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 M6800 Family of parts. This ROM provides an asynchronous communications program, a loader program, and a diagnostic program for use with the MC6800 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.

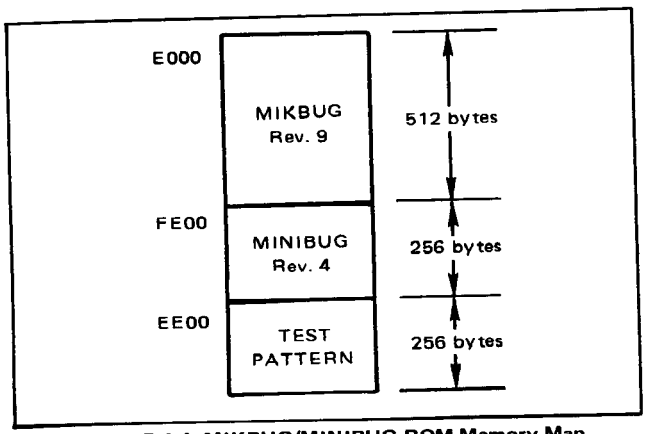

FIGURE 1-1. MIKBUG/MINIBUG ROM Memory Map

NOTE

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 E1FF. The ROM should be connected into a system so that its two top MIKBUG Firmware addresses also will respond to addresses FFFE and FFFF. This is required for the system to restart properly. There should not be any devices in the system at a higher address than 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.

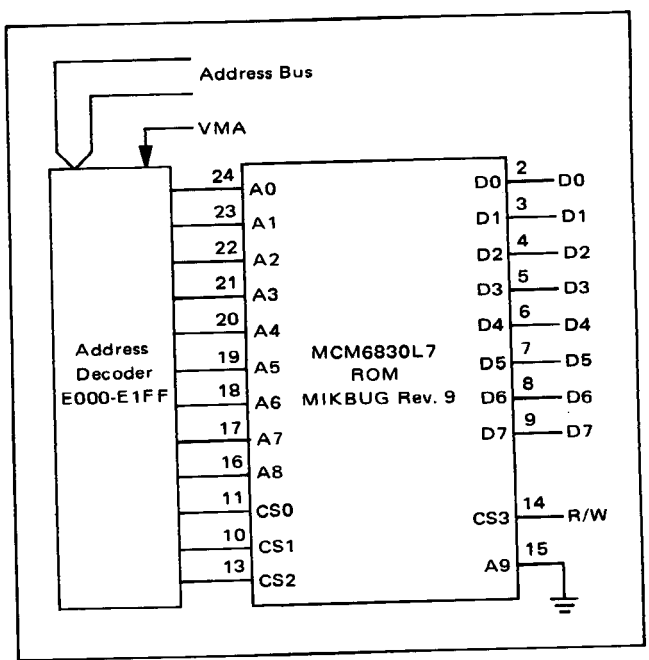

FIGURE 3-1. MCM6830L7 MIKBUG ROM Schematic

The MIKBUG Firmware operates with an MC6820 Peripheral Interface Adapter (PIA) as shown in Figure 3-4. The MC14536 device is used as the interface timer. This timer's interval is set by adjusting the 50 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

The information in this Engineering Note has been carefully reviewed and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described any license under the patent rights of Motorola Inc. or others.
The material in this note is subject to change, and Motorola Inc. reserves the right to change specifications without notice.
MIKBUG and MINIBUG are trademarks of Motorola Inc.

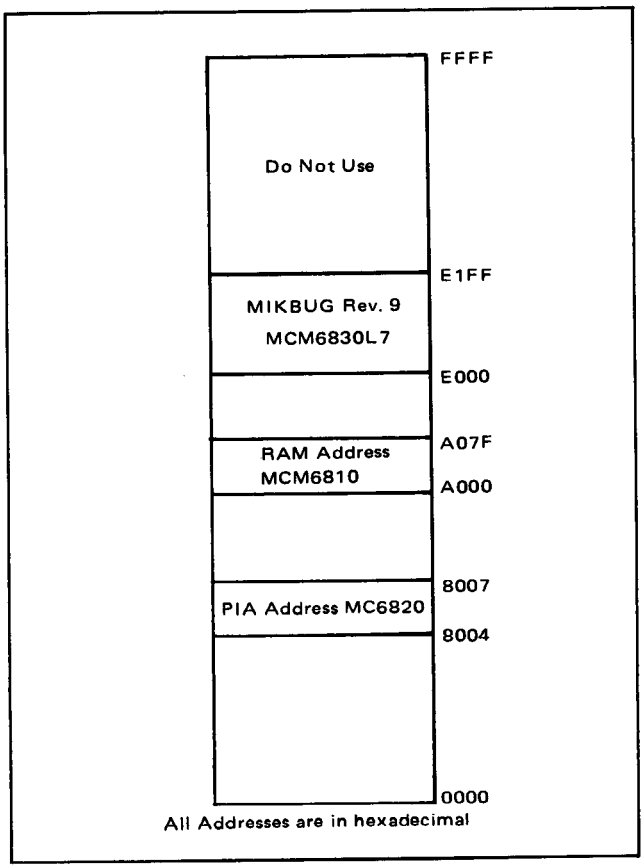

FIGURE 3-2. MIKBUG Rev. 9 Memory Map

for 10 characters 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 MC1488 and MC1489A 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 (Fig-

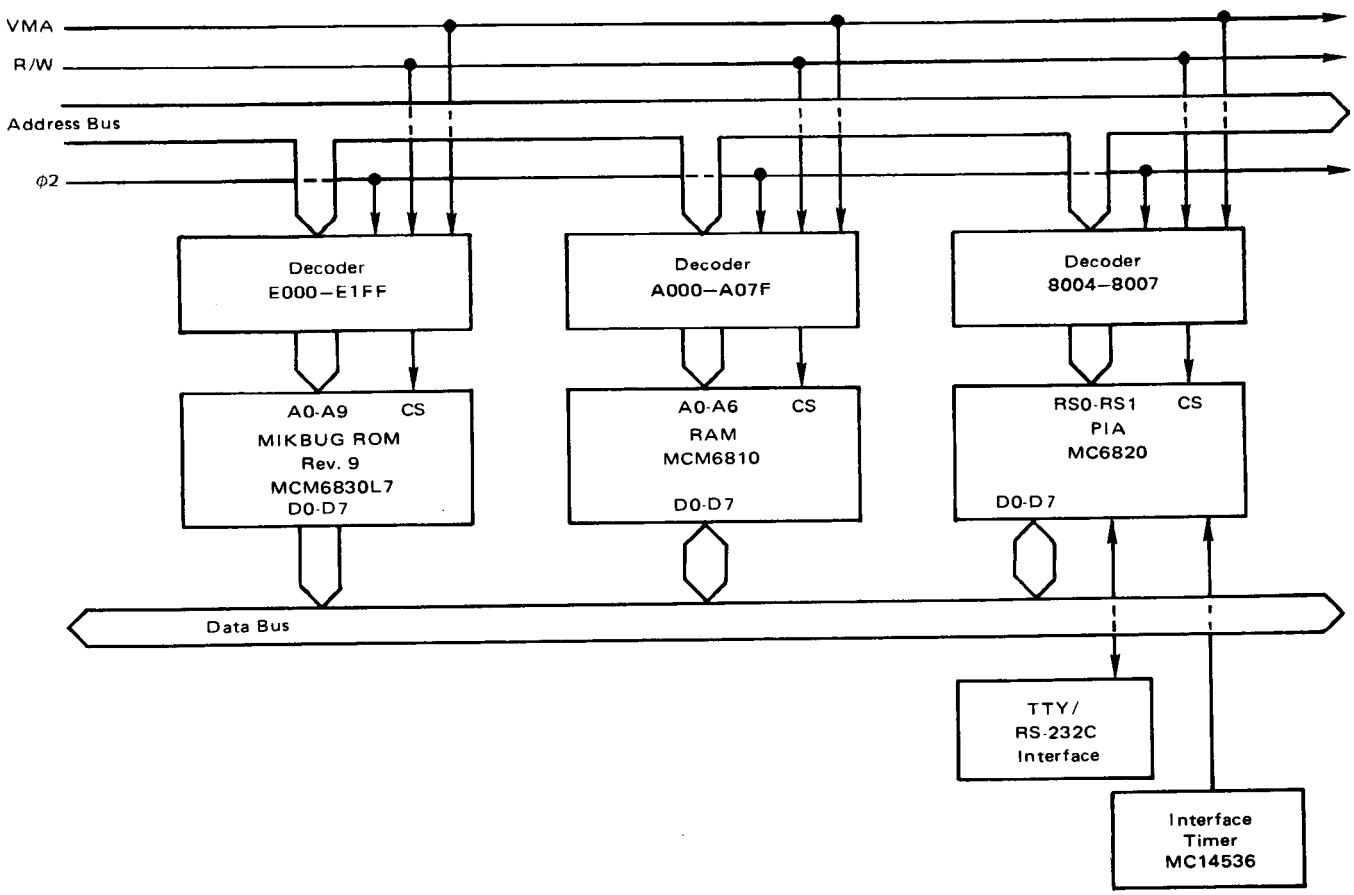

FIGURE 3-3. MIKBUG ROM Rev. 9 System Block Diagram

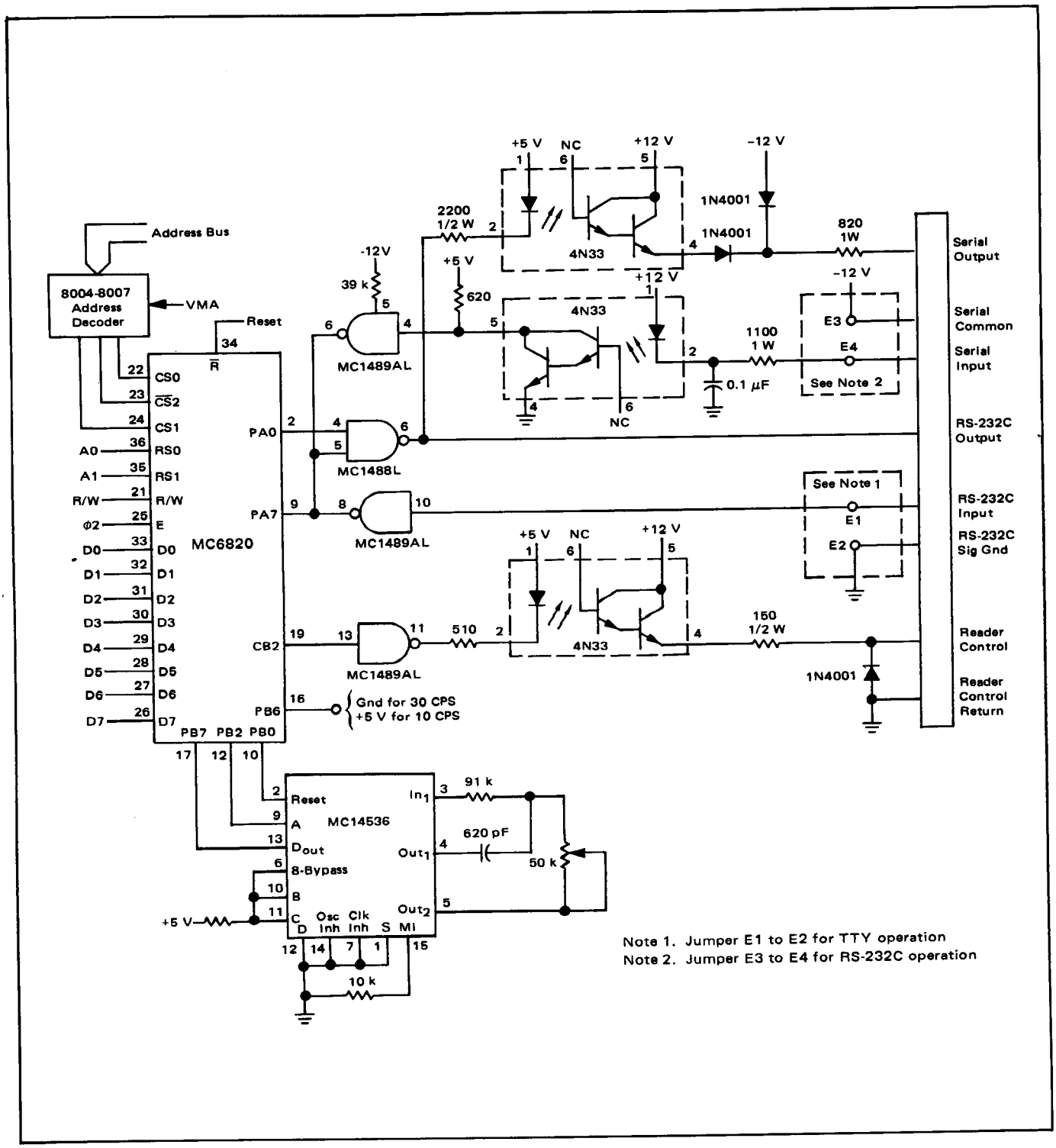

FIGURE 3-4. TTY/RS-232C Interface Used with MIKBUG ROM

ure 3-5) must be configured to respond with address FED6 each time the MPU requests the restart address. As shown, the system also requires an MCM6810 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.

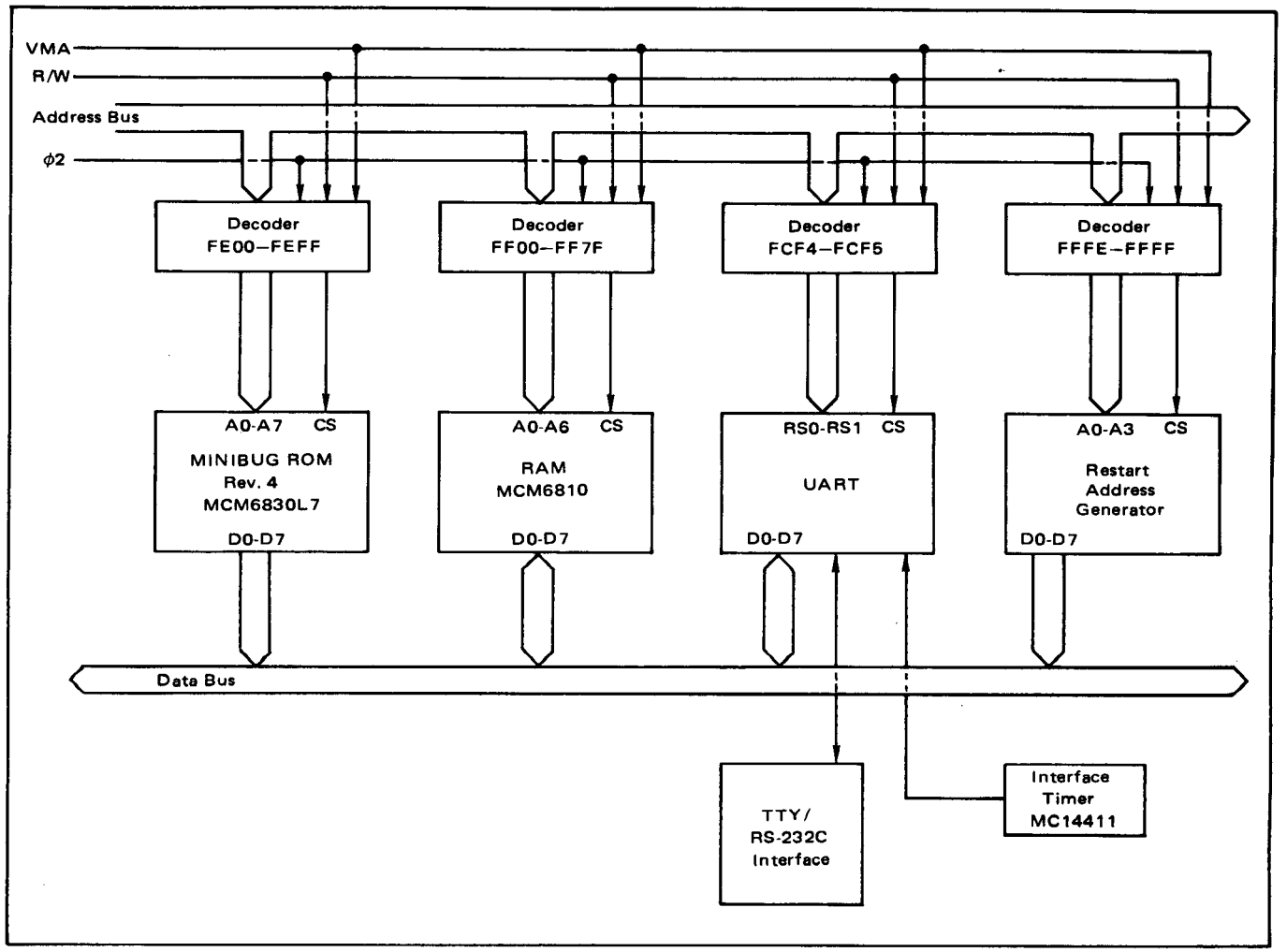

FIGURE 3-5. MINIBUG Rev. 4 System Block Diagram

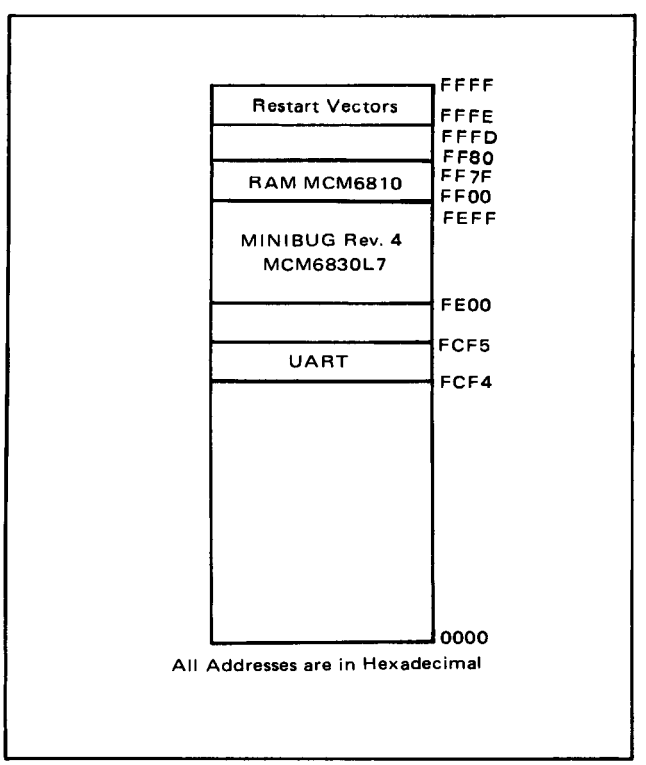

FIGURE 3-6. MINIBUG Rev. 4 Memory Map

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 Loader 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.

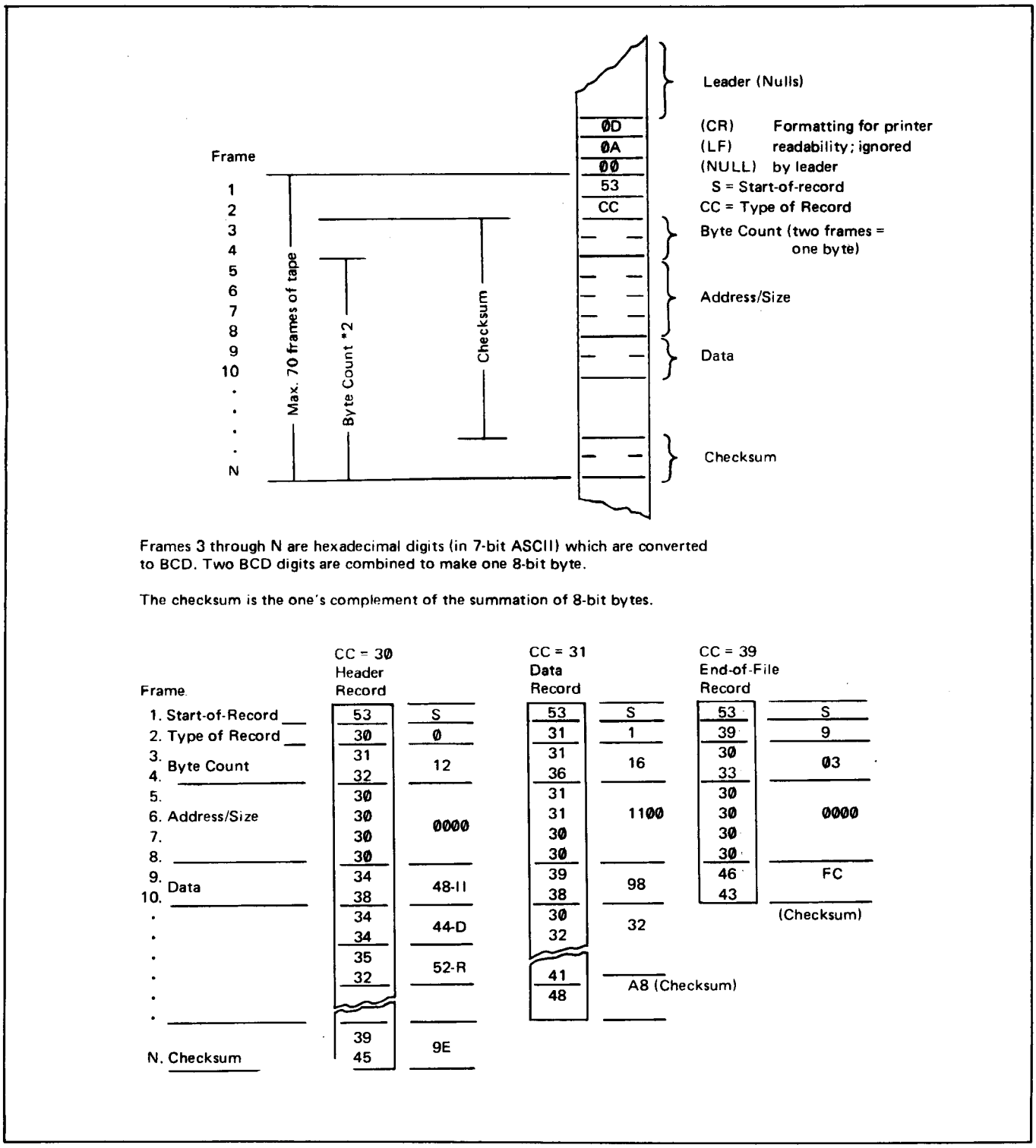

FIGURE 4-1. Paper Tape Format

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.

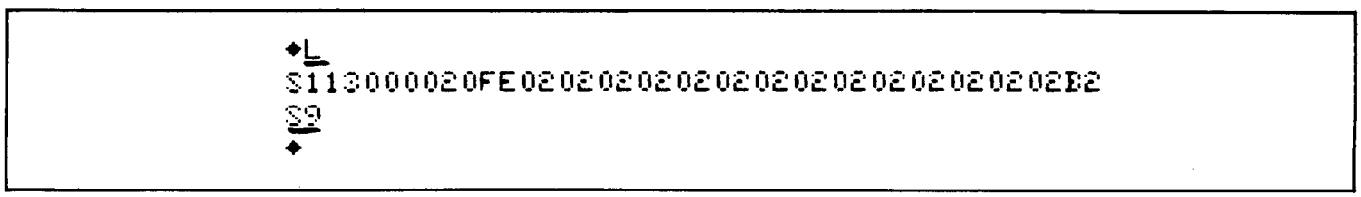

FIGURE 4-2. Typical Memory Loader Function

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.

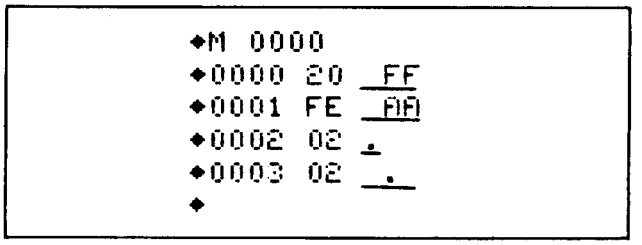

FIGURE 4-3. Typical Memory Examine and Change Function

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
S1130001AA020202020202020202020202020202AC79
◆
```

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 Display 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  00
◆A049 4B  00
◆A04A 9E  __
◆R 8A D6 CE 87AE 0000 A042
◆
```

FIGURE 4-5. Typical Display Contents of MPU Registers 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 on 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 interrup mask is cleared. This interrupt routine may be initiated at any time either through the PIA (if enabled) or the $\overline{IRQ}$ 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 $\overline{IRQ}$ input is discussed below.

f. Ground $\overline{IRQ}$ 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 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 $\overline{NMI}$ input.

f. Ground the $\overline{NMI}$ 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 $\overline{NMI}$ 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 location in 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

S113000020FE0202020202020202020202020202B2

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.

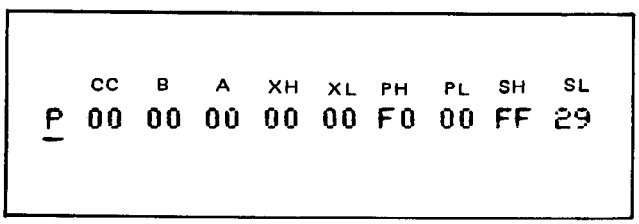

FIGURE 4-8. Typical Contents of MPU Register Function

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.

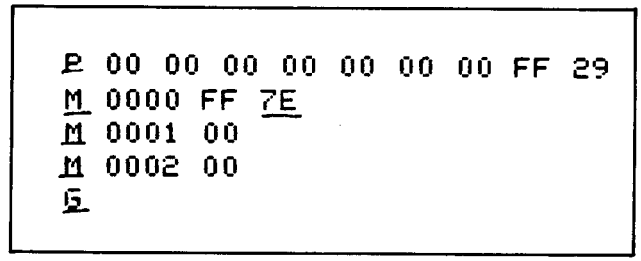

FIGURE 4-9. Typical Go to User's Program Function

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.

5.0 MIKBUG REV. 9 PROGRAM LISTING

```
00100                          NAM    MIKBUG
00200                  *       REV 009
00300                  *       COPYRIGHT 1974 BY MOTOROLA INC

00500                  *       MIKBUG (TM)

00700                  *       L  LOAD
00800                  *       G  GO TO TARGET PROGRAM
00900                  *       M  MEMORY CHANGE
01000                  *       P  PRINT/PUNCH DUMP
01100                  *       R  DISPLAY CONTENTS OF TARGET STACK
01200                  *             CC   B   A   X   P   S

01400        8007      PIASB   EQU    $8007
01500        8006      PIADB   EQU    $8006     B DATA
01600        8005      PIAS    EQU    $8005     PIA STATUS
01700        8004      PIAD    EQU    $8004     PIA DATA
01800                          OPT    MEMORY
01900 E000                     ORG    $E000

02100                  *       I/O INTERRUPT SEQUENCE
02200 E000 FE A000     IO      LDX    IOV
02300 E003 6E 00               JMP    X

  500                  * NMI SEQUENCE
02500 E005 FE A006     POWDWN  LDX    NIO       GET NMI VECTOR
02700 E008 6E 00               JMP    X



03000        E00A      LOAD    EQU    *
03100 E00A 86 3C               LDA A  #$3C
03200 E00C B7 8007             STA A  PIASB     READER RELAY ON
03300 E00F 86 11               LDA A  #@21
03400 E011 8D 62               BSR    OUTCH     OUTPUT CHAR

03600 E013 8D 63     LOAD3     BSR    INCH
03700 E015 81 53               CMP A  #'S
03800 E017 26 FA               BNE    LOAD3     1ST CHAR NOT (S)
03900 E019 8D 5D               BSR    INCH      READ CHAR
04000 E01B 81 39               CMP A  #'9
04100 E01D 27 25               BEQ    LOAD21
04200 E01F 81 31               CMP A  #'1
04300 E021 26 F0               BNE    LOAD3     2ND CHAR NOT (1)
04400 E023 7F A00A             CLR    CKSM      ZERO CHECKSUM
04500 E026 8D 2D               BSR    BYTE      READ BYTE
04600 E028 80 02               SUB A  #2
04700 E02A B7 A00B             STA A  BYTECT    BYTE COUNT
04800                  * BUILD ADDRESS
04900 E02D 8D 18               BSR    BADDR
05000                  * STORE DATA
05100 E02F 8D 24     LOAD11    BSR    BYTE
```

```
05200 E031 7A A00B          DEC     BYTECT
05300 E034 27 05            BEQ     LOAD15    ZERO BYTE COUNT
05400 E036 A7 00            STA A   X         STORE DATA
05500 E038 08               INX
05600 E039 20 F4            BRA     LOAD11

05800 E03B 7C A00A LOAD15   INC     CKSM
05900 E03E 27 D3            BEQ     LOAD3
06000 E040 86 3F   LOAD19   LDA A   #'?       PRINT QUESTION MARK
06100 E042 8D 31            BSR     OUTCH
06200      E044    LOAD21   EQU     *
06300 E044 7E E0E3 C1       JMP     CONTRL


06500                  * BUILD ADDRESS
06600 E047 8D 0C    BADDR  BSR     BYTE      READ 2 FRAMES
06700 E049 B7 A00C         STA A   XHI
06800 E04C 8D 07           BSR     BYTE
06900 E04E B7 A00D         STA A   XLOW
07000 E051 FE A00C         LDX     XHI       (X) ADDRESS WE BUILT
07100 E054 39              RTS


07300                  * INPUT BYTE (TWO FRAMES)
07400 E055 8D 53    BYTE   BSR     INHEX     GET HEX CHAR
07500 E057 48              ASL A
07600 E058 48              ASL A
07700 E059 48              ASL A
07800 E05A 48              ASL A
07900 E05B 16              TAB
08000 E05C 8D 4C           BSR     INHEX
08100 E05E 1B              ABA
08200 E05F 16              TAB
08300 E060 FB A00A         ADD B   CKSM
08400 E063 F7 A00A         STA B   CKSM
08500 E066 39              RTS

08700 E067 44       OUTHL  LSR A             OUT HEX LEFT BCD DIGIT
08800 E068 44              LSR A
08900 E069 44              LSR A
09000 E06A 44              LSR A


09300 E06B 84 0F    OUTHR  AND A   #$F        OUT HEX RIGHT BCD DIGIT
09400 E06D 8B 30           ADD A   #$30
09500 E06F 81 39           CMP A   #$39
09600 E071 23 02           BLS     OUTCH
09700 E073 8B 07           ADD A   #$7

09900                  * OUTPUT ONE CHAR
10000 E075 7E E1D1 OUTCH   JMP     OUTEEE
10100 E078 7E E1AC INCH    JMP     INEEE
```

```
10200                    * PRINT DATA POINTED AT BY X-REG
10300 E07B 8D F8   PDATA2 BSR    OUTCH
10400 E07D 08             INX
10500 E07E A6 00   PDATA1 LDA A  X
10600 E080 81 04          CMP A  #4
10700 E082 26 F7          BNE    PDATA2
10800 E084 39             RTS                STOP ON EOT

11000                    * CHANGE MEMORY (M AAAA DD NN)
11100 E085 8D C0   CHANGE BSR    BADDR      BUILD ADDRESS
11200 E087 CE E19D CHA51  LDX    #MCL
11300 E08A 8D F2          BSR    PDATA1     C/R L/F
11400 E08C CE A00C        LDX    #XHI
11500 E08F 8D 37          BSR    OUT4HS     PRINT ADDRESS
11600 E091 FE A00C        LDX    XHI
11700 E094 8D 34          BSR    OUT2HS     PRINT DATA (OLD)
11800 E096 FF A00C        STX    XHI        SAVE DATA ADDRESS
11900 E099 8D DD          BSR    INCH       INPUT ONE CHAR
12000 E09B 81 20          CMP A  #$20
12100 E09D 26 E8          BNE    CHA51      NOT SPACE
12200 E09F 8D B4          BSR    BYTE       INPUT NEW DATA
12300 E0A1 09             DEX
12400 E0A2 A7 00          STA A  X          CHANGE MEMORY
12500 E0A4 A1 00          CMP A  X
12600 E0A6 27 DF          BEQ    CHA51      DID CHANGE
12700 E0A8 20 96          BRA    LOAD19     NOT CHANGED


13100                    * INPUT HEX CHAR
13200 E0AA 8D CC   INHEX  BSR    INCH
13300 E0AC 80 30          SUB A  #$30
13400 E0AE 2B 94          BMI    C1         NOT HEX
13500 E0B0 81 09          CMP A  #$09
13600 E0B2 2F 0A          BLE    IN1HG
13700 E0B4 81 11          CMP A  #$11
13800 E0B6 2B 8C          BMI    C1         NOT HEX
13900 E0B8 81 16          CMP A  #$16
14000 E0BA 2E 88          BGT    C1         NOT HEX
14100 E0BC 80 07          SUB A  #7
14200 E0BE 39        IN1HG RTS


14300 E0BF A6 00   OUT2H  LDA A  0,X        OUTPUT 2 HEX CHAR
14600 E0C1 8D A4   OUT2HA BSR    OUTHL      OUT LEFT HEX CHAR
14700 E0C3 A6 00          LDA A  0,X
14800 E0C5 08             INX
14900 E0C6 20 A3          BRA    OUTHR      OUTPUT RIGHT HEX CHAR AND R

15100 E0C8 8D F5   OUT4HS BSR    OUT2H      OUTPUT 4 HEX CHAR + SPACE
15200 E0CA 8D F3   OUT2HS BSR    OUT2H      OUTPUT 2 HEX CHAR + SPACE
```

```
15300 E0CC 86 20    OUTS    LDA A   #$20        SPACE
15400 E0CE 20 A5            BRA     OUTCH       (BSR & RTS)

15600                * ENTER POWER ON SEQUENCE
15700       E0D0     START   EQU     *
15800 E0D0 8E A042           LDS     #STACK
15900 E0D3 BF A008           STS     SP          INZ TARGET'S STACK PNTR
16000                * INZ PIA
16100 E0D6 CE 8004           LDX     #PIAD       (X) POINTER TO.DEVICE PIA
16200 E0D9 6C 00             INC     0,X         SET DATA DIR PIAD
16300 E0DB 86 07             LDA A   #$7         00000111
16400 E0DD A7 01             STA A   1,X         INIT CON PIAS
16500 E0DF 6C 00             INC     0,X         MARK COM LINE
16600 E0E1 A7 02             STA A   2,X         SET DATA DIR PIADB
16700 E0E3 86 34    CONTRL  LDA A   #$34        0011 0100
16800 E0E5 B7 8007           STA A   PIASB       SET CONTROL PIASB TURN READ
16900 E0E8 B7 8006           STA A   PIADB       SET TIMER INTERVAL
17000 E0EB 8E A042           LDS     #STACK      SET CONTRL STACK POINTER
17100 E0EE CE E19C           LDX     #MCLOFF

17300 E0F1 8D 8B             BSR     PDATA1      PRINT DATA STRING

17500 E0F3 8D 83             BSR     INCH        READ CHARACTER
17600 E0F5 16                TAB
17700 E0F6 8D D4             BSR     OUTS        PRINT SPACE
17800 E0F8 C1 4C             CMP B   #'L
17900 E0FA 26 03             BNE     *+5
18000 E0FC 7E E00A           JMP     LOAD
18100 E0FF C1 4D             CMP B   #'M
18200 E101 27 82             BEQ     CHANGE
18300 E103 C1 52             CMP B   #'R
18400 E105 27 18             BEQ     PRINT       STACK
18500 E107 C1 50             CMP B   #'P
18600 E109 27 32             BEQ     PUNCH       PRINT/PUNCH
18700 E10B C1 47             CMP B   #'G
18800 E10D 26 D4             BNE     CONTRL
18900 E10F BE A008           LDS     SP          RESTORE PGM'S STACK PTR
19000 E112 3B                RTI                 GO

19200                * ENTER FROM SOFTWARE INTERRUPT
19300       E113     SFE     EQU     *
19400 E113 BF A008           STS     SP          SAVE TARGET'S STACK POINTER
19500                * DECREMENT P-COUNTER
19600 E116 30                TSX
19700 E117 6D 06             TST     6,X
19800 E119 26 02             BNE     *+4
19900 E11B 6A 05             DEC     5,X
20000 E11D 6A 06             DEC     6,X

20200                * PRINT CONTENTS OF STACK
20300 E11F FE A008 PRINT     LDX     SP
20400 E122 08                INX
```

```
20500 E123 8D A5          BSR    OUT2HS   CONDITION CODES
20600 E125 8D A3          BSR    OUT2HS   ACC-B
20700 E127 8D A1          BSR    OUT2HS   ACC-A
20800 E129 8D 9D          BSR    OUT4HS   X-REG
20900 E12B 8D 9B          BSR    OUT4HS   P-COUNTER
21000 E12D CE A008        LDX    #SP
21100 E130 8D 96          BSR    OUT4HS   STACK POINTER
21200 E132 20 AF    C2    BRA    CONTRL


21400              *      PUNCH DUMP
21500              *      PUNCH FROM BEGINING ADDRESS (BEGA) THRU ENDI
21600              *      ADDRESS (ENDA)
21700              *

21900 E134 0D      MTAPE1 FCB    $D,$A,0,0,0,0,'S,'1,4   PUNCH FORMAT
      E135 0A
      E136 00
      E137 00
      E138 00
      E139 00
      E13A 53
      E13B 31
      E13C 04

22100      E13D    PUNCH  EQU    *

22300 E13D 86 12          LDA A  #$12      TURN TTY PUNCH ON
22400 E13F BD E075        JSR    OUTCH     OUT CHAR

22600 E142 FE A002        LDX    BEGA
22700 E145 FF A00F        STX    TW        TEMP BEGINING ADDRESS
22800 E148 B6 A005 PUN11  LDA A  ENDA+1
22900 E14B B0 A010        SUB A  TW+1
23000 E14E F6 A004        LDA B  ENDA
23100 E151 F2 A00F        SBC B  TW
23200 E154 26 04          BNE    PUN22
23300 E156 81 10          CMP A  #16
23400 E158 25 02          BCS    PUN23
23500 E15A 86 0F    PUN22 LDA A  #15
23600 E15C 8B 04    PUN23 ADD A  #4
23700 E15E B7 A011        STA A  MCONT     FRAME COUNT THIS RECORD
23800 E161 80 03          SUB A  #3
23900 E163 B7 A00E        STA A  TEMP      BYTE COUNT THIS RECORD
24000              *      PUNCH C/R,L/F,NULL,S,1
24100 E166 CE E134        LDX    #MTAPE1
24200 E169 BD E07E        JSR    PDATA1
24300 E16C 5F             CLR B            ZERO CHECKSUM
24400              *      PUNCH FRAME COUNT
24500 E16D CE A011        LDX    #MCONT
24600 E170 8D 25          BSR    PUNT2     PUNCH 2 HEX CHAR
24700              *      PUNCH ADDRESS
```

```
24800 E172 CE A00F          LDX     #TW
24900 E175 8D 20            BSR     PUNT2
25000 E177 8D 1E            BSR     PUNT2
25100                 *     PUNCH DATA
25200 E179 FE A00F          LDX     TW
25300 E17C 8D 19    PUN32   BSR     PUNT2     PUNCH ONE BYTE (2 FRAMES)
25400 E17E 7A A00E          DEC     TEMP      DEC BYTE COUNT
25500 E181 26 F9            BNE     PUN32
25600 E183 FF A00F          STX     TW
25700 E186 53               COM B
25800 E187 37               PSH B
25900 E188 30               TSX
26000 E189 8D 0C            BSR     PUNT2     PUNCH CHECKSUM
26100 E18B 33               PUL B             RESTORE STACK
26200 E18C FE A00F          LDX     TW
26300 E18F 09               DEX
26400 E190 BC A004          CPX     ENDA
26500 E193 26 B3            BNE     PUN11
26600 E195 20 9B            BRA     C2        JMP TO CONTRL



26800                 *     PUNCH 2 HEX CHAR, UPDATE CHECKSUM
26900 E197 EB 00    PUNT2   ADD B   0,X       UPDATE CHECKSUM
27000 E199 7E E0BF          JMP     OUT2H     OUTPUT TWO HEX CHAR AND RTS


27020 E19C 13       MCLOFF  FCB     $13       READER OFF
27100 E19D 00       MCL     FCB     $D,$A,$14,0,0,0,'*,4   C/R,L/F,PUNCH
      E19E 0A
      E19F 14
      E1A0 00
      E1A1 00
      E1A2 00
      E1A3 2A
      E1A4 04
27200                 *
27300 E1A5 FF A012  SAV     STX     XTEMP
27400 E1A8 CE 8004          LDX     #PIAD
27500 E1AB 39               RTS
27600                 * INPUT ONE CHAR INTO A-REGISTER
27700 E1AC 37       INEEE   PSH B             SAVE ACC-B
27800 E1AD 8D F6            BSR     SAV       SAV XR
27900 E1AF A6 00    IN1     LDA A   0,X       LOOK FOR START BIT
28000 E1B1 2B FC            BMI     IN1
28100 E1B3 6F 02            CLR     2,X       SET COUNTER FOR HALF BIT TI
28200 E1B5 8D 3C            BSR     DE        START TIMER
28300 E1B7 8D 36            BSR     DEL       DELAY HALF BIT TIME
28400 E1B9 C6 04            LDA B   #4        SET DEL FOR FULL BIT TIME
28500 E1BB E7 02            STA B   2,X
28600 E1BD 58               ASL B             SET UP CNTR WITH 8
```

```
28700 E1BE 8D 2F    IN3     BSR     DEL         WAIT ONE CHAR TIME
28800 E1C0 0D               SEC                 MARK COM LINE
28900 E1C1 69 00            ROL     0,X         GET BIT INTO CFF
29000 E1C3 46               ROR A               CFF TO AR
29100 E1C4 5A               DEC B
29200 E1C5 26 F7            BNE     IN3
29300 E1C7 8D 26            BSR     DEL         WAIT FOR STOP BIT
29400 E1C9 84 7F            AND A   #$7F        RESET PARITY BIT
29500 E1CB 81 7F            CMP A   #$7F
29600 E1CD 27 E0            BEQ     IN1         IF RUBOUT,GET NEXT CHAR
29700 E1CF 20 12            BRA     IOUT2       GO RESTORE REG


29900                   * OUTPUT ONE CHAR
30000 E1D1 37       OUTEEE PSH B                SAV BR
30100 E1D2 8D D1            BSR     SAV         SAV XR
30200 E1D4 C6 0A    IOUT    LDA B   #$A         SET UP COUNTER
30300 E1D6 6A 00            DEC     0,X         SET START BIT
30400 E1D8 8D 19            BSR     DE          START TIMER
30500                   *
30600 E1DA 8D 13    OUT1    BSR     DEL         DELAY ONE BIT TIME
30700 E1DC A7 00            STA A   0,X         PUT OUT ONE DATA BIT
30800 E1DE 0D               SEC                 SET CARRY BIT
30900 E1DF 46               ROR A               SHIFT IN NEXT BIT
31000 E1E0 5A               DEC B               DECREMENT COUNTER
31100 E1E1 26 F7            BNE     OUT1        TEST FOR 0
31200 E1E3 E6 02    IOUT2   LDA B   2,X         TEST FOR STOP BITS
31300 E1E5 58               ASL B               SHIFT BIT TO SIGN
31400 E1E6 2A 02            BPL     IOS         BRANCH FOR 1 STOP BIT
31500 E1E8 8D 05            BSR     DEL         DELAY FOR STOP BITS
31600 E1EA FE A012  IOS     LDX     XTEMP       RES XR
31700 E1ED 33               PUL B               RESTORE BR
31800 E1EE 39               RTS
31900                   *
32000 E1EF 6D 02    DEL     TST     2,X         IS TIME UP
32100 E1F1 2A FC            BPL     DEL
32200 E1F3 6C 02    DE      INC     2,X         RESET TIMER
32300 E1F5 6A 02            DEC     2,X
32400 E1F7 39               RTS

32600 E1F8 E000             FDB     IO
32700 E1FA E113             FDB     SFE
32800 E1FC E005             FDB     POWDWN
32900 E1FE E0D0             FDB     START
33000 A000                  ORG     $A000
33100 A000 0002     IOV     RMB     2           IO INTERRUPT POINTER
33200 A002 0002     BEGA    RMB     2           BEGINING ADDR PRINT/PUNCH
33300 A004 0002     ENDA    RMB     2           ENDING ADDR PRINT/PUNCH
33400 A006 0002     NIO     RMB     2           NMI INTERRUPT POINTER
33500 A008 0001     SP      RMB     1           S-HIGH
33600 A009 0001             RMB     1           S-LOW
33700 A00A 0001     CKSM    RMB     1           CHECKSUM
```

```
33800 A008 0001     BYTECT RMB     1          BYTE COUNT
33900 A00C 0001     XHI    RMB     1          XREG HIGH
34000 A00D 0001     XLOW   RMB     1          XREG LOW
34100 A00E 0001     TEMP   RMB     1          CHAR COUNT (INADD)
34200 A00F 0002     TW     RMB     2          TEMP/
34300 A011 0001     MCONT  RMB     1          TEMP
34400 A012 0002     XTEMP  RMB     2          X-REG TEMP STORAGE
34500 A014 002E            RMB     46
34600 A042 0001     STACK  RMB     1          STACK POINTER


35000                          END

SYMBOL TABLE

PIA5B  8007  PIADB  8006  PIA5   8005  PIAD   8004  IO     E000
POWDWN E005  LOAD   E00A  LOAD3  E013  LOAD11 E02F  LOAD15 E03B
LOAD19 E040  LOAD21 E044  C1     E044  BADDR  E047  BYTE   E055
OUTHL  E067  OUTHR  E06B  OUTCH  E075  INCH   E078  PDATA2 E07B
PDATA1 E07E  CHANGE E085  CHA51  E087  INHEX  E0AA  IN1HG  E0BE
OUT2H  E0BF  OUT2HA E0C1  OUT4HS E0C8  OUT2HS E0CA  OUTS   E0CC
START  E0D0  CONTRL E0E3  SFE    E113  PRINT  E11F  C2     E132
MTAPE1 E134  PUNCH  E13D  PUN11  E148  PUN22  E15A  PUN23  E15C
PUN32  E17C  PUNT2  E197  MCLOFF E19C  MCL    E19D  SAV    E1A5
INEEE  E1AC  IN1    E1AF  IN3    E1BE  OUTEEE E1D1  IOUT   E1D4
OUT1   E1DA  IOUT2  E1E3  IOS    E1EA  DEL    E1EF  DE     E1F3
IOV    A000  BEGA   A002  ENDA   A004  NIO    A006  SP     A008
CKSM   A00A  BYTECT A00B  XHI    A00C  XLOW   A00D  TEMP   A00E
TW     A00F  MCONT  A011  XTEMP  A012  STACK  A042
```

6.0 MINIBUG REV. 4 PROGRAM LISTING

```
00100                            NAM    MINIB
00110                  * MINI-BUG
00120                  * COPYWRITE 1973, MOTOROLA INC
00140                  * REV 004 (USED WITH MIKBUG)
00180       FCF4       ACIACS EQU    @176364   ACIA CONTROL/STATUS
00190       FCF5       ACIADA EQU    ACIACS+1
00200 FE00             ORG    $FE00
00210                  * MINIB
00220                  * INPUT ONE CHAR INTO A-REGISTER
00230 FE00 B6 FCF4 INCH   LDA A  ACIACS
00240 FE03 47             ASR A
00250 FE04 24 FA          BCC    INCH      RECEIVE NOT READY
00260 FE06 B6 FCF5        LDA A  ACIADA    INPUT CHARACTER
00270 FE09 84 7F          AND A  #$7F      RESET PARITY BIT
00280 FE0B 81 7F          CMP A  #$7F
00290 FE0D 27 F1          BEQ    INCH      RUBOUT; IGNORE
00300 FE0F 7E FEAE        JMP    OUTCH     ECHO CHAR

00320                  * INPUT HEX CHAR
00330 FE12 8D EC   INHEX  BSR    INCH
00340 FE14 81 30          CMP A  #$30
00350 FE16 2B 52          BMI    C1        NOT HEX
00360 FE18 81 39          CMP A  #$39
00370 FE1A 2F 0A          BLE    IN1HG
00380 FE1C 81 41          CMP A  #$41
00390 FE1E 2B 4A          BMI    C1        NOT HEX
00400 FE20 81 46          CMP A  #$46
00410 FE22 2E 46          BGT    C1        NOT HEX
00420 FE24 80 07          SUB A  #7
00430 FE26 39      IN1HG  RTS

00450 FE27 86 D1   LOAD   LDA A  #$D1      TURN READER ON
00460 FE29 B7 FCF4        STA A  ACIACS
00470 FE2C 86 11          LDA A  #@21
00480 FE2E 8D 7E          BSR    OUTCH

00500 FE30 8D CE   LOAD3  BSR    INCH
00510 FE32 81 53          CMP A  #'S
00520 FE34 26 FA          BNE    LOAD3     1ST CHAR NOT (S)
00530 FE36 8D C8          BSR    INCH      READ CHAR
00540 FE38 81 39          CMP A  #'9
00550 FE3A 27 25          BEQ    LOAD21
00560 FE3C 81 31          CMP A  #'1
00570 FE3E 26 F0          BNE    LOAD3     2ND CHAR NOT (1)
00580 FE40 7F FF32        CLR    CKSM      ZERO CHECKSUM
00590 FE43 8D 36          BSR    BYTE      READ BYTE
00600 FE45 80 02          SUB A  #2
00610 FE47 B7 FF33        STA A  BYTECT    BYTE COUNT
00620                  * BUILD ADDRESS
00630 FE4A 8D 21          BSR    BADDR
00640                  * STORE DATA
00650 FE4C 8D 2D   LOAD11 BSR    BYTE
00660 FE4E 7A FF33        DEC    BYTECT
```

```
00670 FE51 27 05          BEQ    LOAD15   ZERO BYTE COUNT
00680 FE53 A7 00          STA A  X        STORE DATA
00690 FE55 08             INX
00700 FE56 20 F4          BRA    LOAD11

00720 FE58 7C FF32 LOAD15 INC    CKSM
00730 FE5B 27 D3          BEQ    LOAD3
00740 FE5D 86 3F   LOAD19 LDA A  #'?      PRINT QUESTION MARK
00750 FE5F 8D 4D          BSR    OUTCH
00760 FE61 86 B1   LOAD21 LDA A  #$B1     TURN READER OFF
00770 FE63 B7 FCF4        STA A  ACIACS
00780 FE66 86 13          LDA A  #@23
00790 FE68 8D 44          BSR    OUTCH
00800 FE6A 7E FEDB C1     JMP    CONTRL


00820                * BUILD ADDRESS
00830 FE6D 8D 0C   BADDR  BSR    BYTE     READ 2 FRAMES
00840 FE6F B7 FF34        STA A  XHI
00850 FE72 8D 07          BSR    BYTE
00860 FE74 B7 FF35        STA A  XLOW
00870 FE77 FE FF34        LDX    XHI      (X) ADDRESS WE BUILT
00880 FE7A 39             RTS


00900                * INPUT BYTE (TWO FRAMES)
00910 FE7B 8D 95   BYTE   BSR    INHEX    GET HEX CHAR
00920 FE7D 48             ASL A
00930 FE7E 48             ASL A
00940 FE7F 48             ASL A
00950 FE80 48             ASL A
00960 FE81 16             TAB
00970 FE82 8D 8E          BSR    INHEX
00980 FE84 84 0F          AND A  #$0F      MASK TO 4 BITS
00990 FE86 1B             ABA
01000 FE87 16             TAB
01010 FE88 FB FF32        ADD B  CKSM
01020 FE8B F7 FF32        STA B  CKSM
01030 FE8E 39             RTS


01050                * CHANGE MEMORY (M AAAA DD NN)
01060 FE8F 8D DC   CHANGE BSR    BADDR    BUILD ADDRESS
01070 FE91 8D 34          BSR    OUTS     PRINT SPACE
01080 FE93 8D 30          BSR    OUT2HS
01090 FE95 8D E4          BSR    BYTE
01100 FE97 09             DEX
01110 FE98 A7 00          STA A  X
01120 FE9A A1 00          CMP A  X
01130 FE9C 26 BF          BNE    LOAD19   MEMORY DID NOT CHANGE
01140 FE9E 20 3B          BRA    CONTRL


01160 FEA0 44      OUTHL  LSR A            OUT HEX LEFT BCD DIGIT
01170 FEA1 44             LSR A
```

```
01180 FEA2 44              LSR A
01190 FEA3 44              LSR A

01210 FEA4 84 0F    OUTHR  AND A  #$F       OUT HEX RIGHT BCD DIGIT
01220 FEA6 8B 30           ADD A  #$30
01230 FEA8 81 39           CMP A  #$39
01240 FEAA 23 02           BLS    OUTCH
01250 FEAC 8B 07           ADD A  #$7

01270                 * OUTPUT ONE CHAR
01280 FEAE 37       OUTCH  PSH B           SAVE B-REG
01290 FEAF F6 FCF4  OUTC1  LDA B  ACIACS
01300 FEB2 57              ASR B
01310 FEB3 57              ASR B
01320 FEB4 24 F9           BCC    OUTC1   XMIT NOT READY
01330 FEB6 B7 FCF5         STA A  ACIADA  OUTPUT CHARACTER
01340 FEB9 33              PUL B           RESTORE B-REG
01350 FEBA 39              RTS

01370 FEBB A6 00    OUT2H  LDA A  0,X      OUTPUT 2 HEX CHAR
01380 FEBD 8D E1           BSR    OUTHL   OUT LEFT HEX CHAR
01390 FEBF A6 00           LDA A  0,X
01400 FEC1 8D E1           BSR    OUTHR   OUT RIGHT HEX CHAR
01410 FEC3 08              INX
01420 FEC4 39              RTS


01450 FEC5 8D F4    OUT2HS BSR    OUT2H   OUTPUT 2 HEX CHAR + SPACE
01460 FEC7 86 20    OUTS   LDA A  #$20    SPACE
01470 FEC9 20 E3           BRA    OUTCH   (BSR & RTS)


01500                 * PRINT CONTENTS OF STACK
01510 FECB 30       PRINT  TSX
01520 FECC FF FF30         STX    SP      SAVE STACK POINTER
01530 FECF C6 09           LDA B  #9
01540 FED1 8D F2    PRINT2 BSR    OUT2HS  OUT 2 HEX & SPACE
01550 FED3 5A              DEC B
01560 FED4 26 FB           BNE    PRINT2


01590                 * ENTER POWER ON SEQUENCE
01600       FED6    START  EQU    *
01610                 * INZ ACIA
01620 FED6 86 B1           LDA A  #$B1     SET SYSTEM PARAMETERS
01630 FED8 B7 FCF4         STA A  ACIACS

01650 FEDB 8E FF28  CONTRL LDS    #STACK  SET STACK POINTER
01660 FEDE 86 0D           LDA A  #$D      CARRIAGE RETURN
```

```
01670 FEE0 8D CC          BSR     OUTCH
01680 FEE2 86 0A          LDA A   #$A       LINE FEED
01690 FEE4 8D C8          BSR     OUTCH

01710 FEE6 BD FE00        JSR     INCH      READ CHARACTER
01720 FEE9 16             TAB
01730 FEEA 8D DB          BSR     OUTS      PRINT SPACE
01740 FEEC C1 4C          CMP B   #'L
01750 FEEE 26 03          BNE     *+5
01760 FEF0 7E FE27        JMP     LOAD
01770 FEF3 C1 4D          CMP B   #'M
01780 FEF5 27 98          BEQ     CHANGE
01790 FEF7 C1 50          CMP B   #'P
01800 FEF9 27 D0          BEQ     PRINT     STACK
01810 FEFB C1 47          CMP B   #'G
01820 FEFD 26 DC          BNE     CONTRL
01830 FEFF 3B             RTI               GO
```

```
01860 FF00                ORG     $FF00
01870 FF00 0028           RMB     40
01880 FF28 0001     STACK RMB     1         STACK POINTER
01890               * REGISTERS FOR GO
01900 FF29 0001           RMB     1         CONDITION CODES
01910 FF2A 0001           RMB     1         B ACCUMULATOR
01920 FF2B 0001           RMB     1         A
01930 FF2C 0001           RMB     1         X-HIGH
01940 FF2D 0001           RMB     1         X-LOW
01950 FF2E 0001           RMB     1         P-HIGH
01960 FF2F 0001           RMB     1         P-LOW
01970 FF30 0001     SP    RMB     1         S-HIGH
01980 FF31 0001           RMB     1         S-LOW
01990               * END REGISTERS FOR GO
02000 FF32 0001     CKSM  RMB     1         CHECKSUM
02010 FF33 0001     BYTECT RMB    1         BYTE COUNT
02020 FF34 0001     XHI   RMB     1         XREG HIGH
02030 FF35 0001     XLOW  RMB     1         XREG LOW
02070               END
```

SYMBOL TABLE

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

1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0

MOTOROLA Semiconductor Products Inc.

BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

9789-1 PRINTED IN USA 6-75 IMPERIAL LITHO B51001 10M

EN-100