ENGINEERING NOTE 100

MCM6830L7 MIKBUG/ MINIBUG ROM

Prepared by
Mike Wiles
Computer Systems

Andre Felix
Support Products Group

The MIKBUG/MINIBUG ROM is an MCM6830 ROM of the M8800 Family of parts. This ROM provides an asynchronous communications program, a loader prom, and a diagnostic program for use with the ME8800 Microprocessing Unit.

MOTOROLA
Semiconductor Products Inc.

MCM6830L7 MIKBUG / MINIBUG ROM

1.0 SYSTEMS OVERVIEW

The MIKBUG/MINIBUG ROM provides the user with three separate firmware programs to interface with a serial asynchronous (start-stop) data communications device. They are:

1) MIKBUG Rev. 9
2) MINIBUG Rev. 4
3) Test Pattern

The map of the programs is shown in Figure 1-1.

NOTE

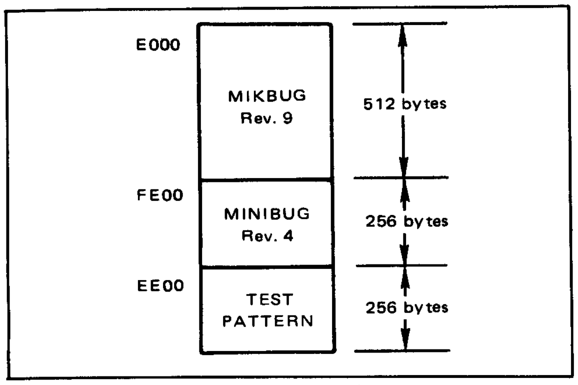


FIGURE 1-1. MIKBUG/MINIBUG ROM Memory Map

All enables for the ROM are active high.

2.0 FEATURES

The more important features of these programs are:

MIKBUG Rev. 9

A. Memory Loader
B. Print Registers of Target Program
C. Print/Punch Dump
D. Memory Change
E. Go to Target Program
F. Operates with PIA for the Parallel-to-Serial Interface
G. Restart/NMI/SWI Interrupt Vectors

MINIBUG Rev. 4

A. Memory Loader
B. Memory Change
C. Print Registers of Target Program
D. Go to Target Program
E. Assumes a UART for the Parallel-to-Serial Interface

3.0 HARDWARE CONFIGURATION

3.1 MIKBUG Hardware

The MIKBUG/MINIBUG ROM is intended for use with the MC6800 Microprocessing Unit in an M6800 Microcomputer system. This ROM, using the MIKBUG Firmware, should be connected into the system as illustrated in Figure 3-1. As shown, all of the enable inputs are high levels and the address line A9 on pin 15 is grounded. The MIKBUG Firmware in this ROM uses addresses E000 through EIFF. The ROM should be connected into a system so that its two top MIKBUG Firmware addresses also will respond to addresses FFFEand FFFF. This is required for the system to restart properly. There should not be any devices in the system at a higher address Man this ROM's addresses. Figure 3-2 depicts a memory map for a system using the MIKBUG Firmware and Figure 3-3 depicts this system's block diagram.

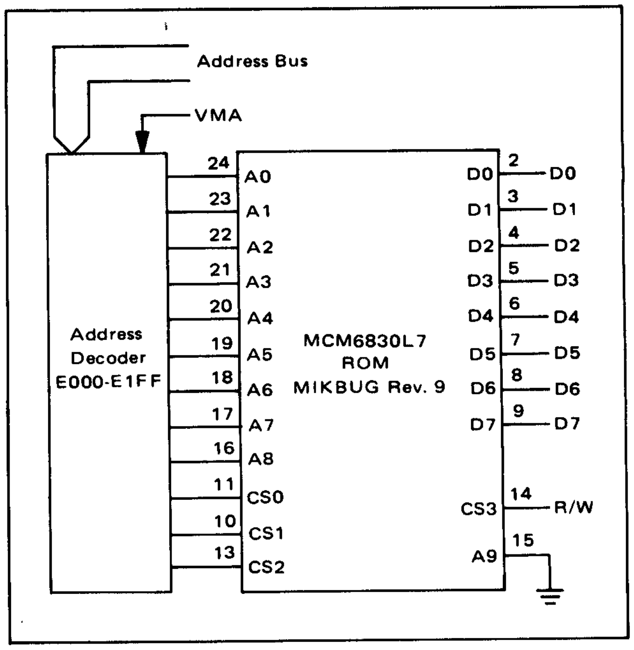


FIGURE 3-1. MCM6830L7 MIKBUG ROM Schematic

The MIKBUG Firmware operates with an MC6820 Peripheral Interface Adapter (PIA) as shown in Figure 3-4. The MC 14536 device is used as the interface timer. This timer's interval is set by adjusting the 5 0 k ohm resistor and monitoring the output signal on pin 13 of the MC14536 device. The zero level of the timing pulse should be 9.1 ms for 10 characters

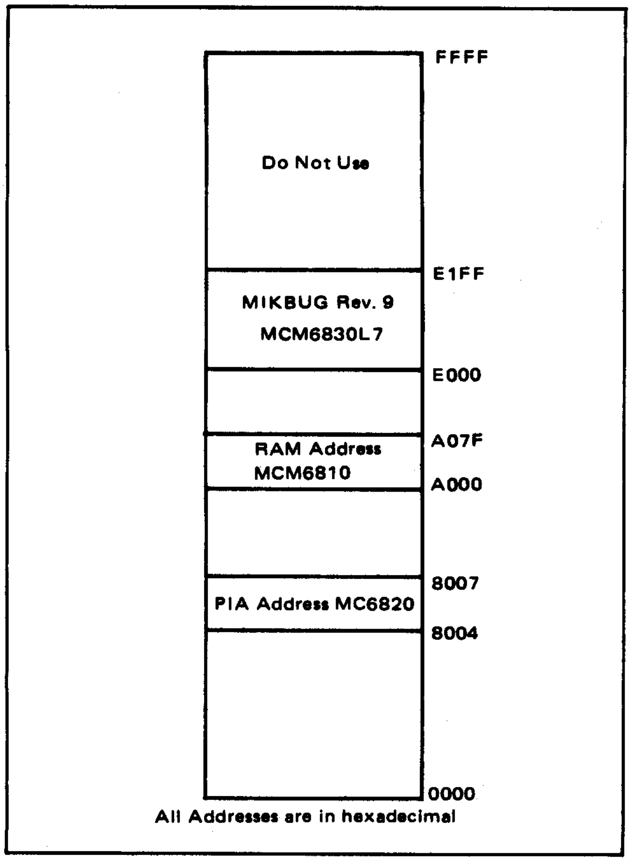


FIGURE 3-2. MIKBUG Rev. 9 Memory Map

per second (CPS) operation and 3.3 ms for 30 CPS operation. Also, pin 16 (PB6) of the MC6820 PIA should be connected to +5 volts for 10 CPS operation and ground for 30 CPS operation.

The MC 1488 and MC 1489A devices provide the system with RS-232C interface capability. If the system is to interface only with an RS-232C terminal, no other interface circuitry is required; however, a jumper should be strapped between E3 and E4. The 4N33 optical isolators and associated circuitry are required to interface with a 20 mA current loop TTY. A jumper should be connected between E1 and E2 for TTY operation.

The MIKBUG Firmware also requires random access memory for a stack and temporary memory storage. The MCM6810 RAM used for this memory should be configured for the base memory address at A000 hexadecimal.

A reset switch is required in the system to provide for restarting the MC6800 MPU and for resetting the MC6820 PIA. The function may be provided by a pushbutton switch and a cross-coupled latch flip-flop.

3.2 MINIBUG Hardware, Rev. 4

The MIKBUG/MINIBUG ROM is intended for use with the MC6800 Microprocessing Unit in an M6800 Microcomputer system. This system, using MINIBUG Firmware Rev. 4, should be set up with the starting ROM address at FE00 hexadecimal. The restart address generator (Figure 3-5)

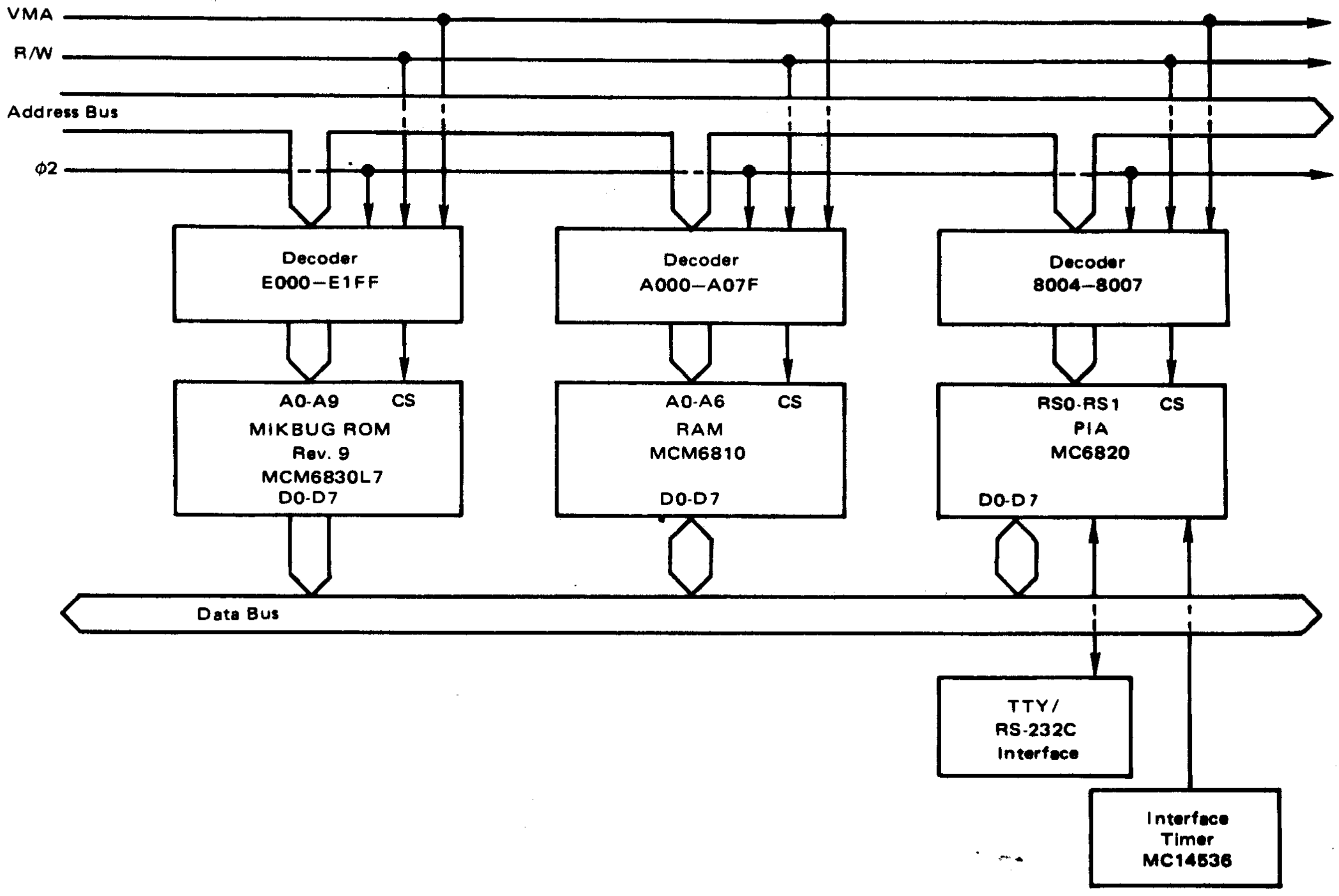


FIGURE 3-3. MIKBUG ROM Rev. 9 System Block Diagram

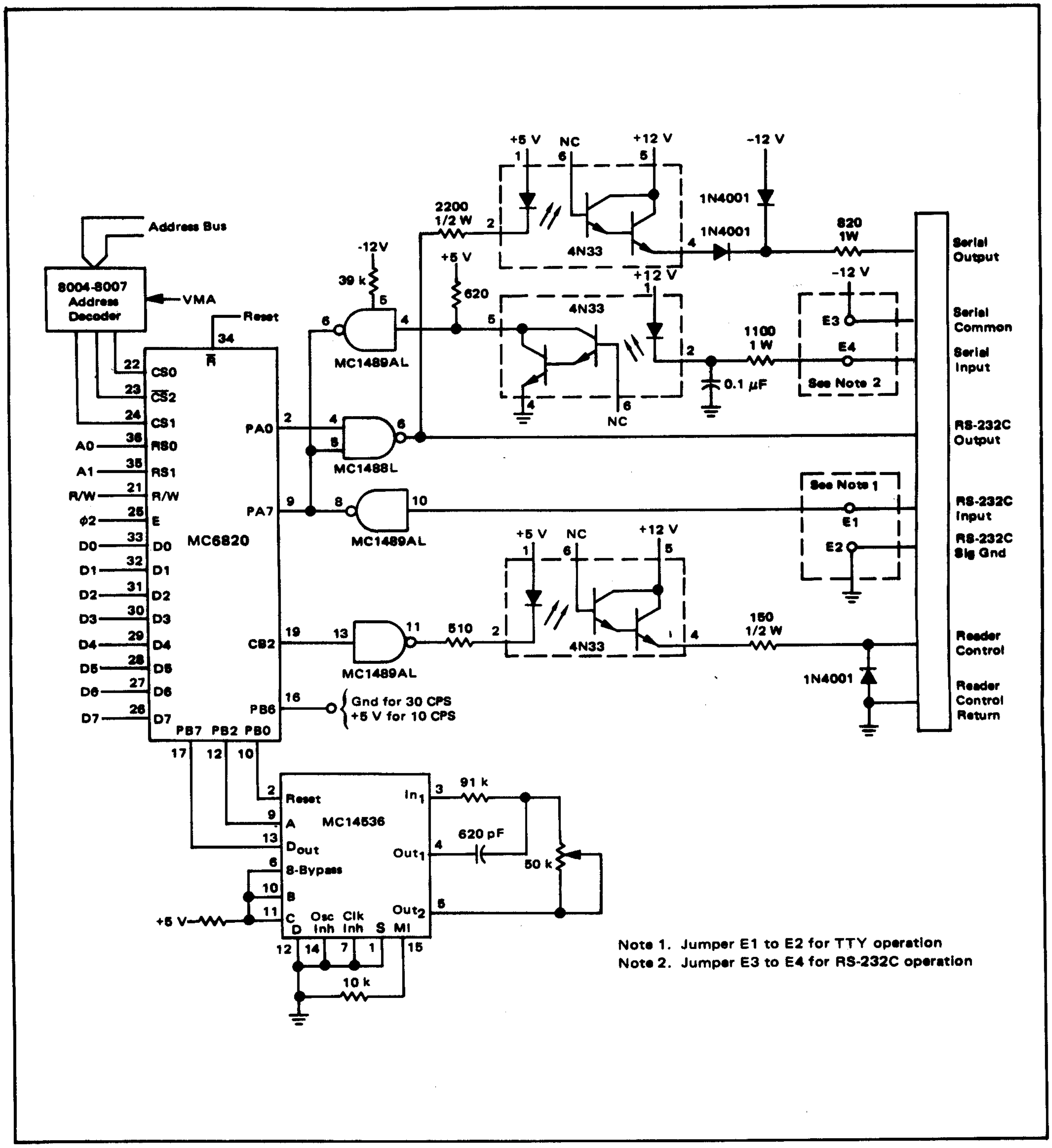


FIGURE 3-4. TTY/RS-232C Interface Used with MIKBUG ROM

must be configured to respond with address FED6 each time the MPU requests the restart address. As shown, the system also requires an MCM68 10 RAM for temporary storage. This RAM shall be configured for a FF00 base memory address. Figure 3-6 depicts a memory map for a system using the MINIBUG Rev. 4 Firmware.

The MINIBUG ROM Rev. 4 also uses a parallel-to-serial data converter to interface with an external terminal. The converter's status register must be located at address FCF4 and the data register at address FCF5. The least significant bit of the status register is used to indicate that the converter has received a character and the second bit indicates that the converter is ready for the next character to be transmitted.

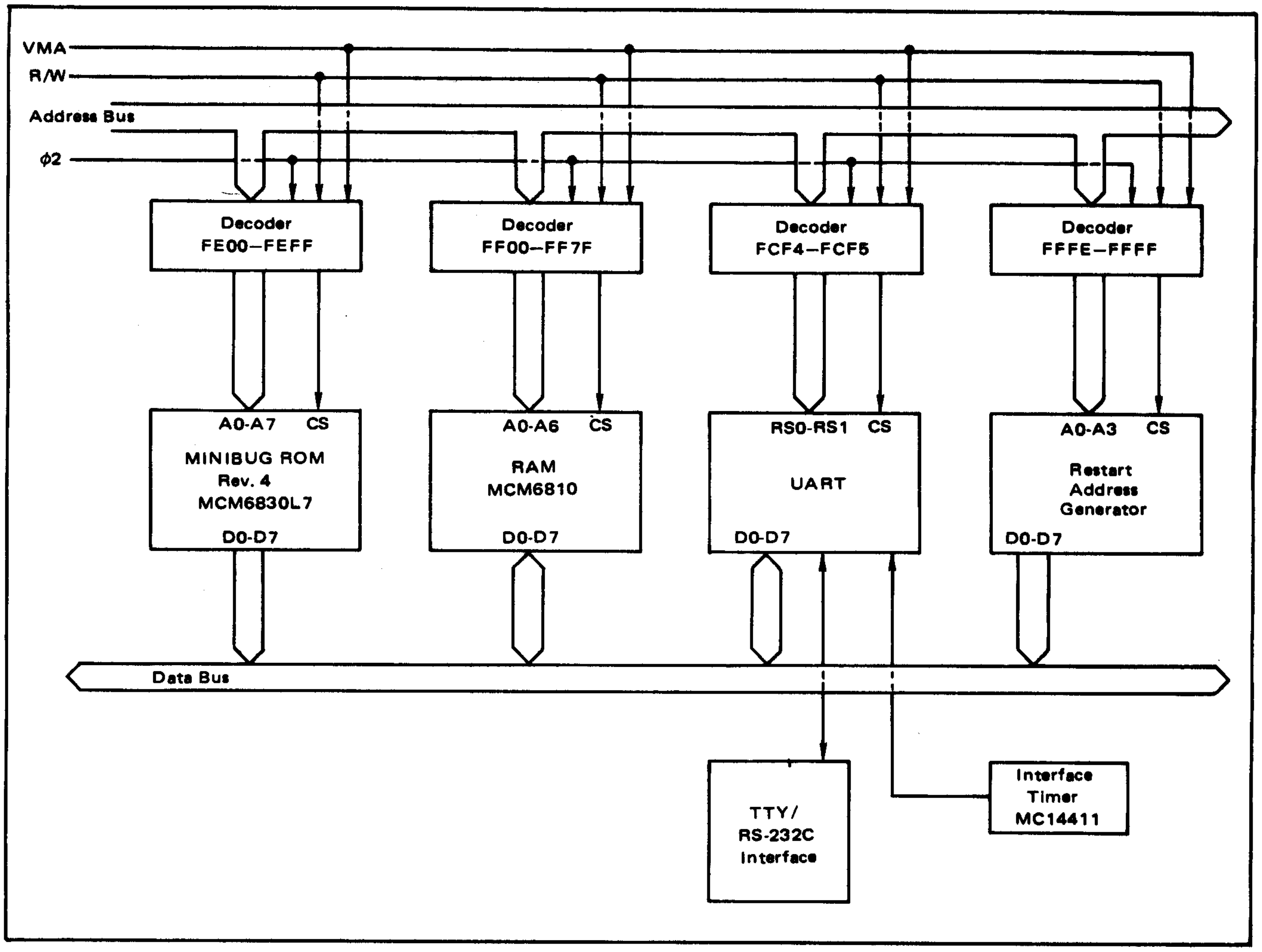


FIGURE 3-5. MINIBUG Rev. 4 System Block Diagram

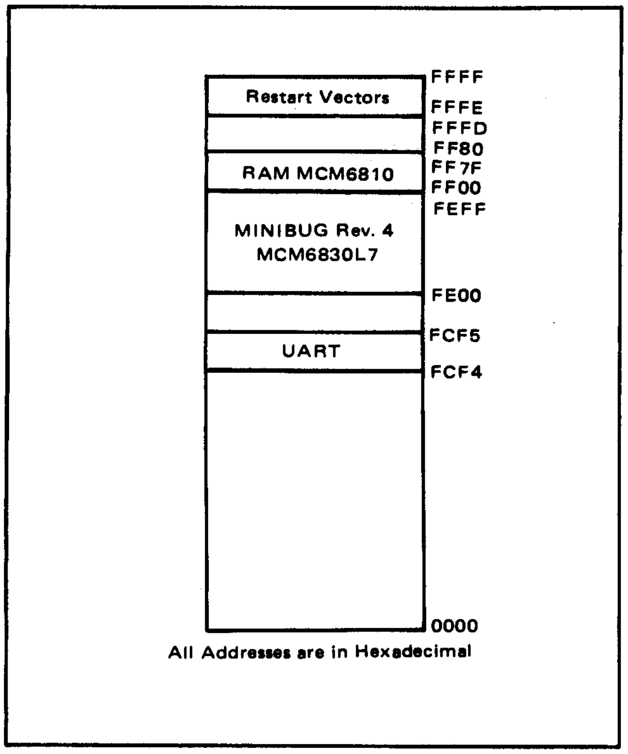


FIGURE 3-6. MINIBUG Rev. 4 Memory Map

4.0 SOFTWARE OPERATION

4.1 MIKBUG Operation

The MIKBUG Firmware may be used to debug and evaluate a user's program. The MIKBUG Firmware enables the user to perform the following functions:

Memory Loader Function
Memory Examine and Change Function
Print/Punch Memory Function
Display Contents of MPU Registers Function
Go to User's Program Function
Interrupt Request Function
Non Maskable Interrupt Function

The operating procedures for each of these routines as well as the Reset Function are discussed in the following paragraphs. The MIKBUG Firmware is inhibited from performing the user's program except in the Go to User's Program Function and the interrupt functions.

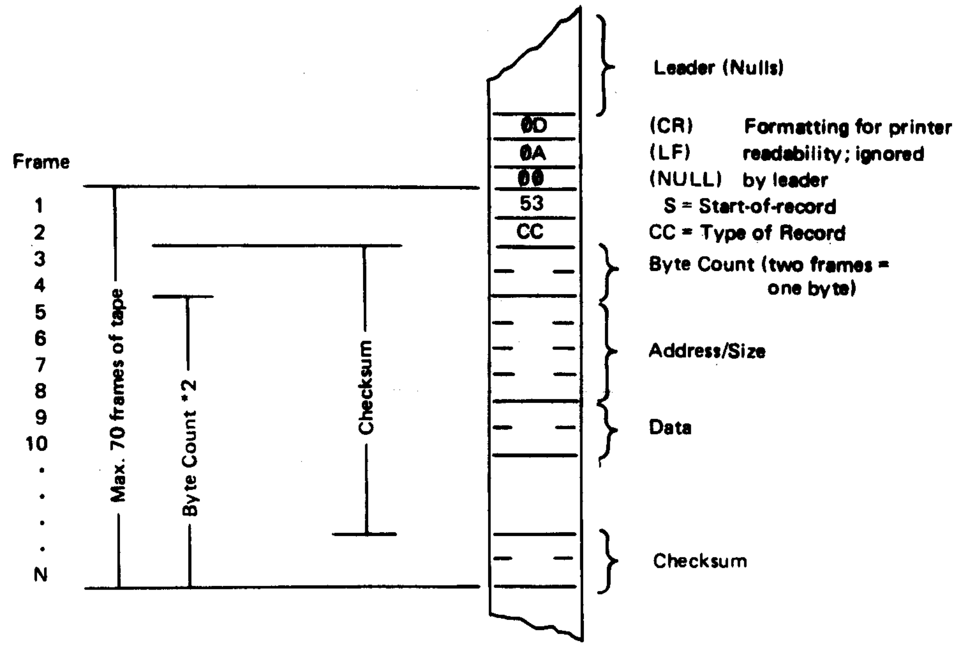
4.1.1 RESET Function

Perform the RESET Function when power is first applied and any time the MIKBUG Firmware loses program control.

Press the RESET pushbutton switch. The MIKBUG Firmware should gain program control and the terminal should respond with a carriage return, a line feed and an asterisk. The MIKBUG control program is ready for an input.

4.1.2 Memory Leader Function

The Memory Loader Function of MIKBUG loads formatted binary object tapes or MIKBUG punched memory dump tapes into memory and if used, external memory modules. Figure 4-1 depicts the paper tape format. It is assumed at the start of this function that the MC6800 MPU is performing its MIKBUG control program and the last data printed by the terminal is an asterisk. Figure 4-2 illustrates a typical Memory Loader Function.



Frames 3 through N are hexadecimal digits (in 7-bit ASCII) which are converted to BCD. Two BCD digits are combined to make one 8-bit byte.

The checksum is the one's complement of the summation of 8-bit bytes.

Frame.	CC = 30 Header Record		CC = 31 Data Record		CC = 39 End-of-File Record	
1. Start-of-Record	53	S	53	S	53	S
2. Type of Record	30	0	31	1	39	9
3. Byte Count	31	12	31	16	30	03
4.	32		36		33	
5.	30	0000	31	1100	30	0000
6. Address/Size	30		31		30	
7.	30		30		30	
8.	30		30		30	
9. Data	34	48-11	39	98	46	FC
10.	38		38		43	(Checksum)
•	34	44-D	30	32		
•	34		32			
•	35	52-R	41	A8 (Checksum)		
•	32		48			
•						
N. Checksum	39	9E				
	45					

FIGURE 4-1. Paper Tape Format

a. Load the tape into the terminal tape reader.
b. Set the tape reader switch to AUTO.
c. Enter the character L after the asterisk. This initiates the MIKBUG loading procedure. The MIKBUG Firmware ignores all characters prior to the start-of-record on the tape.

NOTE

Tapes punched by MIKBUG do not have an end-of-file character at the end of the record; therefore, you must type in the characters S9 to exit from the memory loader function, or push the RESET pushbutton switch.

Checksum Error Detection

If, during the loading function, the MIKBUG Firmware detects a checksum error, it instructs the terminal to print a question mark and then stops the tape reader.

NOTE

Underlined characters indicate user input.

```
*L
S113000020FE02020202020202020202020202020B2
S9
*
```

FIGURE 4-2. Typical Memory Loader Function

d. If a checksum error is present, perform one of the following substeps:

   1) Press the RESET pushbutton switch and abort from the Memory Loader Function. The MPU will return to the MIKBUG control program and the terminal will print a carriage return, a line feed, and an asterisk.

   2) Reposition the tape and enter the character L. The record causing the checksum error is reread.

   3) Ignore the checksum error and enter the character L. The MIKBUG Firmware ignores the checksum error and continues the Memory Loader Function.

CAUTION

If a checksum error is in an address and the continue option in substep 3 is selected, there is no certain way of determining where the data will be loaded into the memory.

4.1.3 Memory Examine and Change Function

The MIKBUG Firmware performs this function in three steps: 1) examining the contents of the selected memory location (opening the memory location); 2) changing the contents of this location, if required; and 3) returning the contents to memory (closing the memory location). The MIKBUG Firmware, in examining a memory location, instructs the terminal to print the contents of this memory location. The MIKBUG Firmware in this function displays each of the program instructions in machine language.

It is assumed at the start of this function that the MPU is performing its MIKBUG control program and the last data printed by the terminal is an asterisk. Figure 4-3 depicts a typical Memory Examine and Change Function.

NOTE

If the memory address selected is in ROM, PROM, or protected RAM, the contents of this memory location cannot be changed and the terminal will print a question mark.

```
*M 0000
*0000 20  FF
*0001 FE  AA
*0002 02 .
*0003 02  ._
*
```

FIGURE 4-3 Typical Memory Examine and Change Function

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

b. Enter in 4-character hexadecimal format the memory address to be opened. The terminal will print on the next line the memory address being opened and the contents of this memory location. The contents are in hexadecimal.

c. The operator must now decide whether to change the data at this memory location. If the data is to be changed, change the data in accordance with step d. If the data is not to be changed, the operator must decide whether to close this location and open the following memory location (step e) or to close this memory location and return to the MIKBUG control program (step f).

d. If the contents of this memory location are to be changed, enter a space code and then the new data (in hexadecimal format) to be stored at this location. The new contents are stored in memory and the terminal prints the following memory address and its contents. Return to step c.

e. To close the present memory and open the following memory location, enter any character except a space character after the displayed memory address contents. The contents are returned to memory and the terminal prints the following memory address and its contents. Return to step c.

f. To close the present memory location and return to the MIKBUG control program, enter a space code followed by a carriage return control character. The contents are returned to memory and the terminal prints an asterisk on the next line.

4.1.4 Print/Punch Memory Function

The Print/Punch Memory Function instructs the MIKBUG Firmware to punch an absolute formatted binary tape and to print the selected memory contents. The tape is formatted as shown in Figure 4-1 except that this tape does not contain an end-of-file control character.

The beginning address and the ending address must be entered into the memory. Memory addresses A002 and A003 are used to store the beginning address and addresses A004 and A005 are used to store the ending address.

It is assumed that the MPU is performing its MIKBUG control program and the last data printed by the terminal is an asterisk. Figure 4-4 illustrates a typical Print/Punch Memory Function.

NOTE

If you do not wish to punch a tape, turn off the terminal's tape reader.

```
*M A002
*A002 F7  00
*A003 6E  01
*A004 99  00
*A005 EE  10
*A006 A0
*P
S1130001AA0202020202020202020202020202AC79
*
```

FIGURE 4-4 Typical Print/Punch Memory Function

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

b. Enter the address A002 after the space code. The terminal will print on the next line the memory address A002 and the contents of the address.

c. Enter a space code and the two most significant hexadecimal bytes of the beginning address after the contents of address A002. These two bytes are stored in memory and the terminal prints address A003 and its contents on the next line.

d. Enter a space code and the two least significant hexadecimal bytes of the beginning address after the contents of address A003. These two bytes are stored in memory and the terminal prints address A004 and its contents on the next line.

e. Enter a space code and the two most significant hexadecimal bytes of the ending address after the contents of address A004. These two bytes are stored in memory and the terminal prints address A005 and its contents on the next line.

f. Enter a space code and the two least significant hexadecimal bytes of the ending address after the contents of address A005. These two bytes are stored in memory and the terminal prints address A006 and its contents on the next line.

g. Enter a space code and carriage return character after the contents of address A006. The control returns to MIKBUG control program and the terminal prints an asterisk.

h. Enter the character P after the asterisk. The MIKBUG Firmware initiates the print/punch operation. At the conclusion of the print/punch operation the terminal prints an asterisk, and returns to the MIKBUG control program.

4.1.5 Delay Contents of MPU Registers Function

The Display Contents of MPU Registers Function enables the MIKBUG Firmware to display the contents of the MC6800 Microprocessing Unit registers for examination and change. It is assumed at the start of this function that

the MPU is performing its MIKBUG control program and the last data printed by the terminal is an asterisk. Figure 4-5 illustrates a typical Display Contents of MPU Registers Function.

```
*R 8A D6 CE 87AE CF4B A042
*M A043
*A043 8A .
*A044 D6 .
*A045 CE .
*A046 87 .
*A047 AE .
*A048 CF .
*A049 4B  00
*A04A 9E  00
*R 8A D6 CE 87AE 0000 A042
*
```

FIGURE 4-5 Typical Display Contents of MPU Register Function

a. Enter the character R after the asterisk. The terminal will print the contents of the MPU registers in the following sequence: condition code register, B accumulator, A accumulator, index register, program counter, and stack pointer. On the following line the terminal prints an asterisk.

b. If the contents of any of the registers are to be changed, change the data in accordance with Paragraph 4.1 .3. It should be noted that the address of the stack pointer is stored last, and it takes eight memory locations to store the contents of the MPU registers on the stack. Figure 4-5 illustrates changing the contents of the MPU registers and identifies the location of each register's data.

4.1.6 Go to User's Program Function

This function enables the MPU to perform the user's program. It is assumed at the start of this function that the MPU is performing its MIKBUG control program and the data printed by the terminal is an asterisk.

Enter the character G after the asterisk. The MC6800 MPU System will perform the user's program until one of the following conditions occurs:

1) The MPU encounters a WAI (WAIt) instruction. The MPU now waits for a non-maskable interrupt or an interrupt request.

2) The MPU encounters a SWI (Software Interrupt) instruction. The MPU stores the data in the MPU registers o n the stack and jumps to the MIKBUG control program. The terminal prints the contents of the MPU registers from the stack.

3) The RESET pushbutton switch is actuated. This switch is to be actuated when the user's program blows and places the MPU under the MIKBUG control program.

4.1.7 Interrupt Request Function

This function enables the user to evaluate a maskable interrupt routine. Steps a through e prepare the firmware to process an interrupt request and step f discusses performing the interrupt routine. It should be noted that this interrupt may be initiated at any time. It is assumed in preparing the MPU to process the interrupt request that the MPU is processing its MIKBUG control program and the last data printed by the terminal is an asterisk.

a. Enter the character M after the asterisk. The terminal will insert a space code after the M.

b. Enter the address A000. The terminal will print on the next line the memory address A000 and the contents of this memory location.

c. Enter a space code and the two most significant hexadecimal bytes of the first interrupt routine's address after the contents of address A000. These two bytes are stored in memory and the terminal prints address A001 and its contents on the next line.

d. Enter a space code and the two least significant hexadecimal bytes of the first interrupt routine's address after the contents of address A001. These two bytes are stored in memory and the terminal prints address A002 and its contents on the next line.

e. Enter a space code and a carriage return character after address A002. The MPU jumps to its MIKBUG control program and the terminal prints an asterisk.

The MPU now is enabled and ready to perform a maskable interrupt routine when the interrupt mask is cleared. This interrupt routine may be initiated at any time either through the PIA (if enabled) or the IRQ input to the MPU. Initiating an interrupt through the PIA is discussed in the MC6820 Peripheral Interface Adapter data sheet while initiating an interrupt through the IRQ input is discussed below.

f. Ground IRQ input. If the interrupt mask is not set, the MPU will jump to the interrupt service routine indirectly through addresses A000 and A001. This is accomplished in MIKBUG by loading the index register with the contents of addresses A000 and A001 and then jumping to the address stored in the index register.

g. Remove the ground from the IRQ input:

4.1.8 Non-Maskable Interrupt Function

This function enables the user to evaluate a non-maskable interrupt routine. Steps a through e prepare the MC6800 MPU System to process a NMI (Non-Maskable Interrupt) input and step f discusses performing the interrupt routine. It is assumed in preparing the MC6800 MPU System to process a non-maskable interrupt that the MC6800 MPU System is processing its MIKBUG control program and the last data printed by the data terminal is an asterisk.

a. Enter the character M after the asterisk. The terminal will insert a space code after the M.

b. Enter the address A006. The terminal will print on the next line the memory address A006 and the contents of this memory location.

c. Enter a space code and the two most significant hexadecimal digits of the first interrupt routine's address after the contents of address A006. These two digits are stored in memory and the terminal prints address A007 and its contents on the next line.

d. Enter a space code and the two least significant hexadecimal digits of the first interrupt routine's address after the contents of address A007. These two digits are stored in memory and the terminal prints address A008 and its contents on the next line.

e. Enter a space code and a carriage return character after address A008. The MC6800 MPU System jumps to its MIKBUG control program and the terminal prints an asterisk.

The MC6800 MPU System now is enabled to perform a non-maskable interrupt routine. This non-maskable interrupt routine may be initiated at any time through the MC6800 MPU System NMI input.

f. Ground the NMI input P1-E. If the non-maskable interrupt is not disabled (E3 to E4), the MPU will jump to the interrupt service routine indirectly through addresses A006 and A007. This is accomplished in MIKBUG by loading the index register with the contents of addresses A006 and A007 and then jumping to the address stored in the index register.

g. Remove the ground from the NMI input P1-E.

4.2 MINIBUG Rev. 4 Operation

The MINIBUG Firmware enables the user's system using the MIKBUG/MINIBUG ROM to perform the following functions:

Memory Loader Function
Memory Examine and Change Function
Display Contents of MPU Registers Function
Go to User's Program Function

The operating procedures for each of these routines as well as the RESET Function are discussed in the following paragraphs.

4.2.1 RESET Function

Perform the RESET Function when power is first applied and any time the MINIBUG Firmware loses program control.

Press the RESET switch (or equivalent). The MINIBUG Firmware should respond with a carriage return and a line feed character. The MINIBUG program control now is ready for an input.

4.2.2 Memory Loader Function

The memory loader function of MINIBUG loads formatted binary object tapes into memory. Figure 4-1 depicts the paper tape format. It is assumed at the start of this function that the MC6800 MPU is performing its MINIBUG control program. Figure 4-6 illustrates a typical memory loader function.

a. Load the tape into the tape reader.
b. Set the tape reader switch to AUTO.
c. Enter the character L. This initiates the MINIBUG loading procedure. The MINIBUG program ignores all characters prior to the start-of-record on the tape.

Checksum Error Detection

If during the loading function, the MINIBUG Firmware detects a checksum error, it instructs the terminal to print a question mark and stops while the MPU performs the MINIBUG control program. To load the tape, the user will have to repeat the memory loader function.

4.2.3 Memory Examine and Change Function

The MINIBUG Firmware performs this function in three steps: 1) examining the contents of the selected memory location (opening the memory location); 2) changing the contents of this location, if required; and 3) returning the contents to memory (closing the memory location). The Firmware, in examining a memory location, instructs the terminal to print the contents of this memory locationin hexadecimal format. The MINIBUG Firmware in this function displays each of the program instructions in Machine language.

It is assumed at the start of this function that the MPU is performing its MINIBUG control program. Figure 4-7 depicts a typical Memory Examine and Change Function.

NOTE

If no memory, a ROM, or a PROM is located at the selected address, the contents of this memory address cannot be changed and the terminal will print a question mark.

```
M FF2E 00 F0
M FF2F 00 00
```

FIGURE 4-7 Typical Memory Examine and Change Function

```
L

S113000020FE02020202020202020202020202020B2

S9
```

FIGURE 4-6. Typical Memory Loader Function

a. Enter the character M. The terminal will insert a space code after the M.

b. Enter in 4-character hexadecimal the memory address to be opened. The terminal will print a space code and then the contents of this memory location. The contents are in hexadecimal.

c. The operator must now decide whether to change the data at this memory location. If the data is to be changed, enter the two new hexadecimal characters to be stored in this location. The new contents are stored in memory and the MPU returns to the MINIBUG control program. If the data is not to be changed, enter a carriage return character; the previous contents are returned to memory and the MPU returns to the MINIBUG control program.

4.2.4 Display Contents of MPU Registers Function

The Display Contents of MPU Registers Function enables the MINIBUG Firmware to display the contents of the MC6800 Microprocessing Unit registers for examination and change. It is assumed at the start of this function that the MPU is performing the MINIBUG control program. Figure 4-8 illustrates a typical Display Contents of MPU Registers Function.

```
  CC B  A  XH XL PH PL SH SL
P 00 00 00 00 00 F0 00 FF 29
```

FIGURE 4-8 Typical Contents of MPU Register Function

a. Enter the character P. The terminal will print the contents of the MPU registers in the following sequence:

SP	Contents	MPU Register
FF29	00	Condition Code Register
FF2A	00	B Accumulator
FF2B	00	A Accumulator
FF2C	00	Index Register High
FF2D	00	Index Register Low
FF2E	F0	Program Counter High
FF2F	00	Program Counter Low

b. Use the Memory Examine and Change Function in paragraph 4.2.3 to change the contents of a register.

4.2.5 Go to User's Program Function

This function enables the MPU to perform the user's program. It is assumed at the start of this function that the MPU is performing its MINIBUG control program. Figure 4-9 illustrates a typical Go to User's Program Function.

Enter the character G. The MPU will load the MPU registers with the contents identified in Paragraph 4.2.4 and then start running the user's program at the address in the program counter (locations FF2E and FF2F). The program counter may be changed using the Memory Examine and Change Function in Paragraph 4.2.3.

```
P 00 00 00 00 00 00 00 FF 29
M 0000 FF 7E
M 0001 00
M 0002
G
```

FIGURE 4-9 Typical Go to Use's Program Function

5.0 MIKBUG REV. 9 PROGRAM LISTING

```
                        NAM    MIKBUG
               *       REV 009
               *       COPYRIGHT 1974 BY MOTOROLA INC

               *       MIKBUG (TM)

               *       L  LOAD
               *       G  GO TO TARGET PROGRAM
               *       M  MEMORY CHANGE
               *       F  PRINTIPUNCH DUMP
               *       R  DISPLAY CONTENTS OF TARGET STACK
               *             CC   B   A   X   P   S
8007           PIASB   EQU    $8007
8006           PIADB   EQU    $8006     B DATA
8005           PIAS    EQU    $8005     PIA STATUS
8004           PIAD    EQU    $8004     PIA DATA
               *       OPT    MEMORY
E000                   ORG    $E000

               *      I/O INTERRUPT SEQUENCE
E000 FE A0 00  IO      LDX    IOV
E003 6E 00             JMP    X

               * NMI SEQUENCE
E005 FE A0 06  POWDWN  LDX    NIO       GET NMI VECTOR
E008 6E 00             JMP    X

E00A           LOAD    EQU    *
E00A 86 3C             LDA A  #$3C
E00C B7 80 07          STA A  PIASB     READER RELAY ON
E00F 86 11             LDA A  #@21
E011 8D 62             BSR    OUTCH     OUTPUT CHAR

E013 8D 63     LOAD3   BSR    INCH
E015 81 53             CMP A  #'S
E017 26 FA             BNE    LOAD3     1ST CHAR NOT (S)
E019 8D 5D             BSR    INCH      READ CHAR
E01B 81 39             CMP A  #'9
E01D 27 25             BEQ    LOAD21
E01F 81 31             CMP A  #'1
E021 26 F0             BNE    LOAD3     2ND CHAR NOT (1)
E023 7F A0 0A          CLR    CKSM      ZERO CHECKSUM
E026 8D 2D             BSR    BYTE      READ BYTE
E028 80 02             SUB A  #2
E02A B7 A0 0B          STA A  BYTECT    BYTE COUNT
               * BUILD ADDRESS
E02D 8D 18             BSR    BADDR
               * STORE DATA
E02F 8D 24     LOAD11  BSR    BYTE
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
E031 7A A0 0B           DEC    BYTECT
E034 27 05              BEQ    LOAD15    ZERO BYTE COUNT
E036 A7 00              STA A  X         STORE DATA
E038 08                 INX
E039 20 F4              BRA    LOAD11

E03B 7C A0 0A  LOAD15   INC    CKSM
E03E 27 D3              BEQ    LOAD3
E040 86 3F     LOAD19   LDA A  #'?       PRINT QUESTION MARK
E042 8D 31              BSR    OUTCH
E044           LOAD21   EQU    *
E044 7E E0 E3  C1       JMP    CONTRL

               * BUILD ADDRESS
E047 8D 0C     BADDR    BSR    BYTE      READ 2 FRAMES
E049 B7 A0 0C           STA A  XHI
E04C 8D 07              BSR    BYTE
E04E B7 A0 0D           STA A  XLOW
E051 FE A0 0C           LDX    XHI       (X) ADDRESS WE BUILT
E054 39                 RTS

               *INPUT BYTE (TWO FRAMES)
E055 8D 53     BYTE     BSR    INHEX     GET HEX CHAR
E057 48                 ASL A
E058 48                 ASL A
E059 48                 ASL A
E05A 48                 ASL A
E05B 16                 TAB
E05C 8D 4C              BSR    INHEX
E05E 1B                 ABA
E05F 16                 TAB
E060 FB A0 0A           ADD B  CKSM
E063 F7 A0 0A           STA B  CKSM
E066 39                 RTS

E067 44        OUTHL    LSR A            OUT HEX LEFT BCD DIGIT
E068 44                 LSR A
E069 44                 LSR A
E06A 44                 LSR A

E06B 84 0F     OUTHR    AND A  #$F       OUT HEX RIGHT BCD DIGIT
E06D 8B 30              ADD A  #$30
E06F 81 39              CMP A  #$39
E071 23 02              BLS    OUTCH
E073 8B 07              ADD A  #$7

               * OUTPUT ONE CHAR
E075 7E E1 D1  OUTCH    JMP    OUTEEE
E078 7E E1 AC  INCH     JMP    INEEE
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
                    * PRINT DATA POINTED AT BY X-REG
      E07B 8D F8    PDATA2  BSR    OUTCH
      E07D 08               INX
      E07E A6 00    PDATA1  LDA A  X
      E080 81 04            CMP A  #4
      E082 26 F7            BNE    PDATA2
      E084 39               RTS              STOP ON EOT

                    * CHANGE MENORY (M AAAA DD NN)
      E085 8D C0    CHANGE  BSR    BADDR     BUILD ADDRESS
      E087 CE E1 9D CHA51   LDX    #MCL
      E08A 8D F2            BSR    PDATA1    C/R L/F
      E08C CE A0 0C         LDX    #XHI
      E08F 8D 37            BSR    OUT4HS    PRINT ADDRESS
      E091 FE A0 0C         LDX    XHI
      E094 8D 34            BSR    OUT2HS    PRINT DATA (OLD)
      E096 FF A0 0C         STX    XHI       SAYE DATA ADDRESS
      E099 8D DD            BSR    INCH      INPUT ONE CHAR
      E09B 81 20            CMP A  #$20
      E09D 26 E8            BNE    CHA51     NOT SPACE
      E09F 8D B4            BSR    BYTE      INPUT NEW DATA
      E0A1 09               DEX
      E0A2 A7 00            STA A  X         CHANGE MEMORY
      E0A4 A1 00            CMP A  X
      E0A6 27 DF            BEQ    CHA51     DID CHANGE
      E0A8 20 96            BRA    LOAD19    NOT CHANGED

                    * INPUT HEX CHAR
      E0AA 8D CC    INHEX   BSR    INCH
      E0AC 80 30            SUB A  #$30
      E0AE 2B 94            BMI    C1        NOT HEX
      E0B0 81 09            CMP A  #$09
      E0B2 2F 0A            BLE    IN1HG
      E0B4 81 11            CMP A  #$11
      E0B6 2B 8C            BMI    C1        NOT HEX
      E0B8 81 16            CMP A  #$16
      E0BA 2E 88            BGT    C1        NOT HEX
      E0BC 80 07            SUB A  #7
      E0BE 39       IN1HG   RTS

      E0BF A6 00    OUT2H   LDA A  0,X       OUTPUT 2 HEX CHAR
      E0C1 8D A4    OUT2HA  BSR    OUTHL     OUT LEFT HEX CHAR
      E0C3 A6 00            LDA A  0,X
      E0C5 08               INX
      E0C6 20 A3            BRA    OUTHR     OUTPUT RIGHT HEX CHAR AND R

      E0C8 8D F5    OUT4HS  BSR    OUT2H     OUTPUT 4 HEX CHAR + SPACE
      E0CA 8D F3    OUT2HS  BSR    OUT2H     OUTPUT 2 HEX CHAR + SPACE
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
E0CC 86 20     OUTS    LDA A  #$20      SPACE
E0CE 20 A5             BRA    OUTCH     (BSR & RTS)

               * ENTER POWER  ON SEQUENCE
E0D0           START   EQU    *
E0D0 8E A0 42          LDS    #STACK
E0D3 BF A0 08          STS    SP        INZ TARGET'S STACK PNTR
               * INZ PIA
E0D6 CE 80 04          LDX    #PIAD     (X) POINTER TO DEVICE PIA
E0D9 6C 00             INC    0,X       SET DATA DIR PIAD
E0DB 86 07             LDA A  #$7
E0DD A7 01             STA A  1,X       INIT CON PIAS
E0DF 6C 00             INC    0,X       MARK COM LINE
E0E1 A7 02             STA A  2,X       SET DATA DIR PIADB
E0E3 86 34     CONTRL  LDA A  #$34
E0E5 B7 80 07          STA A  PIASB     SET CONTROL PIASB TURN READ
E0E8 B7 80 06          STA A  PIADB     SET TIMER INTERVAL
E0EB 8E A0 42          LDS    #STACK    SET CONTRL STACK POINTER
E0EE CE E1 9C          LDX    #MCLOFF

E0F1 8D 8B             BSR    PDATA1    PRINT DATA STRING

E0F3 8D 83             BSR    INCH      READ CHARACTER
E0F5 16                TAB
E0F6 8D D4             BSR    OUTS      PRINT SPACE
E0F8 C1 4C             CMP B  #'L
E0FA 26 03             BNE    *+5
E0FC 7E E0 0A          JMP    LOAD
E0FF C1 4D             CMP B  #'M
E101 27 82             BEQ    CHANGE
E103 C1 52             CMP B  #'R
E105 27 18             BEQ    PRINT     STACK
E107 C1 50             CMP B  #'P
E109 27 32             BEQ    PUNCH     PRINT/PUNCH
E10B C1 47             CMP B  #'G
E10D 26 D4             BNE    CONTRL
E10F BE A0 08          LDS    SP        RESTORE PGM'S STACK PTR
E112 3B                RTI              GO

               * ENTER FROM SOFTVARE INTERRUPT
E113           SFE     EQU    *
E113 BF A0 08          STS    SP        SAVE TARGET'S STACK POINTER
               * DECREMENT P-COUNTER
E116 30                TSX
E117 6D 06             TST    6,X
E119 26 02             BNE    *+4
E11B 6A 05             DEC    5,X
E11D 6A 06             DEC    6,X

               * PRINT CONTENTS OF STACK
E11F FE A0 08  PRINT   LDX    SP
E122 08                INX
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
E123 8D A5              BSR    OUT2HS    CONDITION CODES
E125 8D A3              BSR    OUT2HS    ACC-B
E127 8D A1              BSR    OUT2HS    ACC-A
E129 8D 9D              BSR    OUT4HS    X-REG
E12B 8D 9B              BSR    OUT4HS    P-COUNTER
E12D CE A0 08           LDX    #SP
E130 8D 96              BSR    OUT4HS    STACK POINTER
E132 20 AF      C2      BRA    CONTRL


                * PUNCH DUMP
                * PUNCH FROM BEGINING ADDRESS (BEGA) THRU ENDI
                * ADDRESS (ENDA)
                *
E134 0D         MTAPE1  FCB    $D,$A,0,0,0,0,'S,'1,4 PUNCH FORMAT
E135 0A 00
E137 00 00
E139 00 53
E13B 31 04

E13D            PUNCH   EQU    *

E13D 86 12              LDA A  #$12      TURN TTY PUNCH ON
E13F BD E0 75           JSR    OUTCH     OUT CHAR

E142 FE A0 02           LDX    BEGA
E145 FF A0 0F           STX    TW        TEMP BEGINING ADDRESS
E148 B6 A0 05   PUN11   LDA A  ENDA+1
E14B B0 A0 10           SUB A  TW+1
E14E F6 A0 04           LDA B  ENDA
E151 F2 A0 0F           SBC B  TW
E154 26 04              BNE    PUN22
E156 81 10              CMP A  #16
E158 25 02              BCS    PUN23
E15A 86 0F      PUN22   LDA A  #15
E15C 8B 04      PUN23   ADD A  #4
E15E B7 A0 11           STA A  MCONT     FRAME COUNT THIS RECORD
E161 80 03              SUB A  #3
E163 B7 A0 0E           STA A  TEMP      BYTE COUNT THIS RECORD
                * PUNCH C/R,L/F,NULL,S,1
E166 CE E1 34           LDX    #MTAPE1
E169 BD E0 7E           JSR    PDATA1
E16C 5F                 CLR B            ZERO CHECKSUM
                * PUNCH FRAME COUNT
E16D CE A0 11           LDX    #MCONT
E170 8D 25              BSR    PUNT2     PUNCH 2 HEX CHAR
                * PUNCH ADDRESS
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
E172 CE A0 0F          LDX    #TW
E175 8D 20             BSR    PUNT2
E177 8D 1E             BSR    PUNT2
               * PUNCH DATA
E179 FE A0 0F          LDX    TW
E17C 8D 19     PUN32   BSR    PUNT2     PUNCH ONE BYTE (2 FRAMES)
E17E 7A A0 0E          DEC    TEMP      DEC BYTE COUNT
E181 26 F9             BNE    PUN32
E183 FF A0 0F          STX    TW
E186 53                COM B
E187 37                PSH B
E188 30                TSX
E189 8D 0C             BSR    PUNT2     PUNCH CHECKSUM
E18B 33                PUL B            RESTORE STACK
E18C FE A0 0F          LDX    TW
E18F 09                DEX
E190 BC A0 04          CPX    ENDA
E193 26 B3             BNE    PUN11
E195 20 9B             BRA    C2        JMP TO CONTRL


               * PUNCH 2 HEX CHAR UPDATE CHECKSUM
E197 EB 00     PUNT2   ADD B  0,X       UPDATE CHECKSUM
E199 7E E0 BF          JMP    OUT2H     OUTPUT TWO HEX CHAR AND RTS


E19C 13        MCLOFF  FCB    $13       READER OFF
E19D 0D        MCL     FCB    $D,$A,$14,0,0,0,'*,4 C/R,L/F,PUNCH
E19E 0A 14
E1A0 00 00
E1A2 00 2A
E1A4 04
               *
E1A5 FF A0 12  SAV     STX    XTEMP
E1A8 CE 80 04          LDX    #PIAD
E1AB 39                RTS

               *INPUT   ONE CHAR INTO A-REGISTER
E1AC 37        INEEE   PSH B            SAVE ACC-B
E1AD 8D F6             BSR    SAV       SAV XR
E1AF A6 00     IN1     LDA A  0,X       LOOK FOR START BIT
E1B1 2B FC             BMI    IN1
E1B3 6F 02             CLR    2,X       SET COUNTER FOR HALF BIT TI
E1B5 8D 3C             BSR    DE        START TIMER
E1B7 8D 36             BSR    DEL       DELAY HALF BIT TIME
E1B9 C6 04             LDA B  #4        SET DEL FOR FULL BIT TIME
E1BB E7 02             STA B  2,X
E1BD 58                ASL B            SET UP CNTR WITH 8
```

```
E1BE 8D 2F     IN3     BSR    DEL        WAIT ONE CHAR TIME
E1C0 0D                SEC               NARK CON LINE
E1C1 69 00             ROL    0,X        GET BIT INTO CFF
E1C3 46                ROR A             CFF TO AR
E1C4 5A                DEC B
E1C5 26 F7             BNE    IN3
E1C7 8D 26             BSR    DEL        WAIT FOR STOP BIT
E1C9 84 7F             AND A  #$7F       RESET PARITY BIT
E1CB 81 7F             CMP A  #$7F
E1CD 27 E0             BEQ    IN1        IF RUBOUT, GET NEXT CHAR
E1CF 20 12             BRA    IOUT2      GO RESTORE REG


               * OUTPUT ONE CHAR
E1D1 37        OUTEEE  PSH B             SAV BR
E1D2 8D D1             BSR    SAV        SAV XR
E1D4 C6 0A     IOUT    LDA B  #$A        SET UP COUNTER
E1D6 6A 00             DEC    0,X        SET START BIT
E1D8 8D 19             BSR    DE         START TIMER
E1DA 8D 13     OUT1    BSR    DEL        DELAY ONE BIT TIME
E1DC A7 00             STA A  0,X        PUT OUT ONE DATA BIT
E1DE 0D                SEC               SET CARRY BIT
E1DF 46                ROR A             SHIFT IN NEXT BIT
E1E0 5A                DEC B             DECREMENT COUNTER
E1E1 26 F7             BNE    OUT1       TEST FOR 0
E1E3 E6 02     IOUT2   LDA B  2,X        TEST FOR STOP BITS
E1E5 58                ASL B             SHIFT BIT TO SIGN
E1E6 2A 02             BPL    IOS        BRANCH FOR 1 STOP BIT
E1E8 8D 05             BSR    DEL        DELAY-FOR STOP BITS
E1EA FE A0 12  IOS     LDX    XTEMP      RES XR
E1ED 33                PUL B             RESTORE BR
E1EE 39                RTS

E1EF 6D 02     DEL     TST    2,X        IS TIME UP
E1F1 2A FC             BPL    DEL
E1F3 6C 02     DE      INC    2,X        RESET TIMER
E1F5 6A 02             DEC    2,X
E1F7 39                RTS

E1F8 E0 00             FDB    IO
E1FA E1 13             FDB    SFE
E1FC E0 05             FDB    POWDWN
E1FE E0 D0             FDB    START
A000                   ORG    $A000
A000           IOV     RMB    2          IO INTERRUPT POINTER
A002           BEGA    RMB    2          BEGINING ADDR PRINT/PUNCH
A004           ENDA    RMB    2          ENDING ADDR PRINT/PUNCH
A006           NIO     RMB    2          NMI INTERRUPT POINTER
A008           SP      RMB    1          S-HIGH
A009                   RMB    1          S-LOW
A00A           CKSM    RMB    1          CHECKSUM
```

MIKBUG REV. 9 PROGRAM LISTING (continued)

```
 A00B           BYTECT  RMB    1         BYTE COUNT
 A00C           XHI     RMB    1         XREG HIGH
 A00D           XLOW    RMB    1         XREG LOW
 A00E           TEMP    RMB    1         CHAR COUNT (INADD)
 A00F           TW      RMB    2         TEMP/
 A011           MCONT   RMB    1         TEMP
 A012           XTEMP   RMB    2         X-REG TEMP STORAGE
 A014                   RMB    46
 A042           STACK   RMB    1         STACK POINTER


                        END

NO ERROR(S) DETECTED

   SYMBOL TABLE:

BADDR  E047   BEGA   A002   BYTE   E055   BYTECT A00B   C1     E044
C2     E132   CHA51  E087   CHANGE E085   CKSM   A00A   CONTRL E0E3
DE     E1F3   DEL    E1EF   ENDA   A004   IN1    E1AF   IN1HG  E0BE
IN3    E1BE   INCH   E078   INEEE  E1AC   INHEX  E0AA   IO     E000
IOS    E1EA   IOUT   E1D4   IOUT2  E1E3   IOV    A000   LOAD   E00A
LOAD11 E02F   LOAD15 E03B   LOAD19 E040   LOAD21 E044   LOAD3  E013
MCL    E19D   MCLOFF E19C   MCONT  A011   MTAPE1 E134   NIO    A006
OUT1   E1DA   OUT2H  E0BF   OUT2HA E0C1   OUT2HS E0CA   OUT4HS E0C8
OUTCH  E075   OUTEEE E1D1   OUTHL  E067   OUTHR  E06B   OUTS   E0CC
PDATA1 E07E   PDATA2 E07B   PIAD   8004   PIADB  8006   PIAS   8005
PIASB  8007   POWDWN E005   PRINT  E11F   PUN11  E148   PUN22  E15A
PUN23  E15C   PUN32  E17C   PUNCH  E13D   PUNT2  E197   SAV    E1A5
SFE    E113   SP     A008   STACK  A042   START  E0D0   TEMP   A00E
TW     A00F   XHI    A00C   XLOW   A00D   XTEMP  A012
```

6.0 MINIBUG REV.4 PROGRAM LISTING

```
                          NAM     MINIB
                  * MINI-BUG
                  * COPYWRITE 1973, MOTOROLA IMC
                  * REV 004 (USED WITH MIKBUG)
      FCF4        ACIACS  EQU     @1176364  ACIA CONTROL/STATUS
      FCF5        ACIADA  EQU     ACIACS+1
      FE00                ORG     $FE00
                  * MINIB

                  * INPUT ONE CHAR INTO A-REGISTER
      FE00 B6 FC F4  INCH    LDA A  ACIACS
      FE03 47                ASR A
      FE04 24 FA             BCC    INCH      RECEIVE NOT READY
      FE06 B6 FC F5          LDA A  ACIADA    INPUT CHARACTER
      FE09 84 7F             AND A  #$7F      RESET PARITY BIT
      FE0B 81 7F             CMP A  #$7F
      FE0D 27 F1             BEQ    INCH      RUBOUT; IGNORE
      FE0F 7E FE AE          JMP    OUTCH     ECHO CHAR
                  * INPUT HEX CHAR
      FE12 8D EC     INHEX   BSR    INCH
      FE14 81 30             CMP A  #$30
      FE16 2B 52             BMI    C1        NOT HEX
      FE18 81 39             CMP A  #$39
      FE1A 2F 0A             BLE    IN1HG
      FE1C 81 41             CMP A  #$41
      FE1E 2B 4A             BMI    C1        NOT HEX
      FE20 81 46             CMP A  #$46
      FE22 2E 46             BGT    C1        NOT HEX
      FE24 80 07             SUB A  #7
      FE26 39        IN1HG   RTS

      FE27 86 D1     LOAD    LDA A  #$D1      TURN READER ON
      FE29 B7 FC F4          STA A  ACIACS
      FE2C 86 11             LDA A  #@21
      FE2E 8D 7E             BSR    OUTCH

      FE30 8D CE     LOAD3   BSR    INCH
      FE32 81 53             CMP A  #'S
      FE34 26 FA             BNE    LOAD3     1ST CHAR NOT (S)
      FE36 8D C8             BSR    INCH      READ CHAR
      FE38 81 39             CMP A  #'9
      FE3A 27 25             BEQ    LOAD21
      FE3C 81 31             CMP A  #'1
      FE3E 26 F0             BNE    LOAD3     2ND CHAR NOT (1)
      FE40 7F FF 32          CLR    CKSM      ZERO CHECKSUM
      FE43 8D 36             BSR    BYTE      READ BYTE
      FE45 80 02             SUB A  #2
      FE47 B7 FF 33          STA A  BYTECT    BYTE COUNT
                  * BUILD ADDRESS
      FE4A 8D 21             BSR    BADDR
                  * STORE DATA
      FE4C 8D 2D     LOAD11  BSR    BYTE
      FE4E 7A FF 33          DEC    BYTECT
```

```
FE51 27 05              BEQ    LOAD15    ZERO BYTE COUNT
FE53 A7 00              STA A  X         STORE DATA
FE55 08                 INX
FE56 20 F4              BRA    LOAD11

FE58 7C FF 32  LOAD15   INC    CKSM
FE5B 27 D3              BEQ    LOAD3
FE5D 86 3F     LOAD19   LDA A  #'?       PRINT QUESTION MARK
FE5F 8D 4D              BSR    OUTCH
FE61 86 B1     LOAD21   LDA A  #$B1      TURN READER OFF
FE63 B7 FC F4           STA A  ACIACS
FE66 86 13              LDA A  #@23
FE68 8D 44              BSR    OUTCH
FE6A 7E FE DB  C1       JMP    CONTRL


               * BUILD ADDRESS
FE6D 8D 0C     BADDR    BSR    BYTE      READ 2 FRAMES
FE6F B7 FF 34           STA A  XHI
FE72 8D 07              BSR    BYTE
FE74 B7 FF 35           STA A  XLOW
FE77 FE FF 34           LDX    XHI       (X) ADDRESS WE BUILT
FE7A 39                 RTS

               * INPUT BYTE (TWO FRAMES)
FE7B 8D 95     BYTE     BSR    INHEX     GET HEX CHAR
FE7D 48                 ASL A
FE7E 48                 ASL A
FE7F 48                 ASL A
FE80 48                 ASL A
FE81 16                 TAB
FE82 8D 8E              BSR    INHEX
FE84 84 0F              AND A  #$0F      MASK TO 4 BITS
FE86 1B                 ABA
FE87 16                 TAB
FE88 FB FF 32           ADD B  CKSM
FE8B F7 FF 32           STA B  CKSM
FE8E 39                 RTS

               *CHANGE MEMORY (M AAAA DD NN)
FE8F 8D DC     CHANGE   BSR    BADDR     BUILD ADDRESS
FE91 8D 34              BSR    OUTS      PRINT SPACE
FE93 8D 30              BSR    OUT2HS
FE95 8D E4              BSR    BYTE
FE97 09                 DEX
FE98 A7 00              STA A  X
FE9A A1 00              CMP A  X
FE9C 26 BF              BNE    LOAD19    MEMORY DID NOT CHAMSE
FE9E 20 3B              BRA    CONTRL

FEA0 44        OUTHL    LSR A            OUT HEX LEFT BCD DIGIT
FEA1 44                 LSR A
```

```
      FEA2 44                LSR A
      FEA3 44                LSR A

      FEA4 84 0F     OUTHR   AND A  #$F        OUT HEX RIGHT BCD DIGIT
      FEA6 8B 30             ADD A  #$30
      FEA8 81 39             CMP A  #$39
      FEAA 23 02             BLS    OUTCH
      FEAC 8B 07             ADD A  #$7

                     * OUTPUT ONE CHAR
      FEAE 37        OUTCH   PSH B             SAVE B-REG
      FEAF F6 FC F4  OUTC1   LDA B  ACIACS
      FEB2 57                ASR B
      FEB3 57                ASR B
      FEB4 24 F9             BCC    OUTC1      XMIT NOT READY
      FEB6 B7 FC F5          STA A  ACIADA     OUTPUT CHARACTER
      FEB9 33                PUL B             RESTORE B-REG
      FEBA 39                RTS

      FEBB A6 00     OUT2H   LDA A  0,X        OUTPUT 2 HEX CHAR
      FEBD 8D E1             BSR    OUTHL      OUT LEFT HEX CHAR
      FEBF A6 00             LDA A  0,X
      FEC1 8D E1             BSR    OUTHR      OUT RIGHT HEX CHAR
      FEC3 08                INX
      FEC4 39                RTS

      FEC5 8D F4     OUT2HS  BSR    OUT2H      OUTPUT 2 HEX CHAR + SPACE
      FEC7 86 20     OUTS    LDA A  #$20       SPACE
      FEC9 20 E3             BRA    OUTCH      (BSR & RTS)


                     * PRINT CONTENTS OF STACK.
      FECB 30        PRINT   TSX
      FECC FF FF 30          STX    SP         SAVE STACK POINTER
      FECF C6 09             LDA B  #9
      FED1 8D F2     PRINT2  BSR    OUT2HS     OUT 2 HEX & SPACE
      FED3 5A                DEC B
      FED4 26 FB             BNE    PRINT2


                     * ENTER POWER ON SEQUENCE
      FED6           START   EQU    *
                     * INZ ACIA
      FED6 86 B1             LDA A  #$B1       SET SYSTEM PARAMETERS
      FED8 B7 FC F4          STA A  ACIACS

      FEDB 8E FF 28  CONTRL  LDS    #STACK     SET STACK POINTER
      FEDE 86 0D             LDA A  #$D        CARRIAGE RETURM
```

```
FEE0 8D CC                BSR    OUTCH
FEE2 86 0A                LDA A  #$A       LINE FEED
FEE4 8D C8                BSR    OUTCH

FEE6 BD FE 00             JSR    INCH      READ CHARACTER
FEE9 16                   TAB
FEEA 8D DB                BSR    OUTS      PRINT SPACE
FEEC C1 4C                CMP B  #'L
FEEE 26 03                BNE    *+5
FEF0 7E FE 27             JMP    LOAD
FEF3 C1 4D                CMP B  #'M
FEF5 27 98                BEQ    CHANGE
FEF7 C1 50                CMP B  #'P
FEF9 27 D0                BEQ    PRINT     STACK
FEFB C1 47                CMP B  #'G
FEFD 26 DC                BNE    CONTRL
FEFF 3B                   RTI              GO


FF00                      ORG    $FF00
FF00                      RMB    40
FF28              STACK   RMB    1         STACK POINTER
                  * REGISTERS FOR GO
FF29                      RMB    1         CONDITION CODES
FF2A                      RMB    1         B ACCUMULATOR
FF2B                      RMB    1         A
FF2C                      RMB    1         X-HIGH
FF2D                      RMB    1         X-LOW
FF2E                      RMB    1         P-HIGH
FF2F                      RMB    1         P-LOW
FF30              SP      RMB    1         S-HIGH
FF31                      RMB    1         S-LOW
                  * END REGISTERS FOR GO
FF32              CKSM    RMB    1         CHECKSUM
FF33              BYTECT  RMB    1         BYTE COUNT
FF34              XHI     RMB    1         XREG HIGH
FF35              XLOW    RMB    1         XREG LOW
                          END

NO ERROR(S) DETECTED

   SYMBOL TABLE:

ACIACS FCF4   ACIADA FCF5   BADDR  FE6D   BYTE   FE7B   BYTECT FF33
C1     FE6A   CHANGE FE8F   CKSM   FF32   CONTRL FEDB   IN1HG  FE26
INCH   FE00   INHEX  FE12   LOAD   FE27   LOAD11 FE4C   LOAD15 FE58
LOAD19 FE5D   LOAD21 FE61   LOAD3  FE30   OUT2H  FEBB   OUT2HS FEC5
OUTC1  FEAF   OUTCH  FEAE   OUTHL  FEA0   OUTHR  FEA4   OUTS   FEC7
PRINT  FECB   PRINT2 FED1   SP     FF30   STACK  FF28   START  FED6
XHI    FF34   XLOW   FF35
```