PTMON-69 MONITOR VERSION 1.6

The PTMON-69 Monitor EPROM allows the PT-69 computer to communicate with a terminal for various programming and debugging functions. Although the PTMON-69 was designed for use with the PT-69 computer system it may be used with any 6809 computer with the following characteristics:

1) 56K Memory from 0000 - DFFF
2) ACIA (MC6850) At address E004 for the control port
3) Space for the PTMON program at address F800-FFFF
4) Boot IDE Command - Requires EPROM from F000-F7FF or E800-EFFF
4) Diskette controller compatible with PT's FD-1 OR FD-2.

PTMON-69 is similar to SWTPC's Sbug-E monitor. V1.6 initializes a SWTPC compatible DAT. V1.6 may be used in a SWTPC computer as long as support for DMF2/3 FDC controller is not needed. V1.6 accepts lower as well as upper case letters for commands.

PTMON-69 MONITOR COMMANDS

When PTMON-69 completes power up (or is reset), it will display an identification banner and enter the command mode. When in the command mode, the user is prompted to enter a command. The prompt for command mode is a right arrow, '>'. Commands consist of one character followed by one, two, or three arguments. Commands requiring arguments may be aborted by typing a carriage return instead of a hex digit. A list of available commands follows: ^ - means press the control key and letter at the same time.

```
B [ADDRESS]              - Set Break Point
D [Begin] [End]          - Dump Memory
E [Address]              - Execute Program
F [BEGIN] [END] [DATA]   - Find Data
G                        - CONTINUE EXECUTION after Break Point
H                        - Renter FLEX at CD03
I [BEGIN] [END] [DATA]   - Initialize Memory
L                        - Load S1 Records
M [BEGIN]                - Memory examine/change
P [BEGIN][END]           - Make S records
Q [BEGIN][END]           - Test Memory
R                        - Register Dump
U                        - Boot Floppy
W                        - Boot IDE
X                        - Remove Break Point
^A                       - Change Accumulator A
^B                       - Change Accumulator B
^C                       - Change Condition Codes
^D                       - Change Direct Page Register
^U                       - Change User Stack Pointer
^X                       - Change X Index Register
^Y                       - Change Y Index Register
```

ALTER ACCUMULATOR-A **<CTRL>** "A"

The alter accumulator commands display the current contents of the specified register and then allow the user to alter its value by typing in a new value in hex. Should the user not wish to alter the contents of the register, he can enter a carriage return and PTMON-69 monitor will retain the old value. The alter accumulator-A command uses the following form:

```
>^A   A 55 2A  -   Change Accumulator 'A' From 55 TO 2A
>
```

ALTER ACCUMULATOR-B **<CTRL>** "B"

The alter accumulator-B command uses the following form:

```
>^B   B 55 2A       - Change Accumulator 'B' From 55 TO 2A
>
```

ALTER CONDITION CODES REGISTER **<CTRL>** "C"

The alter conditions code command uses the following form:

```
>^C 04 06       - CHANGE CONDITION CODES REGISTER FROM 04 TO 06
>
```

ALTER DIRECT PAGE REGISTER **<CTRL>** "D"

Alter direct page command uses the following form:

```
>^D 00 01 - CHANGE DIRECT PAGE REGISTER TO 01
>
```

ALTER USER STACK POINTER - **<CTRL>** "U"

The alter user stack pointer uses the following form:

```
>^U C000 B000 - CHANGE USER STACK POINTER TO $B000
>
```

ALTER 'X' INDEX REGISTER - <CTRL> "X"

The alter 'X' register command uses the following form:

```
>^X 1456 4389  - CHANGE 'X' REGISTER TO $4389
>
```

ALTER 'Y' INDEX REGISTER - < CTRL> "Y"

The alter 'Y' register command uses the following form:

```
>^Y 1549 2389  - CHANGE 'Y' REGISTER TO $2389
```

SET BREAKPOINT - >B [ADDRESS)'

The breakpoint command causes the PTMON-69 monitor to save the byte at the specified address and replace it with a software interrupt instruction. When a software interrupt instruction is executed a check is made to see if there is a match with the saved address. If the address matches the SWI instruction is replaced with the saved data byte.

DUMP MEMORY - '>D [BEGIN] [END]'

The dump memory command displays the contents of memory between the first address and the second address entered. The memory is displayed in both hexadecimal and ASCII, and always begins on an even 16 byte boundary. A sample display is shown below:

```
>D 0100 010F
0100 41 42 43 44 45 46 27 20 31 32 33 34 35 36 37 38 ABCDEF12345678
```

The dump command may be terminated in the middle of a dump by typing a carriage return.

EXECUTE PROGRAM COMMAND - '>E [BEGIN]'

This command allows the user to start execution of a program at the specified address. This command is convenient when the user wishes to restart a program that was halted by a breakpoint at some other address than the address of the breakpoint.

FIND DATA COMMAND - '>F [BEGIN] [END] [DATA]'

The find data command allows the user to search memory from the begin address to the end address for a two byte data pattern. All addresses that contain the pattern will be printed. A sample print out is shown below:

```
>F 0100 01FF E1AC
0145
0167
019F
>
```

CONTINUE EXECUTION COMMAND - '>G'

The go command causes PTMON-69 monitor to resume processing of an interrupted program. The interrupt may have been caused by either a breakpoint command or a software interrupt. If the monitor was entered by a breakpoint command the go command starts program execution at the address of the breakpoint. If the monitor was entered by a software interrupt the user must either change the SWI instruction to something else or use the 'E' command to start execution at the address following the SWI instruction.

JUMP TO SUBROUTINE - '>J [BEGIN]'

```
>J 0100    - START EXECUTION OF PROGRAM AT $0100
>
           - RETURN TO MONITOR AFTER COMPLETION OF SUBROUTINE
```

The jump to subroutine routine allows the user to execute a subroutine at the specified address and return to the monitor after execution of an 'RTS' instruction.

INITIALIZE MEMORY COMMAND - 'I [BEGIN] [END] [DATA]'

```
>I 0100 01FF 00          - CLEARS MEMORY BEWTEEN $0100 AND $O1FF
>
```

The initialize memory command allows the user to initialize memory from the beginning address to the ending address with the data byte entered. Note: care should be used not to overwrite the stack memory or it will be necessary to reset the computer.

LOAD TAPE COMMAND - '>L'

The load tape command causes the PTMON-69 monitor to load a Motorola 'S1' formatted tape into memory. At the beginning of the read the monitor sends a reader on control code ($11) to the terminal device. The loading process continues until either an 'S9' record or a bad check sum is detected.

MEMORY EXAMINE/CHANGE - '>M [BEGIN]'

The memory examine/change command allows the user to examine and change the contents of memory beginning with the specified address. Typing a hex value changes the contents of memory to the new data. A carriage return returns to the monitor. A '/' backs up to the last memory location, and any other character selects the next memory location. A sample of the examine/change command follows:

```
>M 0100
0100 44 55      - CHANGE 44 TO 55
0101 22 .       - SELECTS NEXT LOCATION
0102 33 /       - BACKS UP ONE LOCATION
0101 22 <CR>    - RETURNS TO THE MONITOR
```

Note: if the monitor is unable to write to some location for any reason(ROM or bad RAM) an 'X' is printed and the bell is beeped.

PUNCH TAPE COMMAND - '>P [BEGIN][END]'

The punch tape command causes PTMON-69 to punch or print a tape in the Motorola 'S1' format from the beginning and ending address specified by the user. The tape is punched in up to 32 byte blocks of data consisting of an 'S1' header, a byte count, an address, the data block followed by a checksum.

TEST MEMORY COMMAND - '>Q[BEGIN] [END]'

The memory test command allows the user to test memory from the beginning to the ending address as specified by the user. For each successful pass through memory a '+' is printed. If any bad memory is found, the address is printed followed by the number of passes through memory followed by a byte containing the bits that were in error. When testing memory care should be used not to overwrite the PTMON-69 stack RAM. PTMON-69 initially loads its stack pointer at $C6FF, so the user should not test memory in the 6C80-6CFF region. Memory from DFC0-DFFF is also used by PTMON-69 and cannot be tested. The user may abort the memory test by typing a carriage return if desired.

REGISTER DUMP - '>R'

The register dump command causes the PTMON-69 to print the contents of the 6809 registers. The command is executed automatically when a breakpoint instruction is executed.

```
>R

 PC   SP   US    X    Y   A  B DP CC
0000 C073 0000 0000 0000 00 00 00 D0
```

BOOT FLOPPY DISK - '>U'

The boot floppy disk command selects drive zero, issues a restore command to the controller, and attempts to read sector one on track zero into memory at address $C000. If the read is successful, control is transferred to the program loaded at address $C000. If there is no diskette in the drive or there is no drive on the system, the boot command will hang until the reset button is pushed. This command is for an FD2 controller installed in slot 1.

BOOT IDE DRIVE - '>W'

The boot IDE command will boot an IDE type drive connected to an SS30-IDE, PT69-5A or to WD1002/IDE adapter for the PT69-5. The boot reads the link information from sector 1 and loads the file pointed to by the link address. The IDE drive can be any type of device that can attach to an IDE port and supports LBA mode. It should be configured as master. The "W" command will work if WDISK drivers are at E800 or F000.

REMOVE BREAKPOINT COMMAND - '>X'

The 'X' command causes PTMON-69 to remove the breakpoint that was set using the 'B' command. Note: If program execution reached the breakpoint, the breakpoint automatically removes itself.

HOT RESTART TO FLEX - '>H'

This command jumps to address $CD03 which is the warm start address for FLEX, STAR-DOS or REX. No error checking is performed to see if the operating system is present.

ADVANCED PROGRAMMING INFORMATION

PTMON-69 occupies the uppermost 2K of memory space in the system and uses memory from DFC0 to DFFF for internal use. The stack pointer is loaded at address $C6FF which is the top of the user utility space of FLEX. The stack was loaded at this address because Flex version 2:8.3 interferes with the stack if it is loaded at $DFBF. A set of routine addresses is provided at address F800 for use by user programs. These routines are called by the indirect jump to subroutine instruction. A list of significant monitor addresses follows:

```
F800 MONITOR   RE-ENTER MONITOR, COLD START
F802 NEXTCMD   RE-ENTER MONITOR AND PROMPT FOR COMMAND
F804 INCH      GET INPUT CHARACTER FROM TERMINAL
F806 INCHE     GET INPUT CHARACTER FROM TERMINAL AND ECHO
F808 INCHEK    CHECK FOR INPUT CHARACTER
F80A OUTCH     OUTPUT CHARACTER TO TERMINAL
F80C PDATA     PRINT DATA STRING ROUTINE
F80E PCRLF     PRINT A CARRIAGE RETURN AND LINE FEED
F810 PSTRNG    CALL PCRLF, THEN CALL PDATA
F812 RTS       NULL - INCLUDED FOR COMPATIBILITY WITH S-BUG-E
F814 OUT2HS    PRINT TWO HEX CHARACTERS
F816 OUT4HS    PRINT FOUR HEX CHARACTERS
F818 IN2HEX    INPUT TWO HEX CHARACTERS
F81A IN4HEX    INPUT FOUR HEX CHARACTERS
F81C OUT4HEX   PRINT FOUR HEX CHARACTER IN 'X'
DFC2 SWI3-V    SOFTWARE INTERRUPT III VECTOR
DFC4 SWI2-V    SOFTWARE INTERRUPT II VECTOR
DFC6 FIRQ-V    FAST MASKABLE INTERRUPT VECTOR
DFC8 IRQ-V     SLOW MASKABLE INTERRUPT VECTOR
DFCA SWI-V     SOFTWARE INTERRUPT VECTOR
```

CALLING USER ROUTINES

The following is an example of how to call a user callable ROM routine. The user should call the routines through these vectors rather than the internal address of the routine because future releases of the monitor may have different internal addresses. The following is an example of how to print a carriage return and line feed with the PTMON-69 monitor routines:

```
PCRLF     EQU  $F80E     ASSIGN VECTOR TO LABEL
          JSR[PCRLF]     CALL PRINT CRLF ROUTINE
          LDA #4         CONTINUE WITH PROGRAM
```

MONITOR - $F800

Re-enter the PTMON-69 monitor. This routine resets the RAM vector locations, initializes the serial control port, and enters 'NEXTCMD'.

NEXTCMD - $F802

This command enters the PTMON-69 monitor at the point where it prompts for a command. This routine may be called as a subroutine if desired.

INCH - $F804

This routine gets a character from the terminal. The character is returned in the 'A' register with all other registers preserved. The returned character is eight bits long and is not echoed to the terminal.

INCHE - $F806

This routine gets a character from the terminal and echoes it. The character is returned in the 'a' register with all other registers preserved. The most significant bit is masked to a '0'.

INCHEK- $F808

This routine checks to see if a character is available from the terminal. All registers are preserved and the (z) bit is cleared if a character can be read.

OUTCH - $F80A

This routine outputs a character in the 'a' register to the terminal. Return is made with all registers preserved.

PDATA - $F80C

This routine outputs a string to the terminal. On entry 'X' must point to the string to be printed. The print operation is terminated when a byte with the value of '04' is encountered. Return is made with the 'X' register pointing at the byte containing the '04' and all other registers are preserved.

PCRLF - $F80E

This routine prints a carriage return and line feed to the terminal. Return is made with all registers preserved.

PSTRING - $F810

This routine calls 'PCRLF' and then calls 'PDATA'. Return is made with all registers preserved.

LRA - $F812

This vector is included to retain the vector table spacing for compatibility with the SWTPC monitor 'SBUG-E'. Since this routine is used by the DMF2/3 which is not supported, the vector points at an 'RTS' instruction.

OUT2HS - $F814

This routine prints two hex digits pointed at by the 'X' register. On return all registers are preserved except 'A'.

OUT4HS - $F816

This routine prints four hex digits followed by a space. On entry the 'X' register must point at the bytes to be printed. On exit all registers are preserved except 'A' and 'X'.

IN2HEX - $F818

This routine inputs two hex digits. On return the value is returned in the 'A' register. If an invalid digit is entered the (V) bit is set.

IN4HEX - $F81A

This routine inputs four hex digits into the 'X' register. If an invalid number is entered the (V) bit is set.

OUT4HEX - $F81C

This routine prints the number contained in the 'X' register. On exit all registers are preserved.