MOS

MICROCOMPUTERS

TIM
TERMINAL INTERFACE MONITOR
MANUAL

Publication Number 6500-20

MCS6500

MICROCOMPUTER FAMILY

TIM MANUAL

MARCH, 1976

Second Edition

MOS TECHNOLOGY, INC.
950 Rittenhouse Road
Norristown, PA. 19401

TABLE OF CONTENTS

I. INTRODUCTION

TIM is the Terminal Interface Monitor program for MOS Technology's 65XX microprocessors. It is supplied in read-only memory (ROM) as part of the MCS6530-004 multi-function chip. Because the TIM code is nonvolatile, it is available at system power-on and cannot be destroyed inadvertently by user programs. Furthermore, the user is free to use only those TIM capabilities which he needs for a particular program. Both interrupt types, interrupt request (IRQ) and nonmaskable interrupt (NMI) may be set to transfer control to TIM or directly to the user's program.

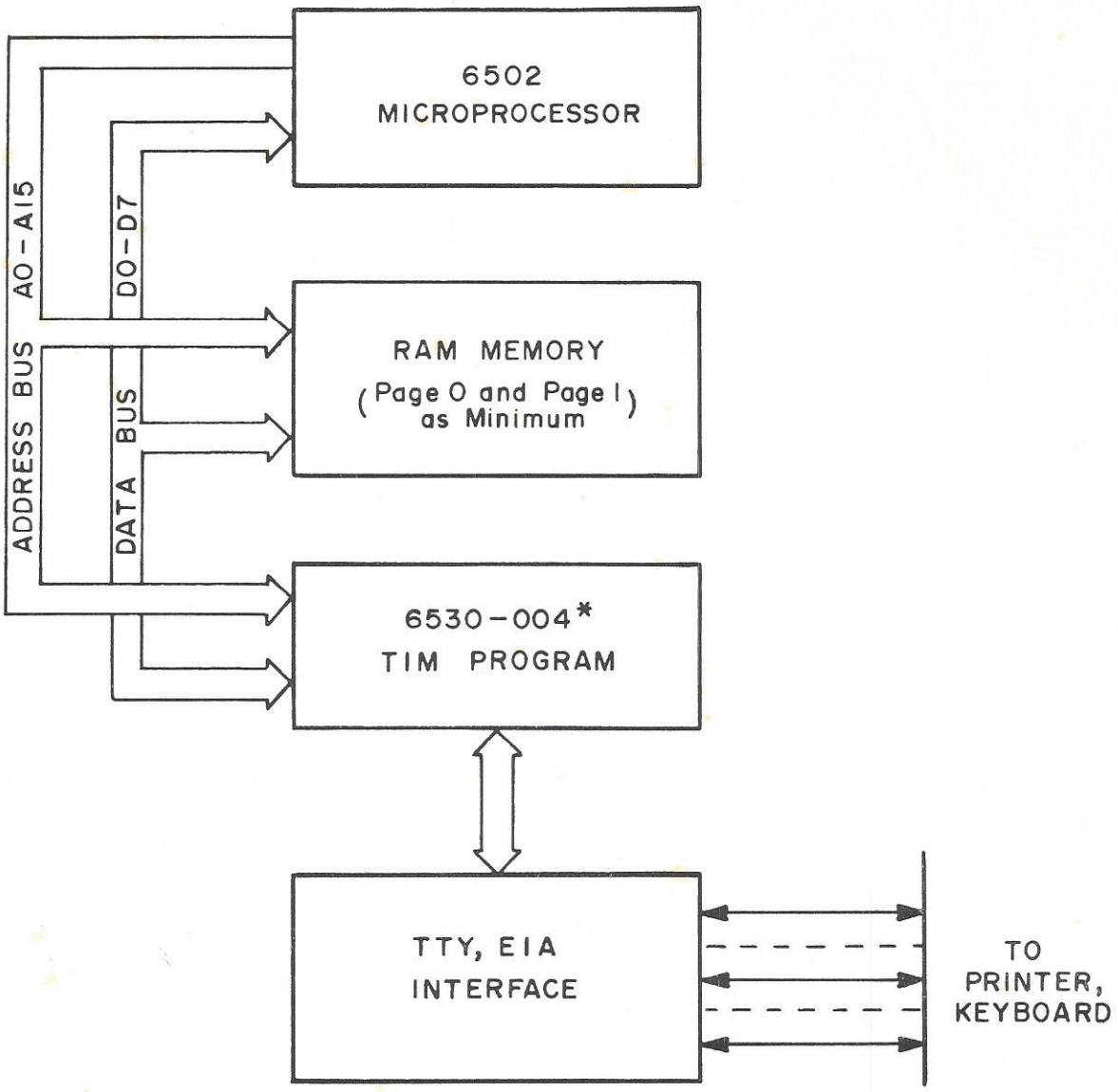
TIM communicates with the user via a serial full-duplex port (using ASCII codes) and automatically adjusts to the speed of the user's terminal. Any speed--even nonstandard ones--can be accommodated. If the user's terminal has a long carriage return time, TIM can be set to perform the proper delay. Commands typed at the terminal can direct TIM to start a program, display or alter registers and memory locations, set breakpoints, and load or punch programs. If available in the system configuration, a high-speed paper tape reader may be used to load programs through a parallel port on the MCS6530-004 chip. Programs may be punched in either of two formats--hexadecimal (assembler output) or BNPF (which is used for programming read-only memories). On loading or modifying memory, TIM performs automatic read-after-write verification to insure that addresses memory exists, is read/write type, and is responding correctly. Operator errors and certain hardware failures may thus be detected using TIM.

TIM also provides several subroutines which may be called by user programs. These include reading and writing characters on the terminal, typing a byte in hexadecimal, reading from high-speed paper tape, and typeing a carriage-return, line-feed sequence with proper delay for the carriage of the terminal being used. Program tapes loaded by TIM may also specify a start address so that programs may be started with a minimum of operator action.

II. SYSTEM CONFIGURATION

Since TIM is a "program" resident in the MCS6530-004 it must be properly configured in a proper system environment.

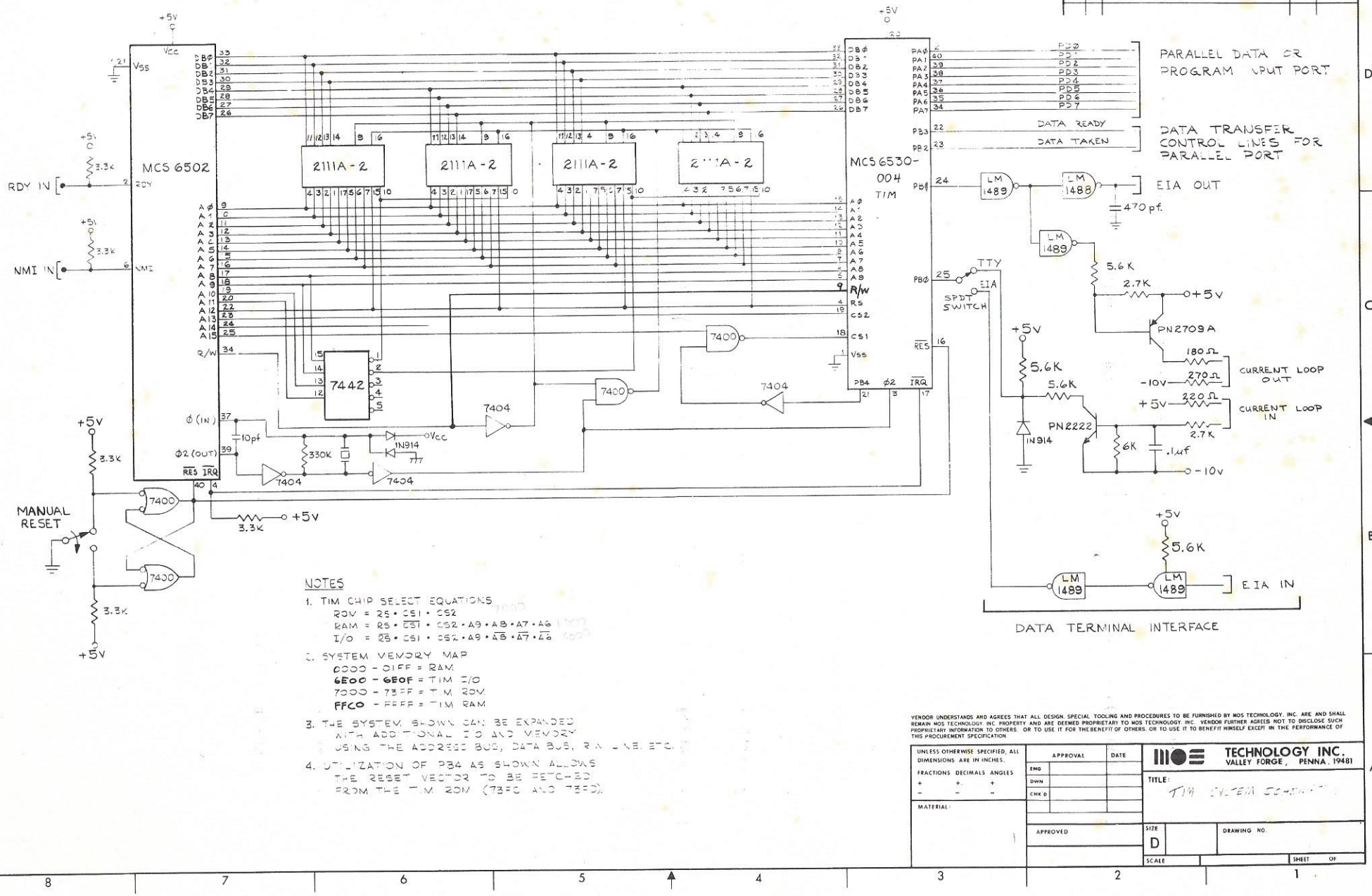
Figure 2-1 represents a block diagram of a minimum system utilizing the TIM program. The MCS6502 is the controlling microprocessor with two pages of memory (pages 0 and 1) representing the minimum RAM requirement. These devices, as well as a representative schematic for the TTY, EIA interfaces, are shown in Figure 2-2 which is a detailed system schematic utilizing the MCS6530-004. Note that the TIM function select equations are found on this schematic.



* Note that the TIM as sold consists only of the MCS6530-004 component accompanied by supporting information to build this system

TYPICAL MINIMUM CONFIGURATION
FOR "TIM" SYSTEM

FIGURE 2-1



NOTES

1. TIM CHIP SELECT EQUATIONS
   ROM = RS • CS1 • CS2
   RAM = RS • $\overline{CS1}$ • CS2 • A9 • A8 • A7 • A6
   I/O = $\overline{RS}$ • CS1 • CS2 • A9 • $\overline{A8}$ • $\overline{A7}$ • $\overline{A6}$

2. SYSTEM MEMORY MAP
   0000 - 03FF = RAM
   6E00 - 6E0F = TIM I/O
   7000 - 73FF = TIM ROM
   FFC0 - FFFF = TIM RAM

3. THE SYSTEM SHOWN CAN BE EXPANDED WITH ADDITIONAL I/O AND MEMORY USING THE ADDRESS BUS, DATA BUS, R/W LINE, ETC.

4. UTILIZATION OF PB4 AS SHOWN ALLOWS THE RESET VECTOR TO BE FETCHED FROM THE TIM ROM (73FC AND 73FD).

REVISIONS			
ZONE	LTR	DESCRIPTION	DATE	APPROVED

UNLESS OTHERWISE SPECIFIED, ALL DIMENSIONS ARE IN INCHES.
FRACTIONS DECIMALS ANGLES
+ + +
− − −

MATERIAL:

APPROVAL	DATE
ENG	
DWN	
CHK'D	
APPROVED	

MOS TECHNOLOGY INC.
VALLEY FORGE, PENNA. 19481

TITLE: TIM SYSTEM SCHEMATIC

SIZE D | DRAWING NO.

SCALE | SHEET OF

III. OPERATIONAL FEATURES OF TIM

A. TIM Commands*

Command	Description
↵	Set line speed. After RESET, a carriage return is typed to allow TIM to measure the line speed.
.R	Display user registers. The format is: PC P A X Y S where: PC is the program counter P is the processor status A is the A (accumulator) register X is the X (index) register Y is the Y (index) register S is the stack pointer low byte (high byte is always 01)
.G	Go. Begin execution at user PC location (see R command).
.M addr	Memory examine. TIM will display the eight bytes beginning at address addr.
.: ADDR data	Alter registers or memory. TIM allows the user to alter registers (if R command precedes) or memory (if M command precedes). Values for registers or memory locations which are not to be changed need not be typed

* Characters typed by the user are underlined. All other characters are typed by the computer. ↵ means carriage-return.

—these fields may be skipped by typing spaces instead of data. The remainder of the fields in a line may be left unchanged by typing carriage return. The : command may be repeated to alter subsequent memory locations without the necessity of typing intervening M commands. Note that TIM automatically types spaces to separate data fields.

.LH

Load Hexadecimal. TIM responds with carriage return, line-feed and loads data in assembler output format from the terminal or high-speed paper tape reader. The format is:

Zero or more leading characters except ";" (usually blank leader)

Any number of records of the form:

;ccaaaadddd....ddssss

where:

cc is the number of bytes in the record in hex

aaaa is the hex address to store the first byte of data

dddd....dd is the data (two hex digits per byte)

ssss is the check-sum, which is the arithmetic sum, to 16 bits, of all the count, address and data bytes represented by the record

A terminating record of zero length, either: ;00 or ;↓

Note that read-after-write and check-sum tests are performed. An error will result in a "?" being typed at the point the error occurred. Data from records with bad check-sums is deposited in memory as received, prior to the error stop.

.H — High-speed/low-speed reader switch. This command switches the load device from the user's terminal to the high-speed reader or vice versa.

.WH addl addh↓ — Write Hexadecimal. An assembler-format tape is generated at the user's terminal. Format is as described above in the LH command description. Note that the address range must be specified with the lower address first. As in the Alter command, TIM types the space between the address fields.

.WB addl addh↓ — Write BNPF. A BNPF format tape is generated at the user's terminal. Format is one or more records as follows:

```
aaaa BddddddddF BddddddddF BddddddddF BddddddddF
```

where:

aaaa is the address of the first of the four bytes specified in the record. (Note: BNPF conventions require that the letter "B" never occur in the address field. Blanks are substituted by TIM.)

B is the letter "B", meaning begin data.

dddddddd is eight data bits—P for logical true, N for logical false.

F is the letter "F", meaning finish.

Note that the BNPF format is output as multiples of four bytes. Thus, a multiple of four bytes will always be punched even if a non-multiple of four bytes is specified.

Cancel Command. While typing any command, its further effect may normally be terminated by typing one or two carriage returns, as required. During alter (:), carriage return means that no further bytes (or registers) are to be altered.

B. TIM Interrupt and Breakpoint Action

BRK

The BRK instruction causes the CPU to interrupt execution, save PC and P registers on the stack. and branch through a vector at locations FFFE and FFFF. TIM initializes this vector to point to itself on RESET. Unless the user modifies this vector, TIM will gain control when a BRK instruction is executed, print an asterisk "*" and the registers (as in R command), and wait for user commands. Note that after a BRK which vectors to TIM, the user's PC points to the byte following the BRK; however, users who choose to handle BRK instructions themselves

should note that BRK acts as a two-byte instruction, leaving the PC (on return via RTI) two bytes past the BRK instruction.

IRQ

Interrupt Request is also vectored through location FFFE. The CPU traps (as with BRK) through this vector when IRQ goes low, provided interrupts are not inhibited. Since this vector is the same as for BRK, TIM examines the BRK bit in the P register after this type of interrupt. If a BRK did not cause the interrupt, then TIM will pass control through the UINT vector. Users should normally put the address of their interrupt service routine in the UINT vector location. If an IRQ occurs and UINT has not been set by the user, TIM reports the unexpected interrupt in the same way as an NMI (see below).

NMI

Non-Maskable Interrupts vector through location FFFA. TIM initializes this vector at RESET to point to itself. If an NMI occurs, a pound-sign character (#) precedes the asterisk and CPU registers printout. This action is the same for IRQ's if the user has not set this vector to point to his own routine.

RESET or POWER-UP

On RESET or POWER-UP, TIM takes control, initializes itself and the system, sets defaults for interrupt vectors and waits for a carriage-return input from the user to determine terminal line speed. After carriage-return is typed, control is passed to the user as in BRK.

C. TIM Monitor Calls and Special Locations

Call	Address	Action	Arg.	Result	Notes
JSR WRT	72C6	Type a character	A	None	A,X cleared Y preserved
JSR RDT	72E9	Read a character	None	A	X cleared Y not preserved
JSR CRLF	728A	Type CR-LF and delay	None	None	A,X cleared Y preserved
JSR SPACE	7377	Type a space character	None	None	A,X,Y preserved
JSR WROB	72B1	Type a byte in hex	A	None	A,X cleared Y preserved
JSR RDHSR	733D	Read a character from high-speed paper tape reader	None	X—char read A—char trimmed to 7 bits	Y preserved

Function	Locations	Notes
Start Address	00F6,00F7	Set with hex tape on load
CR-LF Delay	00E3	Set on load or with user program (in bit times, minimum of 1. Zero means 256 bits-time delay).
UINT	FFF8	User IRQ vector
NMI Vector	FFFA	Hardware NMI vector
RESET Vector	FFFC	Hardware RESET vector
IRQ Vector	FFFE	Hardware IRQ vector

D. TIM Memory Usage

TIM uses the top 29_{10} bytes of page zero (locations 00E3 through 00FF). The user is advised to avoid these locations, except as noted above, if return to TIM or use of TIM subroutines is required before RESETing the processor. TIM also uses the hardware stack when it is in control. Provided the user does not alter the stack pointer during a break, and provided the stack does not overflow, TIM will restore the stack to its original status before returning to the user's program. The user is advised to use page 1 (the stack page) cautiously, leaving at least 20_{10} bytes for TIM use during a break or when using other TIM functions.

IV. TIM CHECKOUT PROCEDURE

The following step-by-step procedure assumes the user has built the TIM hardware system and is now ready to verify its functionality.

( ) 1. Turn power on, or if the power is on, perform a RESET operation. Type a carriage-return on the terminal. TIM should respond with:

```
*  7052  30  18  FF  01  FF
```

(Exact values may vary, although the first and last values should be as shown). If no response or a garbled response occurs, RESET and try again. In case of continued trouble, refer to the diagnostic section of the MOS Hardware Manual.

The "* 7052 30 18 FF 01 FF" printout is TIM's standard breakpoint message format. It consists of an asterisk "*" to identify the breakpoint printout, followed by the CPU register contents in this order: PC, P, A, X, Y, and S, i.e., Program Counter, Processor Status, Accumulator, X index, Y index and Stack Pointer. Note that all TIM inputs and outputs are in base 16 which is referred to as hexadecimal, or just hex. In hexadecimal, the "digits" are 0, 1, 2,..., A, B, C, D, E. F. After printing the CPU registers, TIM is ready to receive commands from you, the operator. TIM indicates this "ready" status by typing the prompting character "." on a new line.

( ) 2. TIM's response to RESET is to wait for a carriage-return and then print the user's registers. TIM uses this carriage-return character to measure the terminal line speed. If you have a settable-rate terminal, change the

rate (any speed between 10 and 30 cps will work) and repeat Step 1. TIM should respond at the new terminal speed.

( ) 3. The user's CPU registers may also be displayed with the R command. Type an R. The monitor should respond as above, but without the asterisk. Presence of the asterisk indicates that an interrupt or break instruction caused the printout.

```
.R  7052  30  18  FF  01  FF
```

( ) 4. Displayed values may be modified using the Alter (:) command. To modify register contents, type a colon (:) followed by the new values. For example:

```
.R  7052  30  18  FF  01  FF
.:  0100  00  00  00  00  FF
.R  0100  00  00  00  00  FF
```

Notice that TIM automatically types spaces to separate data fields. (Note: Characters typed by you, the user, are underlined in this document for clarity. Everything else is typed by the computer.) Examine your registers (R command) to verify the changes.

Memory may be examined and modified, as above, using the M and : commands. Try this:

```
.M  0100  00  66  23  EE  01  A2  41  6E
```

The memory command (M) causes TIM to type the contents of the first eight bytes of memory. (Memory data will be random on startup). Alter and verify these bytes using the Alter command, as above:

```
.M   0100  00  66  23  EE  01  A2  41  6E
.:   0100  00  01  02  03  04  05  06  07
```

If only part of a line is to be altered, items to be left unchanged can be skipped over by typing blanks, and carriage-return (↓). Try this:

```
.M   0100  00  01  02  03  04  05  06  07
.:   0100  FF  __  FF  FF  ↓
.M   0100  FF  01  FF  FF  04  05  06  07
```

( ) 5. Try to alter a location in TIM ROM:

```
.M   7000  85  F9  A9  23  D0  58  A9  16
.:   7000  00?
```

TIM verifies all changes to memory. Since locations 7000 through 7007 are in read-only memory, alteration is not possible. TIM signals write failure with a question mark. Similarly, the monitor will notify you of an attempt to alter a non-existant location:

```
.M   9000  90  90  90  90  90  90  90  90
.:   9000  00?
```

Note that attempts to read non-existant memory will normally yield the high-order byte of the address read.

( ) 6. There are three hardware facilities which may be used to stop a running (or run-away) program without the program itself calling TIM . These are the hardware inputs RESET,

IRQ, and NMI. To test this feature enter the following program at location 0000:

location	contents	instruction
0000	4C	LOOP JMP LOOP
0001	00	
0002	00	

(Use the M and : commands.)

Now, set the program counter (PC) to this location using the R and : commands. Finally, tell TIM to start executing your program using the Go (G) command:

```
.M  0000  FF  11  11  11  91  91  71  91
.:  0000  4C  00  00  ↓
.M  0000  4C  00  00  11  91  91  71  91
.R  0000  30  00  00  00  FF
.:  0000  ↓
.G
```

The computer should now be executing the program. It will continue to run until interrupted. Using the interrupt request line (IRQ), interrupt the processor. It should respond with:

```
*  0000  30  00  00  00  FF
```

Try the same experiment with non-maskable interrupt (NMI). The result should be the same except for a "#" character preceeding, which identifies the NMI printout. Finally, try it with RESET. RESET, however, forces a CPU branch to TIM, losing the old PC and other register contents. Thus NMI is the preferred means for manually interrupting program execution. IRQ may also be

used unless it is required for other functions such as peripheral interrupts.

( ) 7. Use M and : to enter the following test program called CHSET because it prints the character-set on the terminal. Note that Alter (:) commands may be repeated without intervening M commands to set sequential locations:

```
                        ;CHECKOUT PROGRAM -- PRINT THE CHARACTER SET ON USER TERMINAL
                        CRLF    =$728A          ;ADDRESS OF TIM CRLF ROUTINE
                        WRT     =$72C6          ;ADDRESS OF TIM WRITE ROUTINE
                        ;
0000                            *=0             ;VARIABLE STORAGE IN PAGE ZERO
0000                    CHAR    *=*+1           ;STORAGE FOR CHARACTER
                        ;
0001                            *=$0100         ;PROGRAM STARTS ON PAGE ONE
                        ;
0100  20 8A 72          CHSET   JSR CRLF        ;DO CARRIAGE RETURN & LINE FEED
0103  A9 20                     LDA #$20        ;FIRST CHAR IS A SPACE
0105  85 00                     STA CHAR        ;INITIALIZE
                        ;
0107  A5 00             LOOP    LDA CHAR        ;GET CHARACTER
0109  C9 60                     CMP #$60        ;CHECK FOR LIMIT
010B  F0 08                     BEQ DONE        ;DONE IF 60
                        ;
010D  20 C6 72                  JSR WRT         ;PRINT CHAR
0110  E6 00                     INC CHAR        ;NEXT CHAR CODE
0112  4C 07 01                  JMP LOOP        ;CONTINUE
                        ;
0115  00                DONE    BRK             ;STOP & RETURN TO TIM MONITOR
                        ;
0116  4C 00 01                  JMP CHSET       ;DO IT AGAIN
```

```
.M   0100  20  8D  72  20  EC  72  8D  26
.:   0100  20  8A  72  A9  20  85  00  A5
.:   0108  00  C9  60  F0  08  20  C6  72
.:   0110  E6  00  4C  07  01  00  4C  00
.:   0118  01  ↓
```

Now run the program. Do this by setting the PC to 0100 and using the G command. The listing should look like this:

```
.R  0000 30 00 00 00 FF
.:  0100 ↓
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
```

The program may be continued, causing it to execute again, by typing G:

```
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
.G
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
 *  0116 33 60 00 00 FF
```

The CHSET program uses two TIM monitor functions: CRLF is the TIM function which causes a carriage-return and line-feed to be typed on the terminal. WRT is the routine which prints the character whose code is in the A register at the time of the call.

( ) 8. Save the CHSET program on paper tape (if your

terminal has a punch) as follows: First, punch some leader tape with the terminal in local mode. Then return to line mode and enter:

.WH 0100 0118 ↓

Turn the punch on after typing the second address, but before typing carriage-return. Then type carriage-return to punch the tape. When punching stops, turn the terminal back to local and type:

;00

and some blank trailer. This is a zero-length record which terminates your tape. See Appendix II for more information on tape formats.

( ) 9. Try re-loading your program using the LH command:

.LH

Now start the reader to load the program. The tape will be read and printed simultaneously. Stop the tape when the end is reached. (Before loading, you may wish to destroy the program in memory to verify that loading from tape actually works.)

( ) 10. Use the M and : commands to load the following program:

```
                        ;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
                        ;
                        ;
0000                    .       *=0             ;VARIABLE STORAGE IN PAGE ZERO
0000                    BINARY  *=*+1           ;STORAGE FOR CHAR DURING DISSECTION
0001                    COUNT   *=*+1           ;COUNT OF BITS REMAINING TO PRINT
                        ;
0002                            *=$0100         ;PROGRAM BEGINS ON PAGE ONE
                        ;
                        CRLF    =$728A           ;TIM  CRLF ROUTINE
                        WRT     =$72C6           ;TIM  WRITE ROUTINE
                        RDT     =$72E9           ;TIM  READ ROUTINE
                        SPACE   =$7377           ;TIM  SPACE ROUTINE
                        ;
0100  20 8A 72          PBIN    JSR CRLF        ;PRINT CARRIAGE RETURN & LINE FEED
0103  20 E9 72                  JSR RDT         ;GET A CHARACTER
0106  85 00                     STA BINARY      ;SAVE FOR DISSECTION
0108  20 77 73                  JSR SPACE       ;PRINT A SPACE
                        ;
010B  A9 08                     LDA #8          ;INITIALIZE BIT COUNT
010D  85 01                     STA COUNT
                        ;
010F  A9 30             PBLOOP  LDA #'0         ;ASSUME ZERO: LOAD ASCII "0"
0111  06 00                     ASL BINARY      ;C=NEXT BIT
0113  B0 02                     BCS PRINT       ;PRINT ZERO
                        ;
0115  A9 31                     LDA #'1         ;LOAD ASCII "1"
                        ;
0117  20 C6 72          PRINT   JSR WRT         ;PRINT BINARY DIGIT
011A  C6 01                     DEC COUNT       ;COUNT BIT PRINTED
011C  10 F1                     BPL PBLOOP      ;DO NEXT BIT
                        ;
011E  4C 00 01                  JMP PBIN        ;DO IT ALL AGAIN
```

```
.M  0100  20  8D  72  A9  20  85  00  A5
.:  0100  20  8A  72  20  E9  72  85  00
.:  0108  20  77  73  A9  08  85  01  A9
.:  0110  30  06  00  B0  02  A9  31  20
.:  0118  C6  72  C6  01  10  F1  4C  00
.:  0120  01  ↓
```

The purpose of this program is to print the binary representation of an ASCII input character on the terminal. Run the program by starting it at location 0100. Try typing some characters:

```
.R  0116  33  60  00  00  FF
.:  0100  ↓
.G
U  101010101
B  101111011
1  110011101
```

There is obviously something wrong with the program. Bits which should be printed as 1's are 0's and vice versa. (Refer to your 6500 reference card for character codes.) Looking at the program, the problem is that the branch after PBLOOP goes the wrong way! It should be BCC, Branch if Carry Clear (or alternatively, the 1 and 0 loads could be interchanged). Thus, when a one-bit is shifted out of the character, a one should be printed.

Patch the program and try again (the code for BCC is 90).

```
.M   0113  B0  02  A9  31  20  C9  72  C6
.:   0113  90  ↓
.R   7052  31  FC  FF  01  FF
.:   0100  ↓
.G
U  010101010
B  010000100
1  001100010
```

There is, alas, still an error--one too many bits is being printed. The cause of this is a little less obvious. (Do you see it?) To investigate the problem, set a breakpoint at location 011E. Do this by replacing the instruction there with a BRK (code of 00). Then run the program:

```
.M   011E  4C  00  01  EF  4C  00  01  00
.:   011E  00  ↓
.R   7052  31  FC  FF  01  FF
.:   0100  ↓
.G
U  010101010
 *   011F  B0  00  00  AA  FF
```

Once the break has occurred, you are free to investigate the state of the program using TIM. In particular, check the value in location COUNT:

```
.M   0000  00  FF  1B  2E  31  EA  FO  FA
```

Aha! Although COUNT starts out with a value of 8, it is going one step too far (FF is minus 1). This is because the test instruction, BPL PBLOOP is testing to see whether the count is

greater than or equal to zero. Replace it with BNE (code D0), replace your breakpoint with the original contents at that location, and try the program again.

```
.M 011C 10 F1 00 00 01 EF 4C
.: 011C D0 __ 4C ↓
.R 011F B0 00 00 AA FF
.: 0100 ↓
.G
U 01010101
B 01000010
1 00110001
I 01001001
W 01010111
O 01001111
R 01010010
K 01001011
S 01010011
```

```
                        ;CHECKOUT PROGRAM -- PRINT BINARY OF TYPED CHARACTER
                        ;

                        ;
0000                    *=0                     ;VARIABLE STORAGE IN PAGE ZERO
0000            BINARY  *=*+1                   ;STORAGE FOR CHAR DURING DISSECTION
0001            COUNT   *=*+1                   ;COUNT OF BITS REMAINING TO PRINT
                        ;
0002                    *=$01C0                 ;PROGRAM BEGINS ON PAGE ONE
                        ;
                CRLF    =$728A                   ;TIM  CRLF ROUTINE
                WRT     =$72C6                   ;TIM  WRITE ROUTINE
                RDT     =$72E9                   ;TIM  READ ROUTINE
                SPACE   =$7377                   ;TIM  SPACE ROUTINE
                        ;
01C0  20 8A 72  PBIN    JSR CRLF                ;PRINT CARRIAGE RETURN & LINE FEE
01C3  20 E9 72          JSR RDT                 ;GET A CHARACTER
01C6  85 00             STA BINARY              ;SAVE FOR DISSECTION
01C8  20 77 73          JSR SPACE               ;PRINT A SPACE
                        ;
01CB  A9 08             LDA #8                  ;INITIALIZE BIT COUNT
01CD  85 01             STA COUNT
                        ;
01CF  A9 30     PBLOOP  LDA #'0                 ;ASSUME ZERO: LOAD ASCII "0"
01D1  06 00             ASL BINARY              ;C=NEXT BIT
01D3  90 02             BCC PRINT               ;PRINT ZERO
                        ;
01D5  A9 31             LDA #'1                 ;LOAD ASCII "1"
                        ;
01D7  20 C6 72  PRINT   JSR WRT                 ;PRINT BINARY DIGIT
01DA  C6 01             DEC COUNT               ;COUNT BIT PRINTED
01DC  D0 F1             BNE PBLOOP              ;DO NEXT BIT
                        ;
01DE  4C C0 01          JMP PBIN                ;DO IT ALL AGAIN
```

CORRECTED PBIN PROGRAM

( ) 11. Save the corrected program using the WH command. Before punching the terminating record (as above in step 8), turn off the punch and set the PC to the start address of the program (0100). Then punch locations 00F6 and 00F7 on the tape, then the terminator (;00), and finally, some trailer:

```
.R   7052  30  37  FF  01  FF
.:   0100  ↓
.WH  00F6  00F7  ↓
;0200F6000101A2
.;00
```

The resulting tape can be loaded and then started as follows:

```
.LH
  ⋮ (program loads in)
.G
```

Locations 00F6 and 00F7 contain the starting address for programs. You may assemble and load your starting address into these locations to make tapes which can be started with a minimum of operator action. The carriage-return delay time may also be set in this manner. See Appendix II.

( ) 12. It is also possible to punch BNPF-format tapes using TIM. BNPF is the format used by some ROM programmers. The command is similar to that for writing hex tapes:

```
.WB  0100  0127  ↓
```

This command would punch the corrected PBIN program in BNPF

format. Try punching a BNPF tape. (Note that TIM will not load tapes in this format--use hex format (WH) for saving programs for later loading into your 65XX.)

( ) 13. If you have a high-speed paper tape reader attached to your 65XX system, you can use it to load programs in hex format. The H command switches the load device to and from the high speed reader. If you have a high speed reader, try loading a tape as follows:

.H
.LH

Note that control will not return to the user terminal until a terminator record (;00) is read.

APPENDIX A

MEMORY ADDRESS TEST

```
CARD # LOC      CODE        CARD
     1                      ;MEMORY ADDRESS TEST
     2                      ;FOR EACH LOC IN TEST RANGE
     3                              ;CLEAR WHOLE RANGE
     4                      ;       SET LOC TO $FF
     5                      ;       VERIFY WHOLE RAGE $00 EXCEPT (LOC)
     6                      ;       VERIFY (LOC) TO BE $FF
     7                      ;BREAK TO MONITOR ON ERROR WITH LOC IN (0,1)
     8                      ;PRINT ">" ON COMPLETION OF PASS & REPEAT
     9                      ;
    10  00C0                        *=$00C0             ;PAGE 0
    11                      ;
    12                      WRT     =$72C2
    13  0000                LOC     *=*+2               ;TEST CELL ADDR
    14  00C2                LOW     *=*+2                ;LOWER LIMIT OF TEST
    15  00C4                HIGH    *=*+2               ;UPPER LIMIT OF TEST+1
    16  00C6                PTR     *=*+2               ;POINTER TO CELL UNDER TEST
    17                      ;
    18  00C8                        *=$00C1C            ;START ADDR
    19                      ;
    20  0010  A9 0D        MAD     LDA #$0D            ;TYPE CR
    21  0012  20 C2 72              JSR WRT
    22  0015  A9 0A                 LDA #$0A            ;& LF
    23  0017  20 C2 72              JSR WRT
    24                      ;
    25  001A  20 68 00              JSR RSTLOC          ;LOC=LOW

    26  001D  20 71 00              JSR RSTPTR          ;PTR=LOW

    27  002C  A2 00                 LDX #0
    28                      ;
    29                      ;CLEAR MEMORY AREA UNDER TEST
    30  0022  A9 00        ML1     LDA #0
    31  0024  81 C6                 STA (PTR,X)         ;STORE ZERO
    32  0026  20 7A 00              JSR INCPTR          ;INCREMENT & TEST

    33  0029  D0 F7                 BNE ML1             ;NEXT LOC
    34                      ;
    35                      ;PUT $FF IN SELEXTED CELL
    36  0028  A9 FF        TEST    LDA #$FF
    37  002C  81 00                 STA (LOC,X)
    38                      ;VERIFY ALL CELLS ZERO EXCEPT (LOC)
    39  002F  20 71 00              JSR RSTPTR          ;PTR=LOW

    40                      ;
    41  0032  A1 C6        VLOOP   LDA (PTR,X)         ;GET CELL
    42  0034  F0 17                 BEQ NEXTC           ;OK IF ZERO
    43  0036  A4 C6                 LDY PTR             ;NOT ZERO--IS THIS (LOC)?
    44  0038  C4 00                 CPY LOC
    45  003A  F0 01                 BEQ OK1
    46  003C  00                    BRK                 ;NOT (LOC)
    47                      ;
    48  003D  A4 C7        OK1     LDY PTR+1
```

```
CARD #  LOC    CODE        CARD
   49   0C3F   C4 C1            CPY LCC+1
   50   CC41   FC C1            BEQ CK2
   51   0043   CC               BRK                    ;NOT (LCC)
   52                     ;
   53   CC44   C9 FF      CK2   CMP #$FF               ;IS (LCC)--IS DATA OK?
   54   CC46   FC 01            BEQ OK3
   55   CC48   CC               BRK                    ;WRONG DATA
   56                     ;
   57   CC49   A9 CC      OK3   LDA #0                 ;RESET (LOC)
   58   CC4B   81 CC            STA (LCC,X)
   59                     ;
   60   CC4D   2C 7A CC   NEXTC JSR INCPTR             ;NEXT CELL

   61   CC5C   CC EC            BNE VLOOP              ;IF NOT AT LIMIT
   62                     ;
   63   CC52   A5 CC            LDA LCC                ;PRINT STAR EVERY PAGE BOUNDA
   64   CC54   CC C7            BNE NOSTAR
   65   CC5E   A9 2A            LDA #'*
   66   CC58   2C C2 72         JSR WRT
   67   CC5B   A2 CC            LDX #C                 ;FIX X AFTER MON CALL
   68                     ;
   69   CC5D   2C 8B CC   NOSTAR JSR INCLOC            ;NEXT LCC

   70   CC6C   CC C9            BNE TEST
   71                     ;
   72   CC62   2C 68 CC         JSR RSTLCC             ;PASS COMPLETE

   73   CC65   4C 1C CC         JMP MAC                ;NEXT PASS
   74                     ;
   75                     ;RESET LCC TO LOW
   76   CC68   A5 C2      RSTLCC LDA LOW
   77   CC6A   85 CC             STA LCC
   78   CC6C   A5 C3             LDA LOW+1
   79   CC6E   85 C1             STA LCC+1
   80   CC7C   6C                RTS
   81                     ;
   82                     ;RESET PTR TO LOW
   83   CC71   A5 C2      RSTPTR LDA LOW
   84   CC73   85 C6             STA PTR
   85   CC75   A5 C3             LDA LOW+1
   86   CC77   85 C7             STA PTR+1
   87   CC79   6C                RTS
   88                     ;
   89                     ;INCREMENT PTR & CHECK FOR LIMIT
   90   C07A   E6 C6      INCPTR INC PTR               ;INCREMENT
   91   CC7C   DC C2             BNE INC1
   92                     ;
   93   CC7E   E6 C7              INC PTR+1
   94                     ;
   95   CC80   A5 C4      INC1   LDA HIGH              ;CHECK
   96   CC82   C5 C6             CMP PTR
   97   CC84   DC C4             BNE IPRET             ;NOT AT LIMIT
```

```
CARD # LOC   CODE      CARD
   98                ;
   99  0086  A5 C5           LDA HIGH+1
  100  0088  C5 07           CMP PTR+1          ;Z=1 IF AT LIMIT
  101                ;
  102  008A  60      IPRET   RTS
  103                ;
  104                ;INCREMENT LOC & CHECK FOR LIMIT
  105  008B  E6 CC    INCLOC  INC LOC            ;INCR
  106  008D  DC 02           BNE INC2
  107                ;
  108  008F  E6 C1           INC LOC+1
  109                ;
  110  0091  A5 C4    INC2    LDA HIGH           ;CHECK
  111  0093  C5 00           CMP LOC
  112  0095  DC C4           BNE ILRET
  113  0097  A5 05           LDA HIGH+1
  114  0099  C5 01           CMP LOC+1          ;Z=1 IF AT LIMIT
  115
  116  009B  60      ILRET   RTS

END OF MOS/TECHNOLOGY 6501 ASSEMBLY VERSION 3
NUMBER OF ERRORS =   0,  NUMBER OF WARNINGS =   0


        SYMBOL TABLE

  SYMBOL    VALUE  LINE DEFINED        CROSS-REFERENCES

  HIGH      0004          15     95    99   110   113
  ILRET     009B         116    112
  INCLOC    008B         105     65
  INCPTR    007A          90     32    60
  INC1      008C          95     91
  INC2      0091         110    106
  IPRET     008A         102     97
  LOC       000C          13     37    44   49   58   63   77   79   105   108   11
                             114
  LOW       0002          14     76    78   83   85
  MAC       001C          20     73
  ML1       0022          30     33
  NEXTC     004C          60     42
  NOSTAR    005C          69     64
  OK1       003C          48     45
  OK2       0044          53     50
  OK3       004A          57     54
  PTR       0006          16     31    41   43   48   84   86   9C   93   96   10
  RSTLOC    006E          76     25    72
  RSTPTR    0071          83     26    39
  TEST      002B          36     70
  VLOOP     0032          41     61
  WRT       72C2          12     21    23   66
```

APPENDIX B

TIM PROGRAM LISTINGS

```
    TIM VERSION 1.0 - MEM PAGE 0
CARD # LOC     CODE        CARD
     2                     ;
     3                     ;       MOS TECHNOLOGY 650X TERMINAL INTERFACE MONITOR (TIM)
     4                     ;              VERSION 1.0  AUGUST 31, 1975
     5                     ;              COPYRIGHT 1975 MOS TECHNOLOGY
     6                     ;              ALL RIGHTS RESERVED.  UNAUTHORIZED USE
     7                     ;              OF ALL OR PART STRICTLY PROHIBITED.
     8                     ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     9                     ;
    10                     ;  PROMPTING CHARACTER IS A PERIOD (.)
    11                     ;  ----------------------------------
    12                     ;
    13                     ;
    14                     ;  DISPLAY COMMANDS
    15                     ;  ----------------
    16                     ;
    17                     ;  .R           DISPLAY REGISTERS (PC,F,A,X,Y,SP)
    18                     ;  .M  ADDR     DISPLAY MEMORY ( 8 BYTES BEGINNING AT ADDR )
    19                     ;
    20                     ;
    21                     ;  ALTER COMMAND (:)
    22                     ;  -----------------
    23                     ;  .:  DATA       ALTERS PREVIOUSLY DISPLAYED ITEM OR NEXT ITEM
    24                     ;
    25                     ;
    26                     ;  PAPER TAPE I/O COMMANDS
    27                     ;  ----------------------
    28                     ;
    29                     ;  .LH                      LOAD HEX TAPE
    30                     ;  .WB ADDR1 ADDR2          WRITE BNPF TAPE (FROM LOW ADDR1 TO HIGH ADDR2)
    31                     ;  .WH ADDR1 ADDR2          WRITE HEX TAPE (FROM LOW ADDR1 TO HIGH ADDR
    32                     ;
    33                     ;  CONTROL COMMANDS
    34                     ;  ----------------
    35                     ;
    36                     ;  .G                      GO, CONTINUE EXECUTION FROM CURRENT PC ADDRESS
    37                     ;
    38                     ;  .H                      TOGGLES HIGH-SPEED-READER OPTION
    39                     ;                            (IF ITS ON, TURNS IT OFF; IF OFF, TURNS ON
    40                     ;
    41                     ;  BRK AND NMI ENTRY POINTS TO TIM
    42                     ;  ------------------------------
    43                     ;
    44                     ;      TIM IS NORMALLY ENTERED WHEN A 'BRK' INSTRUCTION IS
    45                     ;          ENCOUNTERED DURING PROGRAM EXECUTION.  AT THAT
    46                     ;          TIME CPU REGISTERS ARE OUTPUT:    PC F A X Y SP
    47                     ;          AND CONTROL IS GIVEN TO THE KEYBOARD.
    48                     ;      USER MAY ENTER TIM BY PROGRAMMED BRK OR INDUCED NMI.  NMI
    49                     ;          ENTRIES CAUSE A '#' TO PRECEDE THE '.' IN THE CPU REGISTER
    50                     ;          PRINTOUT FORMAT
    51                     ;
    52                     ;  NON-BRK INTRQ (EXTERNAL DEVICE) INTERRUPT HANDLING
    53                     ;  --------------------------------------------------
```

```
CARD # LOC     CODE        CARD
   54                     ;
   55                     ;       A NON-BRK INTRQ INTERRUPT CAUSES AN INDIRECT JUMP TO THE ADDRESS
   56                     ;           LOCATED AT 'UINT' (HEX FFF8).  THIS LOCATION CAN BE SET
   57                     ;           USING THE ALTER CMD, OR LOADED AUTOMATICALLY IN PAPER TAPE
   58                     ;           FORM WITH THE LH CMD IF THE USER ASSIGNS HIS INTRQ INTERRUPT
   59                     ;           VECTOR TO $FFF8 IN THE SOURCE ASSEMBLY PROGRAM.
   60                     ;       IF NOT RESET BY THE USER, UINT IS SET TO CAUSE EXTERNAL
   61                     ;           DEVICE INTERRUPTS TO ENTER TIM AS NMI'S.  I.E.,
   62                     ;           IF A NMI OCCURS WITHOUT AN INDUCED NMI SIGNAL, IT IS
   63                     ;           AN EXTERNAL DEVICE INTERRUPT.
   64                     ;
   65                     ;   SETTING AND RESETTING PROGRAM BREAKPOINTS
   66                     ;   ----------------------------------------
   67                     ;
   68                     ;       BREAKPOINTS ARE SET AND RESET USING THE MEMORY DISPLAY
   69                     ;           AND ALTER COMMANDS.  BRK HAS A '00' OPERATION CODE.
   70                     ;       TO SET A BREAKPOINT SIMPLY DISPLAY THE MEMORY LOCATION
   71                     ;           (FIRST INSTRUCTION BYTE) AT WHICH THE BREAKPOINT IS
   72                     ;           TO BE PLACED THEN ALTER THE LOCATION TO '00'.  THERE IS
   73                     ;           NO LIMIT TO THE NUMBER OF BREAKPOINTS THAT CAN BE
   74                     ;           ACTIVE AT ONE TIME.
   75                     ;       TO RESET A BREAKPOINT, RESTORE THE ALTERED MEMORY LOCATION
   76                     ;           TO ITS ORIGINAL VALUE.
   77                     ;       WHEN AND IF A BREAKPOINT IS ENCOUNTERED DURING EXECTUION,
   78                     ;           THE BREAKPOINT DATA PRECEDED BY AN '*' IS DISPLAYED.
   79                     ;           THE PROGRAM COUNTER VALUE DISPLAYED IS THE BRK
   80                     ;           INSTRUCTION LOCATION + 1.
   81                     ;
   82                     ;   ---------------------------------------------------------------
   83                     ;
   84                     MDBK    =%CCC1011C          ; X,X,X,POR,DATA-AVAIL,GOT-DATA,SERIAL-OUT,IN
   85                     DAVAIL  =$08
   86                     GOTDAT  =$04
   87                     IOBASE  =$6F00
   88                     MPA     =IOBASE+0
   89                     MDA     =IOBASE+1
   90                     MPB     =IOBASE+2
   91                     MDB     =IOBASE+3
   92                     MCLKIT  =IOBASE+4
   93                     MCLKRD  =IOBASE+4
   94                     MCLKIF  =IOBASE+5
   95                     UINT    =$FFF8
   96                     NCMDS   =7
   97                     MP0     =$7000
   98                     MP1     =$7100
   99                     MP2     =$7200
  100                     MP3     =$7300
  101                     ;
  102                     ;  ZERO PAGE MONITOR RESERVE AREA
  103                     ;
  104                     CRDLY   =227                ;DELAY FOR CR IN BIT-TIMES
  105                     WRAP    =228                ;ADDRESS WRAP-AROUND FLAG
```

```
CARD # LOC   CODE      CARD
  106                 DIFF    =229
  107                 HSPTR   =231
  108                 HSPCP   =232
  109                 PREVC   =233
  110                 MAJORT  =234
  111                 MINCRT  =235
  112                 ACMD    =236
  113                 TMPC    =238
  114                 TMP2    =240
  115                 TMP4    =242
  116                 TMP6    =244
  117                 PCL     =246
  118                 PCH     =247
  119                 FLGS    =248
  120                 ACC     =249
  121                 XR      =250
  122                 YR      =251
  123                 SP      =252
  124                 SAVX    =253
  125                 TMPC    =254
  126                 TMPC2   =255
  127                 RCNT    =TMPC
  128                 LCNT    =TMPC2
  129                 ;
  130                 ;  64 BYTE RAM MONITER RESERVE AREA
  131                 ;
  132                 RAM64   =$FFCC
  133  0000                   *=RAM64
```

```
CARD #  LOC   CODE        CARD
   135                    ;
   136                    ;
   137                    ;  TIM PAGE 0 (RELATIVE)
   138  FFC0                  *=MPC
   139                    ;
   140  7000  85 F9       NMINT  STA ACC          ; SAVE A
   141  7002  A9 23              LDA #'#          ; SET A=# TO INDICATE NMINT ENTRY
   142  7004  DC 55              BNE B3           ; JMP B3
   143                    ;
   144  7006  A9 16       RESET  LDA #MDBK        ; INIT DIR REG, PCR TC 1 RELOCATES
   145                    ;
   146  7008  8D 03 6E           STA MDB
   147                    ;
   148  700B  A2 08              LDX #8           ; X=0
   149  700D  BC F7 73    R1     LDA INTVEC-1,X   ; INITIALIZE INT VECTORS
   150  7010  9D F7 FF           STA UINT-1,X
   151  7013  CA                 DEX
   152  7014  D0 F7              BNE R1
   153                    ;
   154  7016  86 EA              STX MAJORT       ; INIT MAJOR T COUNT TO ZERO
   155  7018  86 E7              STX HSPTR        ; CLEAR HSPTR FLAGS
   156  701A  86 E8              STX HSRCP
   157  701C  CA                 DEX              ; X=FF
   158  701D  9A                 TXS              ; SP=FF
   159                    ;
   160                    ;                      ; COMPUTE BIT-TIME CONSTANT, X=FF
   161                    ;
   162  701E  A0 01              LDY #1           ; SET TO MEASURE 2 BITS
   163  7020  84 E3              STY CRDLY        ;INIT CR DELAY TIME PARAMETER
   164  7022  AD 02 6E    R0     LDA MPB          ; WAIT FOR START
   165  7025  4A                 LSR A
   166  7026  90 FA              BCC R0
   167                    ;
   168  7028  8E 04 6E    R2     STX MCLKIT       ; START CLOCK INITIALLY WITH FF
   169  702B  AD 05 6E    R3     LDA MCLKIF
   170  702E  10 04              BPL R4
   171  7030  E6 EA              INC MAJORT       ; COUNT MAJOR T
   172  7032  D0 F4              BNE R2           ; GO RESTART CLOCK WITH X = FF
   173                    ;
   174  7034  98          R4     TYA
   175  7035  4D 02 6E           EOR MPB
   176  7038  29 01              AND #1
   177  703A  F0 EF              BEQ R3           ; WAIT FOR Y BIT 0 AND SERIAL-IN NOT EQU
   178  703C  88                 DEY
   179  703D  10 EC              BPL R3           ; LOOP UNTIL START OF BIT 2
   180                    ;
   181  703F  AD 04 6E           LDA MCLKRD
   182  7042  49 FF              EOR #$FF         ; COMPLEMENT RESIDUE
   183  7044  4A          R5     LSR A            ; HALF IT
   184  7045  46 EA              LSR MAJORT       ; HALF MAJOR
   185  7047  90 02              BCC R6
   186  7049  09 80              ORA #$80         ; PROPAGATE HC TO LC
```

```
CARD # LOC   CODE        CARD
  187  7C4B  C8          R6      INY
  188  7C4C  F0 F6               BEQ R5
  189  7C4E  85 EB               STA MINORT
  190                    ;
  191  7C50  58                 CLI                 ; ENABLE INTS
  192  7C51  00                 BRK                 ; ENTER TIM BY BRK
  193                    ;
  194  7C52  85 F9       INTRQ   STA ACC             ; SAVE ACC
  195  7C54  68                 PLA                 ; FLAGS TO A
  196  7C55  48                 PHA                 ; RESTORE STACK STATUS
  197  7C56  29 10              AND #$10            ; TEST BRK FLAG
  198  7C58  F0 27              BEQ BX              ; USER INTERRUPT
  199                    ;
  200  7C5A  0A                 ASL A               ; SET A=SPACE (10 X 2 = 20)
  201  7C5B  85 FE       B3      STA TMPC            ; SAVE INT TYPE FLAG
  202  7C5D  D8                 CLD                 ; CLEAR DECIMAL MODE
  203  7C5E  4A                 LSR A               ; # IS ODD, SPACE IS EVEN
  204                                              ; SET CY FOR PC BRK CORRECTION
  205                    ;
  206  7C5F  86 FA              STX XR              ; SAVE X
  207  7C61  84 FB              STY YR              ; Y
  208  7C63  68                 PLA
  209  7C64  85 F8              STA FLGS            ; FLAGS
  210  7C66  68                 PLA
  211  7C67  69 FF              ADC #$FF            ; CY SET TO PC-1 FOR BRK
  212  7C69  85 F6              STA PCL
  213  7C6B  68                 PLA
  214  7C6C  69 FF              ADC #$FF
  215  7C6E  85 F7              STA PCH
  216  7C70  BA                 TSX
  217  7C71  86 FC              STX SP              ; SAVE ORIG SP
  218                    ;
  219  7C73  20 8A 72    B5      JSR CRLF
  220  7C76  A6 FE              LDX TMPC
  221                    ;
  222  7C78  A9 2A              LDA #'*
  223  7C7A  20 C0 72           JSR WRTWO
  224  7C7D  A9 52              LDA #'R             ; SET FOR R DISPLAY TO PERMIT
  225  7C7F  D0 16              BNE S0              ;   IMMEDIATE ALTER FOLLOWING BREAKPOINT.
  226                    ;
  227  7C81  A5 F9       BX      LDA ACC
  228  7C83  6C F8 FF           JMP (UINT)          ; CONTROL TO USER INTRQ SERVICE ROUTINE
  229                    ;
  230  7C86  A9 00       START   LDA #0              ;NEXT COMMAND FROM USER
  231  7C88  85 E7              STA HSPTR           ;CLEAR H. S. PAPER TAPE FLAG
  232  7C8A  85 E4              STA WRAP            ;CLEAR ADDRESS WRAP-AROUND FLAG
  233  7C8C  20 8A 72           JSR CRLF
  234  7C8F  A9 2E              LDA #'.             ; TYPE PROMPTING '.'
  235  7C91  20 C6 72           JSR WRCC
  236  7C94  20 E9 72           JSR RDCC            ; READ CMD, CHAR RETURNED IN A
  237                    ;
  238  7C97  A2 06       S0      LDX #NCMDS-1        ; LOOK-UP CMD
```

```
MPJ TIM PAGE 0

CARD # LOC   CODE        CARD
  239 7C99  DD 06 71  S1    CMP CMDS,X
  240 7C9C  D0 19           BNE S2
  241                 ;
  242 7C9E  A5 FD           LDA SAVX          ; SAVE PREVIOUS CMD
  243 7CA0  85 E9           STA PREVC
  244 7CA2  86 FD           STX SAVX          ; SAVE CURRENT CMD INDEX
  245 7CA4  A9 71           LDA #MP1/256      ; JMP INDIRECT TO CMD CODE
  246 7CA6  85 ED           STA ACMD+1        ;   ALL CMD CODE BEGINS ON MP1
  247 7CA8  BD 0D 71        LDA ADRS,X
  248 7CAB  85 EC           STA ACMD
  249 7CAD  E0 03           CPX #3            ; IF :, R OR M (0, 1, OR 2)  SPACE ?
  250 7CAF  B0 03           BCS IJMP
  251 7CB1  20 74 73        JSR SPAC2
  252                 ;
  253 7CB4  6C EC 00  IJMP  JMP (ACMD)
  254                 ;
  255 7CB7  CA        S2    DEX
  256 7CB8  10 DF           BPL S1            ; LOOP FOR ALL CMDS
  257                 ;
  258 7CBA  A9 3F     ERROPR LDA #'?          ; OPERATOR ERR, TYPE '?', RESTART
  259 7CBC  20 C6 72        JSR WBOC
  260 7CBF  90 C5           BCC START         ; JMP START (WBOC RETURNS CY=0)
  261                 ;
  262 7CC1  38        DCMP  SEC               ; TMP2-TMP0 DOUBLE SUBTRACT
  263 7CC2  A5 F0           LDA TMP2
  264 7CC4  E5 EE           SBC TMP0
  265 7CC6  85 E5           STA DIFF
  266 7CC8  A5 F1           LDA TMP2+1
  267 7CCA  E5 EF           SBC TMP0+1
  268 7CCC  A8              TAY               ;RETURN HIGH ORDER PART IN Y
  269 7CCD  05 E5           ORA DIFF          ; OR LO FOR EQU TEST
  270 7CCF  60              RTS
  271                 ;
  272 7CD0  A5 EE     PUTP  LDA TMP0          ; MOVE TMP0 TO PCH,PCL
  273 7CD2  85 F6           STA PCL
  274 7CD4  A5 EF           LDA TMP0+1
  275 7CD6  85 F7           STA PCH
  276 7CD8  60              RTS
  277                 ;
  278 7CD9  A9 00     ZTMP  LDA #0            ; CLEAR REGS
  279 7CDB  95 EE           STA TMP0,X
  280 7CDD  95 EF           STA TMP0+1,X
  281 7CDF  60              RTS
  282                 ;
  283                 ;  READ AND STORE BYTE.  NO STORE IF SPACE OR RCNT=0.
  284                 ;
  285 7CE0  20 B3 73  BYTE  JSR RDOB            ; CHAR IN A, CY=0 IF SP
  286 7CE3  90 10           BCC BY3             ; SPACE
  287                 ;
  288 7CE5  A2 00           LDX #0              ; STORE BYTE
  289 7CE7  81 EE           STA (TMP0,X)
  290                 ;
```

```
CARD #  LOC    CODE        CARD
  291   7CE5  C1 EE            CMP (TMPD,X)        ; TEST FOR VALID WRITE (RAM)
  292   7CE8  F0 05            BEQ BY2
  293   7CEC  68               PLA                 ; ERR, CLEAR JSR ADR IN STACK
  294   7CED  68               PLA
  295   7CEE  4C BA 70         JMP ERROPR
  296                  ;
  297   7DF2  20 7C 72  BY2    JSR DADD            ; INCR CKSUM
  298   7CF5  20 97 73  BY3    JSR INCTMP          ; GO INCR TMPC ADR
  299   7CF8  C6 FE            DEC RCNT
  300   7CFA  60               RTS
  301                  ;
  302   7CFB  A9 F8    SETR    LDA #FLGS           ; SET TO ACCESS REGS
  303   7CFD  85 EE            STA TMPC
  304   7CFF  A9 00            LDA #0
  305   7101  85 EF            STA TMPC+1
  306   7103  A9 05            LDA #5
  307   7105  60               RTS
  308                  ;
  309   7106  3A       CMDS    .BYTE ':'
  310   7107  52               .BYTE 'R'
  311   7108  4D               .BYTE 'M'
  312   7109  47               .BYTE 'G'
  313   710A  48               .BYTE 'H'
  314   710B  4C               .BYTE 'L'
  315   710C  57               .BYTE 'W'           ; W MUST BE LAST CMD IN CHAIN
  316   710D  3A       ADRS    .BYTE ALTER-MP1
  317   710E  14               .BYTE DSPLYR-MP1
  318   710F  1C               .BYTE DSPLYM-MP1
  319   7110  5C               .BYTE GO-MP1
  320   7111  6F               .BYTE HSP-MP1
  321   7112  74               .BYTE LH-MP1
  322   7113  C2               .BYTE WO-MP1
```

Verified 8-7-78

7100

```
MPI TIM PAGE 1

CARD # LOC      CODE        CARD
  324                       ;
  325                       ;
  326                       ;  NOTE -- ALL CMD CODE MUST BEGIN ON MPI
  327                       ;
  328                       ;  DISPLAY REG CMD - A,F,X,Y, AND SP
  329                       ;
  330   7114  20 A6 72      DSPLYR JSR WRPC          ; WRITE PC
  331   7117  20 FB 70             JSR SETR
  332   711A  D0 07                BNE MC           ; USE DSPLYM
  333                       ;
  334   711C  20 A4 73      DSPLYM JSR RDDA          ; READ MEM ADR INTO TMPC
  335   711F  90 16                BCC ERRS1        ; ERR IF NO ADDR
  336   7121  A9 08                LDA #8
  337   7123  85 FE          MC    STA TMPC
  338   7125  A0 00                LDY #0
  339   7127  20 77 73      M1    JSR SPACE        ; TYPE 8 BYTES OF MEM
  340   712A  B1 EE                LDA (TMPC),Y     ; (TMPC) PRESERVED FOR POSS ALTER
  341   712C  20 B1 72             JSR WRDR
  342   712F  C8                  INY              ; INCR INDEX
  343   7130  C6 FE                DEC TMPC
  344   7132  D0 F3                BNE M1
  345   7134  4C E6 70      BEGS1 JMP START
  346                       ;
  347   7137  4C BA 70      ERRS1 JMP ERRCPR
  348                       ;
  349                       ;  ALTER LAST DISPLAYED ITEM (ADR IN TMPC)
  350                       ;
  351   713A  C6 E9         ALTER DEC PREVC        ; R INDEX = 1
  352   713C  D0 0D                BNE A3
  353                       ;
  354   713E  20 A4 73             JSR RDDA         ; CY=C IF SP
  355   7141  90 03                BCC A2           ; SPACE
  356   7143  20 D0 70             JSR PUTP         ; ALTER PC
  357   7146  20 FB 70      A2     JSR SETR         ; ALTER R'S
  358   7149  D0 05                BNE A4           ; JMP A4 (SETR RETURNS ACC = 5)
  359   714B  20 9A 72      A3     JSR WRDA         ; ALTER M, TYPE ADR
  360   714E  A9 08                LDA #8           ; SET CNT=8
  361                       ;
  362   7150  85 FE         A4     STA RCNT
  363   7152  20 77 73      A5     JSR SPACE        ; PRESERVES Y
  364   7155  20 E0 70             JSR BYTE
  365   7158  D0 F8                BNE A5
  366   715A  F0 D8         A9     BEQ BEGS1
  367                       ;
  368   715C  A6 FC         GO     LDX SP
  369   715E  9A                   TXS              ; CHG OR NEW SP VALUE TO SP
  370   715F  A5 F7                LDA PCH
  371   7161  48                  PHA
  372   7162  A5 F6                LDA PCL
  373   7164  48                  PHA
  374   7165  A5 F8                LDA FLGS
  375   7167  48                  PHA
```

```
CARD #  LOC    CODE          CARD
  376   7168  A5 F9           LDA ACC
  377   716A  A6 FA           LDX XR
  378   716C  A4 FB           LDY YR
  379   716E  40              RTI
  380                 ;
  381   716F  E6 E8     HSP   INC HSFLG       ; TOGGLE BIT C
  382   7171  4C 86 70        JMP START
  383                 ;
  384   7174  20 E9 72  LH    JSR RDCC        ; READ SECOND CMD CHAR
  385   7177  20 8A 72        JSR CRLF
  386   717A  A6 E8           LDX HSFLG       ; ENABLE PTR OPTION IF SET
  387   717C  86 E7           STX HSPTR
  388   717E  20 E9 72  LH1   JSR RDCC
  389   7181  C9 3B           CMP #';         ; FIND NEXT RCD MARK (;)
  390   7183  D0 F9           BNE LH1
  391                 ;
  392   7185  A2 04           LDX #4
  393   7187  20 D9 70        JSR ZTMP        ; CLEAR CKSUM REGS TMP4
  394   718A  20 B3 73        JSR RDDB
  395   718D  D0 06           BNE LH2
  396                 ;
  397   718F  A2 00           LDX #0          ; CLEAR HS RCD FLAG
  398   7191  86 E7           STX HSPTR
  399   7193  F0 9F           BEQ BEGS1       ; FINISHED
  400                 ;
  401   7195  85 FE     LH2   STA RCNT        ; RCNT
  402   7197  20 7C 72        JSR DADD        ; RCD LNGH TO CKSUM
  403   719A  20 B3 73        JSR RDDB        ; SA HO TO TMPC+1
  404   719D  85 EF           STA TMPC+1
  405   719F  20 7C 72        JSR DADD        ; ADD TO CKSUM
  406   71A2  20 B3 73        JSR RDDB        ; SA LO TO TMPC
  407   71A5  85 EE           STA TMPC
  408   71A7  20 7C 72        JSR DADD        ; ADD TO CKSUM
  409                 ;
  410   71AA  20 E0 70  LH3   JSR BYTE        ; BYTE SUB/R DECRS RCNT ON EXIT
  411   71AD  D0 FB           BNE LH3
  412   71AF  20 A4 73        JSR RDCA        ; CKSUM FROM HEX RCD TO TMPC
  413   71B2  A5 F2           LDA TMP4        ; TMP4 TO TMP2 FOR COMP
  414   71B4  85 F0           STA TMP2
  415   71B6  A5 F3           LDA TMP4+1
  416   71B8  85 F1           STA TMP2+1
  417   71BA  20 C1 70        JSR CCMP
  418   71BD  F0 BF           BEQ LH1
  419   71BF  4C BA 70  ERRP1 JMP ERRCPR
  420                 ;
write 421   71C2  20 E9 72  WC    JSR RDDC        ; RD 2ND CMD CHAR
(tape)  422   71C5  85 FE           STA TMPC
  423   71C7  20 77 73        JSR SPACE
  424   71CA  20 A4 73        JSR RDDA        starting address
  425   71CC  20 87 73        JSR T2T2        ; SA TO TMP2
  426   71CC  20 77 73        JSR SPACE       ; SPACE BEFORE NEXT ADDRESS
  427   71D3  20 A4 73        JSR RDDA
```

```
MP1 TIM PAGE 1

CARD #  LOC   CODE        CARD
  428   71D6  20 87 73          JSR T2T2          ; SA TO TMP0, EA TO TMP2
  429   71D9  20 E5 72          JSR RDOC          ; DELAY FOR FINAL CR
  430   71DC  A5 FE             LDA TMPC
  431                     ;
  432   71DE  C9 48             CMP #'H
  433   71E0  D0 59             BNE WB
  434                     ;
  435   71E2  A6 E4       WH0   LDX WRAP          ;IF ADDR HAS WRAPPED AROUND
  436   71E4  D0 52             BNE BCCST         ;THEN TERMINATE WRITE OPERATION
  437                     ;
  438   71E6  20 8A 72          JSR CRLF
  439   71E9  A2 18             LDX #24
  440   71EB  86 FE             STX RCNT          ; RCNT=24
  441   71ED  A2 04             LDX #4            ; CLEAR CKSUM
  442   71EF  20 D9 70          JSR ZTMP
  443                     ;
  444   71F2  A9 3B             LDA #';
  445   71F4  20 C6 72          JSR WROC          ; WR RCD MARK
  446                     ;
  447   71F7  20 C1 70          JSR DCMP          ; EA-SA (TMP0+2-TMP0) DIFF IN LOC DIFF,+1
  448   71FA  98                TYA               ; MS BYTE OF DIFF
  449   71FB  D0 0A             BNE WH1
  450   71FD  A5 F5             LDA DIFF
  451   71FF  C9 17             CMP #23
  452   7201  B0 04             BCS WH1           ; DIFF GT 24
  453   7203  85 FE             STA RCNT          ; INCR LAST RCNT
  454   7205  E6 FE             INC RCNT
  455   7207  A5 FE       WH1   LDA RCNT
  456   7209  20 7C 72          JSR DADC          ; ADD TO CKSUM
  457   720C  20 B1 72          JSR WRCB          ; RCD CNT IN A
  458   720F  A5 EE             LDA TMPC+1        ; SA HC
  459   7211  20 7C 72          JSR DADC
  460   7214  20 B1 72          JSR WRCB
  461   7217  A5 EE             LDA TMPC          ; SA LC
  462   7219  20 7C 72          JSR DADC
  463   721C  20 B1 72          JSR WRCB
  464                     ;
  465   721F  A0 00       WH2   LDY #0
  466   7221  B1 EE             LDA (TMP0),Y
  467                     ;
  468   7223  20 7C 72          JSR DADC          ; INC CKSUM, PRESERVES A
  469   7226  20 B1 72          JSR WROB
  470   7229  20 97 73          JSR INCTMP        ; INC SA
  471   722C  C6 FE             DEC RCNT
  472   722E  D0 EF             BNE WH2           ; LOOP FOR UP TO 24 BYTES
  473                     ;
  474   7230  20 9E 72          JSR WRCA4         ; WRITE CKSUM
  475                     ;
  476   7233  20 C1 70          JSR DCMP
  477   7236  B0 AA             BCS WH0           ; LOOP WHILE EA GT OR = SA
  478   7238  4C 86 70    BCCST JMP START
  479                     ;
```

```
MPI TIM PAGE 1

CARD #  LOC   CODE        CARD
  480                     ;
  481  723B  E6 FD        WB      INC SAVX          ; SAVX TO = NCMDS FOR ASCII SUB/R
  482  723D  A5 E4        WB1     LDA WRAP          ;IF ADDR HAS WRAPPED AROUND
  483  723F  D0 F7                BNE BCCST         ;THEN TERMINATE WRITE OPERATION
  484                     ;
  485  7241  A9 04                LDA #4
  486  7243  85 EC                STA ACMD
  487  7245  20 8A 72             JSR CRLF
  488  7248  20 9A 72             JSR WRDA          ; OUTPUT HEX ADR
  489                     ;
  490  724B  20 77 73     WBNPF   JSR SPACE
  491  724E  A2 09                LDX #9
  492  7250  86 FE                STX TMPC          ; LOOP CNT =9
  493  7252  A1 E5                LDA (TMPC-9,X)
  494  7254  85 FF                STA TMPC2         ; BYTE TO TMPC2
  495  7256  A9 42                LDA #'B
  496  7258  D0 08                BNE WBF2          ; WRITE B
  497                     ;
  498  725A  A9 50        WBF1    LDA #'P
  499  725C  06 FF                ASL TMPC2
  500  725E  B0 02                BCS WBF2
  501  7260  A9 4E                LDA #'N
  502                     ;
  503  7262  20 C6 72     WBF2    JSR WRCC          ; WRITE N OR P
  504  7265  C6 FE                DEC TMPC
  505  7267  D0 F1                BNE WBF1          ; LOOP
  506  7269  A9 46                LDA #'F
  507  726B  20 C6 72             JSR WRDC          ; WRITE F
  508                     ;
  509  726E  20 97 73             JSR INCTMP
  510                     ;
  511  7271  C6 EC                DEC ACMD          ; TEST FOR MULTIPLE OF FOUR
  512  7273  D0 D6                BNE WBNPF
  513                     ;
  514  7275  20 C1 70             JSR DCMP
  515  7278  B0 C3                BCS WB1           ; LOOP WHILE EA GT OR = SA
  516  727A  90 BC                BCC BCCST
  517                     ;
  518  727C  48           CADD    PHA               ; SAVE A
  519  727D  18                   CLC
  520  727E  65 F2                ADC TMP4    LSB
  521  7280  85 F2                STA TMP4
  522  7282  A5 F3                LDA TMP4+1  MSB
  523  7284  69 00                ADC #0
  524  7286  85 F3                STA TMP4+1
  525  7288  68                   PLA               ; RESTORE A
  526  7289  60                   RTS
  527                     ;
  528  728A  A2 0D        CRLF    LDX #$0D
  529  728C  A9 0A                LDA #$0A
  530  728E  20 C0 72             JSR WRTWO
  531  7291  A6 E3                LDX CRDLY         ;BIT-TIME COUNT FOR DELAY
```

OK 8-10-78

```
CARD #  LOC   CODE        CARD
   532  7293  20 1D 73  CR1     JSR DLY2          ;DELAY OF ONE BIT-TIME
   533  7296  CA              DEX
   534  7297  D0 FA           BNE CR1
   535  7299  60              RTS
   536                    ;
   537                    ;  WRITE ADR FROM TMPC STORES
   538                    ;
   539  729A  A2 01       WRCA    LDX #1
   540  729C  D0 0A           BNE WRCA1
   541  729E  A2 05       WRCA4   LDX #5
   542  72A0  D0 06           BNE WRCA1
   543  72A2  A2 07       WRCA6   LDX #7
   544  72A4  D0 02           BNE WRCA1
   545  72A6  A2 09       WRPC    LDX #9
   546  72A8  B5 ED       WRCA1   LDA TMPC-1,X
   547  72AA  48              PHA
   548  72AB  B5 EE           LDA TMPC,X
   549  72AD  20 B1 72        JSR WRCB
↑OK550  72B0  68              PLA
   551                    ;
   552                    ;  WRITE BYTE - A = BYTE
   553                    ;  UNPACK BYTE DATA INTO TWO ASCII CHARS. A=BYTE; X,A=CHARS
   554                    ;
   555  72B1  48          WRCB    PHA
   556  72B2  4A              LSR A
   557  72B3  4A              LSR A
   558  72B4  4A              LSR A
   559  72B5  4A              LSR A
   560  72B6  20 58 73        JSR ASCII         ; CONVERT TO ASCII
   561  72B9  AA              TAX
   562  72BA  68              PLA
   563  72BB  29 0F           AND #$0F
   564  72BD  20 58 73        JSR ASCII
   565                    ;
   566                    ;  WRITE 2 CHARS - X,A = CHARS
   567                    ;
   568  72C0  48          WRTWO   PHA
   569  72C1  8A              TXA
   570  72C2  20 C6 72        JSR WRT
   571  72C5  68              PLA
   572                    ;
   573                    ;  WRITE SERIAL OUTPUT
   574                    ;  A = CHAR TO BE OUTPUT
   575                    ;
   576  72C6  20 1D 73    WRT     JSR DLY2
   577  72C9  A2 09           LDX #9
   578                    WROC    =WRT
   579  72CB  49 FF           EOR #$FF          ; COMPLEMENT A
   580  72CD  38              SEC
   581                    ;
OK 582  72CE  20 DA 72    WRT1    JSR CLT
   583  72D1  20 1D 73        JSR DLY2
```

```
CARD #  LOC    CODE          CARD
   584  72D4  4A            LSR A
   585  72D5  CA            DEX
   586  72D6  D0 F6         BNE WRT1
   587  72D8  F0 3F         BEQ RDT5
   588                                        ; ELSE BNE?
   589                    ;
   590  72DA  48         OUT   PHA             ; SAVE A
   591  72DB  AD 02 6E         LDA MPB         ; OUTPUT BIT FROM CY
   592  72DE  29 FD            AND #%11111101
   593  72E0  90 02            BCC OUT1
   594  72E2  09 02            ORA #%00000010
   595  72E4  8D 02 6E   OUT1  STA MPB
   596  72E7  68               PLA             ; RESTORE A
   597  72E8  60               RTS
   598                    ;
   599                    ;  OUTPUT  RETURNS CHAR IN A
   600                    ;
   601  72E9  A5 E7      RDT   LDA HSPTR        ; TEST HS PTR OPTION
   602  72EB  4A               LSR A
   603  72EC  B0 4F            BCS RDHSR
   604                    RDOC  =RDT
   605  72EE  A2 08            LDX #8
   606                    ;
   607  72F0  AD 02 6E   RDT1  LDA MPB
   608  72F3  4A               LSR A           ; WAIT FOR START BIT
   609  72F4  90 FA            BCC RDT1
   610                    ;
   611  72F6  20 20 73         JSR DLY1
   612  72F9  20 DA 72         JSR OUT         ; ECHO START BIT
   613                    ;
   614  72FC  20 1D 73   RDT2  JSR DLY2
   615  72FF  AD 02 6E         LDA MPB         ; CY = NEXT BIT
   616  7302  4A               LSR A
   617  7303  20 DA 72         JSR OUT         ; ECHO
   618                    ;
   619  7306  08               PHP             ; SAVE BIT
   620  7307  98               TYA             ; Y CONTAINS CHAR BEING FORMED
   621  7308  4A               LSR A
   622  7309  28               PLP             ; RECALL BIT
   623  730A  90 02            BCC RDT4
   624  730C  09 80            ORA #$80        ; ADD IN NEXT BIT
   625  730E  A8         RDT4  TAY
   626  730F  CA               DEX
   627  7310  D0 EA            BNE RDT2        ; LOOP FOR 8 BITS
   628  7312  49 FF            EOR #$FF        ; COMPLEMENT DATA
   629  7314  29 7F            AND #$7F        ; CLEAR PARITY
   630                    ;
   631  7316  20 1D 73         JSR DLY2
   632  7319  18         RDT5  CLC
   633  731A  20 DA 72         JSR OUT         ;AND DELAY 2 HALF-BIT-TIMES
   634                    ;
   635  731D  20 20 73   DLY2  JSR DLY1
```

```
CARD #  LOC   CODE        CARD
   636  7320  48          DLY1   PHA                 ; SAVE FLAGS AND A
   637  7321  08                 PHP
   638  7322  8A                 TXA                 ; SAVE X
   639  7323  48                 PHA
   640  7324  AE EA              LDX MAJCRT
   641  7326  A5 EB              LDA MINCRT
   642                     ;
   643  7328  8D 04 6E     DL2    STA MCLK1T
   644                     ;
   645  732B  AD 05 6E     DL3    LDA MCLK1F
   646  732E  10 FB        B      BPL DL3
   647  7330  CA                  DEX
   648  7331  08                 PHP
   649  7332  AD 04 6E           LDA MCLKRD          ; RESET TIMER INT FLAG
   650  7335  28                 PLP
   651  7336  10 F3              BPL DL3
   652                     ;
   653  7338  68                 PLA                 ; RESTORE REGS
   654  7339  AA                 TAX
   655  733A  28                 PLP
   656  733B  68                 PLA
   657  733C  60           DLX    RTS
   658                     ;
   659  733D  AD 02 6E     RDHSR  LDA MPB             ; LOOP ON DATA AVAIL
   660  7340  29 08              AND #DAVAIL
   661  7342  F0 F9              BEQ RDHSR
   662                     ;
   663  7344  AE 00 6E           LDX MPA             ; READ DATA
   664  7347  AD 02 6E           LDA MPB             ; SEND GOT-DATA PULSE
   665  734A  09 04              ORA #GOTDAT
   666  734C  8D 02 6E           STA MPB
   667  734F  29 FB              AND #%11111011
   668  7351  8D 02 6E           STA MPB
   669  7354  8A                 TXA
   670  7355  29 7F              AND #$7F
   671  7357  60                 RTS
   672                     ;
   673  7358  18           ASCII  CLC
   674  7359  69 06              ADC #6
   675  735B  69 F0              ADC #$F0
   676  735D  90 02              BCC ASC1
   677  735F  69 06              ADC #$06
   678                     ;
   679  7361  69 3A        ASC1   ADC #$3A
   680  7363  48                 PHA                 ; TEST FOR LETTER B IN ADR DURING WBNPF
   681  7364  C9 42              CMP #'B
   682  7366  D0 0A              BNE ASCX
   683  7368  A5 FD              LDA SAVX
   684  736A  C9 07              CMP #NCMDS
   685  736C  D0 04              BNE ASCX            ; NOT WB CMD
   686  736E  68                 PLA
   687  736F  A9 20              LDA #'              ; FOR WB, BLANK B'S IN ADR
```

B-10-78 OK

HIGH SPEED READER

6E00 PARALLEL INPUT

+ 28160 DECIMAL

```
    MPL TIM PAGE 1

CARD # LOC   CODE      CARD
  688  7371  48            PHA
  689  7372  68       ASCX PLA
  690  7373  60            RTS
  691                 ;
  692  7374  20 77 73 SPAC2 JSR SPACE
  693  7377  48       SPACE PHA                ; SAVE A,X,Y
  694  737B  8A             TXA
  695  737C  48             PHA
  696  737D  98             TYA
  697  737E  48             PHA
  698  737F  A9 20          LDA #'
  699  7381  20 C6 72       JSR WRT            ; TYPE SP
  700  7384  68             PLA                ; RESTORE A,X,Y
  701  7385  A8             TAY
  702  7386  68             PLA
  703  7387  AA             TAX
  704  7388  68             PLA
  705  7389  60             RTS
  706                 ;
  707  738A  A2 02    T2T2  LDX #2
  708  738C  B5 ED    T2T21 LDA TMPC-1,X
  709  738E  48             PHA
  710  738F  B5 EF          LDA TMP2-1,X
  711  7391  95 ED          STA TMPC-1,X
  712  7393  68             PLA
  713  7394  95 EF          STA TMP2-1,X
  714  7396  CA             DEX
  715  7397  D0 F3          BNE T2T21
  716  7399  60             RTS
  717                 ;
  718                 ;INCREMENT (TMPC,TMPC+1) BY 1
  719  739A  E6 EE    INCTMP INC TMPC           ;LOW BYTE
  720  739C  F0 01          BEQ INCT1
  721  739E  60             RTS
  722                 ;
  723  739F  E6 EF    INCT1  INC TMPC+1         ;HIGH BYTE
  724  73A1  F0 01          BEQ SETWRP
  725  73A3  60             RTS
  726                 ;
  727  73A4  E6 E4    SETWRP INC WRAP           ;PCINTER HAS WRAPPED ARCUND - SET FLAG
  728  73A6  60             RTS
  729                 ;
  730                 ; READ HEX ADR, RETURN HO IN TMPC, LC IN TMPC+1 AND CY=1
  731                 ;    IF SP CY=C
  732                 ;
  733  73A7  20 B6 73 RDDA  JSR RDDB           ; READ 2 CHAR BYTE
  734  73AA  90 02          BCC RDDA2          ; SPACE
  735                 ;
  736  73AC  85 EF          STA TMPC+1
  737  73AE  20 B6 73 RDDA2 JSR RDDB
  738  73B1  90 02          BCC RDEXIT         ; SP
  739  73B3  85 EE          STA TMPC
```

```
MP1 TIM PAGE 1

CARD # LOC   CODE      CARD
  740 73B2  60        RDEXIT RTS
  741                 ;
  742                 ;  READ HEX BYTE AND RETURN IN A, AND CY=1
  743                 ;     IF SP CY=0
  744                 ;     Y REG IS PRESERVED
  745                 ;
  746 73B3  98        RDCB    TYA                 ; SAVE Y
  747 73B4  48                PHA
  748 73B5  A9 0C             LDA #C              ; SET DATA = C
  749 73B7  85 EC             STA ACMD
  750 73B9  20 E9 72          JSR RDOC
  751 73BC  C9 0D             CMP #$0D            ; CR?
  752 73BE  D0 06             BNE RDCB1
  753 73C0  68                PLA                 ;YES - GO TO START
  754 73C1  68                PLA                 ;CLEANING STACK UP FIRST
  755 73C2  68                PLA
  756 73C3  4C 86 7D          JMP START
  757                 ;
  758 73C6  C9 20     RDCB1   CMP #' '            ; SPACE
  759 73C8  D0 0A             BNE RDCB2
  760 73CA  20 E9 72          JSR RDOC            ; READ NEXT CHAR
  761 73CD  C9 20             CMP #' '
  762 73CF  D0 0F             BNE RDCB3
  763 73D1  18                CLC                 ; CY=0
  764 73D2  90 12             BCC RDCB4
  765                 ;
  766 73D4  20 EB 73  RDCB2   JSR HEXIT           ; TO HEX
  767 73D7  0A                ASL A
  768 73D8  0A                ASL A
  769 73D9  0A                ASL A
  770 73DA  0A                ASL A
  771 73DB  85 EC             STA ACMD
  772 73DD  20 E9 72          JSR RDOC            ; 2ND CHAR ASSUMED HEX
  773 73E0  20 EB 73  RDCB3   JSR HEXIT
  774 73E3  05 EC             ORA ACMD
  775 73E5  38                SEC                 ; CY=1
  776 73E6  AA        RDCB4   TAX
  777 73E7  68                PLA                 ; RESTORE Y
  778 73E8  A8                TAY
  779 73E9  8A                TXA                 ;SET Z & N FLAGS FOR RETURN
  780 73EA  60                RTS
  781                 ;
  782 73EB  C9 3A     HEXIT   CMP #$3A
  783 73ED  08                PHP                 ; SAVE FLAGS
  784 73EE  29 0F             AND #$0F
  785 73F0  28                PLP
  786 73F1  90 02             BCC HEX09           ; 0-9
  787 73F3  69 08             ADC #8              ; ALPHA ADD 8+CY=9
  788 73F5  60        HEX09   RTS
  789                 ;
  790 73F6                    *=MP3+$F8
  791                 ;
```

```
    MPI TIM PAGE 1

CARD #  LOC     CODE        CARD
   792  73F8  00 70      INTVEC .WORD NMINT        ; DEFAULT USER INTRQ TO NMINT
   793  73FA  00 70             .WORD NMINT
   794  73FC  0E 70             .WORD RESET
   795  73FE  52 70             .WORD INTRQ
   796                    ;

END OF MOS/TECHNOLOGY 6501 ASSEMBLY VERSION 3
NUMBER OF ERRORS =    0,  NUMBER OF WARNINGS =    0
```

SYMBOL TABLE

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES							
ACC	00F9	120	140	194	227	376				
ACMD	00EC	112	246	248	253	486	511	749	771	774
ACRS	71CC	316	247							
ALTER	713A	351	316							
ASCII	735E	673	560	564						
ASCX	7372	689	682	685						
ASC1	7361	679	676							
A2	7146	357	355							
A3	714B	359	352							
A4	715C	362	358							
A5	7152	363	365							
A9	715A	366								
BCCST	7238	478	436	483	516					
BEQS1	7134	345	366	399						
BX	7081	227	198							
BYTE	70EC	285	364	410						
BY2	70F2	297	292							
BY3	70F5	298	286							
B3	7058	201	142							
B5	7073	219								
CMDS	71C6	309	235							
CRDLY	C0F3	104	163	531						
CRLF	728A	528	219	233	385	438	487			
CR1	7293	532	534							
DADD	727C	518	297	402	405	408	456	459	462	468
DAVAIL	0008	85	660							
DCMP	70C1	262	417	447	476	514				
DIFF	00E5	106	265	269	450					
DLX	733C	657								
DLY1	732C	636	611	635						
DLY2	731D	635	532	576	583	614	631			
DL2	7328	643								
DL3	732B	645	646	651						
DSPLYM	711C	334	318							
DSPLYR	7114	330	317							
ERROPR	70BA	258	295	347	419					
ERRP1	71BF	419								
ERRS1	7137	347	335							
FLGS	C0F8	119	209	302	374					
GC	715C	368	319							
GOTDAT	CCC4	86	665							
HEXIT	73EB	782	766	773						
HEXC9	73F5	788	786							
HSP	716F	381	320							
HSPTR	00E7	107	155	231	387	398	601			
HSROP	CCF8	108	156	381	386					
IJMP	70B4	253	250							
INCTMP	7397	719	298	470	509					
INCT1	739C	723	720							
INTRQ	7052	194	795							
INTVEC	73F8	792	149							

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES									
ICBASE	6FCC	87	88	89	90	91	92	93	94			
LCNT	COFF	128										
LH	7174	384	321									
LH1	717E	388	390	418								
LH2	7195	401	395									
LH3	71AA	410	411									
MAJORT	C0EA	110	154	171	184	640						
MCLKIF	6E05	94	169	645								
MCLKRD	6FC4	93	181	649								
MCLKIT	6EC4	92	168	643								
MDA	6FC1	89										
MDB	6FC3	91	146									
MDBK	C016	84	144									
MINORT	C0FB	111	189	641								
MPA	6FCC	88	663									
MPB	6E02	90	164	175	591	595	607	615	659	664	666	668
MPC	70CC	97	138									
MP1	71CC	98	245	316	317	318	319	320	321	322		
MP2	72CC	99										
MP3	73CC	100	790									
M0	7123	337	332									
M1	7127	339	344									
NCMDS	C0C7	96	238	684								
NMINT	70CC	140	792	793								
OUT	72DA	590	582	612	617	633						
OUT1	72E4	595	593									
PCH	C0F7	118	215	275	370							
PCL	C0F6	117	212	273	372							
PREVC	C0E9	109	243	351								
PUTP	70DC	272	356									
RAM64	FFCC	132	133									
RCNT	C0FE	127	299	362	401	440	453	454	455	471		
RDEXIT	73B2	740	738									
RDFSR	733D	659	603	661								
RDCA	73A4	733	334	354	412	424	427					
RDCA2	73AB	737	734									
RDCB	73B3	746	285	394	403	406	733	737				
RDCB1	73C6	758	752									
RDCB2	73D4	766	759									
RDCB3	73EC	773	762									
RDCB4	73E6	776	764									
RDCC	72E9	6C4	236	384	388	421	429	75C	76C	772		
RDT	72E9	6C1	6C4									
RDT1	72FC	6C7	6C9									
RDT2	72FC	614	627									
RDT4	73CF	625	623									
RDT5	7319	632	587									
RESET	7006	144	794									
RC	7022	164	166									
R1	70CC	149	152									
R2	7028	168	172									
R3	702B	169	177	179								
R4	7034	174	17C									
R5	7044	183	188									

SYMBOL	VALUE	LINE DEFINED	CROSS-REFERENCES									
R6	704B	187	185									
SAVX	00FD	124	242	244	481	683						
SETR	7CFB	302	331	357								
SETWRP	73A1	727	724									
SP	00FC	123	217	368								
SPACE	7377	693	339	363	423	426	490	692				
SPAC2	7374	692	251									
START	7086	230	260	345	382	478	756					
SC	7097	238	225									
S1	7099	239	256									
S2	7087	255	240									
TMPC	00FE	125	127	201	220	337	343	422	430	492	504	
TMPC2	00FF	126	128	494	499							
TMPC	00FE	113	264	267	272	274	279	280	289	291	303	305
			340	404	407	458	461	466	493	546	548	708
			711	719	723	736	739					
TMP2	00FC	114	263	266	414	416	710	713				
TMP4	00F2	115	413	415	520	521	522	524				
TMP6	00F4	116										
T2T2	7387	707	425	428								
T2T21	7389	708	715									
UINT	FFF8	95	150	228								
WB	723B	481	433									
WBF1	725A	498	505									
WBF2	7262	503	496	500								
WBNPF	724B	490	512									
WB1	723C	482	515									
WFC	71E2	435	477									
WH1	72C7	455	449	452								
WH2	721F	465	472									
WC	71C2	421	322									
WRAP	00E4	105	232	435	482	727						
WRCA	729A	539	359	488								
WRCA1	72AB	546	540	542	544							
WRCA4	729B	541	474									
WRCA6	72A2	543										
WRCB	7281	555	341	457	460	463	469	549				
WRCC	72C6	578	235	259	445	503	507					
WRPC	72AB	545	330									
WRT	72C6	576	570	578	699							
WRTWO	72CC	568	223	530								
WRT1	72CB	582	586									
XR	00FA	121	206	377								
YR	00FB	122	207	378								
ZTMP	7009	278	393	442								

INSTRUCTION	COUNT
ADC	9
AND	9
ASL	6
BCC	15
BCS	6
BEQ	11
BIT	0
BMI	0
BNE	33
BPL	5
BRK	1
BVC	0
BVS	0
CLC	4
CLD	1
CLI	1
CLV	0
CMP	11
CPX	1
CPY	2
DEC	6
DEX	8
DEY	1
EOR	4
INC	7
INX	3
INY	2
JMP	9
JSR	99
LDA	65
LDX	24
LDY	4
LSR	12
NOP	0
ORA	6
PHA	18
PHP	4
PLA	23
PLP	4
ROL	0
RTI	1
RTS	19
SBC	2
SEC	3
SED	0
SEI	0
STA	45
STX	11
STY	2
TAX	4
TAY	4
TSX	1
TXA	5
TXS	2
TYA	5