SBUG-E©

6809 ROM MONITOR

Version 1.5

USER'S GUIDE

IMPORTANT NOTE

Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Southwest Technical Products Corporation specifically disclaims any responsibility for any damages incurred or generated by such material. Southwest Technical Products Corporation reserves the right to revise this material at any time without obligation to notify any person of such revisions.

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY SAN ANTONIO, TEXAS 78216

6809 SBUG–E© Monitor ROM Version 1.5

The 6809 SBUG monitor ROM is provided to enable the computer to communicate with a terminal for the purpose of various programming and debugging functions. It has been designed to work in a SWTPC MP-09 processor board. It will not work in other processor boards. SBUG requires an MP-S interface installed in I/O port 1, at least 4K of RAM memory installed at any address at or below address DØØØ (52K) and an MP-B or MP-B2 motherboard patched to address I/O devices at 56K (EØØØ hex). Refer to the MP-09 instruction set for changes that must be made to several computer system boards for proper system operation. While SBUG provides some functional compatibility with 6800 DISKBUG®, SWTBUG®, and MIKBUG® monitors; in general, programs utilitzing the ROM functions of the older monitors will need to be changed. Note that while the 6800 is a predecessor to the 6809, 6800 object code will not execute properly on a 6809. A cross-assembler is available to aid in processor transition.

If the full 56K memory capability is not desired, it is possible to modify the SBUG–E© Version 1.5 code to allow the MP-09 processor board to run in an unmodified 6800 mainframe. With the modified code, the system memory capacity is limited to 40K bytes, and the system clock frequency is limited to 1 MHz. The SBUG monitor is supplied in a 16K masked ROM that is pinout compatible with the INTEL 5V-only 2716 EPROM. This compatibility provides the user the capability of duplicating the SBUG monitor in EPROM with custom modifications to meet his particular needs. A detailed explanation of how to make this change is provided later in this document.

SBUG–E© Monitor Commands

When the SBUG–E© monitor completes power up (or reset) processing, it will display its identification and enter the command mode. In this mode, it prompts the user for monitor commands with a right arrow character, ">". Commands consist of one character and may be followed by one or two arguments. All arguments are entered in hexadecimal, with the number of digits corresponding to the precision required of the argument. For example, a 16-bit argument requires four hex digits. Commands requiring arguments may be aborted by typing a carriage return character instead of a hex digit. A list of available commands follows:

Alter A-accumulator – control "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 register value, he can enter a carriage return at the terminal device and the SBUG–E© monitor will retain the old register value. The alter A-accumulator command takes the following form:

```
>^A     A=43 6F                    –change accumulator from 43 to 6F
>
```

Alter B-accumulator – control "B"

The alter B-accumulator command takes the following form:

```
>^B     B=7F 2A                    –change accumulator from 7F to 2A
>
```

Alter Condition Codes – control "C"

The alter condition code command first displays the current condition flags in binary, as described under Display Registers. The monitor will then allow the contents of the condition flags to be changed by typing a a new value in hex. Note that the "E" flag cannot be cleared. The alter condition code command takes the following form:

```
>^C     CC: E F – I – – – – D4     –set the zero condition
>^C     CC: E F – I – Z – – [cr]   –no change
>
```

Alter Direct Page Register – control "D"

The alter direct page register command takes the following format:

```
>^D     DP=ØØ Ø1                    –alter direct page register to $Ø1
>
```

Alter Program Counter – control "P"

The alter program counter command has the following format:

```
>^P     PC=4655 434B                –change program counter to $434B
>
```

Alter User Stack Pointer – control "U"

The alter user stack command has the following format:

```
>^U     US=5348 4954                –change user stack to $4954
>
```

Alter X Index Register – control "X"

The alter X-register command has the following format:

```
>^X     IX=4355 4E54                –change X-register to $4E54
>
```

Alter Y Index Register – control "Y"

The alter Y-register command has the following format:

```
>^Y     IY=434F 434B                –change Y-register to $434B
>
```

Set Breakpoint – "B" [addr] "

The breakpoint command causes the SBUG–E© monitor to save the byte at the specified address in its internal breakpoint table and replace it with a software interrupt instruction. Whenever the monitor is entered via a software interrupt instruction, the internal breakpoint table is searched for a matching program counter entry. If this entry is found, the saved byte replaces the software interrupt instruction, the entry is removed from the breakpoint table, and the program counter is adjusted to point back at the restored byte. The user can continue execution of the interrupted program via the Go monitor command.

Up to five breakpoints may be active at one time. If the monitor finds that there is no space in the breakpoint table, it will ouput a question mark after the address and the breakpoint will not be set. Breakpoints may be explicitly cleared via the "X" command.

Disk Bootstrap – "D"

The disk boot command invokes the SBUG–E© bootstrap program for the DMAF1 or DMAF2 8-inch floppy disks. The bootstrap program issues a restore to drive zero, then attempts to read sector one of track zero into memory at address $ØØØØ (logical). If the read indicates a CRC failure, the bootstrap program will continue retrying the read until the sector is read properly or the reset switch is pressed. When the sector is successfully loaded, control is passed to the Flex internal bootstrap program in the loaded sector. Note that if there is no disk in the drive, or if no disk interface is attached, the bootstrap program will hang until the reset button is pressed.

Since the ROM-based disk read function does not contain the extensive error detection and retry functions found in the disk operating system, it is possible to have a disk boot fail on an undetected soft read error. If this is the case, the disk drive will deselect without booting the operating system. If this occurs, the reset button should be pressed and the bootstrap repeated. This type of soft failure should be very rare–if it should occur with any regularity, it indicates a problem with either the disk media,disk drive or controller.

Examine Memory – "E [addr] –[addr] "

The examine memory command displays the contents of memory between the first address specification and the second address specification. The memory is displayed in both hexadecimal and ASCII, and is always displayed in multiples of 16 bytes. The format of the display is shown below:

```
>E  579B–57A2
 –  5790    ØØ ØØ 22 53 . . . 38 3Ø 39 2Ø    . . "SWTPC S/6809
 –  57AØ    2Ø 69 73 2Ø . . . 21 22 ØD ØA    i s  t he best !". .
>
```

Since the display of large blocks of memory can take considerable time at low baud rates, the display may be terminated at the end of a line by typing the carriage return at the terminal device.

Continue Execution – "G"

The Go command causes the SBUG–E© monitor to resume processing of an interrupted program. The interrupt may have been caused by a software interrupt or breakpoint. In the case of a reset interrupt, the Go command simply re-enters the monitor code and requests another command. If the monitor was entered via a software interrupt, execution resumes at the first instruction after the software interrupt. In the case of a breakpoint, which is a special case of software interrupt, the monitor removes the breakpoint and resets the program counter. Execution will resume at the instruction at the breakpoint address. If the user wishes to change the address at which execution will resume, the program counter can be changed prior to issuing the Go command.

Load MIKBUG® Format Tape – "L"

The tape load command causes the SBUG–E© monitor to load a MIKBUG® formatted paper tape (or cassette) into memory. At the beginning of the load, the monitor sends a reader-on control character ($11) to the terminal device. The tape is then read and any object record blocks loaded into memory. The loading process stops when either an end of tape indicator is read, or a bad checksum is detected on a load block. For information on block formats, see the Punch command.

Memory Alter – "M [addr] "

The memory alter command allows the user to examine and alter the contents of (logical) memory on a byte-by-byte basis. When the memory alter command is invoked, the specified address is made the "pointed" address. For each byte, the SBUG–E© monitor displays the "pointed" address, and contents of memory at that address, and then allows the user to alter that byte, and/or move forward or backward through memory. An example is provided as explanation:

```
>M  93F3
 –  93F3    46 .      – period moves forward in memory
 –  93F4    55 ,      – so does a comma
 –  93F5    43        – in fact, most characters move forward
 –  93F6    4B 48     – typing a hex value alters the byte
 –  93F7    49 ^      – typing an up-arrow goes backward
 –  93F6    48 [cr]   – typing carriage return ends the alter
>
```

If a new value is specified for a memory alter, and for some reason or other the monitor cannot write the new value into memory (write into ROM, protected memory, etc.), the alter function displays a question mark after the new value, like this – F833 3C 2A ? – and continues.

Punch MIKBUG® Tape – "P [addr] – [addr]"

The "P" command causes the SBUG–E© monitor to output the specified range of memory addresses as a MIKBUG® formatted ASCII tape. The tape is punched in 32-byte blocks starting at the first specified address and continuing through the second specified address, inclusive. Each block consists of an object record header, a block count field, a block address field, the data block itself, and a checksum. The checksum is calculated by taking the compliment of the sum of the byte count, the two bytes of address, and all of the data bytes output in the block. The punch routine outputs a punch-on control character ($12) at the beginning of the punch operation and outputs a punch-off control character ($14) at the end of the punch operation.

Memory Test – "Q [addr] –[addr]"

The "Q" command causes the SBUG–E© monitor to perform an address variable memory test starting at the first address specified and continuing through the second specified address, inclusive. The memory test will display a plus sign "+" after completing each pass through memory, and will perform 65,535 passes over the address range before terminating. Since this large number of passes can take a substantial period of time (just under 90 hours for 56K of memory), the test can be interrupted at the end of each pass by typing a carriage return at the terminal.

The memory test is performed in two parts. The first part computes a test pattern for each byte of memory by adding together the MSP and LSP of the memory address and the MSP and LSP of the pass number. This test pattern is stored into the memory and subsequently checked by the second part of the memory test. This test will detect address convergence errors, row or column decode errors, and bit write or read failures with high reliability.

If the memory test detects an error, the following message is displayed on the terminal device:

– ERROR AT XXXX=> YYYYY, PASS ZZZZ, BITS IN ERROR BBBBBBBB

The logical address of the failing memory is specified by XXXX while the actual physical address is specified as YYYYY. The pass number is shown (in hex) as ZZZZ. The bits-in-error field is displayed as a series of eight characters, each specifying a bit in the memory word. Bits that were read (or written) incorrectly are specified by their bit number, valid bits are shown as hyphens. If the memory problem is due to slow memories, the bits-in-error field may be shown as all hyphens.

For example, a user having 20K of memory in his 6809 system suspects that he has a problem with memory. He would then run the monitor memory diagnostic and receives the following indications:

```
>Q  C000–DF7F ++++++++++
 –  ERROR AT D13F  => 0713F, PASS 000C, BITS IN ERROR --5-----
 –  ERROR AT D17F  => 0717F, PASS 000C, BITS IN ERROR --5-----
 –  ERROR AT D1BF  => 071BF, PASS 000C, BITS IN ERROR --5-----
 –  ERROR AT D1FF  => 071FF, PASS 000C, BITS IN ERROR --5-----
```

This display shows that the memory test detected a memory malfunction at logical address D13F hex (in the system area) on pass 12. The physical address of this memory board is 0713F hex. The high order digit of the address is zero, since extended addressing is not being used in this system, and the particular bank of memory chips responds at 7000 hex. In the case of an MP-8M memory board, this is the upper row of memory chips. Since in each case, bit 5 is the bit found to be in error, the third memory chip from the left in the top row should be suspected.

Since the SBUG–E© monitor assigns its stack storage starting at $DFBF, it is suggested that the uppermost address specified in the memory test command be lower than $DF80. If the memory test overruns its own stack, it will appear to continue operation but the results of the test are no longer valid.

Display Registers – "R"

The display register command causes the SBUG–E© monitor to display the contents of all accessible processor registers. Most register values are displayed in hexadecimal, however, the condition code register is shown in an expanded binary format. If a flag bit in the condition code register is set the corresponding flag name is shown, otherwise the flag position is denoted with a hyphen. The format of the display is as follows:

– SP=hhhh US=hhhh DP=hh IX=hhhh IY=hhhh
– PC=hhhh A=hh B=hh CC: E F H I N Z V C

The condition flag letters are "E" – for "entire state" (always set), "F" for "FIRQ masked", "H" for "half-carry", "I" for "IRQ masked", "N" for "negative", "Z" for "zero" (or equal), "V" for "two's compliment overflow", and "C" for "carry".

Display Stack – "S"

The display stack command causes the SBUG–E© monitor to enter the memory examine function using the current stack pointer as a lower limit and using its default internal stack pointer ($DFCØ) as an upper limit. This results in the stack contents being displayed, if the stack space assigned by the SBUG–E© monitor is utilized. If the stack pointer has been changed via a load stack instruction, the operation of this command may be unpredictable. The format of the stack display is the same as that of memory examine.

Boot MF-68 Minifloppy – "U"

The "U" command invokes the SBUG–E© bootstrap program for the MF-68 5-inch floppy disk. The disk interface should be installed in processor interface slot 6 with appropriate modifications to MP-B or MP-B2 motherboards. The bootstrap program issues a restore command to drive zero, then attempts to read sector one on track zero into memory at $CØØØ (logical). If the sector is read successfully, control is transferred to the disk boot sector. Note that if there is no disk in the drive, or no interface attached, the disk boot command will hang until the reset switch is pressed.

Remove Outstanding Breakpoints – "X"

The "X" command causes the SBUG–E© monitor to remove any breakpoints that may have been set in memory. Note that a reset (or NMI) interrupt implicitly calls the "X" command, clearing any left-over breakpoints.

Information for Advanced Programmers

The SBUG–E© monitor resides in the uppermost 2K of memory space in the system (from F8ØØ hex to FFFF hex) and allocates RAM memory from DFCØ hex to DFFF for internal use as vectors, tables, and pointers. A set of routine addresses is provided at address F8ØØ hex for use by user programs. These routines may be called via Indirect Jump to Subroutine instruction. A group of vectors is provided in RAM in locations DFCØ to DFCF hex to provide for user control of interrupt processing. A list of significant monitor addresses follows:

Addr	Name	Description of Address
$F8ØØ	MONITOR	Re-enter the monitor and initialize memory.
$F8Ø2	NEXTCMD	Re-enter monitor and prompt for command.
$F8Ø4	INCH	Get input character from terminal device.
$F8Ø6	INCHE	Get input character from terminal and echo.
$F8Ø8	INCHEK	Check for input character.
$F8ØA	OUTCH	Output character to terminal device.
$F8ØC	PDATA	Mikbug® compatible print data string.
$F8ØE	PCRLF	Print carriage return, line feed, nulls.
$F81Ø	PSTRNG	Call PCRLF, then print data string.
SF812	LRA	Load real address of memory byte.
$D8CØ	USER–V	User Interrupt Vector
$D8C2	SW13–V	Software Interrupt III Vector
$D8C4	SW12–V	Software Interrupt II Vector.
$D8C6	FIRQ–V	Fast Maskable Interrupt Vector.
$D8C8	IRQ–V	Slow Maskable Interrupt Vector.
$DFCA	SWI–V	Software Interrupt Vector
$DFCC	SVC–VO	Supervisor Call Vector Origin Address.
$DFCE	SVC–VL	Supervisor Call Vector Limit Address.

Note that these vector entries are **addresses**, and jumping directly into the address vector is invalid and will produce unpredictable results.

User-callable ROM Routines

The following paragraphs provide entry parameterization and functional descriptions of the provided user-callable ROM routines in the SBUG–E© monitor. It is inadvisable to call directly into the ROM routines as opposed to using the entry vector since future releases of the SBUG ROM may have different internal addresses while the entry vector will remain valid. The proper method for calling a routine in SBUG is to use an indirect subroutine call, e.g.

```
INCHE   EQU     $F8Ø6           INPUT ROUTINE ADDRESS
        JSR     [INCHE]         CALL INPUT CHARACTER ROUTINE
        STA      ,X+            STORE INTO BUFFER, etc.
```

MONITOR – $F8ØØ

Re-enter the SBUG–E© monitor. Re-establish the RAM interrupt vectors and initialize the serial interface to the terminal device. Establish a dummy stack, compute memory size, and display the signon message, then enter NEXTCMD.

NEXTCMD – $F8Ø2

Enter the SBUG–E© monitor at the point where it prompts for a command. This entry point does not initialize the stack, which implies that the SBUG monitor can be called as a subroutine.

INCH – $F8Ø4

Get a character from the terminal device. The character is returned in the A register with all other registers preserved. The input character is 8-bits long, and is not echoed to the terminal device.

INCHE – $F806

Get a character from the terminal device and echo. The character is returned in the A register with all other registers preserved. The input is masked to 7-bits in length, and then echoed back to the terminal device.

INCHEK – $F808

Check for a character available from the terminal device. The serial interface is checked to see if it has become read ready. All registers are preserved, and the routine will return zero clear if a character can be read from the terminal device.

OUTCH – $F80A

Output a character to the terminal device. The character to be output is passed in the A register, and return is made with all registers preserved.

PDATA – $F80C

Output a MIKBUG® compatible string to the terminal device. The routine is entered with the IX register pointing at the first character of the string to be output. The string is terminated with a byte having the value 04 hex. Return is made with the A register containing the 04, the IX register pointing at the byte after the 04, and all other registers preserved.

PCRLF – $F80E

This routine outputs a carriage return, a line feed, and three null characters to the terminal device. The A register returns wih a 04, all other registers are preserved.

PSTRNG – $F810

This routine is entered with the IX register contining the address of a string to be output. PCRLF is called to obtain a new line, then PDATA to print the string. The A register contains a 04 on exit, the IX register points to the byte after the 04, and all other registers are preserved.

LRA – $F812

Load the 20-bit physical address of a memory byte into the A and X registers. This routine is entered with the logical address of a memory byte in the IX register. Exit is made with the high-order four bits of the 20-bit physical address in the A register, and the low-order 16-bits of the 20-bit physical address in the IX register. All other registers are preserved. This routine is required since the DMAF1 and DMAF2 disk controllers must present physical addresses on the system bus.

User-referable RAM Locations

The SBUG monitor reserves memory from DFC0 hex to DFFF for use as internal tables and addresses. Included in this area are the interrupt handler addresses for the maskable and software interrupts. These interrupts are available to the user by placing the address of an interrupt service routine in the appropriate vector address. Note that RESET and NMI are not vectored through RAM and hence are not available to the user. The interrupt vector addresses and explanations follow:

Software Interrupt Vector – $DFCA

This vector contains the address of the software interrupt service routine entered whenever an SWI instruction is executed. After power up, this vector points to SBUG breakpoint processing, however, if breakpoints are not being used, this vector may be altered by the user.

Maskable Interrupt Vector – $DFC8

This vector contains the address of the IRQ interrupt service routine. After power up processing, this vector points at a return from interrupt instruction.

Fast Maskable Interrupt Vector – $DFC6

This vector contains the address of the FIRQ interrupt service routine. After power up processing, this vector points at a return from interrupt instruction.

Software Interrupt II Vector – $DFC4

This vector contains the address of the secondary software interrupt service routine entered whenever an SWI2 instruction is executed. After power up, this vector points to a return from interrupt instruction. The secondary software interrupt is not used on a system level to make it available at a user level.

Software Interrupt III Vector – $DFC2

This vector contains the address of the tertiary software interrupt service routine entered whenever an SWI3 instruction is executed and there is no supervisor call vector specified (see below). After power up, this vector points to a return from interrupt instruction.

Supervisor Call Vector Origin – $DFCC, and
Supervisor Call Vector Limit – $DFCE

The supervisor call (SVC) vectors are used in conjunction with the SWI3 instruction to provide a second level decode for interrupt processing. When an SWI3 interrupt occurs, the SBUG monitor checks the SVC Vector Origin for an address of $FFFF. If this value is contained in the vector, SWI3 processing defaults to the SWI3 vector (see above), otherwise, the byte of data following the SWI3 instruction is fetched and the program counter adjusted to bypass this byte. The byte is multiplied by two, and used as an index into the SVC vector. The indexed address is compared to the address contained in the SVC Vector Limit, and if it is not greater, an indirect jump is made to the SVC routine. If the address is found to be out of bounds, the processing proceeds through the SWI3 vector.

When the SVC routine is entered, the US register contains the value of the stack pointer on entry to the SVC handler. The stack pointer should be restored to this value before returning from the SVC routine. The supervisor call service routine usually terminates its processing via a TFR U,S: RTI sequence.

Note that an invalid parameter (either vector or SVC number) passed to the SVC service routine will cause the SWI3 vector to be used. In this manner, a supervisor program can detect invalid parameterization. After power up processing, both the origin and limit vectors contain $FFFF.

Duplicating the SBUG–E© Monitor in EPROM

Since the SBUG–E© monitor ROM is pinout compatible with 2716 EPROM's, the MP-R PROM programmer can be used to duplicate the SBUG code in PROM. In addition, since the MP-09 board has on-board address translation, it is possible to modify the SBUG–E© code to recognize I/O devices addressed at 32K (8000 hex), and hence to run an MP-09 board in a system with an unmodified motherboard. This provides the user with the capability of easily changing from a 6800 system to a 6809 system and back, but costs the user an additional 16K of memory capability, and limts the 6809 system to running at 1MHz, and using the MF-68 minifloppy. If the user wants to use the DMAF1 or DMAF2 disk controller, the controller must be modified to respond at F000 hex (60K) in order to properly run DMAF software.

The SBUG–E© monitor ROM should be placed in the MP-R PROM programmer, and read into the program buffer using the MP-R software read command. The entire ROM must be read. The edit command can then be used to change the byte at FF79 (hex) from F1 (hex) to F7 (hex). This changes the way the SBUG monitor sets up the address translation table at power up time to recongize I/O at 8000 (hex). A blank 2716 is inserted into the PROM programmer and a new PROM burned, which will then replace the SBUG ROM in the MP-09 board.