROUTINE: ROUT1 (TERMINAL OUTPUT)
                 * OUTPUT THE BYTE AT (R11).  IF IT IS
                 * A CARRIAGE RETURN, DELAY ACCORDING TO
                 * THE VALUE (DELAY)
                 *
037A 020A 03EC   ROUT1     LI   R10,WAITA          ; R10=INDEX TO WAIT
037E D21B                  MOVB *R11,R8            ; R8=CHARACTER
0380 069A                  BL   *R10               ; TWO STOP BITS
0382 069A                  BL   *R10
0384 0209 0008             LI   R9,8               ; R9=CHARACTER COUNT
0388 1E00                  SBZ  TTYO               ; START BIT
038A 069A                  BL   *R10               ; WAIT FOR START BIT
038C 3048        R120      LDCR R8,1               ;(22) OUTPUT ONE BIT
038E 069A                  BL   *R10               ;(16) WAIT FOR IT
0390 0918                  SRL  R8,1               ;(14) GET NEXT BIT
0392 0609                  DEC  R9                 ;(10) CONTINUE TILL DONE
0394 16FB                  JNE  R120               ;(10)
0396 1D00                  SBO  TTYO               ; STOP BIT
0398 06C8                  SWPB R8                 ; REPOSITION BYTE
039A 1018                  JMP  R250               ; GO CHECK BREAK, ETC.
                 *
                 * ROUTINE: ROUT2 (TERMINAL ECHO)
                 * INPUT ONE CHARACTER FROM TERMINAL AND
                 * RETURN IT IN (R11).  IF CARRIAGE RETURN
                 * DELAY ACCORDING TO THE VALUE IN (DELAY)
                 *
039C 04CC        ROUT2     CLR  R12                ; RESET CRU BASE
039E C28B                  MOV  R11,R10            ; SAVE POINTER
```

```
 03A0 3448      R210   STCR R8,1          ; WAIT FOR START
  3A2 16FE             JNE  R210
 03A4 C2E0 0024        MOV  @>24,R11      ; WAIT ABOUT 1/2
 03A8 064B      R220   DECT R11
 03AA 15FE             JGT  R220
 03AC 0209 0009        LI   R9,9          ; R9=BIT COUNT
 03B0 C2E0 0022 R230   MOV  @>22,R11      ;(22) ECHO?
 03B4 1501             JGT  R240          ;(8) NO
 03B6 3048             LDCR R8,1          ;(22) YES
 03B8 06A0 03F0 R240   BL   @WAITB        ;(20) WAIT ONE BIT
 03BC 0918             SRL  R8,1          ;(14) REPOSITION
 03BE 3448             STCR R8,1          ;(42) GET ONE BIT
 03C0 0609             DEC  R9            ;(10) CONTINUE TILL DONE
 03C2 16F6             JNE  R230          ;(10)
 03C4 1D00             SBO  TTYO          ; STOP BIT
 03C6 0A98             SLA  R8,9          ; STRIP PARITY
 03C8 0918             SRL  R8,1
 03CA D688             MOVB R8,*R10       ; STORE CHAR
 03CC 9220 00BB R250   CB   @CRET,R8      ; RETURN?
 03D0 1604             JNE  R270          ; NO
 03D2 C320 0020        MOV  @>20,R12      ; YES-DELAY
 03D6 060C      R260   DEC  R12
 03D8 15FE             JGT  R260
 03DA 1F00      R270   TB   TTYI          ; BREAK?
 03DC 1306             JEQ  R290          ; NO
 03DE 1F00      R280   TB   TTYI          ; WAIT TILL END
  3E0 16FE             JNE  R280
  3E2 028E 0000        CI   R14,START     ; IN MONITOR?
 03E6 14A5             JHE  BRANCH        ; YES-NO BREAK
 03E8 041C      RETN   BLWP *R12          ; RESET (R12 IS ZERO)
 03EA 0380      R290   RTWP               ; RETURN TO CALLER
                *
                * ROUTINE: WAIT
                * WAIT ONE BIT TIME
                * WAITA IS FOR OUTPUT AND WAITB
                * IS FOR INPUT
                *
 03EC 09FC      WAITA  SRL  R12,15        ; 72CYCLES HERE, SO
 03EE 09BC             SRL  R12,11        ; USE 76 MORE
 03F0 C320 0024 WAITB  MOV  @>24,R12      ; 148 CYCLES HERE,
 03F4 1F00      WAITC  TB   0             ;(12) DUMMY
 03F6 060C             DEC  R12           ;(10) 32CYCLES/COUNT
 03F8 15FD             JGT  WAITC         ;(10)
 03FA 045B             B    *R11          ; EXIT
 03FC 0080             DATA MTRWP
 03FE 0000             DATA START         ; LOAD VECTOR
 0400                  END
```

```
0248 ADDIN    0278 ALT      027A ALT1     029E ALT2     01B6 BDISP
   2 BDISPS   015C BKDSP    018A BKDSP2   01A0 BKDSP3   014C BKIN
0028 BKRTN    0332 BRANCH   0026 BREAK    0050 BRK      0068 BRK1
0076 BRK2     0084 BRK3     0062 BRKXOP   02D5 COLON    011A CONT
025A CONT1    012A COPY     012C COPY10   000D CR       00BB CRET
0A0D CRLF     00BE DASH     0020 DELAY    01E4 DISPW    01EC DISPW1
01FC DISPW2   021A DISPW3   00D4 DISRA    00D2 DISRG    00C0 DMEMN
```

```
 02AE DUMP     02B0 DUMP1    02B4 DUMP2    00BF EQUAL    0276 EXIT
 00FC FINDC    009E GET      0158 GO       031E HEX     *0000 IDTMM
 02DE INSP     02EC INSP1    02C8 LOAD     02CA LOAD1    001C MAX
 02F0 MODIF    02F8 MODIF1   00F2 MTR      00EA MTRN     0080 MTRWP
 0116 NEWL     0260 NOHEX    0252 NONEW    1000 NOOP     0001 PRG
 02FC PROG     0308 PROG1    0310 PROG2    0142 QUEST    0000 R0
 0001 R1       000A R10      000B R11      000C R12      038C R120
 000D R13      000E R14      000F R15      0002 R2       03A0 R210
 03A8 R220     03B0 R230     03B8 R240     03CC R250     03D6 R260
 03DA R270     03DE R280     03EA R290     0003 R3       0004 R4
 0005 R5       0006 R6       0007 R7       0008 R8       0009 R9
 021C RDNUM    0224 RDNUMA   0222 RDNUMB   03E8 RETN     037A ROUT1
 039C ROUT2    0134 SNAP     028B SPACE    0000 START    022A STRT
 0012 STRT10   0030 STRT20   0034 STRT30   0336 TABC     0270 TSTND
 0000 TTYI     0000 TTYO     0094 TYPE     0096 TYPE1    01C2 TYPEH
 01D8 TYPEH2   00B8 TYPEN    01BE TYPEWD   009C TYPEX    00D0 USRWP
 03EC WAITA    03F0 WAITB    03F4 WAITC    2C00 XOP0     036C XOPTB
 0090 XOPWS
EDIT/ASM/LOAD?
```