Mikbug Operating System

One of the outstanding features of the SWTPC 6800 Microprocessor/System is its use of Motorola's Mikbug firmware package stored in the MCM6830L7 ROM. It is thru the software stored in this ROM that the user may use the control terminal to initiate various functions such-as load programs, execute programs, dump programs and print or change the contents of the CPU registers. Motorola's Engineering Note 100 describes in detail the operation of the Mikbug ROM, however, there are some additional things which should be noted before reading this note.

Also contained within the Mikbug ROM a program called Minibug and a section called test pattern. The Minibug program is an old and less elaborate version of what is now called Mikbug and it along with the test pattern have been disabled from ROM access. The ROM was designed so that either Mikbug or Minibug could be used and since the MIKBUG was by far superior, it was chosen for our system. Although Engineering Note 100 has instructions for the operation of Minibug, they should not be read in order to avoid confusion.

Another interesting note is that although Mikbug is used with an MC6810 128_{10} word Random Access Memory (RAM) for temporary storage, there are at least 54_{10} RAM locations that are unassigned located from address $A04A_{16}$ to A97F inclusive which may be used for user program Use. There are also 46_{10} locations from address A014 to A041 inclusive which have been reserved for the push-down stack. This means that a 46_{10} deep push down stack may be maintained in the MC6810 RAM or that if a short stack is used, the lower portion of these 46_{10} locations may be used for user program space. It is a good idea, however, not to use any locations between A037 and A049 inclusive.

When you start reading about the "Display Contents of MPU Registers Function" in Engineering Note 100 it should be pointed out that the system always assumes the push-down stack is located between address locations A043 and A049, with the program counter stored in locations A048 and A049. Whenever you enter a "G" for the Go to User's Program Function" the first thing the processor does is execute a "return from interrupt" instruction (RTI) which loads the data stored in address locations A043 to A049 into the processor's condition code register, accumulators, index register and program counter. If you forget to use the "Memory Examine and Change Function" to load the program counter locations (A048 and A049) with the starting address of your loaded program, or for that matter any of the processor registers which must be initialized to some value, your program of course will not run. At the completion of the "return from interrupt" instruction (RTI) the processor jumps to the starting address of your program. The stack pointer is set to A049 which means any interrupt, branch to subroutine, jump to subroutine or push instruction in your program will change the data stored at location A049 and depending upon the instruction, A¢48, A047, etc.. This means that if you abort the program with the RESET button, you will probably have to go back and reload pertinent data in locations A043 thru A049 before restarting the program using the "Go to User's Program Function" again. If all of this seems confusing reread this paragraph after reading Engineering Note 100 contained in this section of the notebook.

Another important note is that there are two subroutines within the Mikbug package which greatly aid the programmer in his control terminal input/ output routines. They are called INEEE, address ElAC, and OUTEEE, address E1D1 whose descriptions follow:

INEEE (E1AC) - The INEEE subroutine should be used to receive incoming the control terminal's keyboard via the control interface (serial), I/0 port 1.The subroutine loops within itself until a character is received, at which time it is deposited into the A accumulator in ASCII form. Bit 7, the parity bit of the received data is automatically zeroed out and is not checked in any way for accuracy before it is loaded into accumulator A. The B accumulator and register are used in the subroutine, how ever the original data in these locations is stored and then restored at the completion of the subroutine.

OUTEEE (E1D1) - The OUTEEE subroutine should be used to transmit characters out of the control interface (serial), I/0 port 1, the control terminal. The ASCII coded character to be transmitted must be loaded into the A accumulator before the subroutine is called. The entire eight bits are transmitted out as they are loaded into the accumulator. No partly bit determinations are made by the subroutine. The B accumulator and index register are used in the subroutine, however, the original data in these locations is stored and then restored at the completion of the subroutine.

The simplest way to get to these subroutines is to do a dump to subroutine extended (JSR_{asm}) or (BD_{16}) followed by the address of the subroutine.

Entry back into the Mikbug control program can be done automatically at the end of by inserting a dump to address E0E316 (JMP_{asm}) or ($7E_{16}$) followed by the address $E0E3_{16}$.

One final note is that the Motorola Note 100 does not cover the software interrupt function, which is a tool for debugging problems. It is used as follows:

SOFWARE INTERRUPT (BREAKPOINT) FUNCTION - This software interrupt function provides you with a method of entering breakpoints your program. Assume that you are debugging your program and wish to verify that your program has reached a particular program instruction. You can, by using the SWI (software interrupt) enter a breakpoint at this program instruction's address. To enter the breakpoint you load the instruction at the selected address SWI instruction. Now, when the SWTPC 6800 System executes this SWI instruction in the user's program, it returns program control to the MIKBUG software interrupt routine, prints the contents of the MPU registers, and proceeds to the MIKBUG control program. This software interrupt mode (SWI) which displays the contents of the MPU registers does not assume the stack is located between address A043 and A049 like the "Display contents of MPU Registers Function" of the MIKBUG control program. The register data printout will be accurate no matter where the stack is positioned in memory. The paragraphs discuss entering the breakpoint into and removing a breakpoint from a user's program.

ENTERING A SWI BREAKPOINT - Use the following procedures to enter a software interrupt breakpoint into the user's program. It is assumed prior to these procedures that the user's program has been loaded into memory, the SWTPC 6800 System is performing its MIKBUG control program, and the last character printed by the data terminal is an asterisk.

a. Enter a M after the asterisk to open a memory location. The terminal will insert a space after the M.

b. Enter in 4-character hexadecimal the memory address you have selected to enter a breakpoint. The terminal will print on the next line this memory address and its contents. Record the memory contents.

c. Enter a space code and the hexadecimal characters 3F (SWI instruction). The SWI instruction is now stored in memory and the terminal prints the next address and its contents on the next line.

d. Enter a space code followed by carriage return control character. The SWTPC 6800 System returns to the MIKBUG control program and the terminal prints an asterisk on the next line.

e. Run the user's program in accordance with the Go To User's Program Function (described in Engineering Note 100). When the SWTPC 6800 System executes the SWI instruction, it returns program control to the MIKBUG program, prints the contents of the MPU registers, and advances to the MIKBUG control program.

REMOVING A SWI BREAKPOINT AND RESTORING THE PROGRAM - Use the following procedures to remove a software interrupt breakpoint from the user's program. It is assumed at the start of these procedures that SWI instruction has been loaded to a known memory location, the SWTPC 6800 System is performing its MIKBUG control program, and the last character printed by the terminal is an asterisk.

a. Enter a M after the asterisk to open a memory location. The terminal will insert a space code after the M.

b. Enter in 4-character hexadecimal code the address whose SWI instruction is to be removed. The terminal will print on the next line this memory and address and 3F (SWI instruction).

c. Enter a space code and restore the original instruction removed from this address previously. The new contents are stored in memory and the terminal prints the following memory address and its contents on the next line.

d. Enter a space code followed by a carriage re urn control character. The SWTPC 6800 System returns to the MIKBUG control program and the terminal prints an asterisk on the next line.