Terminal Interface Monitor (TIM)
for the 6500 Microprocessor Family

Oliver Holt
Old Nashua Road
Amherst, NH 03031

TIM is a unique monitor program for the 6500 microprocessor family. TIM is the forerunner to KIM and is still used today in many configurations — ready made and homebrew. TIM is supplied by MOS Technology on an MCS6530 multi-function chip. This chip contains ROM, RAM, an interval timer, and I/O. Using this chip, MOS Technology was able to squeeze the complete monitor function into a single IC. The 1K of ROM in the 6530 contains the monitor program; the 64 bytes of RAM are used for storage and vector interrupt addresses; the timer is used for timing the serial I/O; the 13 I/O lines are used to communicate with a serial I/O device and a parallel device. The TIM part number is MCS6530-004.

TIM has a couple of unique features not incorporated in most monitors. The first feature is the ability to reconfigure the TIM memory locations during resets. During reset all I/O lines on the 6530 are set up as inputs and look like high signals to external devices. One of these I/O lines is used with address line A15 to make A15 a "don't care" condition. 6500 type microprocessors fetch the reset vector address from FFFC and FFFD. Because A15 is a "don't care", the vector address is fetched from 7FFD instead of FFFC and FFFD. Locations 7FFC and 7FFD contain the TIM entry point for a reset condition.

Figure 1 is a block diagram of a minimum TIM-based system including the circuitry required to accomplish the reset operation. The I/O line used is PB4. This signal is inverted and NANDed with A15. During reset PB4 is high making $\overline{PB4}$ low. A low input to the NAND gate causes a high output, always enabling CS1 on the 6530. When the I/O ports are initialized in the reset service routine, PB4 goes low making $\overline{PB4}$ a high. Now the output of the NAND gate is $\overline{A15}$ and CS1 is only high when A15 is low. CS1 along with the other chip selects and the address lines give the 6530 a set of unique addresses below 8000 but the software is set up for the address space between 7000 and 73FF.

The other unique feature of the TIM is that the terminal interface speed is adaptive. After the system is reset, the user types a carriage return. TIM measures the terminal speed using the data stream generated by the carriage return signal. This speed information is stored and used as the terminal speed for all following communication with the external device until the next time the system is reset.

After the reset and carriage return, TIM responds with an "*" and prints the contents of the registers, followed by an automatic carriage return and a ".". The period indicates that TIM is now ready to accept user commands. TIM commands allow displaying registers, executing programs, examining and altering memory, reading hexadecimal data from either a high speed reader or a TTY and writing either hexadecimal or BNPF data to a TTY. [BNPF is a tape format used by some of the older PROM programmers.].

Using the BRK instruction the user can set up breakpoints to monitor the execution of a program. The user inserts a BRK instruction [00] where the breakpoints are required. Upon execution of a BRK instruction TIM is entered and the registers are printed. The vector address for a BRK instruction is stored in RAM at FFFE and FFFF. The user may alter these locations and write his own routine for handling debug operations.

All TIM operations are performed in hex unless a BNPF tape is required. The memory is displayed in hex in groups of eight memory locations as shown:

```
.M 0000 00 01 02 03 04 05 06 07
```

command address data

TIM will respond with a period "." after each command is completed. If a user wants to modify data, he first opens memory with the "M" command and then types a colon ":" as follows: [Underlined data is what the user types.]

```
.M 0000 00 01 02 03 04 05 06 07
.: 0000 00 01 25 03 99 (carriage return)
```

The carriage return terminates the operation. The 6500 registers may be examined:

```
.R 7052 31 27 F0 01 FF
   PC   P  A  X  Y  SP
```

After the registers have been opened for examination, they may be changed using the colon ":" as shown:

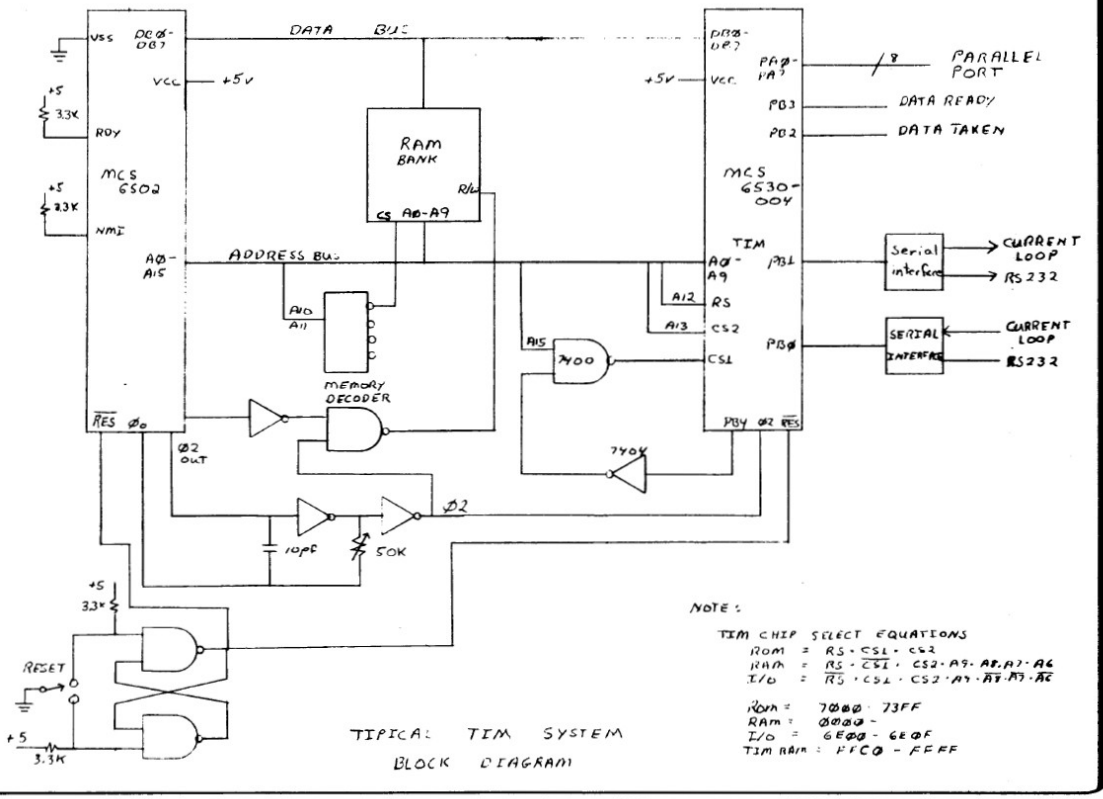
```
.R 7052 31 27 F0 01 FF
.: 0100 00 00 00 00 FF (carriage return)
```

The other commands for reading and punching tapes operate in a similar manner. TIM also has a switch which is set by the "H" command that specifies whether or not a high speed reader or TTY is the source of paper tape input.

TIM, like KIM, also has many useful subroutines that can be called by a users program. A set of useable subroutines to type characters, read characters, type a line feed and carriage return, type a space, and to type a byte in hex are completely documented in the TIM manual. There are other subroutines that can be used that are not documented and these include double precision addition, output a bit, input a bit, ASCII conversion, and input eight bits.

The TIM manual contains a complete software listing and a memory test program. The manual also includes example programs to aid the user in becoming familiar with the TIM commands. TIM is a very useful building block for anyone interested in building their own 6500 system. It has been used as the monitor for a number of systems available in kit and/or assembled versions. These include the CGRS Microtech 6000 system, the DATAC 1000, and others.

If you are interested in building your own homebrew system, the figure on the following page is a block diagram for a basic system. TIM is available from MOS Technology representative.



TIPICAL TIM SYSTEM BLOCK DIAGRAM

TIM MEETS THE S100 BUS

Gary L. Tater
7925 Nottingham Way
Ellicott City, MD 21043

Hardly a computer meeting goes by without a discussion of which bus structure is best. While the S100 bus may not be optimum for the 6502 microprocessor, its use does make purchasing RAM and ROM boards easy.

With this in mind, I purchased a 6502 CPU board for the S100 bus from CGRS Microtech. This CPU board is almost a complete system with its onboard 2K RAM and 4K ROM. But in order to use my CT-64 Southwest Technical Products video terminal with this CPU, I needed an S100 terminal interface monitor (TIM) board. While CGRS markets a very nice TIM board, I elected to build a bare bones S100 TIM board which is described in this article.

In addition to serving as a serial I/O port for a terminal, TIM contains an operating system for 6500 microcomputers. The OCT-NOV issue of MICRO (page 5) contains an article on the operation of the TIM program. In summary, TIM is a read-only memory and I/O device that is self adapting to terminal speeds between 10 - 30 cps. With TIM you can display and alter CPU and memory location using a keyboard and video display; you can read and write hex formatted data from a paper tape or a cassette interface such as the Southwest Technical Products AC-30; and you have an eight bit parallel I/O port where each bit of the eight can be programmed as either input or output.

As you can see from the schematic diagram (Figure 2), only the TIM chip (6530-004) and four integrated circuits are needed, excluding voltage regulators. For the perfectionist, buffering could be added to the address lines, data lines, and parallel output port, but two CGRS Microtech systems are now successfully using this TIM design. Integrated circuits U2 and U3 are used during resets to reconfigure TIM memory locations as described in the previously referenced TIM article. The MC 1488 and MC 1489 are Motorola devices which convert TTL levels to RS 232 levels and RS 232 levels to TTL respectively.

A memory map of this TIM design is provided in Figure 1. For proper operation of a 6502 microprocessor and this TIM board, you will need both page zero and page one memory. Page one is needed by the 6502 microprocessor for its software stack. Page zero memory is used in the TIM program to store the baud rate of your terminal (locations 00EA and 00EB).

To operate a TIM based system you need only momentarily ground pin 16 of TIM (pin #75 of the S100 bus) using a switch on your front panel. After you send a carriage return to the computer, you should see a TIM message such as:

```
7052 30 2E FF 01 FF
```

This message contains first the program counter (7052), processor status register (30), accumulator (2E), X register (FF), Y register (01), and stack pointer (FF). The actual values will vary from machine to machine.

```
7000 - 73FF  TIM ROM
FFC0 - FFFF  TIM RAM
6E00 - 6E0F  TIM I/O
6E02         Serial Port
```

Figure 1
TIM Board Memory Map

If you have a problem, first check all of your wiring and the +5, +12, and -12 voltages. Then insure that your reset switch is controlling pin 16 of TIM. Next, using an oscilloscope, check for a carriage return character at pin 25 of TIM and pin 24 for the TIM message. With a good signal at pin 25 but no answer at pin 24, the last two things to check are the address lines including pin 21, PB4, and finally, check your TIM chip in a working system. The two systems built using this design on prototype boards came up immediately. Hopefully, you will have the same good fortune.

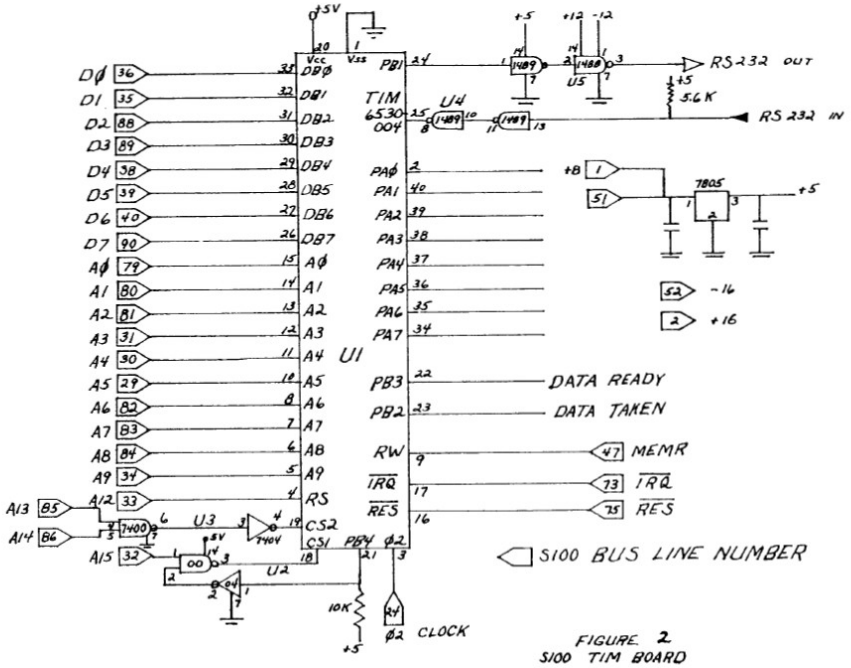


FIGURE 2
S100 TIM BOARD

TWO SHORT TIM PROGRAMS

Gary L. Tater
7925 Nottingham Way
Ellicott City, MD 21043

A Fast Talking TIM

If you have used both KIM and TIM with a terminal, you know that TIM has many nice features. For instance you can enter eight bytes at a time with TIM, and TIM has many more subroutines you can call in your programs than KIM does. However, KIM can adapt to terminal frequencies up to 2400 baud whereas TIM was designed to work from 100 to 300 baud. This article describes a program which allows you to communicate with TIM at 1200 baud or higher.

After a reset TIM automatically measures the speed of your terminal and deposits the bit times representative of the baud rate in two zero page locations, 00EA and 00EB. To increase the baud rate above 300 baud, the procedure is to place the correct values into EA and EB and change your terminal to that speed.

```
0100 20 A4 73  NEWVAL JSR   $73A4  READ TWO BYTES VIA TIM MONITOR
0103 A5 EE            LDA   $00EE  PUT EE INTO EB
0105 85 EB            STA   $00EB
0107 A5 EF            LDA   $00EF  PUT EF INTO EA
0109 85 EA            STA   $00EA
010B 00               BRK
010C 4C 00 01         JMP   NEWVAL TYPE G FOR NEW VALUES
```

Figure 1
Program to Change 00EA and 00EB.
Type Major Value 00EA First

By using the short program of Figure 1, I was able to find the correct values for 600 and 1200 baud operation (See Table 1) for my CT-64 and CGRS CPU board which has a 6502 operating with a one megahertz crystal. For each baud rate there is a range of values that is acceptable for EB. I have attempted to find the center of the range for my system. You will probably need to experiment to find the best numbers for your computer.

Baud Rate	00EA	00EB
1200	01	50
600	03	13
300	06	3C

Table 1
Zero page memory values for three baud rates.

Using this basic information I wrote the program of Figure 2. The programs begins at 157E and asks:

SPEED 300 600 1200?

At this point you should type 3, 6, or 1 and change your terminal to the correct rate. The program determines what you have entered and stores the correct values in EA and EB. By inspection of the program, you should be able to expand it to 2400 baud if you have a faster terminal. For a one megahertz system typical values are 00 in EA and 75 in EB for 2400 baud.

A TIM Operating System Menu

If you have written a collection of utility programs, assemblers, disassemblers and application programs, you will need a directory program with which you can easily call your desired program. The short program in Figure 3 uses the alphabet to call 26 programs. When the programs finish, they should return to the beginning of the directory program at location 0100.

You may choose to keep the program in ROM as I do. Only locations 0116 and 011B need be changed to do this provided you start the program at the beginning of a page.

The program prints a prompting "-" so that you'll know its in command and not TIM. If you type a nonalphabetic character, it will restart. After you type a letter, say a C for compare or M for move, the program finds the appropriate starting address stored between 0122 and 0155. After the starting address is stored in 00F6 and 00F7, the program calls the "GO" subroutine in TIM which causes your program to be executed.

```
          THIS PROGRAM IS RELOCATABLE AS LONG AS THE POINTER TO
          THE TEXT MESSAGE IS CHANGED IN LINE "PRINT"
157E D8         START  CLD          CLEAR DECIMAL MODE
157F A0 00             LDYIM $00    INITIALIZE INDEX
1581 B9 B3 15   PRINT  LDAY  TEXT   GET ASCII CHARACTERS
1584 F0 06             BEQ   PDONE  DONE IF NULL CHARACTER
1586 20 C6 72          JSR   $72C6  PRINT VIA TIM OUTPUT ROUTINE
1589 C8                INY          BUMP POINTER
158A D0 F5             BNE   PRINT  UNCONDITIONAL BRANCH TO PRINT NEXT
158C 20 E9 72   PDONE  JSR   $72E9  READ CHOICE VIA MONITOR
158F C9 31             CMPIM '1     ASCII 1 ?
1591 F0 1A             BEQ   HIGH   1200 BAUD
1593 C9 36             CMPIM '6
1595 F0 10             BEQ   MEDIUM
1597 C9 33             CMPIM '3
1599 D0 E3             BNE   START  NOT VALID CHARACTER

159B A2 3C      LOW    LDXIM $3C    GET VALUES FOR 300 BAUD
159D A9 06             LDAIM $06

159F 85 EA      FIXIT  STA   $00EA  SAVE FOR TIM TIMING ROUTINES
15A1 86 EB             STX   $00EB  SAVE SECOND VALUE
15A3 00                BRK          RETURN TO MONITOR
15A4 18                CLC          CLEAR CARRY
15A5 B0 D7             BCS   START  UNCONDITIONAL BRANCH

15A7 A2 13      MEDIUM LDXIM $13    GET VALUES FOR 600 BAUD
15A9 A9 03             LDAIM $03
15AB D0 F2             BNE   FIXIT  UNCONDITIONAL BRANCH TO FIXIT

15AD A2 50      HIGH   LDXIM $50    GET VALUES FOR 1200 BAUD
15AF A9 01             LDAIM $01
15B1 D0 EC             BNE   FIXIT  UNCONDITIONAL BRANCH TO FIXIT

15B3 53         TEXT   =     'S       "SPEED  300 600 1200 ?"
15B4 50                =     'P
15B5 45                =     'E
15B6 45                =     'E
15B7 44                =     'D
15B8 20                =     '
15B9 20                =     '
15BA 33                =     '3
15BB 30                =     '0
15BC 30                =     '0
15BD 20                =     '
15BE 36                =     '6
15BF 30                =     '0
15C0 30                =     '0
15C1 20                =     '
15C2 31                =     '1
15C3 32                =     '2
15C4 30                =     '0
15C5 30                =     '0
15C6 20                =     '
15C7 3F                =     '?
15C8 20                =     '
15C9 00                =     $00
```

Figure 2
6502 Program to Change Speed

```
0100 20 8A 72  START  JSR   $728A  CRLF VIA TIM MONITOR
0103 A9 2D            LDAIM '-     PRINT "-"
0105 20 C6 72         JSR   $72C6  VIA TIM MONITOR
0108 20 EE 72         JSR   $72EE  READ A CHARACTER VIA TIM
010B C9 5B            CMPIM $5B    TEST FOR GREATER THAN Z
010D 10 F1            BPL   START  BRANCH IF TOO LARGE
010F 38               SEC          SET TO CONVERT ASCII TO INDEX
0110 E9 41            SBCIM 'A     BY SUBTRACTING VALUE OF ASCII A
0112 30 EC            BMI   START  IF MINUS, THEN CHARACTER LESS THAN A
0114 0A               ASLA         MULTIPLY BY TWO FOR INDEX
0115 AA               TAX          PUT CONVERTED VALUE INTO INDEX
0116 BD 24 01         LDAX  LOWADR GET START ADDRESS LOW
0119 85 F6            STA   $00F6  SAVE FOR TIM
011B BD 25 01         LDAX  HGHADR GET START ADDRESS HIGH
011E 85 F7            STA   $00F7  SAVE START ADDRESS HIGH
0120 20 5C 71         JSR   $715C  GO TO SUBROUTINE VIA TIM
0123 00               BRK

0124 00        LOWADR =     $00    LOW ADDRESS FOR A, FILLED IN BY USER
0125 00        HGHADR =     $00    HIGH ADDRESS FOR A, FILLED IN BY USER
0126 00               =     $00    LOW ADDRESS FOR B
0127 00               =     $00    HIGH ADDRESS FOR B
               AND SO FORTH THROUGH
               LOW AND HIGH PAIR FOR Z
```

Figure 3
A TIM Directory Program