MOS

MICROCOMPUTERS

TIM TERMINAL INTERFACE MONITOR MANUAL

Publication Number 6500-20

MCS6500

MICROCOMPUTER FAMILY

TIM MANUAL

MARCH, 1976

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.

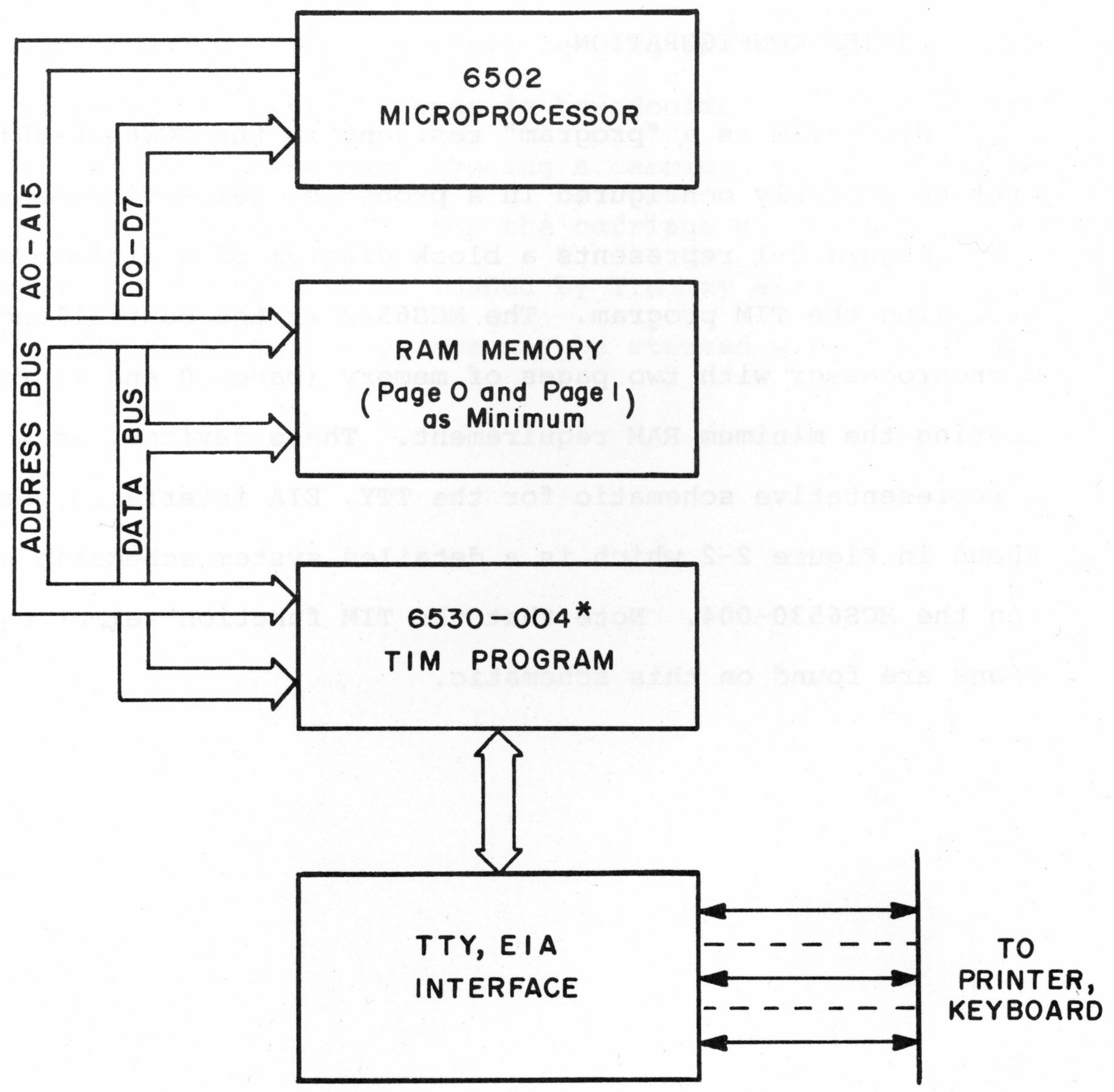
TIM communicates with the user via a serial full-duplex port (using ASCII codes) and automatically adjusts to the speed of the user's terminal. Any speed--even nonstandard ones--can be accommodated. If the user's terminal has a long carriage return time, TIM can be set to perform the proper delay. Commands 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.

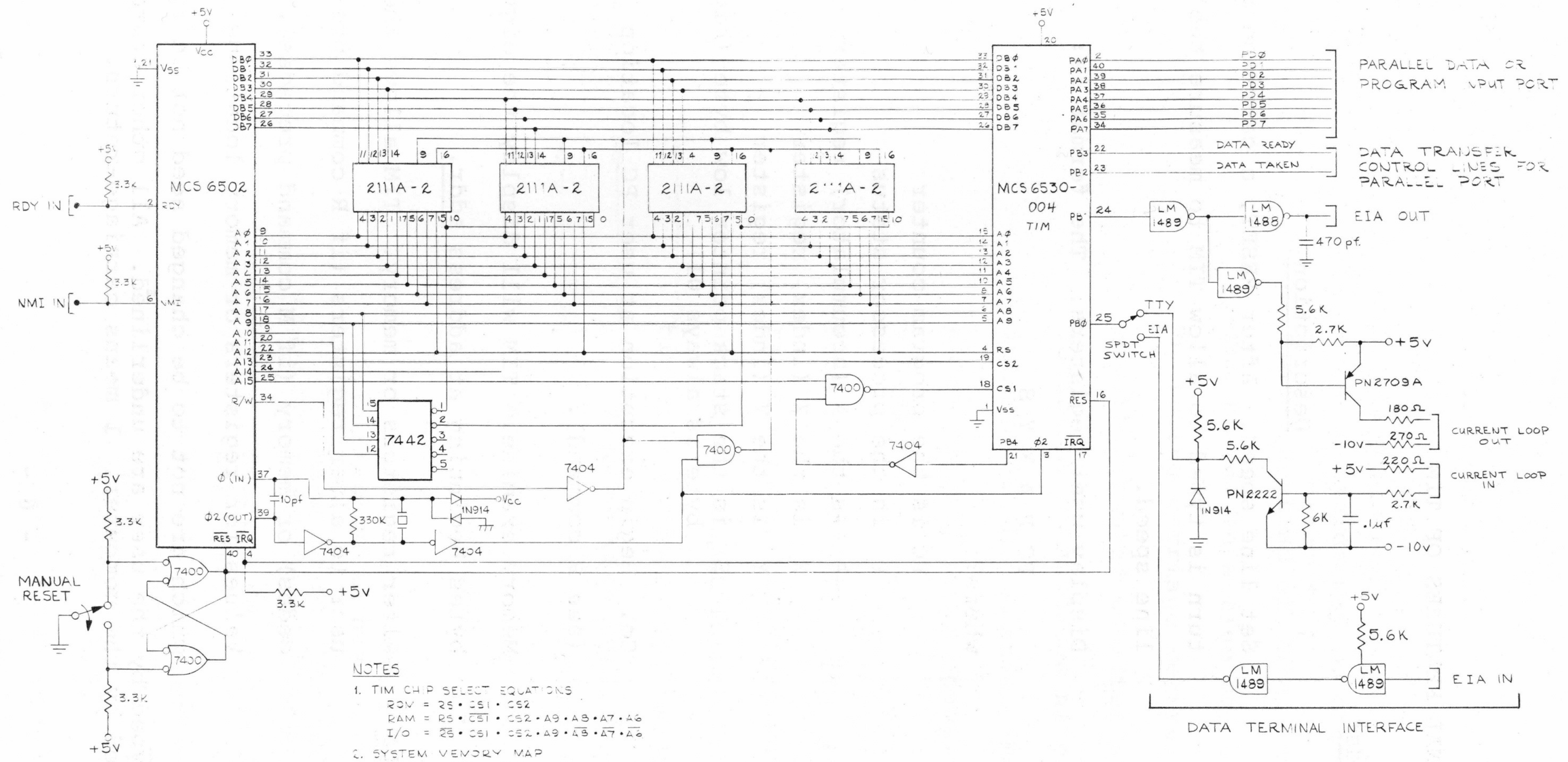
Figure 2-1 represents a block diagram of a minimum system utilizing the TIM program. The MCS6502 is the controlling microprocessor with two pages of memory (pages 0 and 1) representing the minimum RAM requirement. These devices, as well as a representative schematic for the TTY, EIA interfaces, are shown 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
   $ROM = RS \cdot CS1 \cdot CS2$
   $RAM = RS \cdot \overline{CS1} \cdot CS2 \cdot A9 \cdot A8 \cdot A7 \cdot A6$
   $I/O = \overline{RS} \cdot CS1 \cdot CS2 \cdot A9 \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6}$

2. SYSTEM MEMORY MAP
   0000 - 01FF = RAM
   6300 - 630F = TIM I/O
   7000 - 73FF = TIM ROM
   FFA0 - 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).

"TIM" SYSTEM SCHEMATIC

FIGURE 2-2

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.

.<u>LH</u> 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.

.<u>H</u> 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.

.<u>WH</u> <u>addl</u> <u>addh</u>↵ 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.

.<u>WB</u> <u>addl</u> <u>addh</u>↵ 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 <u>before</u> 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
                    ;
                    ;
CCCC                .      *=C             ;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
                    ;
C1CC  20 8A 72      PBIN   JSR CRLF        ;PRINT CARRIAGE RETURN & LINE FEED
 103  20 E9 72             JSR RDT         ;GET A CHARACTER
 1C6  85 CC                STA BINARY      ;SAVE FOR DISSECTION
 1C8  2C 77 73             JSR SPACE       ;PRINT A SPACE
                    ;
01CB  A9 C8                LDA #8          ;INITIALIZE BIT COUNT
01CD  85 01                STA COUNT
                    ;
01CF  A9 3C         PBLOOP LDA #'0         ;ASSUME ZERO: LOAD ASCII "0"
0111  C6 CO                ASL BINARY      ;C=NEXT BIT
0113  BC 02                BCS PRINT       ;PRINT ZERO
                    ;.
0115  A9 31                LDA #'1         ;LOAD ASCII "1"
                    ;
C117  2C C6 72      PRINT  JSR WRT         ;PRINT BINARY DIGIT
C11A  C6 01                DEC COUNT       ;COUNT BIT PRINTED
C11C  1C F1                BPL PBLOOP      ;DO NEXT BIT
                    ;
011E  4C CO 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   F0   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                   *=$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
                ;
  00  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  90 02            BCC 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  D0 F1            BNE PBLOOP   ;DO NEXT BIT
                ;
011E  4C 00 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 locátions 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 <u>not</u> 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   0000                           *=$0000              ;PAGE 0
  11                          ;
  12                          WRT     =$72C2
  13   0000                   LOC     *=*+2                ;TEST CELL ADDR
  14   0002                   LOW     *=*+2                 ;LOWER LIMIT OF TEST
  15   0004                   HIGH    *=*+2                ;UPPER LIMIT OF TEST+1
  16   0006                   PTR     *=*+2                ;POINTER TO CELL UNDER TEST
  17                          ;
  18   0008                           *=$0010              ;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   0020    A2 00                  LDX #0
  28                          ;
  29                          ;CLEAR MEMORY AREA UNDER TEST
  30   0022    A9 00          ML1     LDA #0
  31   0024    81 06                  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 $FF IN SELEXTED CELL
  36   002B    A9 FF          TEST    LDA #$FF
  37   002D    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 06          VLOOP   LDA (PTR,X)          ;GET CELL
  42   0034    F0 17                  BEQ NEXTC            ;CK IF ZERO
  43   0036    A4 06                  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 07          OK1     LDY PTR+1
```

```
CARD # LOC      CODE        CARD
   49  003F   C4 01                CPY LOC+1
   50  0041   F0 01                BEQ OK2
   51  0043   00                   BRK                ;NOT (LOC)
   52                       ;
   53  0044   C9 FF         OK2    CMP #$FF           ;IS (LOC)--IS DATA OK?
   54  0046   F0 01                BEQ OK3
   55  0048   00                   BRK                ;WRONG DATA
   56                       ;
   57  0049   A9 00         OK3    LDA #0             ;RESET (LOC)
   58  004B   81 00                STA (LOC,X)
   59                       ;
   60  004D   20 7A 00      NEXTC  JSR INCPTR         ;NEXT CELL

   61  0050   D0 E0                BNE VLOOP          ,IF NOT AT LIMIT
   62                       ;
   63  0052   A5 00                LDA LOC            ;PRINT STAR EVERY PAGE BOUNDA
   64  0054   D0 07                BNE NOSTAR
   65  0056   A9 2A                LDA #'*
   66  0058   20 C2 72             JSR WRT
   67  005B   A2 00                LDX #0             ;FIX X AFTER MON CALL
   68                       ;
   69  005D   20 8B 00      NOSTAR JSR INCLOC         ;NEXT LOC

   70  0060   D0 C9                BNE TEST
   71                       ;
   72  0062   20 68 00             JSR RSTLOC         ;PASS COMPLETE

   73  0065   4C 10 00             JMP MAD            ;NEXT PASS
   74                       ;
   75                       ;RESET LOC TO LOW
   76  0068   A5 02         RSTLOC LDA LOW
   77  006A   85 00                STA LOC
   78  006C   A5 03                LDA LOW+1
   79  006E   85 01                STA LOC+1
   80  0070   60                   RTS
   81                       ;
   82                       ;RESET PTR TO LOW
   83  0071   A5 02         RSTPTR LDA LOW
   84  0073   85 06                STA PTR
   85  0075   A5 03                LDA LOW+1
   86  0077   85 07                STA PTR+1
   87  0079   60                   RTS
   88                       ;
   89                       ;INCREMENT PTR & CHECK FOR LIMIT
   90  007A   E6 06         INCPTR INC PTR            ;INCREMENT
   91  007C   D0 02                BNE INC1
   92                       ;
   93  007E   E6 07                INC PTR+1
   94                       ;
   95  0080   A5 04         INC1   LDA HIGH           ;CHECK
   96  0082   C5 06                CMP PTR
   97  0084   D0 04                BNE IPRET          ;NOT AT LIMIT
```

```
CARD # LOC      CODE        CARD
   98                    ;
   99  0086  A5 05              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 00       INCLOC INC LOC            ;INCR
  106  008D  D0 02              BNE INC2
  107                    ;
  108  008F  E6 01              INC LOC+1
  109                    ;
  110  0091  A5 04       INC2   LDA HIGH           ;CHECK
  111  0093  C5 00              CMP LOC
  112  0095  D0 04              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     69
 INCPTR    007A          90     32   60
 INC1      0080          95     91
 INC2      0091         110    106
 IPRET     008A         102     97
 LOC       0000          13     37   44   49    58    63    77    79   105   108   111
                               114
 LOW       0002          14     76   78   83    85
 MAD       001C          20     73
 ML1       0022          30     33
 NEXTL     004D          60     42
 NOSTAR    005D          69     64
 CK1       003D          48     45
 CK2       0044          53     50
 CK3       0049          57     54
 PTR       0006          16     31   41   43    48    84    86    90    93    96   100
 RSTLOC    0068          76     25   72
 RSTPTR    0071          83     26   39
 TEST      0028          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
                        ;
                        ;       MOS TECHNOLOGY 650X TERMINAL INTERFACE MONITOR (TIM)
                        ;               VERSION 1.0  AUGUST 31, 1975
                        ;               COPYRIGHT 1975 MOS TECHNOLOGY
                        ;               ALL RIGHTS RESERVED.  UNAUTHORIZED USE
                        ;               OF ALL OR PART STRICTLY PROHIBITED.
                        ; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ;
                        ;  PROMPTING CHARACTER IS A PERIOD (.)
                        ;  -----------------------------------
                        ;
                        ;
                        ;  DISPLAY COMMANDS
                        ;  ----------------
                        ;
                        ;  .R           DISPLAY REGISTERS (PC,F,A,X,Y,SP)
                        ;  .M  ADDR     DISPLAY MEMORY ( 8 BYTES BEGINNING AT ADDR )
                        ;
                        ;
                        ;  ALTER COMMAND (:)
                        ;  -----------------
                        ;  .:  DATA       ALTERS PREVIOUSLY DISPLAYED ITEM OR NEXT ITEM
                        ;
                        ;
                        ;  PAPER TAPE I/O COMMANDS
                        ;  -----------------------
                        ;
                        ;  .LH                        LOAD HEX TAPE
                        ;  .WB ADDR1 ADDR2            WRITE BNPF TAPE (FROM LOW ADDR1 TO HIGH ADDR2)
                        ;  .WH ADDR1 ADDR2            WRITE HEX TAPE (FROM LOW ADDR1 TO HIGH ADDR2)
                        ;
                        ;  CONTROL COMMANDS
                        ;  ----------------
                        ;
                        ;  .G                         GO, CONTINUE EXECUTION FROM CURRENT PC ADDRESS
                        ;
                        ;  .H                         TOGGLES HIGH-SPEED-READER OPTION
                        ;                               (IF ITS ON, TURNS IT OFF; IF OFF, TURNS ON
                        ;
                        ;  BRK AND NMI ENTRY POINTS TO TIM
                        ;  -------------------------------
                        ;
                        ;      TIM IS NORMALLY ENTERED WHEN A 'BRK' INSTRUCTION IS
                        ;          ENCOUNTERED DURING PROGRAM EXECUTION.  AT THAT
                        ;          TIME CPU REGISTERS ARE OUTPUT:  * PC F A X Y SP
                        ;          AND CONTROL IS GIVEN TO THE KEYBOARD.
                        ;      USER MAY ENTER TIM BY PROGRAMMED BRK OR INDUCED NMI.  NMI
                        ;          ENTRIES CAUSE A '#' TO PRECEDE THE '*' IN THE CPU REGISTER
                        ;          PRINTOUT FORMAT
                        ;
                        ;  NON-BRK INTRQ (EXTERNAL DEVICE) INTERRUPT HANDLING
                        ;  --------------------------------------------------
```

```
CARD # LOC      CODE        CARD
                            ;
                            ;       A NON-BRK INTRQ INTERRUPT CAUSES AN INDIRECT JUMP TO THE ADDRESS
                            ;           LOCATED AT 'UINT' (HEX FFF8).  THIS LOCATION CAN BE SET
                            ;           USING THE ALTER CMD, OR LOADED AUTOMATICALLY IN PAPER TAPE
                            ;           FORM WITH THE LH CMD IF THE USER ASSIGNS HIS INTRQ INTERRUPT
                            ;           VECTOR TO $FFF8 IN THE SOURCE ASSEMBLY PROGRAM.
                            ;       IF NOT RESET BY THE USER, UINT IS SET TO CAUSE EXTERNAL
                            ;           DEVICE INTERRUPTS TO ENTER TIM AS NMI'S.  I.E.,
                            ;           IF A NMI OCCURS WITHOUT AN INDUCED NMI SIGNAL, IT IS
                            ;           AN EXTERNAL DEVICE INTERRUPT.
                            ;
                            ;   SETTING AND RESETTING PROGRAM BREAKPOINTS
                            ;   ------------------------------------------
                            ;
                            ;       BREAKPOINTS ARE SET AND RESET USING THE MEMORY DISPLAY
                            ;           AND ALTER COMMANDS.  BRK HAS A '00' OPERATION CODE.
                            ;       TO SET A BREAKPOINT SIMPLY DISPLAY THE MEMORY LOCATION
                            ;           (FIRST INSTRUCTION BYTE) AT WHICH THE BREAKPOINT IS
                            ;           TO BE PLACED THEN ALTER THE LOCATION TO '00'.  THERE IS
                            ;           NO LIMIT TO THE NUMBER OF BREAKPOINTS THAT CAN BE
                            ;           ACTIVE AT ONE TIME.
                            ;       TO RESET A BREAKPOINT, RESTORE THE ALTERED MEMORY LOCATION
                            ;           TO ITS ORIGINAL VALUE.
                            ;       WHEN AND IF A BREAKPOINT IS ENCOUNTERED DURING EXECTUION,
                            ;           THE BREAKPOINT DATA PRECEDED BY AN '*' IS DISPLAYED.
                            ;           THE PROGRAM COUNTER VALUE DISPLAYED IS THE BRK
                            ;           INSTRUCTION LOCATION + 1.
                            ;
                            ;   -------------------------------------------------------------------
                            ;
                            MDBK    =%00010110          ; X,X,X,POR,DATA-AVAIL,GOT-DATA,SERIAL-OUT,IN
                            DAVAIL  =$08
                            GOTDAT  =$04
                            IOBASE  =$6E00
                            MPA     =IOBASE+0
                            MDA     =IOBASE+1
                            MPB     =IOBASE+2
                            MDB     =IOBASE+3
                            MCLKIT  =IOBASE+4
                            MCLKRD  =IOBASE+4
                            MCLKIF  =IOBASE+5
                            UINT    =$FFF8
                            NCMDS   =7
                            MP0     =$7000
                            MP1     =$7100
                            MP2     =$7200
                            MP3     =$7300
                            ;
                            ;  ZERO PAGE MONITOR RESERVE AREA
                            ;
                            CRDLY   =227                ;DELAY FOR CR IN BIT-TIMES
                            WRAP    =228                ;ADDRESS WRAP-AROUND FLAG
```

```
     TIM VERSION 1.0 - MEM PAGE 0

CARD # LOC       CODE         CARD
  106                       DIFF    =229
  107                       HSPTR   =231
  108                       HSRCP   =232
  109                       PREVC   =233
  110                       MAJORT  =234
  111                       MINCRT  =235
  112                       ACMD    =236
  113                       TMP0    =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   =$FFC0
  133   0000                        *=RAM64
```

```
    MP0 TIM PAGE 0

CARD # LOC      CODE          CARD
  135                       ;
  136                       ;
  137                       ;  TIM PAGE 0 (RELATIVE)
  138    FFC0                       *=MP0
  139                       ;
  140    7000   85 F9       NMINT   STA ACC          ; SAVE A
  141    7002   A9 23               LDA #'#          ; SET A=# TO INDICATE NMINT ENTRY
  142    7004   D0 55               BNE B3           ; JMP B3
  143                       ;
  144    7006   A9 16       RESET   LDA #MDBK        ; INIT DIR REG, PCR TO 1 RELOCATES
  145                       ;
  146    7008   8D 03 6E            STA MDB
  147                       ;
  148    700B   A2 08               LDX #8           ; X=0
  149    700D   BD 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 HO TO LO
```

```
CARD # LOC      CODE          CARD
  187   704B   C8          R6      INY
  188   704C   F0 F6               BEQ R5
  189   704E   85 EB               STA MINORT
  190                      ;
  191   7050   58                  CLI                   ; ENABLE INTS
  192   7051   00                  BRK                   ; ENTER TIM BY BRK
  193                      ;
  194   7052   85 F9       INTRQ   STA ACC               ; SAVE ACC
  195   7054   68                  PLA                   ; FLAGS TO A
  196   7055   48                  PHA                   ; RESTORE STACK STATUS
  197   7056   29 10               AND #$10              ; TEST BRK FLAG
  198   7058   F0 27               BEQ BX                ; USER INTERRUPT
  199                      ;
  200   705A   0A                  ASL A                 ; SET A=SPACE (10 X 2 = 20)
  201   705B   85 FE       B3      STA TMPC              ; SAVE INT TYPE FLAG
  202   705D   D8                  CLD                   ; CLEAR DECIMAL MODE
  203   705E   4A                  LSR A                 ; # IS ODD, SPACE IS EVEN
  204                                                    ; SET CY FOR PC BRK CORRECTION
  205                      ;
  206   705F   86 FA               STX XR                ; SAVE X
  207   7061   84 FB               STY YR                ; Y
  208   7063   68                  PLA
  209   7064   85 F8               STA FLGS              ; FLAGS
  210   7066   68                  PLA
  211   7067   69 FF               ADC #$FF              ; CY SET TO PC-1 FOR BRK
  212   7069   85 F6               STA PCL
  213   706B   68                  PLA
  214   706C   69 FF               ADC #$FF
  215   706E   85 F7               STA PCH
  216   7070   BA                  TSX
  217   7071   86 FC               STX SP                ; SAVE ORIG SP
  218                      ;
  219   7073   20 8A 72    B5      JSR CRLF
  220   7076   A6 FE               LDX TMPC
  221                      ;
  222   7078   A9 2A               LDA #'*
  223   707A   20 C0 72            JSR WRTWO
  224   707D   A9 52               LDA #'R               ; SET FOR R DISPLAY TO PERMIT
  225   707F   D0 16               BNE S0                ;   IMMEDIATE ALTER FOLLOWING BREAKPOINT.
  226                      ;
  227   7081   A5 F9       BX      LDA ACC
  228   7083   6C F8 FF            JMP (UINT)            ; CONTROL TO USER INTRQ SERVICE ROUTINE
  229                      ;
  230   7086   A9 00       START   LDA #0                ;NEXT COMMAND FROM USER
  231   7088   85 E7               STA HSPTR             ;CLEAR H. S. PAPER TAPE FLAG
  232   708A   85 E4               STA WRAP              ;CLEAR ADDRESS WRAP-AROUND FLAG
  233   708C   20 8A 72            JSR CRLF
  234   708F   A9 2E               LDA #'.               ; TYPE PROMPTING '.'
  235   7091   20 C6 72            JSR WROC
  236   7094   20 E9 72            JSR RDOC              ; READ CMD, CHAR RETURNED IN A
  237                      ;
  238   7097   A2 06       S0      LDX #NCMDS-1          ; LOOK-UP CMD
```

```
    MPO TIM PAGE 0

CARD # LOC      CODE        CARD
  239  7099   DD 06 71  S1      CMP CMDS,X
  240  709C   D0 19             BNE S2
  241                   ;
  242  709E   A5 FD             LDA SAVX          ; SAVE PREVIOUS CMD
  243  70A0   85 E9             STA PREVC
  244  70A2   86 FD             STX SAVX          ; SAVE CURRENT CMD INDEX
  245  70A4   A9 71             LDA #MP1/256      ; JMP INDIRECT TO CMD CODE
  246  70A6   85 ED             STA ACMD+1        ;   ALL CMD CODE BEGINS ON MP1
  247  70A8   BD 0D 71          LDA ADRS,X
  248  70AB   85 EC             STA ACMD
  249  70AD   E0 03             CPX #3            ; IF :, R OR M (0, 1, OR 2)  SPACE 2
  250  70AF   B0 03             BCS IJMP
  251  70B1   20 74 73          JSR SPAC2
  252                   ;
  253  70B4   6C EC 00  IJMP    JMP (ACMD)
  254                   ;
  255  70B7   CA        S2      DEX
  256  70B8   10 DF             BPL S1            ; LOOP FOR ALL CMDS
  257                   ;
  258  70BA   A9 3F     ERROPR  LDA #'?           ; OPERATOR ERR, TYPE '?', RESTART
  259  70BC   20 C6 72          JSR WROC
  260  70BF   90 C5             BCC START         ; JMP START (WROC RETURNS CY=0)
  261                   ;
  262  70C1   38        DCMP    SEC               ; TMP2-TMP0 DOUBLE SUBTRACT
  263  70C2   A5 F0             LDA TMP2
  264  70C4   E5 EE             SBC TMP0
  265  70C6   85 E5             STA DIFF
  266  70C8   A5 F1             LDA TMP2+1
  267  70CA   E5 EF             SBC TMP0+1
  268  70CC   A8                TAY               ;RETURN HIGH ORDER PART IN Y
  269  70CD   05 E5             ORA DIFF          ; OR LO FOR EQU TEST
  270  70CF   60                RTS
  271                   ;
  272  70D0   A5 EE     PUTP    LDA TMP0          ; MOVE TMP0 TO PCH,PCL
  273  70D2   85 F6             STA PCL
  274  70D4   A5 EF             LDA TMP0+1
  275  70D6   85 F7             STA PCH
  276  70D8   60                RTS
  277                   ;
  278  70D9   A9 00     ZTMP    LDA #0            ; CLEAR REGS
  279  70DB   95 EE             STA TMP0,X
  280  70DD   95 EF             STA TMP0+1,X
  281  70DF   60                RTS
  282                   ;
  283                   ;  READ AND STORE BYTE.  NO STORE IF SPACE OR RCNT=0.
  284                   ;
  285  70E0   20 B3 73  BYTE    JSR RDOB          ; CHAR IN A, CY=0 IF SP
  286  70E3   90 10             BCC BY3           ; SPACE
  287                   ;
  288  70E5   A2 00             LDX #0            ; STORE BYTE
  289  70E7   81 EE             STA (TMP0,X)
  290                   ;
```

```
CARD # LOC      CODE         CARD
  291  70E9   C1 FE                 CMP (TMPO,X)        ; TEST FOR VALID WRITE (RAM)
  292  70EB   F0 05                 BEQ BY2
  293  70ED   68                    PLA                 ; ERR, CLEAR JSR ADR IN STACK
  294  70EE   68                    PLA
  295  70EF   4C BA 70              JMP ERROPR
  296                        ;
  297  70F2   20 7C 72      BY2     JSR DADD            ; INCR CKSUM
  298  70F5   20 97 73      BY3     JSR INCTMP          ; GO INCR TMPO ADR
  299  70F8   C6 FE                 DEC RCNT
  300  70FA   60                    RTS
  301                        ;
  302  70FB   A9 F8         SETR    LDA #FLGS           ; SET TO ACCESS REGS
  303  70FD   85 FE                 STA TMPO
  304  70FF   A9 00                 LDA #0
  305  7101   85 FF                 STA TMPO+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   50                    .BYTE GO-MP1
  320  7111   6F                    .BYTE HSP-MP1
  321  7112   74                    .BYTE LH-MP1
  322  7113   C2                    .BYTE WO-MP1
```

```
CARD # LOC     CODE        CARD
                           ;
                           ;
                           ;  NOTE -- ALL CMD CODE MUST BEGIN ON MP1
                           ;
                           ;  DISPLAY REG CMD - A,F,X,Y, AND SP
                           ;
       7114   20 A6 72   DSPLYR JSR WRPC          ; WRITE PC
       7117   20 FB 70          JSR SETR
       711A   D0 07             BNE M0            ; USE DSPLYM
                           ;
       711C   20 A4 73   DSPLYM JSR RDOA          ; READ MEM ADR INTO TMPC
       711F   90 16             BCC ERRS1         ; ERR IF NO ADDR
       7121   A9 08             LDA #8
       7123   85 FE      M0     STA TMPC
       7125   A0 00             LDY #0
       7127   20 77 73   M1     JSR SPACE         ; TYPE 8 BYTES OF MEM
       712A   B1 EE             LDA (TMPO),Y      ; (TMPO) PRESERVED FOR POSS ALTER
       712C   20 B1 72          JSR WROB
       712F   C8                INY               ; INCR INDEX
       7130   C6 FE             DEC TMPC
       7132   D0 F3             BNE M1
       7134   4C 86 70   BEQS1  JMP START
                           ;
       7137   4C BA 70   ERRS1  JMP ERROPR
                           ;
                           ;  ALTER LAST DISPLAYED ITEM (ADR IN TMPC)
                           ;
       713A   C6 E9      ALTER  DEC PREVC         ; R INDEX = 1
       713C   D0 0D             BNE A3
                           ;
       713E   20 A4 73          JSR RDOA          ; CY=0 IF SP
       7141   90 03             BCC A2            ; SPACE
       7143   20 DC 70          JSR PUTP          ; ALTER PC
       7146   20 FB 70   A2     JSR SETR          ; ALTER R'S
       7149   D0 C5             BNE A4            ; JMP A4 (SETR RETURNS ACC = 5)
       714B   20 9A 72   A3     JSR WROA          ; ALTER M, TYPE ADR
       714E   A9 08             LDA #8            ; SET CNT=8
                           ;
       7150   85 FE      A4     STA RCNT
       7152   20 77 73   A5     JSR SPACE         ; PRESERVES Y
       7155   20 E0 70          JSR BYTE
       7158   D0 F8             BNE A5
       715A   F0 D8      A9     BEQ BEQS1
                           ;
       715C   A6 FC      GO     LDX SP
       715E   9A                TXS               ; ORIG OR NEW SP VALUE TO SP
       715F   A5 F7             LDA PCH
       7161   48                PHA
       7162   A5 F6             LDA PCL
       7164   48                PHA
       7165   A5 F8             LDA FLGS
       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 HSROP           ; TOGGLE BIT 0
  382   7171   4C 86 70          JMP START
  383                      ;
  384   7174   20 E9 72   LH     JSR RDOC            ; READ SECOND CMD CHAR
  385   7177   20 8A 72          JSR CRLF
  386   717A   A6 E8             LDX HSROP           ; ENABLE PTR OPTION IF SET
  387   717C   86 E7             STX HSPTR
  388   717E   20 E9 72   LH1    JSR RDOC
  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 RDOB
  395   718D   D0 06             BNE LH2
  396                      ;
  397   718F   A2 00             LDX #0              ; CLEAR HS RDR FLAG
  398   7191   86 E7             STX HSPTR
  399   7193   F0 9F             BEQ BEQS1           ; 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 RDOB            ; SA HO TO TMP0+1
  404   719D   85 EF             STA TMP0+1
  405   719F   20 7C 72          JSR DADD            ; ADD TO CKSUM
  406   71A2   20 B3 73          JSR RDOB            ; SA LO TO TMP0
  407   71A5   85 EE             STA TMP0
  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 RDOA            ; CKSUM FROM HEX RCD TO TMP0
  413   71B2   A5 F2             LDA TMP4            ; TMP4 TO TMP2 FOR DCMP
  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 DCMP
  418   71BD   F0 BF             BEQ LH1
  419   71BF   4C BA 70   ERRP1  JMP ERROPR
  420                      ;
  421   71C2   20 E9 72   WO     JSR RDOC            ; RD 2ND CMD CHAR
  422   71C5   85 FE             STA TMPC
  423   71C7   20 77 73          JSR SPACE
  424   71CA   20 A4 73          JSR RDOA
  425   71CD   20 87 73          JSR T2T2          ; SA TO TMP2
  426   71D0   20 77 73          JSR SPACE           ; SPACE BEFORE NEXT ADDRESS
  427   71D3   20 A4 73          JSR RDOA
```

```
CARD # LOC      CODE        CARD
 428   71D6   20 87 73           JSR T2T2             ; SA TO TMP0, EA TO TMP2
 429   71D9   20 E9 72           JSR RDOC             ; DELAY FOR FINAL CR
 430   71DC   A5 FE              LDA TMP0
 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 E5              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 DADD             ; ADD TO CKSUM
 457   720C   20 B1 72           JSR WROB             ; RCD CNT  IN A
 458   720F   A5 EF              LDA TMP0+1           ; SA HO
 459   7211   20 7C 72           JSR DADD
 460   7214   20 B1 72           JSR WROB
 461   7217   A5 EE              LDA TMP0             ; SA LO
 462   7219   20 7C 72           JSR DADD
 463   721C   20 B1 72           JSR WROB
 464                      ;
 465   721F   A0 00       WH2    LDY #0
 466   7221   B1 EE              LDA (TMP0),Y
 467                      ;
 468   7223   20 7C 72           JSR DADD             ; 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 WROA4            ; 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                      ;
```

```
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 WROA             ; 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 WROC             ; 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 WROC             ; 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 7C             JSR DCMP
  515   7278   B0 C3                BCS WB1              ; LOOP WHILE EA GT OR = SA
  516   727A   90 BC                BCC BCCST
  517                          ;
  518   727C   48          DADD     PHA                  ; SAVE A
  519   727D   18                   CLC
  520   727E   65 F2                ADC TMP4
  521   7280   85 F2                STA TMP4
  522   7282   A5 F3                LDA TMP4+1
  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
```

```
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 TMPO STORES
  538                     ;
  539   729A   A2 01      WROA     LDX #1
  540   729C   D0 0A               BNE WROA1
  541   729E   A2 05      WROA4    LDX #5
  542   72A0   D0 06               BNE WROA1
  543   72A2   A2 07      WROA6    LDX #7
  544   72A4   D0 02               BNE WROA1
  545   72A6   A2 09      WRPC     LDX #9
  546   72A8   B5 ED      WROA1    LDA TMPO-1,X
  547   72AA   48                  PHA
  548   72AB   B5 EE               LDA TMPO,X
  549   72AD   20 B1 72            JSR WROB
  550   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         WROB     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                     ;
  582   72CE   20 DA 72   WRT1     JSR OUT
  583   72D1   20 1D 73            JSR DLY2
```

```
        LOC     CODE          CARD
        72D4    4A                       LSR A
        72D5    CA                       DEX
        72D6    D0 F6                    BNE WRT1
        72D8    F0 3F                    BEQ RDT5
                                                             ; *USE BNE?
                              ;
        72DA    48            OUT        PHA                 ; SAVE A
        72DB    AD 02 6E                 LDA MPB             ; OUTPUT BIT FROM CY
        72DE    29 FD                    AND #%11111101
        72E0    90 02                    BCC OUT1
        72E2    09 02                    ORA #%00000010
        72E4    8D 02 6E      OUT1       STA MPB
        72E7    68                       PLA                 ; RESTORE A
        72E8    60                       RTS
                              ;
                              ;  OUTPUT  RETURNS CHAR IN A
                              ;
        72E9    A5 E7         RDT        LDA HSPTR           ; TEST HS PTR OPTION
        72EB    4A                       LSR A
        72EC    B0 4F                    BCS RDHSR
                              RDOC       =RDT
        72EE    A2 08                    LDX #8
                              ;
        72F0    AD 02 6E      RDT1       LDA MPB
        72F3    4A                       LSR A               ; WAIT FOR START BIT
        72F4    90 FA                    BCC RDT1
                              ;
        72F6    20 20 73                 JSR DLY1
        72F9    20 DA 72                 JSR OUT             ; ECHO START BIT
                              ;
        72FC    20 1D 73      RDT2       JSR DLY2
        72FF    AD 02 6E                 LDA MPB             ; CY = NEXT BIT
        7302    4A                       LSR A
        7303    20 DA 72                 JSR OUT             ; ECHO
                              ;
        7306    08                       PHP                 ; SAVE BIT
        7307    98                       TYA                 ; Y CONTAINS CHAR BEING FORMED
        7308    4A                       LSR A
        7309    28                       PLP                 ; RECALL BIT
        730A    90 02                    BCC RDT4
        730C    09 80                    ORA #$80            ; ADD IN NEXT BIT
        730E    A8            RDT4       TAY
        730F    CA                       DEX
        7310    D0 EA                    BNE RDT2            ; LOOP FOR 8 BITS
        7312    49 FF                    EOR #$FF            ; COMPLEMENT DATA
        7314    29 7F                    AND #$7F            ; CLEAR PARITY
                              ;
        7316    20 1D 73                 JSR DLY2
        7319    18            RDT5       CLC
        731A    20 DA 72                 JSR OUT             ;AND DELAY 2 HALF-BIT-TIMES
                              ;
        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  A6 EA              LDX MAJORT
  641  7326  A5 EB              LDA MINORT
  642                    ;
  643  7328  8D C4 6E    DL2    STA MCLKIT
  644                    ;
  645  732B  AD C5 6E    DL3    LDA MCLKIF
  646  732E  10 FB              BPL DL3
  647  7330  CA                 DEX
  648  7331  08                 PHP
  649  7332  AD C4 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
```

```
CARD # LOC     CODE        CARD
       7371  48                   PHA
       7372  68          ASCX     PLA
       7373  60                   RTS
                         ;
       7374  20 77 73    SPAC2    JSR SPACE
       7377  48          SPACE    PHA                   ; SAVE A,X,Y
       7378  8A                   TXA
       7379  48                   PHA
       737A  98                   TYA
       737B  48                   PHA
       737C  A9 20                LDA #'
       737E  20 C6 72             JSR WRT               ; TYPE SP
       7381  68                   PLA                   ; RESTORE A,X,Y
       7382  A8                   TAY
       7383  68                   PLA
       7384  AA                   TAX
       7385  68                   PLA
       7386  60                   RTS
                         ;
       7387  A2 02       T2T2     LDX #2
       7389  B5 ED       T2T21    LDA TMP0-1,X
       738B  48                   PHA
       738C  B5 EF                LDA TMP2-1,X
       738E  95 ED                STA TMP0-1,X
       7390  68                   PLA
       7391  95 EF                STA TMP2-1,X
       7393  CA                   DEX
       7394  D0 F3                BNE T2T21
       7396  60                   RTS
                         ;
                         ;INCREMENT (TMP0,TMP0+1) BY 1
       7397  E6 EE       INCTMP   INC TMP0              ;LOW BYTE
       7399  F0 01                BEQ INCT1
       739B  60                   RTS
                         ;
       739C  E6 EF       INCT1    INC TMP0+1            ;HIGH BYTE
       739E  F0 01                BEQ SETWRP
       73A0  60                   RTS
                         ;
       73A1  E6 E4       SETWRP   INC WRAP              ;POINTER HAS WRAPPED AROUND - SET FLAG
       73A3  60                   RTS
                         ;
                         ; READ HEX ADR, RETURN HO  IN TMP0, LO IN TMP0+1 AND CY=1
                         ;     IF SP CY=0
                         ;
       73A4  20 B3 73    RDOA     JSR RDOB              ; READ 2 CHAR BYTE
       73A7  90 02                BCC RDOA2             ; SPACE
                         ;
       73A9  85 EF                STA TMP0+1
       73AB  20 B3 73    RDOA2    JSR RDOB
       73AE  90 02                BCC RDEXIT            ; SP
       73B0  85 EE                STA TMP0
```

```
CARD # LOC      CODE        CARD
       73B2   60          RDEXIT RTS
                          ;
                          ;  READ HEX BYTE AND RETURN IN A, AND CY=1
                          ;     IF SP CY=0
                          ;     Y REG IS PRESERVED
                          ;
       73B3   98          RDOB    TYA                ; SAVE Y
       73B4   48                  PHA
       73B5   A9 00               LDA #0             ; SET DATA = 0
       73B7   85 EC               STA ACMD
       73B9   20 E9 72            JSR RDOC
       73BC   C9 0D               CMP #$0D           ; CR?
       73BE   D0 06               BNE RDOB1
       73C0   68                  PLA                ;YES - GO TO START
       73C1   68                  PLA                ;CLEANING STACK UP FIRST
       73C2   68                  PLA
       73C3   4C 86 70            JMP START
                          ;
       73C6   C9 20       RDOB1   CMP #'             ; SPACE
       73C8   D0 0A               BNE RDOB2
       73CA   20 E9 72            JSR RDOC           ; READ NEXT CHAR
       73CD   C9 20               CMP #'
       73CF   D0 0F               BNE RDOB3
       73D1   18                  CLC                ; CY=0
       73D2   90 12               BCC RDOB4
                          ;
       73D4   20 EB 73    RDOB2   JSR HEXIT          ; TO HEX
       73D7   0A                  ASL A
       73D8   0A                  ASL A
       73D9   0A                  ASL A
       73DA   0A                  ASL A
       73DB   85 EC               STA ACMD
       73DD   20 E9 72            JSR RDOC           ; 2ND CHAR ASSUMED HEX
       73E0   20 EB 73    RDOB3   JSR HEXIT
       73E3   05 EC               ORA ACMD
       73E5   38                  SEC                ; CY=1
       73E6   AA          RDOB4   TAX
       73E7   68                  PLA                ; RESTORE Y
       73E8   A8                  TAY
       73E9   8A                  TXA                ;SET Z & N FLAGS FOR RETURN
       73EA   60                  RTS
                          ;
       73EB   C9 3A       HEXIT   CMP #$3A
       73ED   08                  PHP                ; SAVE FLAGS
       73EE   29 0F               AND #$0F
       73F0   28                  PLP
       73F1   90 02               BCC HEX09          ; 0-9
       73F3   69 08               ADC #8             ; ALPHA ADD 8+CY=9
       73F5   60          HEX09   RTS
                          ;
       73F6                       *=MP3+$F8
                          ;
```

```
CARD # LOC      CODE        CARD
  792   73F8  00 70      INTVEC .WORD NMINT        ; DEFAULT USER INTRQ TO NMINT
  793   73FA  00 70             .WORD NMINT
  794   73FC  06 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  [illegible]  774
ADRS      710D         316   247
ALTER     713A         351   316
ASCII     7358         673   560  564
ASCX      7372         689   682  685
ASC1      7361         679   676
A2        7146         357   355
A3        714B         359   352
A4        7150         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      70E0         285   364  410
BY2       70F2         297   292
BY3       70F5         298   286
B3        705B         201   142
B5        7073         219
CMDS      7106         309   239
CRDLY     00E3         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      00F8         119   209  302  374
GO        715C         368   319
GOTDAT    00C4          86   665
HEXIT     73EB         782   766  773
HEXC9     73F5         788   786
HSP       716F         381   320
HSPTR     00E7         107   155  231  387  398  601
HSROP     00E8         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

IOBASE   6E00        87    88    89    90    91    92    93    94
LCNT     00FF       128
LH       7174       384   321
LH1      717E       388   390   418
LH2      7195       401   395
LH3      71AA       410   411
MAJORT   00EA       110   154   171   184   640
MCLKIF   6E05        94   169   645
MCLKRD   6E04        93   181   649
MCLKIT   6E04        92   168   643
MDA      6E01        89
MDB      6E03        91   146
MDBK     0016        84   144
MINORT   00EB       111   189   641
MPA      6E00        88   663
MPB      6E02        90   164   175   591   595   607   615   659   664   666   668
MPC      7000        97   138
MP1      7100        98   245   316   317   318   319   320   321   322
MP2      7200        99
MP3      7300       100   790
M0       7123       337   332
M1       7127       339   344
NCMDS    0007        96   238   684
NMINT    7000       140   792   793
OUT      72DA       590   582   612   617   633
OUT1     72E4       595   593
PCH      00F7       118   215   275   370
PCL      00F6       117   212   273   372
PREVC    00E9       109   243   351
PUTP     70DC       272   356
RAM64    FFC0       132   133
RCNT     00FE       127   299   362   401   440   453   454   455   471
RDEXIT   73B2       740   738
RDHSR    733D       659   603   661
RDOA     73A4       733   334   354   412   424   427
RDOA2    73AB       737   734
RDOB     73B3       746   285   394   403   406   733   737
RDOB1    73C6       758   752
RDOB2    73D4       766   759
RDOB3    73E0       773   762
RDOB4    73E6       776   764
RDOC     72E9       604   236   384   388   421   429   750   760   772
RDT      72E9       601   604
RDT1     72FC       607   609
RDT2     72FC       614   627
RDT4     730E       625   623
RDT5     7319       632   587
RESET    7006       144   794
R0       7022       164   166
R1       700D       149   152
R2       7028       168   172
R3       702B       169   177   179
R4       7034       174   170
R5       7044       183   188
```

```
SYMBOL   VALUE   LINE DEFINED       CROSS-REFERENCES

R6        704B         187    185
SAVX      00FD         124    242   244   481   683
SETR      70FB         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
S0        7097         238    225
S1        7099         239    256
S2        70B7         255    240
TMPC      00FE         125    127   201   220   337   343   422   430   492   504
TMPC2     00FF         126    128   494   499
TMP0      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       723D         482    515
WHC       71E2         435    477
WH1       72C7         455    449   452
WH2       721F         465    472
WC        71C2         421    322
WRAP      00E4         105    232   435   482   727
WROA      729A         539    359   488
WROA1     72A8         546    540   542   544
WROA4     729E         541    474
WROA6     72A2         543
WROB      72B1         555    341   457   460   463   469   549
WROC      72C6         578    235   259   445   503   507
WRPC      72A6         545    330
WRT       72C6         576    570   578   699
WRTWO     72C0         568    223   530
WRT1      72CE         582    586
XR        00FA         121    206   377
YR        00FB         122    207   378
ZTMP      70D9         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         0
     DEC         6
     DEX         8
     DEY         1
     EOR         4
     INC         7
     INX         0
     INY         2
     JMP         9
     JSR        89
     LDA        65
     LDX        24
     LDY         4
     LSR        13
     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
```