



# OPERATOR'S MANUAL

## FOR THE EM-800 DIAGNOSTIC EMULATOR NSC800™ SERIES MICROPROCESSOR

920-11484-00  
February, 1985

 **Applied  
Microsystems**  
CORPORATION

5020 - 148th Avenue N.E.  
P.O. Box C-1002  
Redmond, WA 98073  
(206) 882-2000  
Toll Free Service: 1-800-426-3925

*Copyright © 1983 Applied Microsystems Corporation. All rights reserved.*  
Diagnostic Emulator is a Trademark of Applied Microsystems Corporation.  
NSC800™ is a Trademark of National Semiconductor Corporation

---

*Applied Microsystems Corporation has made every effort to document this product accurately and completely. However, Applied Microsystems assumes no liability for errors or for any damages that result from use of this manual or the equipment it accompanies. Applied Microsystems reserves the right to make changes to this manual without notice at any time.*

*Because this configuration of the EM-800 Diagnostic Emulator is intended for use in developing, debugging, and testing NSC800™ microprocessor-based systems, it is assumed that the user is familiar with the terminology of the NSC800™ microprocessor.*

**WARNING** — *This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.*

---

# TABLE OF CONTENTS

---

## I. INTRODUCTION

1.1	SYSTEM CONCEPT	1-1
1.2	TRANSPARENCY	1-2
1.3	WARRANTY	1-2
1.4	GENERAL SPECIFICATIONS	1-3

## II. EM-800 COMPONENTS

2.1	OPERATOR'S STATION	2-2
2.2	EMULATOR PROBE	2-2
2.3	KEYBOARD	2-2
2.4	DIAGNOSTIC EPROM SOCKET	2-2
2.5	DISPLAY PANEL	2-3
2.6	TRACE MEMORY	2-6
2.7	BACK PANEL CONTROLS AND CONNECTORS	2-7
2.8	RAM OVERLAY	2-7
2.9	DISASSEMBLER	2-7

## III. BASIC OPERATING INSTRUCTIONS

3.1	OPERATING VOLTAGE	3-2
3.2	SAFETY INFORMATION	3-2
3.3	CONNECTION TO TARGET EQUIPMENT	3-2

## IV. EM-800 FUNCTIONS

4.1	EXECUTION CONTROL	4-2
4.1.0	RESET Keyswitch	4-2
4.1.1	RUN Keyswitch	4-2
4.1.2	RUN BKPT Keyswitch	4-3
4.1.3	STEP Keyswitch	4-3
4.1.4	Breakpoint System	4-6
4.1.5	Trace Memory	4-10
4.2	EXAMINATION AND ALTERATION OF CPU REGISTERS	4-11
4.3	EXAMINATION AND ALTERATION OF MEMORY LOCATIONS	4-13
4.4	EXAMINATION AND ALTERATION OF I/O PORTS	4-15

## V. RAM OVERLAY

5.1	OVERVIEW	5-2
5.2	INSTALLATION	5-4
5.3	CONTROLS	5-5
5.4	UPLOADING/DOWNLOADING	5-7

---

## VI. TRACE DISASSEMBLY

6.1	OVERVIEW	6-2
	6.6.1 Operation Preparation Procedures	6-2
6.2	FORMAT DEFINITION	6-3
6.3	LINE ASSEMBLER	6-9
	6.3.1 How to Use the Line Assembler	6-10
	6.3.2 Features Supported	6-10
6.4	MEMORY DISASSEMBLER	6-12
	6.4.1 Overview	6-12
	6.4.2 How to Use the Disassembler	6-12
	6.4.3 Disassembler Output Format	6-13
	6.4.4 Errors	6-13

## VII. BUILT-IN DIAGNOSTICS (CODE FUNCTIONS)

7.1	GROUP A: MEMORY TESTS	7-4
7.2	GROUP B: OSCILLOSCOPE LOOPS	7-7
7.3	GROUP C: MEMORY LOAD AND DUMP	7-9
7.4	GROUP D: MISCELLANEOUS	7-15
7.5	GROUP E: CHANGE DEFAULT PARAMETERS	7-18
7.6	GROUP F: INTROSPECTION MODE	7-19

## VIII. USER-IMPLEMENTED CODE FUNCTIONS

8.1	OVERVIEW	8-2
8.2	INTERNAL ENVIRONMENT	8-3
	8.2.1 ROM	8-4
	8.2.2 Front Panel EPROM Socket	8-4
	8.2.3 Scratchpad RAM	8-4
	8.2.4 I/O Devices	8-6
8.3	ENTRY TO USER CODE FUNCTIONS	8-15
8.4	INTROSPECTION MODE	8-16
	8.4.1 Code F	8-16
8.5	GETTING TO AND FROM THE TARGET SYSTEM	8-16
	8.5.1 Examine and Store	8-17
	8.5.2 Pause to Run	8-18
	8.5.3 Run to Pause	8-18
	8.5.4 Re-Entry Jump	8-19
8.6	USER-ACCESSIBLE SUBROUTINES	8-21
8.7	INTERRUPTS	8-25
8.8	CODE FUNCTION EXAMPLE	8-25

## IX. SUPPLEMENTARY INFORMATION

9.1	AUXILIARY CONNECTOR	9-2
9.2	OPTION SWITCHES	9-4
9.3	SERIAL INTERFACE	9-6
9.4	UPLOAD/DOWNLOAD PROTOCOL	9-2
9.5	EXTERNAL BREAKPOINT	9-7
9.6	TRACE HOLD	9-6
	9.6.1 Window Mode	9-11
	9.6.2 Selective Trace	9-11
9.7	SIGNATURE ANALYSIS	9-12
9.8	SOFT SHUTDOWN	9-14



---

## X. MAINTENANCE & TROUBLESHOOTING

10.1	MAINTENANCE	10-2
10.1.1	Power Supply	10-2
10.1.2	Cables	10-2
10.1.3	Probe Tip Assembly	10-3

### APPENDIX A.

NULL TARGET — A SOFTWARE SIMULATION TOOL	A-1
--	-----

### APPENDIX B.

SYSTEM ERROR CODES	B-1
--------------------	-----

### INDEX



---

# LIST OF FIGURES

---

2.1-1	Operator's Station	2-3
2.7-1	Back Panel	2-4
3.3-1	Installing 40-Pin Plug	3-2
5.1-1	RAM Overlay Decals	5-3
5.2-1	RAM Overlay Installation	5-5
5.3-1	RAM Overlay Controls	5-6
6.2-1	Code E1 80-Character Single-Line Disassembly Format	6-6
6.2-2	Code E1 72-Character Single-Line Disassembly Format	6-7
6.2-3	Code E1 72-Character Two-Line Disassembly Format	6-8
8.2-1	EM-800 Internal Memory Map	8-3
8.2-2	Map of Internal Scratchpad RAM	8-5
8.2-3	Keyboard Input Locations	8-7
8.2-4	Serial Port Data and Status Locations	8-7
8.2-4	Trace Memory Format	8-11
9.1-1	J3-Auxiliary Connector Pinout (D-Subminiature, Female)	9-2
9.2-1	Option Jumpers	9-5
9.3-1	Serial Word Format	9-7
9.5-1	Timing Relationships	9-9
9.6-1	Trace Hold and Timing	9-10
9.6-2	Window Mode Circuit	9-11
9.7-1	Simplified Microprocessor Diagram	9-13



---

# LIST OF TABLES

---

4.1-1	Display Panel Indicators	4-4
4.1-2	Breakpoint Qualifiers	4-6
4.2-1	Keyboard Designators	4-13
5-1	Memory Block Address A and B	5-7
6-1	Disassembly Format Selection	6-3
7.1-1	Code Functions	7-2
7.3-1	C3 and C7 Code Function Error Codes	7-11
7.3-2	Memory Dump Format	7-14
8.6-1	User-Accessible Subroutines	8-21
9.2-1	Set-Up Characteristics for Serial Port	9-4
10-2	Troubleshooting	10-3



---

## **EDITION NOTE:**

This is the first edition of this EM-800 Operator's Manual. Updates and revisions will be announced on this page in future editions.

There is one change in manual syntax that you may notice if you are using other documentation from Applied Microsystems Corporation. The obsolete term "carriage return" (abbreviated <cr>) has been replaced as follows:

in text: RETURN

in examples: <return>

As other documentation is reissued, it will be revised to reflect this change.





---

# **SECTION 1 INTRODUCTION**

---

<b>1.1</b>	<b>SYSTEM CONCEPT</b>
<b>1.2</b>	<b>TRANSPARENCY</b>
<b>1.3</b>	<b>WARRANTY</b>
<b>1.4</b>	<b>GENERAL SPECIFICATIONS</b>

---

---

## 1.1 SYSTEM CONCEPT

The EM-800 Diagnostic Emulator is a microprocessor test and diagnosis instrument designed to emulate the NSC800™ microprocessor. The Diagnostic Emulator consists of an Operator's Station with keyboard and display panel, and an emulator's pod and cables for connection to your system. The EM-800 is fast and easy to use and includes many diagnostic capabilities for troubleshooting problems in your system.

## 1.2 TRANSPARENCY

The EM-800 Diagnostic Emulator is transparent to the normal operation of the target system in that emulation is in real-time, with no additional processor cycles required as a result of the emulation process. There are no target system addresses or I/O ports needed or used by the EM-800, and there are no programs or other software objects that are required to be in the target address space. As a consequence of this transparency, you should not experience difficulties in using the EM-800 Diagnostic Emulator with your system, even if there are critical software timing constraints in the system.

## 1.3 LIMITED WARRANTY

Applied Microsystems Corporation warrants that the equipment accompanying this document is free from defects in materials and workmanship, and will perform to the applicable published Applied Microsystems Corporation specifications for one year from the date of shipment. THIS WARRANTY IS IN LIEU OF, AND REPLACES ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING THE WARRANTY OF MERCHANTABILITY AND THE WARRANTY OF FITNESS FOR PARTICULAR PURPOSE. In no event will Applied Microsystems Corporation be liable for special or consequential damages as a result of a breach of this warranty provision. The liability of Applied Microsystems Corporation shall be limited to replacing or repairing at its option any defective unit which is returned F.O.B. to Applied Microsystems Corporation's manufacturing plant. Equipment or parts which, in Applied Microsystems Corporation's opinion, have been subjected to abuse, misuse, accident, alteration, neglect, unauthorized repair, or improper installation are not covered by this warranty. Applied Microsystems Corporation shall have the right to determine the existence and cause of any defect. When items are repaired or replaced, this warranty will remain in effect for the balance of the warranty period or 90 days following the date of shipment, whichever period is longer.

## 1.4 GENERAL SPECIFICATIONS

### Input Power

90 to 140 Vac  
60 Hz  
less than 50 watts

### Optional

180 to 280 Vac  
50 Hz  
less than 50 watts

### Physical:

#### Operator's Station

Width: 292 mm (11.5 inches)  
Height: 117 mm (4.6 inches)  
Depth: 356 mm (14 inches)

#### Target System Connection (Ribbon Cable)

Total Length (including Pod):  
1.5M (58 inches)

#### Emulator Cable Pod

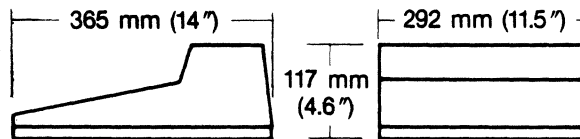
Length: 173 mm (6.8 inches)  
Width: 90 mm (3.6 inches)  
Depth: 33 mm (1.3 inches)

**Total Weight:** 4.5 Kg (11 lbs);  
Shipping 6.3 Kg (14 lbs)

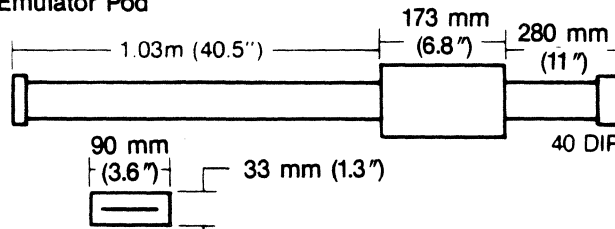
### Environmental:

Operating Temperature: 0°C to 40°C (32°F to 104°F)  
Storage Temperature: -40°C to 70°C (-40°F to 158°F)  
Humidity: 5% to 95% RH non-condensing

#### Operator's Station



#### Emulator Pod





---

# SECTION 2

## EM-800 COMPONENTS

---

- 2.1 OPERATOR'S STATION
  - 2.2 EMULATOR PROBE
  - 2.3 KEYBOARD
  - 2.4 DIAGNOSTIC EPROM SOCKET
  - 2.5 DISPLAY PANEL
  - 2.6 TRACE MEMORY
  - 2.7 BACK PANEL CONTROLS  
AND CONNECTORS
    - Main Power Switch • Baud Rate Selector*
    - Switch • Auxiliary Connector • Option Switches*
    - RAM Overlay Bank A and Bank B Address Switches*
    - RAM Overlay Bank Enable Switches*
  - 2.8 RAM OVERLAY
  - 2.9 DISASSEMBLER
-

## 2.1 OPERATOR'S STATION

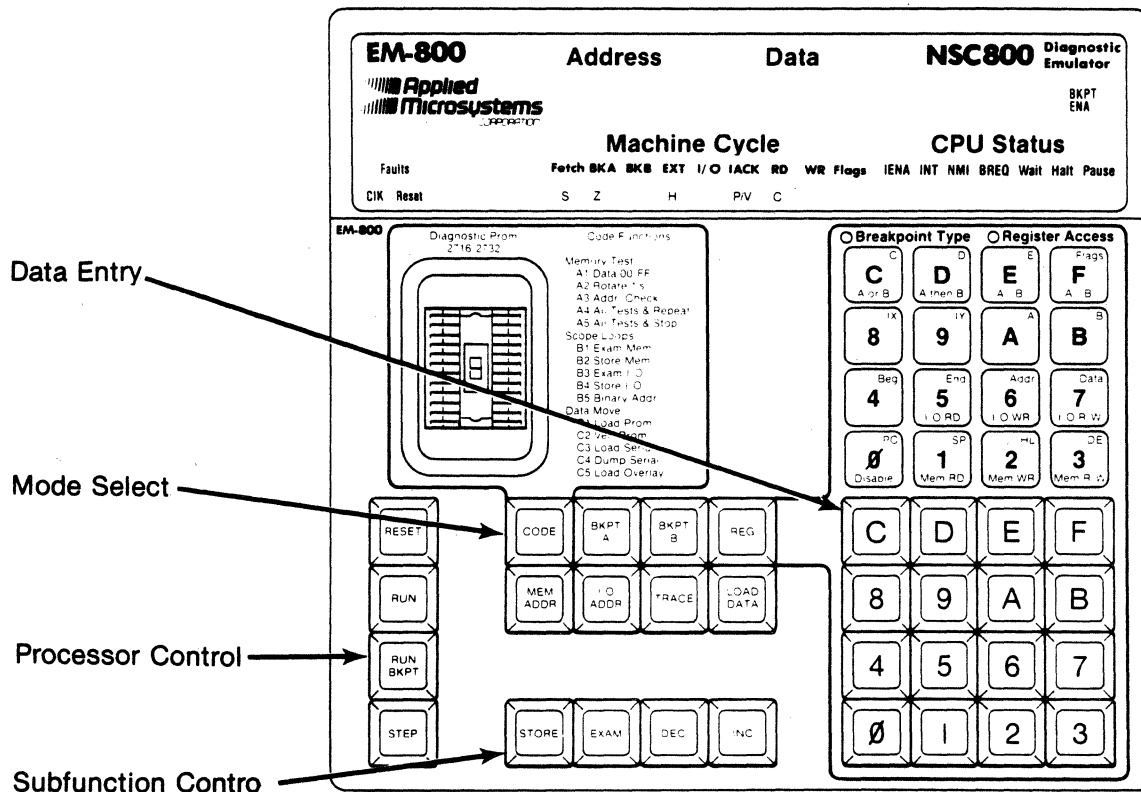
The EM-800 Operator's Station consists of a keyboard, display panel, diagnostic EPROM socket, back panel controls, and connectors. The operator's station contains most of the system electronics, including the emulation control circuitry, Trace Memory, Breakpoint comparators, plus control firmware with preprogrammed test routines. A RAM Overlay option may be included. See Figure 2-1.1.

## 2.2 EMULATOR PROBE

The EM-800 is the emulator probe for the NSC800 microprocessor. The probe contains the CPU and associated circuitry and buffers. It connects to the Operator's Station via 40-inch ribbon cables and to the target system CPU socket via 11-inch ribbon cables and a 40-contact DIP connector.

## 2.3 KEYBOARD

The keyboard has 32 keyswitches divided into four groupings: Processor Control, Mode Select, Subfunction Control and Data Entry.



## 2.4 DIAGNOSTIC EPROM SOCKET

A low-insertion force EPROM socket to accept EPROMs compatible with Intel 2716 or 2732 types (single +5 power supply and Intel pinout). You may create your own system test and diagnosis routines, program the EPROM with these routines, insert the EPROM into the EM-800 front panel socket and then execute the routines in a convenient manner from the EM-800 keyboard. See Section 8: USER-IMPLEMENTED CODE FUNCTIONS.

## 2.5 DISPLAY PANEL

The display panel consists of LED dot-matrix address and data displays and of individual LED indicators. Address and data information are displayed in hexadecimal notation. The individual indicator LEDs are divided into five groupings:

- Fault indicators (CLK.RESET) show loss of system clock or a continuous RESET condition.
- Machine Cycle indicators (Fetch. BK A. BK B. EXT. I/O. IACK. RD. WR) read out the control bus and other information acquired during target program execution.
- The microprocessor condition code bits (S. Z. H. P/V. C) are also displayed on the machine cycle indicators.
- CPU Status indicators (IENA. INT. Trap. NMI. BREQ. Halt. Pause) show the condition of the emulated target system CPU.
- Breakpoint Enable (BKPT ENA) is illuminated if the breakpoint System is enabled.

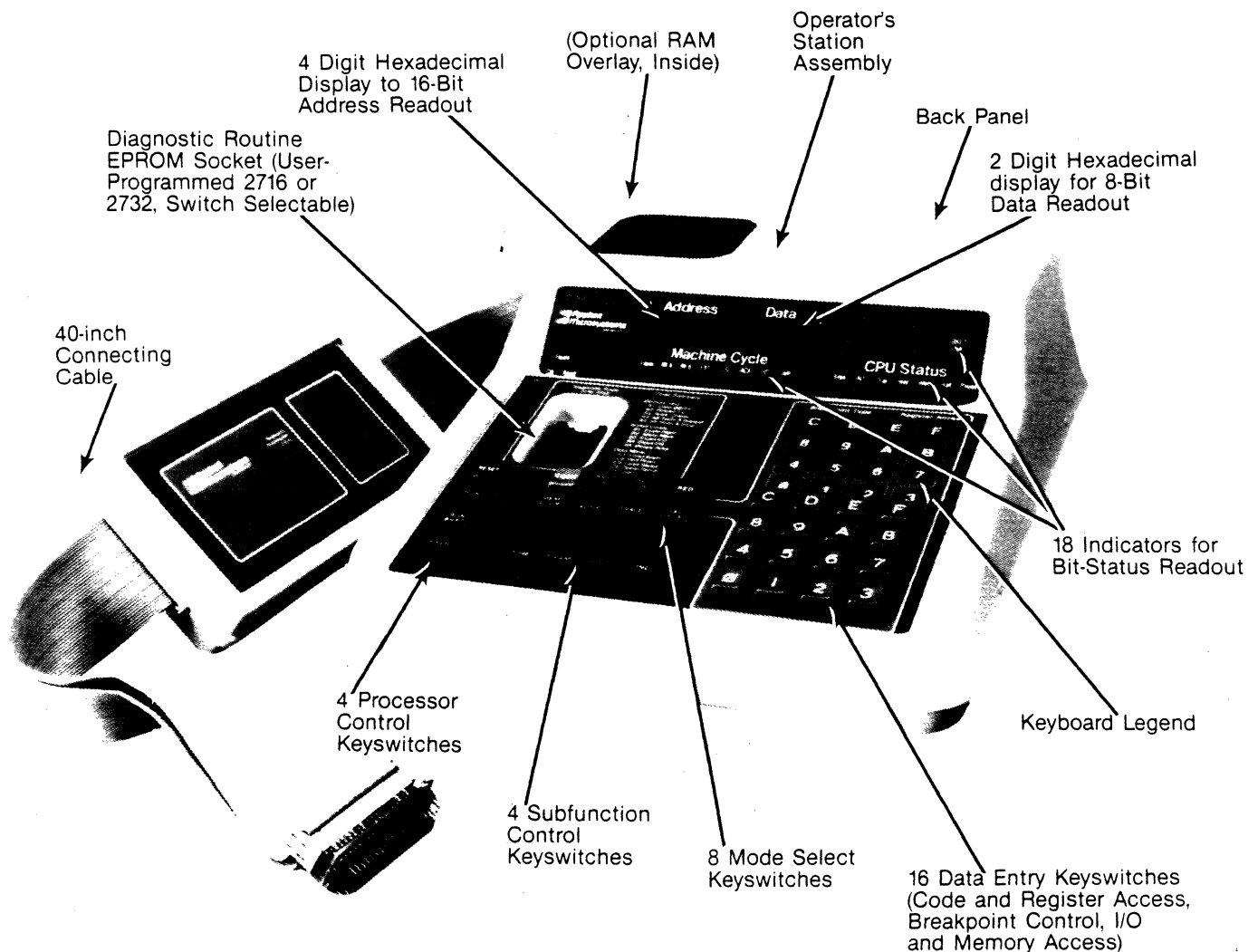
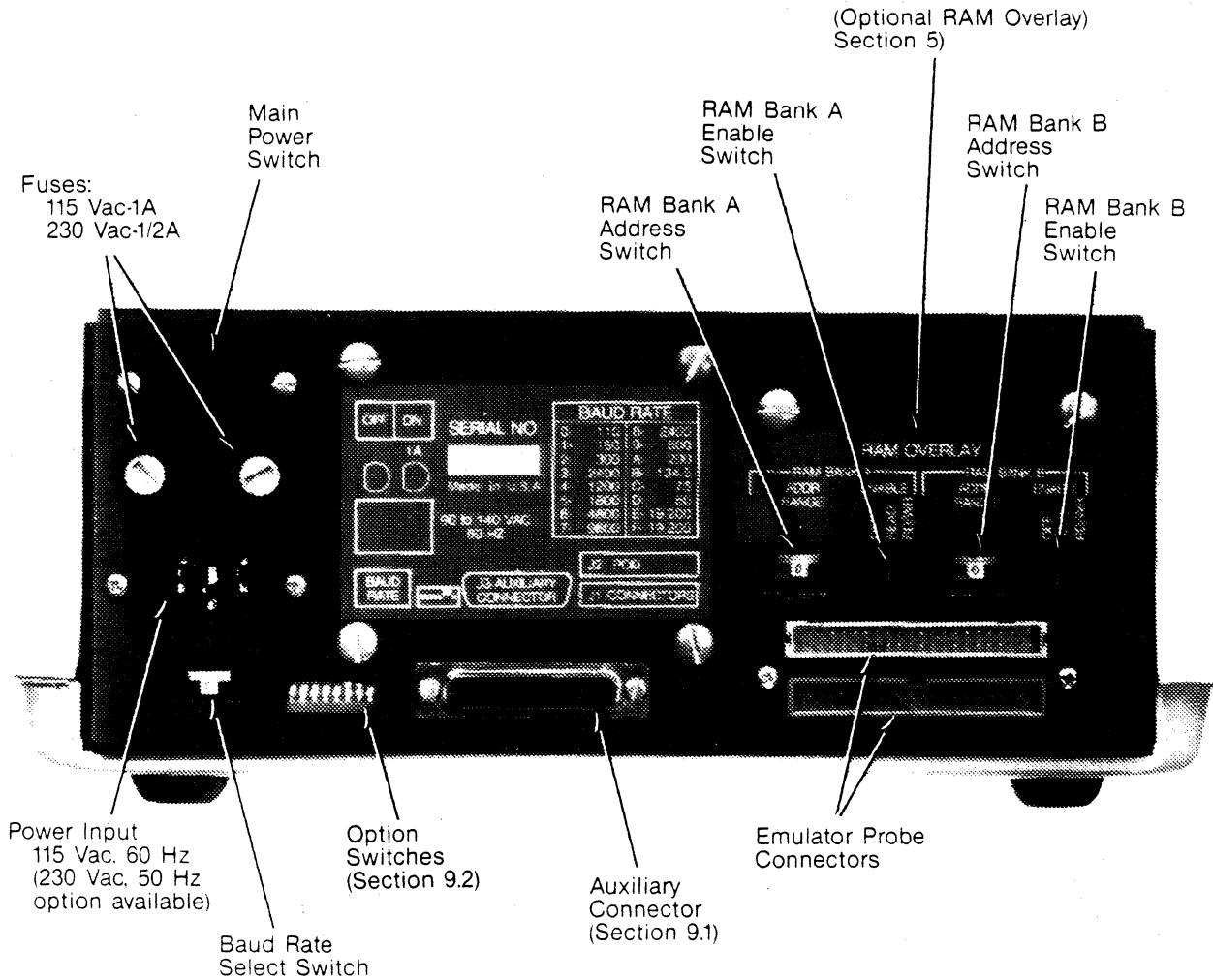


Figure 2.1-1  
Operator's Station

## 2.6 TRACE MEMORY

The Trace Memory is a 252-word by 32-bit memory that captures information from each bus cycle of the emulated target system microprocessor: the 16 address bits, 8 data bits, CPU read and write signals, the type of bus cycle (i.e., op-code fetch, I/O, or Interrupt Acknowledge) and two possible breakpoint sources—the Breakpoint Comparators and the External Breakpoint input.





---

## 2.7 BACK PANEL CONTROLS AND CONNECTORS

The back panel of the EM-800 includes the controls and various connectors used to connect the Diagnostic Emulator with power, the emulator probe and other external equipment:

### **Main Power Switch**

Controls the primary power to the unit.

### **Baud Rate Selector Switch**

A 16-position switch controls the transmission rate of serial data flow between the Diagnostic Emulator and peripheral equipment. The baud rate selection options are visible on the back panel template shown in Figure 2-7.1

### **Auxiliary Connector**

A 25-pin, D subminiature female connector. It provides RS-232C signals and additional control signals to auxiliary equipment (i.e., signature analyzer, oscilloscope, target system, development system). See section 9.1.

### **Option Switches**

These switches control characteristics of the EM-800 RESET circuitry and communications interface. The switches are shown in Figure 2-7.1, and switch options are discussed and illustrated in Section 9.2.

### **RAM Overlay Bank A and Bank B Address Switches**

### **RAM Overlay Bank Enable Switches**

These switches allow you to select the address range and to enable RAM Overlay Memory as off, read only or read/write. See Section 5, RAM Overlay, if you have the RAM Overlay option.

## 2.8 RAM OVERLAY

The EM-800 may be equipped with an optional RAM overlay feature. Either 8K bytes of 200 nS static RAM, or 16K or 64K bytes of 150 nS static RAM are available on a circuit board that includes the appropriate addressing and buffering.

This memory can be mapped into the target address space, overlaying the user system in the address block selected. The overlay memory may be loaded from the target system memory, front panel EPROM or external device by executing the appropriate Code Function. See Section 5, RAM Overlay.

## 2.9 DISASSEMBLER

The EM-800 is configured with a firmware package that provides for formatting and output of system information to an ASCII terminal device with RS-232 interface such as a CRT or hard copy terminal. The disassembly firmware extracts information from the EM-800 Trace Memory and emulation processor registers, formats data for display with instruction op-codes given in standard NSC mnemonic form (JP, ADD, PUSH, SET, LD, etc.) and outputs data through the serial port. See Section 6.



---

# **SECTION 3**

## **BASIC OPERATING INSTRUCTIONS**

- 
- 3.1 OPERATING VOLTAGE**
  - 3.2 SAFETY INFORMATION**
  - 3.3 CONNECTION TO TARGET EQUIPMENT**
-

### 3.1 OPERATING VOLTAGE

The EM-800 Diagnostic Emulator is normally supplied for operation from 90 to 140 Volts AC at 58 to 62 Hz line. The unit is also available for operation from 180 to 280 Volts AC at 48 to 52 Hz line if specified at time of order. The EM-800 uses a regulating transformer that also has the advantage of providing good blocking of conducted noise sometimes present on the power input to the unit.

### 3.2 SAFETY INFORMATION

The EM-800 is supplied with a 3-wire cord with a 3-terminal polarized plug for connection to the power source and protective ground. The ground terminal of the plug is connected to the metal chassis parts of the instrument. Electric-shock protection is provided if the plug is connected to a mating outlet with a protective ground contact that is properly grounded.

The internal (logic) ground of the EM-800 is not connected to the protective ground, but floats to the same potential as the equipment to which the unit is connected.

**CAUTION:**

*GROUNDING CONFLICTS MAY OCCUR IF THE EM-800 IS CONNECTED TO TWO ITEMS OF EQUIPMENT WITH DIFFERING GROUND POTENTIALS. SUCH AS THE TARGET EQUIPMENT AND A RS-232C TERMINAL*

### 3.3 CONNECTION TO TARGET EQUIPMENT

First, connect the emulator probe to the EM-800 operator station. Then remove the target system microprocessor from its socket and plug in the 40-pin plug. Be sure to observe correct pin 1 orientation. (See Figure 3-3.1.)

**CAUTION: NOTE CORRECT PIN 1 ORIENTATION.**

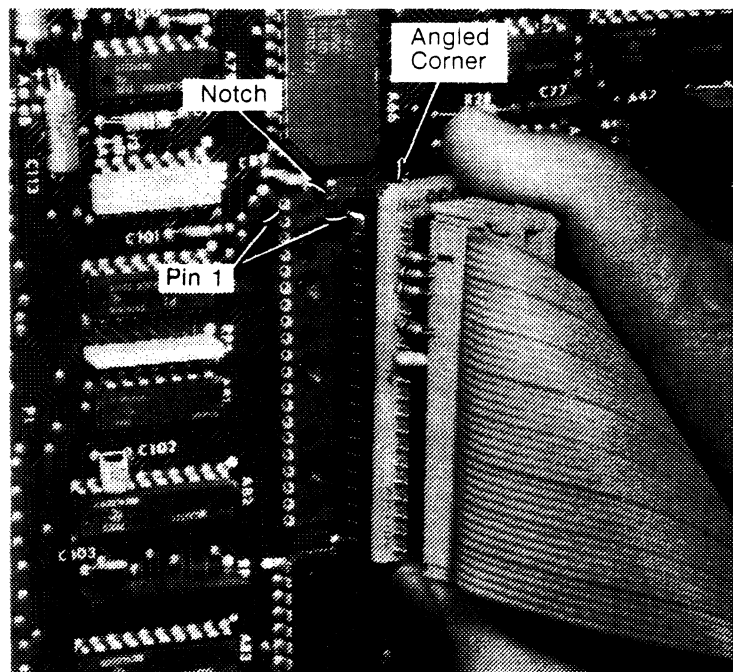


Figure 3.3-1  
Installing 40-pin plug

---

Apply power to the EM-800 and the target system after the unit is properly connected to the target circuitry. Once power is applied to the Diagnostic Emulator and the clock begins operating, it performs a power-on-reset operation during which the following functions are performed:

1. Reset CPU.
2. Clear Trace Memory and CPU Registers.
3. Clear the program starting address to zero (the default starting address) and display the address.
4. The Diagnostic Emulator awaits further operator input.

After the EM-800 is connected to the target system and power is applied to both the EM-800 and the target system, perform the following checks to verify that the unit is operating correctly:

1. The Clock Fault and Reset Fault indicators are not illuminated. This means that the system clock oscillator is operating and is being received by the EM-800 and that there is no continuous RESET signal from the target system.
2. The PAUSE indicator should be illuminated. This indicates that no target program is executing and that the EM-800 is awaiting operator commands.
3. At power-on time, the ADDRESS Display should indicate 0000. If it does, the EM-800 internal control program is operating.

If all items check out, you can begin operating the emulator. Details of EM-800 functions are discussed in the next section.

If you have experienced difficulty setting up your emulator, read Section 10 and then call your Applied Microsystems Corporation representative, if necessary.



---

# SECTION 4

## EM-800 FUNCTIONS

---

- 4.1 **EXECUTION CONTROL**  
*RESET Keyswitch • RUN Keyswitch*  
*RUN BKPT Keyswitch • STEP Keyswitch*  
*Breakpoint System • Trace Memory*
  - 4.2 **EXAMINATION AND ALTERATION  
OF CPU REGISTERS**
  - 4.3 **EXAMINATION AND ALTERATION  
OF MEMORY LOCATIONS**
  - 4.4 **EXAMINATION AND ALTERATION  
OF I/O PORTS**
-

---

## **EM-800 FUNCTIONS**

A basic function of the EM-800 is to emulate the target system microprocessor. Effectively, the Diagnostic Emulator is a pin-compatible functional replacement for the microprocessor in the target system. The unit is designed to meet the timing specifications of the emulated processor and to minimize the increase in electrical loading of the user system.

The EM-800 is always in one of two modes: RUN or PAUSE. If in the RUN mode, the EM-800 is emulating the target system microprocessor and executing the target system program at full system speed. The trace memory will be active (unless inhibited by external control), and all bus cycles of the emulated microprocessor are recorded for possible later display. In the PAUSE mode, emulation of the target system microprocessor is suspended and you may perform other functions, such as manually examining or altering memory locations, I/O ports or internal registers of the emulated microprocessor; you may also review the history of the target program execution from the Trace Memory or execute one of the code Function routines.

### **4.1 EXECUTION CONTROL**

You control the EM-800 primarily through the Operator's Station Keyboard. Keyswitch groupings are designed for easy use. The EM-800 display provides information about program execution, CPU status and EM-800 conditions. Table 4-1.1 lists the display panel indicators.

#### **4.1.0 RESET Keyswitch**

The red RESET Keyswitch resets the microprocessor and initializes the EM-800 in the PAUSE mode. At this time the Address Display shows the program starting address. The program starting address may be changed by entering digits with the hexadecimal keyswitches, or the current program starting address may be used.

The option switches on the back panel may be used to set up one of several options concerning the RESET circuitry of the EM-800 and the target system. See Section 9.2.

#### **4.1.1 RUN Keyswitch**

Pressing the RUN Keyswitch causes the EM-800 to execute the target program beginning at the preset address or continuing from the last instruction executed. Execution is at full system speed with no extra wait states beyond those commanded by the target system. The activity of the executing program is recorded continuously in the Trace Memory. It is also possible to obtain a general view of the program activity by watching the displays. For example, it is possible to tell if the program is in a tight loop or ranging widely in the program address space by observing changes in the Address Display.



---

### 4.1.2 RUN BKPT Keyswitch

This Keyswitch starts the EM-800 running the target program in real time, as does RUN, and also enables the breakpoint-stop circuitry. If a breakpoint is detected, the EM-800 will pause before executing the next instruction, and the display will show the cycle where the breakpoint was detected. Pressing RUN BKPT again will cause execution to resume until the breakpoint is again detected. The breakpoint-stop circuitry may be disabled during program execution by pressing RUN.

### 4.1.3 STEP Keyswitch

Pressing the STEP Keyswitch while the program is running causes the program execution to stop. The displays show the operation code fetch cycle of the last instruction executed, with the address, op-code (data) and control signals visible. When the Diagnostic Emulator stops executing the target program, the following events take place:

1. The processor stops executing the target program.
2. The processor registers are saved in internal scratch pad memory and are accessible for display or alteration.

You, in effect, freeze the target program execution at the point reached when STEP was depressed. You then have several options:

1. Continue executing the program at full speed by pressing RUN.
2. Continue executing the program one instruction at a time by pressing STEP for each additional instruction execution.
3. Examine or change the contents of any of the processor registers.
4. Examine any memory or I/O address, and if the location is writable, store new data in it.
5. Review the last 252 bus cycles performed by the processor by decrementing through the Trace Memory.

The state of the target program is not changed by any of these operations unless you have purposely altered it and program execution may be continued from the point where it stopped.

The program may be executed one instruction at a time by pressing STEP once for each instruction. If STEP is pressed and held down, the emulator begins stepping at about seven instructions per second. The step rate then accelerates gradually from 7 steps per second to about 75 steps per second. Execution stops again if the keyswitch is released.

Table 4.1-1  
Display Panel  
Indicators

**FAULT GROUP**

ILLUMINATES IF:

- CLK Target system clock not operating.  
Target system clock is low in frequency  
EM-800 not connected to target system.
- RESET Processor and Diagnostic Emulator held in Reset  
by a low on the RESET IN terminal of the  
microprocessor socket.

**MACHINE CYCLE GROUP**

ILLUMINATES IF:

- FETCH Displayed machine cycle is an op-code fetch cycle.
- BK A Breakpoint A. Conditions set up for an output  
from Breakpoint A Comparator were satisfied dur-  
ing the displayed machine cycle.
- BK B Breakpoint B. Conditions set up for an output from  
Breakpoint B Comparator were satisfied during the  
displayed machine cycle.
- EXT External. External Breakpoint input low (active) dur-  
ing displayed machine cycle.
- I/O Input/Output. Machine cycle being displayed is a  
data transfer from an input port address or to an  
output port address.
- IACK Interrupt Acknowledge. Machine cycle being  
displayed is an interrupt acknowledge cycle.
- RD Read. Machine cycle being displayed is read from  
memory or read from I/O cycle.
- WR Write. Machine cycle being displayed is write to  
memory or write to I/O cycle.

---

## FLAGS

### ILLUMINATES IF:

S	Sign bit is true.
Z	Zero bit is true.
H	Auxiliary Carry bit is true.
P/V	Parity bit is true.
C	Carry bit is true.

## CPU STATUS

### ILLUMINATES IF:

IENA	Emulated processor is ready to respond to an interrupt. (Interrupts enabled.)
INT	Interrupt. One of the following inputs to NSC800 is active: INTR (PIN 10); RST 5.5 (PIN 9); RST 6.5 (PIN 7).
NMI	NMI input to NSC800 is active.
BREQ	BREQ input of the NSC800 is active.
Wait	WAIT is active. CPU is waiting. If the CPU is working with a system that requires some wait states, the indicator may be more or less dimly lit.
Halt	Processor has executed a HALT instruction and has entered the HALT state.
Pause	Real-time emulation of the target program is suspended and the Diagnostic Emulator is awaiting another command.
BKPT ENA	Breakpoint Enable. Illuminates if the RUN BKPT keyswitch is depressed.

---

#### 4.1.4 Breakpoint System

The Diagnostic Emulator incorporates a Regional/Relational breakpoint generation system to enable you to monitor the operation of your program and to stop execution of your program when desired. The EM-800 contains two independent address comparators. Each of these comparators continuously monitors the 16-bit address bus of the microprocessor. In addition, each comparator may be qualified to respond to read cycles only, to write cycles only, or to both read and write cycles. The comparators may also be configured to respond to memory addresses or to I/O port addresses.

It is also possible to configure the breakpoint system so that a specified relationship must hold between the A and B breakpoint comparators before PAUSE occurs. The relationships that may be specified are the following:

1. A or B Break if condition A or condition B is found (or both).
2. A then B. Break if condition A is found followed some time later by condition B.
3. A↔B Break if any address in the range from A to B (inclusive) is found.
4. ←A—B→ Break if any address outside of the range from A to B is found. (Including addresses A and B.)

Table 4.1-2  
Breakpoint Qualifiers  
and Features

---

0 - Disable	4 - Not Used	C - A or B
1 - Memory Read	5 - I/O Read	D - A then B
2 - Memory Write	6 - I/O Write	E - Range A to B
3 - Memory Read/Write	7 - I/O Read/Write	F - Range outside A to B

---




The various breakpoint qualifiers and relationships are set up by simple keystroke sequences. Some examples of these sequences follow.


---

**EXAMPLE:**

Set up breakpoint comparator A to respond to read or write cycles at address 4300<sub>16</sub>; disable comparator B.

**KEYSTROKE SEQUENCE:**

     Set breakpoint address.

  Set qualifier 3 (memory R/W).

Press and hold down BKPT A Key while qualifier is entered.

  Set qualifier 0 (Disable).






Press and hold down BKPT B Key while qualifier is entered.

On power-up, the EM-800 sets the qualifiers for both breakpoint comparators for the A OR B relation (comparators operating independently of each other) and the memory read/write qualifier. The address to which each comparator is initialized is 0000<sub>16</sub>. In the preceding example it was not necessary to alter the relationship holding between the two comparators, so the default A OR B relationship was not altered.

**EXAMPLE:**






Set up breakpoint comparator A to respond to read cycles only at memory address 8A72<sub>16</sub>, and breakpoint comparator B to respond to write cycles to I/O port 13<sub>16</sub>.

**KEYSTROKE SEQUENCE:**

     Set A breakpoint address.

  Set A qualifier 1 (MEMORY read).

Press and hold down BKPT A Key while qualifier is entered.

     Set B breakpoint address.

  Set B qualifier to 6 (I/O WRITE)

Press and hold down BKPT B Key while qualifier is entered.

---

When the breakpoint circuitry is set up as desired, program execution may be started using RUN BKPT. The function RUN BKPT is the same as the function of RUN except that when the breakpoint condition occurs, program execution stops (after finishing the instruction cycle). You can also start program execution using the RUN Key, and then later arm the breakpoint-stop circuitry by depressing RUN BKPT even while the target program is executing. Breakpoints may be disabled while the target program is executing while depressing RUN. The BKPT ENA indicator on the display shows the current breakpoint enable status of the emulator.

**EXAMPLE:**






Set up breakpoint range from 4307<sub>16</sub> to FFFF<sub>16</sub>.

**KEYSTROKE SEQUENCE:**

     Set A to range beginning (4307<sub>16</sub>).

  Set qualifier to E (Range A to B).

Press and hold down BKPT A Key while qualifiers are entered.

     Set B to range end (FFFF<sub>16</sub>).

With the specifications made as shown, the breakpoint circuitry will respond to any cycle to any address in the range of 4307<sub>16</sub> to FFFF<sub>16</sub>. Note that it was not necessary to specify any qualifiers for the B comparator; this is because the two comparators are linked together to provide address range detection, and the qualifiers entered for A apply also to B.

---

**EXAMPLE:**

Set up sequential breakpoint detection such that target program execution will halt after the A comparator has detected address EB22<sub>16</sub> and then the B comparator has detected address 48<sub>16</sub>.

**KEYSTROKE SEQUENCE:**

     Set A to address EB22<sub>16</sub>.

  Set A qualifier to D (A then B).

Press and hold down BKPT A Key while qualifier is entered.

   Set B to address 48<sub>16</sub>.

---

## 4.1.5 TRACE MEMORY

One of the most useful EM-800 features is its 252 bus cycle Trace Memory. The Trace Memory is organized as a ring buffer that records all target program activity. It operates in both real-time and single-step modes, and its contents remain in the correct sequence, regardless of whether you operate the program wholly or partly in either of these two modes.

To review the Trace Memory contents, the EM-800 must be paused. The PAUSE mode is entered automatically when the program encounters a breakpoint, or it can be entered manually by depressing STEP. When the program enters PAUSE as a consequence of depressing the STEP Key, the Display shows the fetch cycle address and data for the last instruction recorded.

When a breakpoint triggers PAUSE\*, the Display shows the cycle where the breakpoint was detected, and you can easily review the program activities leading up to the event. Depressing DEC allows you to examine the last 252 bus cycles of the program activity prior to the breakpoint. Depressing INC allows you to review forward up to the last cycle traced. Depressing STEP advances the target program past the breakpoint event, one instruction at a time. Depressing TRACE allows you to return to Trace Memory again after selecting another mode (i.e., MEM ADDR, I/O ADDR, etc.) and return the original program event or bus cycle to the display. The TRACE Key has no effect unless the program is already in PAUSE. STEP actually causes the emulator to execute another program instruction and this instruction is entered into the Trace Memory like all others.

The NSC800 machine instructions may have one or several bus cycles per instruction. The following two examples illustrate displayed Trace Memory contents, first after executing a simple instruction and then after a more complex one.

### EXAMPLE 1: LD B, C

Cycle	Addr	Data	Fetch	RD	WR
1	4000	41	X	X	

*Single bus cycle instruction: Move contents of C register to B register. Assume the instruction location is address 4000<sub>16</sub> in the target memory. The Trace Memory records a bus cycle with the address of 4000<sub>16</sub>, data of 41<sub>16</sub>, and control bits indicating that it is a fetch operation and a read cycle.*

\*The EM-800 finishes executing the instruction cycle before it pauses.



---

**EXAMPLE 2: LD (07055H), HL**

Cycle	Addr	Data	Fetch	RD	WR
1	4000	22	X	X	
2	4001	55		X	
3	4002	70		X	
4	7055	34			X
5	7056	12			X

Five bus cycle instruction: Cycle one fetches op-code 22 of the LD instruction located at address 4000<sub>16</sub>. Cycles two and three read low-order and high-order bytes (55<sub>16</sub> and 70<sub>16</sub>) of the 16-bit address located at 4001<sub>16</sub> and 4002<sub>16</sub>. Cycles four and five write the contents of the HL register pair (34<sub>16</sub> and 12<sub>16</sub>). The Trace Memory records all five bus cycles of the instruction. The address location, program data and op-code cycle are shown on the Display Panel for each bus cycle of the instruction. If the EM-800 had entered PAUSE and displayed Cycle 1 (the OP-CODE fetch), then the INC Key would be used to advance through the Trace Memory and observe the subsequent bus cycles.

Normally, the INC and DEC Keys move the trace index one cycle at a time. However, if you depress and hold down the TRACE Key, the INC and DEC Keys will cause the next or previous fetch cycle to be displayed without stopping on other machine cycles.

#### 4.2 EXAMINATION AND ALTERATION OF CPU REGISTERS

The NSC800 register contents may be examined, and if desired, overwritten with new data.

Register data is displayed by using the blue REG Keyswitch, followed by one of the hexadecimal keyswitches to designate which register should be displayed. Table 4-2.1 shows the registers selected by the various keyswitches. Note that 4 through 7 do not correspond to actual NSC800™ registers. These keyswitches are used to set up parameters for the Built-In test routines or user Code functions. These Code Functions are described in later sections. (See Section 7 and Section 8).

---

Examples of readout and alteration of CPU registers:

**EXAMPLE:**



B register contents displayed on address.



B register is accessed and then overwritten with data  $3F_{16}$ .



Stack Pointer is accessed and displayed on 16-bit address display



HL Register Pair is accessed, and then the contents are overwritten with  $3C00_{16}$ .

The prime registers are accessed by depressing the hexadecimal key a second time. For example, the B-prime register is accessed in the following example:

**EXAMPLE:**



The first key depression will access and display the B register, but depressing the key a second time switches the display to the prime register. If the register has 16 bits, then all four of the digits of the display will be illuminated. If the register has eight bits, then the value is shown on the two low-order digits of the Address Display.

The Data Display is used for feedback about the register that has been selected. If you select the stack pointer, the Data Display will show a '1' since the 1 digit key is used to select that register. If you select the B register (as in the example preceding) the Data Display will show a 'B'; if you select the B-prime register, then a '1B' will be displayed.

Table 4-2.1  
Keyboard Designators  
(After REG Keyswitch  
Is Pressed)

KEY	(first key push) REGISTER	(second key push) REGISTER	DESCRIPTION
0	PC		Program Counter
1	SP		Stack Pointer
2	HL	HL'	HL Register Pair and Prime
3	DE	DE'	DE Register Pair and Prime
4	BEG*		Begin Address (for programmed tests)
5	END*		End Address (for programmed tests)
6	ADDR*		Address (programmed test parameter)
7	DATA*		Data (programmed test parameter)
8	IX		IX Register
9	IY		IY Register
A	A	A'	Accumulator and Prime
B	B	B'	B Register and Prime
C	C	C'	C Register and Prime
D	D	D'	D Register and Prime
E	E	E'	E Register and Prime
F	FLAGS	FLAGS'	Flags and Prime

\*Not an actual NSC800™ register

### 4.3 EXAMINATION AND ALTERATION OF MEMORY LOCATIONS

Any memory location accessible to the emulated microprocessor may be accessed and displayed by the EM-800. If desired, new data may be written to the location.

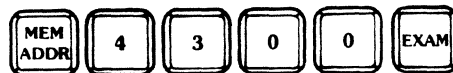
**EXAMPLE:**



Address 431A<sub>16</sub> is entered, and when EXAM is pressed, the EM-800 will read from address 431A<sub>16</sub> and display the data obtained.

If you wish to review a group of sequential memory locations, enter the initial address and examine that location as illustrated above; then examine successive locations by depressing INC.

**EXAMPLE:**



Examine data at 4300<sub>16</sub>.



Examine data at 4301<sub>16</sub>.

etc.

Examine data at successive locations, etc.

A memory location may be altered by entering an address, as shown above, then entering data using LOAD DATA and finally storing the data to the selected memory address using STORE.

---

**EXAMPLE:**



The above sequence writes the data  $55_{16}$  to memory address  $13FE_{16}$  in the target system.

Sequential locations may be quickly altered by incrementing the address after each data entry operation. For example, the following keystroke sequence enters a short program fragment into memory:

**EXAMPLE:**



Enter initial address  $0800_{16}$ .



Enter data  $C3_{16}$  then store the data to  $0800_{16}$  and increment to  $0801_{16}$ .



Enter data  $00_{16}$ , then store the data to  $0801_{16}$  and increment to  $0802_{16}$ .



Enter data  $08_{16}$  and store to  $0802_{16}$ ; increment to  $0803_{16}$ , etc.

The EM-800 does not require redundant keystrokes. The unit assumes that if you have entered new data while a particular memory address is accessed, that you want to store that data before going to the next address.

In all of the above examples in which INC was used, DEC (decrement) could also have been used.

---

#### 4.4 EXAMINATION AND ALTERATION OF I/O PORTS

Input/Output ports of the NSC800™ may be accessed and displayed in a similar manner to that described for memory addresses, with two differences. The first is that the I/O ports respond to an eight-bit address and consequently only eight-bit addresses need be entered. The second difference is that the INC and DEC keyswitches do not perform an automatic read of the next (or previous) I/O port address. The intent of this characteristic is to help avoid unintended reading of an I/O port, which sometimes results in a change of state of complex I/O devices. For example, a complex interface circuit such as the National PIO will change state when the input data register is read. The following is a keystroke sequence that may be used to examine data at an input port:

#### EXAMPLE:



Read and display data at Input Port 3.



Data A0<sub>16</sub> is written to Output Port 1B<sub>16</sub>. (No READ cycle was performed at Port 1B.)

#### NOTE:

To enable or disable interrupt masks, you must execute an I/O instruction storing data at location BB. The I/O function on the EM-800 cannot be used to change the interrupt mask because this is an internal function of the NSC800.



---

# **SECTION 5 RAM OVERLAY**

- 
- 5.1 OVERVIEW**
  - 5.2 INSTALLATION**
  - 5.3 CONTROLS**
  - 5.4 UPLOADING/DOWNLOADING**
-

---

## 5.1 OVERVIEW

The emulator may be equipped with an optional RAM overlay feature. The following options are available:

- 8K bytes of 200nS static RAM
- 16K bytes of 150nS static RAM
- 64 bytes of 150nS static RAM

The RAM overlay feature is available on a circuit board which includes the appropriate addressing and buffering.

The 8K-byte memory is divided into two independent 4K-byte memory banks. Each bank has independent control circuitry and may be enabled as read/write memory or read only memory, or disabled. Each 4K bank may be independently set to occupy a specific address range.

The 16K-byte memory is also divided into two independent 8K-byte memory banks. Each bank has independent control circuitry and may be enabled as read/write memory or read only memory, or disabled. Each 8K bank is further divided into two 4K blocks that can each be independently set to occupy a specific address range. If only a single 4K block is required, set both blocks to the same address range.

The 64K-byte memory is also divided into sixteen 4K-byte memory banks that can overlay the entire memory space. Each bank may be enabled as read/write memory or read only memory, or disabled by one of the sixteen switches. The address for each 4K is hardwired so that it is only necessary to enable those banks that are needed. Refer to Figure 5-1.1.

The RAM overlay memory can be used for patching software in simulated PROMs or adding further memory space to your test system. For instance, a program that is normally in ROM can be loaded into RAM and debugged with no need to burn PROMs until production. This kind of feature is very helpful if you are doing a field test with no computer available; the RAM overlay memory allows you to do a patch or try option values and test until the program is correct.

### NOTE

You should be aware that when the RAM overlay is enabled, it replaces CPU memory space.





---

## 5.2 INSTALLATION

To install a RAM Overlay option in your emulator (units without the two hex stand-offs, as shown in Figure 5-2.1), the procedure is as follows:

1. Unplug the power cord from the back of the emulator.
2. Unplug the pod cables attached to the back of the emulator, noting the proper positioning.
3. Turn the emulator upside down and place it on a soft surface to prevent scratching the top cover.
4. Remove the four top cover screws and the four rubber feet from the bottom of the emulator.
5. Remove the bottom cover.
6. Turn the emulator upright and carefully remove the top cover.
7. Remove the four display assembly screws, being careful not to scratch the display plex panel. Note the location of the spacers behind the plex panel.
8. Remove the display assembly.
9. Remove the four keyboard assembly screws.
10. Remove the keyboard assembly.
11. Unplug the four-contact Molex connector from the power supply regulator board, located next to the power switch.
12. Remove the two screws located on either side of the pod cable connector at the rear of the unit.
13. Slide the main logic assembly forward and out of the emulator frame. Note that it is necessary to install the new mounting bracket included in your 64K RAM kit onto the Main Logic Board.
14. Install the two hex standoffs supplied into the position shown in Figure 5-2.1. The standoffs will be secured into place by using two screws supplied in the installation kit. (Install the screws from the bottom side of the assembly.)
15. Cut the lower half of the back panel decal to expose the cutouts as shown in Figure 5-1.1a for the 8K and 16K RAM Overlays only. For the 64K RAM Overlay, remove the old decal and replace it with the new one supplied in your 64K RAM Kit.
16. Plug the RAM Overlay connector into the main logic assembly.
17. Reassemble the emulator by reversing steps 1 through 13.
18. Connect the probe tip assembly to a known-good target system.

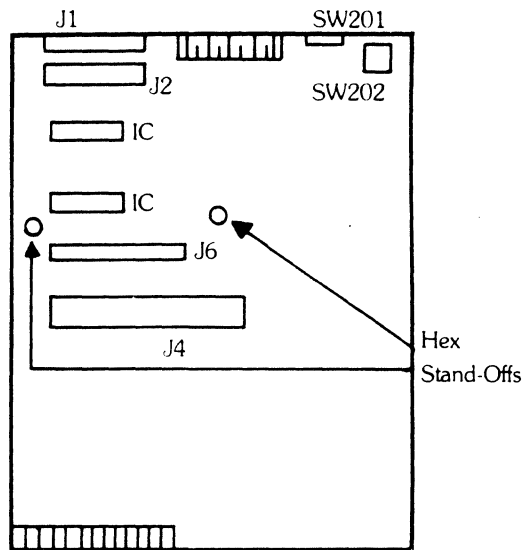


Figure 5.2-1  
Overlay Installation

If a RAM Overlay block is set up to respond to a range of addresses, say  $000_{16}$  to  $0FF_{16}$ , then target system memory in the same address range becomes inaccessible to the emulation processor. The memory block has "overlaid" the corresponding target system addresses. (See, however, the description of Code Function C5 for an exception to this characteristic of the emulator.)

The contents of the RAM Overlay are retained as long as power is applied to the emulator. It is possible to load the RAM Overlay with data, turn the enable switches off and retrieve the data at a later time. To retrieve data, turn the enable switches to either the READ or RE WR position.

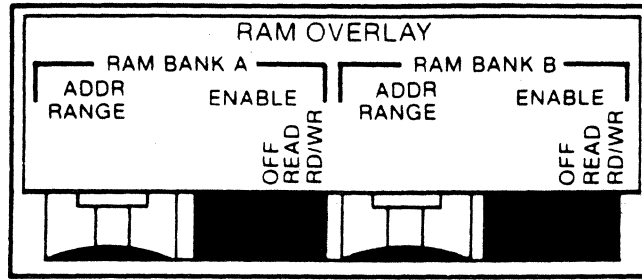
Each 4K byte block of memory for 8K and 16K RAM Overlays has an associated Address Range switch. The 64K RAM Overlay also has an associated control switch for each 4K block of memory. See Figure 5.3-1.

### 5.3 CONTROLS

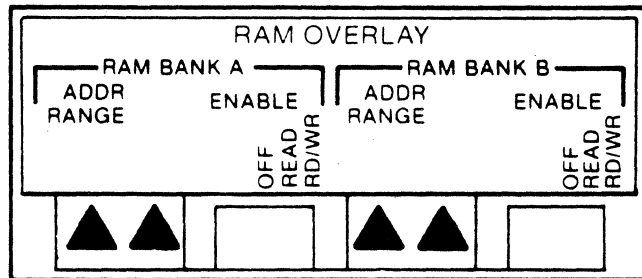
The enable switches are three-position switches that place the memory bank into one of three conditions:

- **OFF**  
The memory bank is disabled and is effectively removed from the system.
- **Read/Only**  
In this mode, it is not possible for the target system program to alter the contents of the memory. Note, however, that the emulator is still able to write to the memory bank from the keyboard or from a Code function routine such as Code Function C3 (download).
- **Read/Write**  
The memory bank is placed in a Read/Write configuration. Both the target system and the emulator are able to read the memory and write new information to it.

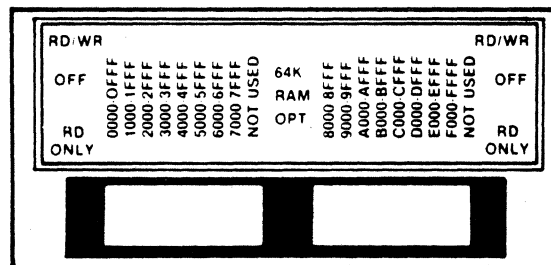
Figure 5.3-1  
RAM Overlay Controls



a) 8K



b) 16K



c) 64K

If a memory bank is disabled (toggle switch in the off position), the memory will nevertheless continue to retain data. The data will reappear in the target address space whenever the memory is again enabled.

The address range switches for the 8K and 16K RAM Overlays are 16-position rotary (thumbwheel) switches used to select the address range where the 4K memory blocks will reside in the target address space. Each of the 4K memory blocks can be moved to any of 16 positions, beginning at a 4K boundary. See Table 5.3-1.

Table 5.3-1  
Memory Block Address  
A and B

SWITCH POSITION	MEMORY BLOCK ADDRESS	SWITCH POSITION	MEMORY BLOCK ADDRESS
0	0000 <sub>16</sub> -0FFF <sub>16</sub>	8	8000 <sub>16</sub> -8FFF <sub>16</sub>
1	1000 <sub>16</sub> -1FFF <sub>16</sub>	9	9000 <sub>16</sub> -9FFF <sub>16</sub>
2	2000 <sub>16</sub> -2FFF <sub>16</sub>	A	A000 <sub>16</sub> -AFFF <sub>16</sub>
3	3000 <sub>16</sub> -3FFF <sub>16</sub>	B	B000 <sub>16</sub> -BFFF <sub>16</sub>
4	4000 <sub>16</sub> -4FFF <sub>16</sub>	C	C000 <sub>16</sub> -CFFF <sub>16</sub>
5	5000 <sub>16</sub> -5FFF <sub>16</sub>	D	D000 <sub>16</sub> -DFFF <sub>16</sub>
6	6000 <sub>16</sub> -6FFF <sub>16</sub>	E	E000 <sub>16</sub> -EFFF <sub>16</sub>
7	7000 <sub>16</sub> -7FFF <sub>16</sub>	F	F000 <sub>16</sub> -FFFF <sub>16</sub>

Because of the large memory capacity, the card setup for the 64K RAM is slightly different than the 8K and the 16K RAM. The memory control switches consist of sixteen three-position switches.

- The up position enables the RD/WR memory.
- The down position enables the RD memory only.
- The center position disables the memory.

Eighteen switches are present; however two are not used. Each switch represents a 4K byte segment of memory. The decal on the back of the EM denotes the switch positions and memory range.

## 5.4 UPLOADING/ DOWNLOADING

Programs can be transferred to the RAM Overlay from the target system, the front panel EPROM socket or the RS-232C serial port on the auxiliary connector. Programs in the RAM overlay can be dumped to the RS-232C port. Both uploading and downloading are accomplished by enabling the RAM Overlay for read only or read/write, selecting the desired address range via the rotary switches, and then executing the appropriate Code Functions C1 through C6. A summary of these Code Functions is given below. (For more information see Section 7, Built-in Diagnostic Functions.) Note that the overlay block involved must be enabled; otherwise, these code functions will involve the target system memory in place of the RAM Overlay.

### CODE C1—LOAD RAM OVERLAY FROM FRONT PANEL PROM

This Code Function transfers data from the front panel diagnostic EPROM to the Overlay RAM. To use this Code Function, first enter the starting and ending address values in the BEG and END registers, then start the routine.

### CODE C2—VERIFY RAM OVERLAY AGAINST FRONT PANEL PROM

Code C2 compares the front panel PROM with the address range you specify. The address range is loaded into the BEG and END register. If a non-verify occurs, the Diagnostic Emulator emits three beeps and pauses. The address and the data that failed to verify are displayed. By depressing and holding the EXAM keyswitch, the correct data will be displayed.

---

### **CODE C3—LOAD RAM OVERLAY FROM SERIAL LINK**

Code C3 transfers hex data from the RS-232C input to the RAM Overlay. To use this Code Function, connect the RS-232C input to the source of information, start the routine and then enable the source to download the appropriate data.

### **CODE C4—DUMP RAM OVERLAY TO SERIAL LINK**

Code C4 transfers data from selected areas of RAM Overlay to the serial RS-232C output. To use the Code Function, first specify the address limits in the BEG and END registers, next prepare the receiving device to accept data, then start the routine.

### **CODE C5—LOAD RAM OVERLAY FROM TARGET MEMORY**

Code C5 transfers data from selected areas of target memory space to the equivalent areas in Overlay Memory. The BEG and END registers are set to the range of addresses from which data is to be transferred.

### **CODE C6—VERIFY RAM OVERLAY WITH TARGET MEMORY**

Code C6 compares data from selected areas of target program memory to the equivalent areas in Overlay Memory. The BEG and END registers are set to the desired target memory address range.

If a non-verify occurs, the Diagnostic Emulator emits three beeps and pauses. The address and the data that failed to verify are displayed. By depressing and holding the EXAM Keyswitch, the correct data will be displayed.

---

# **SECTION 6**

## **DISASSEMBLY**

---

- 6.1 OVERVIEW**
  - 6.2 FORMAT DEFINITION**
- 

- 6.3 LINE ASSEMBLER**
  - 6.3.1 How to Use the Line Assembler**
  - 6.3.2 Features Supported**
- 6.4 MEMORY DISASSEMBLER**
  - 6.4.1 Overview**
  - 6.4.2 How to Use the Disassembler**
  - 6.4.3 Disassembler Output Format**
  - 6.4.4 Errors**

---

## 6.1 OVERVIEW

The disassembler firmware gives the EM-800 the ability to output the contents of the Trace Memory and emulation processor registers to the serial port. In this way, a readable and attractive display may be created on a CRT or hardcopy terminal.

The disassembly firmware is disabled when the EM-800 is first powered up, so you must enable it before use. The following procedure will make the EM-800 ready to operate with an ASCII terminal and disassembly firmware:

### 6.1.1 OPERATION PREPARATION PROCEDURES

1. Connect the terminal to the EM-800 using an appropriate cable. The minimum circuits that must be connected are:

Pin 1—Protective Ground

Pin 2—Serial Data Out

Pin 7—Signal Ground

Some RS-232 terminals may also require the following connection:

Pin 20—Data Terminal Ready

Take care that Pins 10, 11, 12, 13, 22, 24 and 25 are not connected to incompatible circuits. See 9.1, Auxiliary Connector.

2. Set the Baud Rate Selector switches of the EM-800 and the terminal to compatible settings.
3. Check the setting of Option Switch 3. If Option Switch 3 is open (up), then the EM-800 will not output serial data unless the Clear-to-Send signal (Pin 5) is high. If the Clear-to-Send signal is not important in your application, set Option Switch 3 to the CLOSED (down) position and the EM-800 will output data on command regardless of the state of Pin 5.
4. Enable the disassembly firmware by executing the Code Function E1, E2 or E3. The EM-800 is now ready to output to the terminal device. See Table 6.1-1 for format selection information.

The disassembly firmware may be turned off by executing Code Function E0.

Operate the EM-800 in the normal manner. Any time that the EM-800 transfers from RUN to PAUSE, the disassembly firmware will format and dump a part of the contents of the Trace Memory to the terminal; normally 24 lines of output are produced. The last line output represents the last instruction executed and the firmware will then output the register display.



---

If the EM-800 is operated in single-step mode, the firmware will output the register display after every instruction.

The disassembly software is designed so that 20 lines of output are produced each time the emulator transfers from RUN to PAUSE; this amount of data provides approximately a full screen on most CRT terminals. If output of the entire contents of the Trace Memory is desired, execute Code Function D8. This Code Function will execute even if disassembly is not enabled, but the disassembly firmware option *must* be installed on the machine. All of the data in the Trace Memory will be formatted and output. Data output may be suspended for a moment by depressing the EXAM Key; when the Key is released, data output will continue.

#### NOTE

It is possible that the data recorded in Trace Memory does not represent actual machine execution of a program (for example, a block of data left by a memory diagnostic Code Function or a data transfer Code Function). In such a case, the disassembler will not format and output the data.

Table 6-1-1  
Disassembly Format  
Selection

---

CODE FUNCTION	REGISTERS DISPLAYED	FORMAT
E0	---	(Disable Disassembly)
E1	Flags, A, BC, DE, HL, SP	80-Character Line
E2	Flags, A, BC, DE, HL	72-Character Line
E3	Flags, A, BC, DE, HL, IX IX, SP, Flags, A, BC, DE, HL	Two 72-Character Lines

---

## 6.2 FORMAT DEFINITION

Figures 6.2-1, 6.2-2, and 6.2-3 show some lines from a printer connected to an EM-800. The various fields of the disassembly presentation are identified in the figure. All numbers that are output by the disassembler are in hexadecimal representation. Additional information about the fields of the display follows:

### Address

The address of the first op-code byte of the instruction.

### Op-Code

The operation code of the instruction. Double op-code instructions are displayed with the prefix byte extending to the left of the column of op-code bytes. The two-line register display format will show the IX register contents underneath this field.

### Operand

The operand bytes of the instruction (if any).

---

### Op-Code Mnemonic

The operation code of the instruction given in mnemonic form. The two-line register display format will show the IY register contents underneath this field.

### Operand

The operand field of the instruction is symbolic format, except that addresses and constants are given as hexadecimal numbers.

### Data Transfer

Any data transfer operations that occur as a consequence of the instruction are shown here. The most common formats are:

AAAA > DD

AAAA < DD

The first format means that the processor wrote data 'DD' to address 'AAAA.' The second format means that the processor read data 'DD' from address 'AAAA.' The other formats are associated with I/O instructions and take the form:

- PP > DD

- PP < DD

The first format means that the processor wrote data 'DD' to output port 'PP.' The two-line register display format will show the stack pointer contents underneath this field.

Some instructions transfer more than one byte of data. The second byte of a data transfer will be shown in this field. If there are more than two bytes transferred, the additional bytes are shown in fields 6 and 7 on the following line. See, for example, the EX (SP), HL instruction, which reads two bytes from the top of stack and then writes two other bytes to the top of stack. All of these transfers are easily seen from the display.

### Breakpoint

If a breakpoint occurred during the execution of the instruction on this line, it will be identified in this column by an asterisk (\*).

---

### **Flag Register**

The CPU flag register (condition code register) is shown in this field. Each of the five characters in this field represents one of the condition code bits as follows:

- First - 'S' if sign bit is true.
- Second - 'Z' if zero bit is true.
- Third - 'H' if half carry bit is true.
- Fourth - 'P' if parity/overflow bit is true.
- Fifth - 'N' if subtract flag bit is true.
- Sixth - 'C' if carry bit is true.

If any of the condition code bits is not true the letter is replaced by a period. The two-line register display format will show the alternate (prime) flag register underneath this field.

### **Accumulator**

The content of the accumulator after the execution of the instruction. The two-line register display format will show the alternate (prime) accumulator underneath this field.

### **BC Register Pair**

The content of the BC register pair following the execution of the instruction. The two-line register display format will show the alternate (prime) BC register pair underneath this field.

### **DE Register Pair**

The content of the DE register pair following the execution of the instruction. The two-line register display format will show the alternate (prime) DE register pair underneath this field.

### **HL Register Pair**

The content of the HL register pair following the execution of the instruction. The two-line register display format will show the alternate HL register pair underneath this field.

### **Stack Pointer**

The content of the stack pointer following the execution of the instruction. This field is not displayed in the 72-character line formats.

Figure 6.2-1  
Code E1 80-Character  
Single-Line  
Disassembly Format

Address	Op-Code	Operand	Op-Code Mnemonic	Operand	Data Transfer	Data Transfer	Breakpoint	Flag Register	Accumulator	BC Register Pair	DE Register Pair	HL Register Pair	Stack Pointer
0000	C3	2B00	JF	002B									
002B	31	E030	LD	SP,30E0									
002E	18	03	JR	0033									
0033	2A	B130	LD	HL,(30B1)	30B1<23	30B2<9D							
0036	18	0C	JR	0044									
0044	11	DD62	LD	DE,62DD									
0047	19		ADD	HL,DE									
0048	7C		LD	A,H									
0049	B5		OR	L									
004A	20	08	JR	NZ,0054									
004C	3A	8034	LD	A,(3480)	3480<FF								
004F	E6	40	AND	40									
0051	C2	0D01	JF	NZ,010D									
010D	DF		RST	1B	30DF>01	30DE>0E							
0018	C3	100E	JF	0E10									
0E10	E3		EX	(SP),HL	30DE<67	30DF<01							
0E11	D5		PUSH	DE	30DF>00	30DE>00							
0E12	F5		PUSH	AF	30DD>62	30DC>DD							
0E13	33		INC	SP	30DB>40	30DA>10							
0E14	EB		EX	DE,HL									
0E15	3E	CB	LD	A,CB									
0E17	32	7430	LD	(3074),A	3074>CB								
0E1A	1A		LD	A,(DE)	0167<3E								
0E1B	32	7530	LD	(3075),A	3075>3E								
0E1E	3E	C9	LD	A,C9									
0E20	32	7630	LD	(3076),A	3076>C9								
0E23	13		INC	DE									
0E24	21	9F30	LD	HL,309F									
0E27	CD	7430	CALL	3074	30DA>0E	30D9>2A							
3074	CBFE		SET	7,(HL)	309F<C0	309F>C0							
3076	C9		RET		30D9<05	30DA<70							
7005	FF		RST	3B	7006<FF	30DA>70*..H... C9 003D 0168 309F 30DB							

Figure 6.2-2  
 Code E2 72-Character  
 Single-Line  
 Disassembly Format

Address	Op-Code	Operand	Op-Code Mnemonic	Operand	Data Transfer	Data Transfer	Breakpoint	Flag Register	Accumulator	BC Register Pair	DE Register Pair	HL Register Pair
0000	C3	2B00	JP	002B								
002B	31	E030	LD	SP,30E0								
002E	18	03	JR	0033								
0033	2A	B130	LD	HL,(30B1)	30B1<23	30B2<9D						
0036	18	0C	JR	0044								
0044	11	DD62	LD	DE,62DD								
0047	19		ADD	HL,DE								
0048	7C		LD	A,H								
0049	B5		OR	L								
004A	20	08	JR	NZ,0054								
004C	3A	8034	LD	A,(3480)	3480<FF							
004F	E6	40	AND	40								
0051	C2	0D01	JP	NZ,010D								
010D	DF		RST	18	30DF>01	30DE>0E						
0018	C3	100E	JP	0E10								
0E10	E3		EX	(SP),HL	30DE<67	30DF<01						
					30DF>00	30DE>00						
0E11	D5		PUSH	DE	30DD>62	30DC>DD						
0E12	F5		PUSH	AF	30DB>40	30DA>10						
0E13	33		INC	SP								
0E14	EB		EX	DE,HL								
0E15	3E	CB	LD	A,CB								
0E17	32	7430	LD	(3074),A	3074>CB							
0E1A	1A		LD	A,(DE)	0167<3E							
0E1B	32	7530	LD	(3075),A	3075>3E							
0E1E	3E	C9	LD	A,C9								
0E20	32	7630	LD	(3076),A	3076>C9							
0E23	13		INC	DE								
0E24	21	9F30	LD	HL,309F								
0E27	CD	7430	CALL	3074	30DA>0E	30D9>2A						
3074	CBFE		SET	7,(HL)	309F<E0	309F>E0						
3076	C9		RET		30D9<05	30DA<70						
7005	FF		RST	38	7006<FF	30DA>70*..H... C9 003D 0168 309F						

Figure 62-3  
 Code E3 72-Character  
 Single-Line  
 Disassembly Format

Address	Op-Code (IX Register)	Operand	Op-Code Mnemonic (IY Register)	Operand	Data Transfer (Stack Pointer)	Data Transfer	Breakpoint	Flag Register (Flag Register)	Accumulator (Accumulator)	BC Register Pair (BC Register Pair)	DE Register Pair (DE Register Pair)	HL Register Pair (HL Register Pair)
07E4	6F		LD	L,A								
07E5	7C		LD	A,H								
07E6	CE 00		ADC	A,00								
07E8	27		DAA									
07E9	67		LD	H,A								
07EA	3A 7534		LD	A,(3475)	3475<4E							
07ED	E6 08		AND	08								
07EF	C9		RET		305A<BB	305B<07						
07BB	C2 B807		JP	NZ,07BB								
07B8	CD D007		CALL	07D0	305B>07	305A>BB						
07D0	3D		DEC	A								
07D1	78		LD	A,B								
07D2	A7		AND	A								
07D3	CA F007		JP	Z,07F0								
07D6	3A 7E30		LD	A,(307E)	307E<57							
07D9	83		ADD	A,E								
07DA	5F		LD	E,A								
07DB	3A 7F30		LD	A,(307F)	307F<4E							
07DE	8A		ADC	A,D								
07DF	57		LD	D,A								
07E0	7D		LD	A,L								
07E1	CE 00		ADC	A,00								
07E3	27		DAA									
07E4	6F		LD	L,A								
	IX:3000	IY:0000			SF:305A			..... 16 1B00 7D11 2416				
								..... 00 0000 0000 0000				

---

### 6.3 LINE ASSEMBLER

The NSC800 Line Assembler allows you to enter and assemble NSC800/Zilog Z80 instructions into either the target system's memory or the EM-800's overlay memory. The line assembler recognizes all standard Z80 mnemonics as well as certain "assembler directives" detailed in Section 6.3.2. The line assembler gives you a powerful software tool to aid in hardware/software debugging and software patching. It is a tool for creating small hardware/software checkout routines, patching existing software, developing software, debugging software, etc. It is not designed as an all-purpose editor/assembler software development package.

NOTE:

The assembler uses EM-800 scratch RAM addresses 3100H--32BFH.

#### 6.3.1 How to Use the Line Assembler

The line assembler assumes that a terminal is attached to the EM as described in Section 6.1.

NOTE:

Many terminals generate only a RETURN when the RETURN key is pressed. Some terminals generate a RETURN/LINEFEED. Often this function is programable through a switch. If your terminal does not generate a LINEFEED, then the In-line Assembler will overwrite the mnemonic you typed in with the assembled output. If your terminal does generate a LINEFEED, then the mnemonic you typed remains and the assembled output is displayed on the next line.

In the examples shown here, all entries and assembled responses are shown as if there is a RETURN/LINEFEED.

To invoke the line assembler, enter on the emulator keyboard:

```
<code> <C> <0>
```

The following response will appear on the terminal screen:

```
APPLIED MICROSYSTEMS CORPORATION  
NSC800 SINGLE LINE ASSEMBLER  
VERSION X.X  
(C) COPYRIGHT 1983
```

```
0000>
```

At this point, lines may now be entered and assembled into target memory. When you want to stop using the line assembler, type the pseudo-operation "END" after the address prompt as in this example:

```

F004 > LD A,D
F004 7A
F005 > END
*** End of Line Assembly ***

```

The line assembler may be used to invoke the memory disassembler by entering a RETURN as the only entry on a line. This causes the memory disassembler to display the instruction located at the current value of the location counter and increment the location counter.

Example:

```

F000 > <return >
F000 20F2 JR NZ, EFF4
F002 > <return >
F002 F620 OR 20
F004 > END <return >
*** End of Line Assembly ***

```

### 6.3.2 Features Supported

All standard mnemonics are supported. The following assembler directives are supported by the NSC800 Line Assembler:

Table 6.3-1  
Assembler Directives

Assembler Directives	
ORG	Sets the address pointer
=	Sets the value of a LBL
END	Terminates the line assembler
DEFT	Defines an ASCII text string
DEFB	Defines a byte of memory
DEFW	Defines a word of memory
LBL	A temporary line assembler value
\$	Represents the value of the address pointer
PRE	Turns preview on or off
ORG	Initially the address pointer is 0000H. Use ORG to change the address pointer.  -To change the address pointer to 0064H, enter ORG 100
=	Use the equal sign with LBL to set or recall a local label value.  -To set label 1 to 100H, enter LBL1 = 100H <return >  -To see the value of label 3, enter LBL3 = <return >
END	Terminates the line assembler and returns control to the emulator keyboard.



- 
- DEFT** To enter a target string into memory, type:  
DEFT text text text <return>
- DEFB** To enter a single byte directly into memory, type:  
DEFB AFH <return>
- DEFW** To enter a two-byte word, type:  
DEFW AFAFH <return>
- \$** The current value of the address pointer.
- PRE** Toggles between the two forms of the line assembler prompt:  
-The simple address pointer >  
-A preview of the code at the address pointer.
- LBL** The Single Line Assembler provides up to ten local labels in the form of LBL0 through LBL9.

**NOTE:**

No labels are initialized upon entering the Line Assembler. You must define a label before using it. The allowable label operations are defined below.

- LBLn =** Prints the current label value
- LBLn = number** Assigns the value "number" to the label
- LBLn = \$** Assigns the value of the current location counter to the label

Any mnemonic and all assembler directives except "ORG" and "END" may be preceded by a label.

**Line Editor**

Because the line assembler operates on a single line of text at any one time, there are only simple line editing functions. These are:

- Ctrl-H or Backspace** Deletes the character before the cursor
- Ctrl-I or Tab** Enters spaces to the next tab stop in the line (tab stops are 10,20)

---

## Numeric Bases

Numbers may be entered in either decimal, hexadecimal, binary or octal forms. All numbers output by the Line Assembler are in hex format. The following table illustrates the input formats:

<u>Form</u>	<u>Format</u>
Decimal	100
Hexadecimal	100H
Binary	10110B
Octal	7342O or 7342Q

## 6.4 MEMORY DISASSEMBLER

### 6.4.1 Overview

The NSC800 Memory Disassembler allows you to disassemble target memory (either overlay or user memory) and display the disassembled code on a terminal. The range of display is specified by loading the beginning and end registers prior to entering the disassembler (see below).

### 6.4.2 How to Use the Disassembler

To invoke the memory disassembler, first load the begin and end registers with the desired address range, i.e.

```
<reg> <4> <load data> ... enter starting address  
<reg> <5> <load data> ... enter ending address
```

Next, start the disassembly by entering on the emulator keyboard:

```
<code> <C> <A>
```

You can control the display by entering

Ctrl-S (stop)

or

Ctrl-Q (resume)

---

### 6.4.3 Disassembler Output Format

The format of disassembled target memory is illustrated here:

F000	C206F0	JP NZ, F006
F003	E3	EX (SP), HL
F004	C5	PUSH BC
F005	C9	RET

### 6.4.4 Errors

Upon detection of an error, the line assembler displays an error message and then prints a prompt. The location counter is not updated. The two possible types of error messages are:

Syntax error            Indicates a syntax error in the input line

Target write error      Indicates failure of store operation to target system (or overlay)

If you have a syntax error, retype your input line. If you have a target write error, there are several possible conditions:

- The RAM overlay switches are not on.
- You may be trying to store code in an incorrect area, such as an address range where no memory is available.
- There is a possible problem in the target system, such as defective RAM or an addressing error (trying to write to ROM).



---

# **SECTION 7**

## **BUILT-IN DIAGNOSTICS**

### **(CODE FUNCTIONS)**

---

- 7.1      GROUP A: MEMORY TESTS**
  - 7.2      GROUP B: OSCILLOSCOPE LOOPS**
  - 7.3      GROUP C: MEMORY LOAD AND DUMP**
  - 7.4      GROUP D: MISCELLANEOUS**
  - 7.5      GROUP E: CHANGE DEFAULT PARAMETERS**
  - 7.6      GROUP F: INTROSPECTION MODE**
-

The Diagnostic Emulator contains built-in test functions and utility routines designed to be convenient and useful for testing systems and verifying their proper operation. These test and utility routines have been named Code Functions because they are accessed by depressing the CODE keyswitch, followed by hexadecimal digits designating the routine desired. Table 7-1 lists all of the Code Functions programmed into the Diagnostic Emulator.

Table 7.1-1  
Code Functions

---

<b>GROUP A: MEMORY TESTS</b>			
A1	RAM TEST	(00/FF)	
A2	RAM TEST	(Rotating 1s)	Repeating Tests
A3	RAM TEST	(Addresses)	
A4	ALL RAM TESTS		
A5	ALL RAM TESTS		
A6	RAM TEST	(00/FF)	
A7	RAM TEST	(Rotating 1s)	One Pass and Stop
A8	RAM TEST	(Addresses)	
<b>GROUP B: OSCILLOSCOPE LOOPS</b>			
B1	Repetitive Memory Read		
B2	Repetitive Memory Write		
B3	Repetitive I/O Read		
B4	Repetitive I/O Write		
B5	Continuous Address Increment, 64K Range		
B6	Repetitive Memory Write (Data/Panel PROM)		
B7	Repetitive I/O Write (Data/Data)		
BC	Repetitive Store/Examine Memory		
BD	Repetitive Store/Examine I/O Port		
BE	Store Rotating 1-Bit to Memory		
BF	Store Rotating 1-Bit to I/O Port		
<b>GROUP C: MEMORY LOAD AND DUMP</b>			
C0	Line Assembly Mode (See Section 6.3)		
C1	Load Target from Front Panel PROM		
C2	Verify Target with Front Panel PROM		
C3	Load Target from Serial Link (DOWNLOAD)		
C4	Dump Target to Serial Link (UPLOAD)		
C5	Load RAM Overlay from Target		
C6	Verify RAM Overlay from Target		
C7	Verify Target against Serial Link		
C8	Fill Target with Specified Data		
C9	Verify Target with Specified Data		
CA	Disassemble Memory (See Section 6.4)		
CB	Block Move Target System Data		
CC	Dump Target to Serial Link in User Viewable Format		
CE	Repeat Segment of Data over an Entire Block		

---

---

## GROUP D: MISCELLANEOUS

D0	Clear Interrupt Enable Flip-Flop
D1	Set Interrupt Enable Flip-Flop
D2	Display Clock Frequency
D3	Display PROM/ROM Signature
D4	Output 50 Nulls from Serial Link
D5	Call User Routine in Internal RAM at 3000 <sub>16</sub>
D6	Call User Routine in Internal RAM at 3003 <sub>16</sub>
D7	Clear Trace Memory
D8	Dump Entire Content of Trace Memory
D9	Halt CPU (for changing Front Panel PROM)
DA	Display Revision Number for Control PROM
DB	Display Revision Number for Disassembly PROM
DC	Calculate Branch Offset and Display
DD	Do Self-Test of Control PROM and Disassembly PROM
DE	Output (CR) (LF)(NUL) from Serial Link
DF	Count Hours, Minutes and Seconds on Display

## GROUP E: CHANGE DEFAULT PARAMETERS

E0	Disable Disassembly (Default at Power-On)
E1	Enable Disassembly, 80 Character Line, One Line of Registers
E2	Enable Disassembly, 72 Character Line, One Line of Registers
E3	Enable Disassembly, 72 Character Line, Two Lines of Registers (IX, IY, SP and Prime registers on Line 2)
EF	Call Remote Control Software

## GROUP F: INTROSPECTION MODE

F	Set Basic "Introspection" Mode
F0-F9	Set Introspection Mode and Initialize Emulator for Debug of User Code Function

## CODE A2—ROTATE 1's

Code Function A2 memory test routine performs a test on all data bits in the range specified. The range tested is from the address contained in the BEG register through the address contained in the END register. The routine starts with the first location in the range and tests the location by writing and checking a bit, one bit at a time, in all of the positions of the word under test. The routine writes and checks by writing and reading the following data patterns:

Binary Pattern	Hexadecimal
0 0 0 0 0 0 0 1	01 <sub>16</sub>
0 0 0 0 0 0 1 0	02 <sub>16</sub>
0 0 0 0 0 1 0 0	04 <sub>16</sub>
0 0 0 0 1 0 0 0	08 <sub>16</sub>
0 0 0 1 0 0 0 0	10 <sub>16</sub>
0 0 1 0 0 0 0 0	20 <sub>16</sub>
0 1 0 0 0 0 0 0	40 <sub>16</sub>
1 0 0 0 0 0 0 0	80 <sub>16</sub>

---

## 7.1 GROUP A: MEMORY TESTS

### CODE A1—00/FF DATA TEST

The Code Function A1 memory test routine quickly determines whether all location within a specified range can be set to 00<sub>16</sub> and FF<sub>16</sub>. The range tested is from the address specified by the BEG (begin) register through the address specified by the END register. The routine operated by setting the first location of the range to 00<sub>16</sub> and reading the location to see if a 00<sub>16</sub> is returned. Then the routine stores an FF<sub>16</sub> to this location and reads the location to see if an FF<sub>16</sub> is returned. Finally, the routine increments the address and tests the next location in the range. During the execution of this test, the address and data activity are visible on the displays and stored in the Trace Memory.

If a memory error is encountered, the routine emits three beeps and displays the address of the failure and the erroneous data read. At this time, you have three options:

1. Depress EXAM to display the data the routine expected to read from the memory. Release the keyswitch to again display the bad data.
2. Depress INC to continue testing at the next address in the range. If additional problems are found, the program will stop again and any of the options listed may be taken.
3. Exit the test routine by using any of the mode keys (MEM, I/O, REG, RUN, etc.) or RESET.

After testing all locations in the specified range, the EM-800 emits one short beep and repeats the test. The RESET keyswitch is used to terminate the test at any time.

After a location has been successfully tested, all bit positions in the location may be set and cleared independently of each other. The program then increments to the next sequential address in the range and proceeds to test in the same manner. If an error is detected, the test stops, the EM-800 emits the three beeps that signify an error, and the Display Panel shows the defective memory address and the bad data. At this point you have three options:



- 
1. Depress EXAM to display the good data the diagnostic routine expected to read. Release EXAM to return the bad data to the display.
  2. Depress INC to continue testing. If additional problems are found, the test stops and any of the options listed may be taken again.
  3. To terminate the test, depress RESET, RUN, or any of the mode select keyswitches.

After testing all locations in the specified range, the EM-800 emits one short beep and repeats the test. The RESET keyswitch is used to terminate the test at any time.

### **CODE A3—ADDRESS TEST**

Code Function A3 memory test determines whether an address decoding failure exists in the memory system under test. It tests the memory range from the address contained in the BEG register to the address contained in the END register. The routine prepares for operation by clearing all locations in the range to  $00_{16}$ . Next, the first location is set to  $FF_{16}$  and then a check is made of all address-related locations in the range to determine if any of them have been altered by the writing of the  $FF_{16}$ . After all locations in the range that are address-related to the first location have been checked, the program resets the first location to  $00_{16}$ , then the next sequential location in the range is set to  $FF_{16}$ , and the address-related locations checked. The test proceeds until all locations in the specified range have been set to  $FF_{16}$ , and the respective address-related locations checked.

For the purposes of this test, an address is said to be related to a second address if it differs from it by only one bit (in any bit position). The test checks all possible address-related combinations as long as a generated address does not fall outside the specified range.

If an addressing error is found the test stops, the EM-800 emits three beeps, and the display shows the erroneous data and its address. At this point you have three options:

1. Depress EXAM to display the data the diagnostic routine expected to read. Release EXAM to return the erroneous data back to display.

- 
2. Depress INC to continue testing. If additional problems are found, the test will stop and any of the options listed may be taken again.
  3. To terminate the test depress RESET, or RUN or any of the mode selection keyswitches.

After all locations in the specified range have been tested, the EM-800 emits one short beep and repeats the test. To exit this code function at any time, depress RESET.

#### **CODE A4—ALL TESTS AND REPEAT**

Code Function A4 executes the A1, A2, and A3 diagnostic functions in sequence, then emits a short beep and repeats. The test may be terminated by depressing RESET. If an error is found, you may respond in any of the ways described for the individual diagnostic functions.

#### **CODE A5—ALL TESTS AND STOP**

Code Function A5 executes the A1, A2, and A3 diagnostic functions in sequence, emits a short beep and stops. If an error is found, you may respond in any of the ways described for the individual diagnostic functions. When the test is complete, the displays will read CODE A5 and the Trace Memory will contain a record of the last 252 bus transfers.

#### **CODE A6—00 FF DATA TEST**

Code Function A6 is identical to the Code Function A1 except the function stops after a single pass through the test. When the test is complete, the displays will read CODE A6 and the Trace Memory will contain a record of the last 252 bus transfers.

#### **CODE A7—ROTATE 1's**

Code Function A7 is identical to the Code Function A2 except that the function stops after a single pass through the test. When the test is complete, the displays will read CODE A7 and the Trace Memory will contain a record of the last 252 bus transfers.

#### **CODE A8—ADDRESS TEST**

Code Function A8 is identical to the Code Function A3 except the function stops after a single pass through the test. When the test is complete, the displays will read CODE A8 and the Trace Memory will contain a record of the last 252 bus transfers.

---

## 7.2 GROUP B: OSCILLOSCOPE LOOPS

The Oscilloscope Loop Functions are a group of functions that provide several types of repetitive stimuli to a target system. They provide repetitive waveforms in the target system hardware that may easily be examined at various circuit points with an oscilloscope or other test equipment. Many of the functions are also useful as stimulus routines for Signature Analysis testing.

### NOTE:

ADDR register referred to below is  
"Hex Keyswitch - #6."

### CODE B1—REPETITIVE MEMORY READ

This function repetitively reads the single memory location addressed by the ADDR register. These address, data and RD signals are all shown on the Display. A high-going pulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the memory location is read. Depress RESET to exit this function.

### CODE B2—REPETITIVE MEMORY WRITE

This function repetitively writes the data contained in the DATA register to the single memory location addressed by the ADDR register. The address, data and RD signals are all shown on the Display. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the memory location is read. Depress RESET to exit this function.

### CODE B3—REPETITIVE I/O READ

This function repetitively reads the single I/O port location addressed by the ADDR register. The address, data and RD signals are all shown on the Display. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the I/O port is read. Depress RESET to exit this function.

### CODE B4—REPETITIVE I/O WRITE

This function repetitively writes the data contained in the DATA register to the I/O port location addressed by the ADDR register. The address, data and WR signals are all shown on the Display. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the I/O port is read. Depress RESET to exit this function.

### CODE B5—CONTINUOUS ADDRESS INCREMENT

This function places the EM-800 in a special mode in which it outputs successive addresses from  $0000_{16}$  to  $FFFF_{16}$  at a very high rate. Internal to the EM-800 this is accomplished by forcing a NOP instruction to the processor on every fetch cycle. Externally the EM-800 appears to be doing a fetch cycle at each address at the full speed of the processor (as determined by the clock frequency).

In this mode, the processor does **not** respond to target system WAIT commands.

---

The Continuous Address Increment function is used to check out address decoding networks in hardware systems and as a stimulus for signature analysis trouble-shooting.

It is possible to obtain a sync pulse for triggering an oscilloscope or a signature analyzer from either the Breakpoint A or Breakpoint B output at the back Panel Auxiliary Connector; the output pulse occurs each time the processor reads from the breakpoint address. (The processor does not stop.)

Depress RESET to terminate the Continuous Address Increment mode.

#### **CODE B6—REPETITIVE MEMORY WRITE (DATA/ $\overline{\text{DATA}}$ )**

This function repetitively writes data to the address designated by the ADDR register. The data written is that contained in the DATA register except it is complemented every other time data is written. The address, data and RD signals are all displayed. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the address is accessed. Depress RESET to exit this function.

#### **CODE B7—REPETITIVE I/O WRITE (DATA/ $\overline{\text{DATA}}$ )**

This function repetitively writes data to the I/O port designated by the ADDR register. The data written is that contained in the DATA register except it is complemented every other time data is written. The address, data and WR signals are all displayed. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the address is accessed. Depress RESET to exit this function.

#### **CODE BC—REPETITIVE MEMORY WRITE/READ**

This Code Function writes the data contained in the DATA register to the address designated by the ADDR register, then reads the same address; this process is repeated at a high rate. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the memory address is accessed for either the write cycle or the read cycle. Depress RESET to exit this function.

#### **CODE BD—REPETITIVE I/O WRITE/READ**

This function writes the data contained in the DATA register to the I/O port specified in the ADDR register, then reads the same port address; this process is repeated at a high rate. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the I/O port is accessed for either the write cycle or the read cycle. Depress RESET to exit this function.

---

### **CODE BE—WRITING ROTATING BITS TO MEMORY**

This function repetitively writes a rotating bit pattern to the memory location selected by the ADDR register. The data contained in the DATA register is written to the memory location. Then each bit in the DATA register is shifted left one bit position each time the memory location is written to. When the bit shifts out of the high-order position, it is re-entered in the low-order position. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the I/O port is read. Depress RESET to exit this function.

### **CODE BF—WRITING ROTATING BITS TO I/O PORT**

This function repetitively writes a rotating bit pattern to the I/O port selected by the ADDR register. The data contained in the DATA register is written to the I/O port. Then each bit in the DATA register is shifted left one bit position each time the memory location is written to. When the bit shifts out of the high-order position, it is re-entered in the low-order position. A high-going impulse is output from the BKPT A pin (pin 12) of the Auxiliary Connector each time the I/O port is read. Depress RESET to exit this function.

## **7.3 GROUP C: MEMORY LOAD AND DUMP**

### **CODE C1—LOAD TARGET FROM FRONT PANEL PROM**

The Code Function C1 transfers data from the Front Panel Diagnostic PROM to the target system. This routine requires you to specify the destination address range in the target system by entering the first address of the range in register BEG and the last address of the range in register END. To use this Code Function, first enter the appropriate address values in the BEG and END Panel PROM into the target address space with the first location in the PROM transferred to the first address of the specified range. After the transfer is complete, the Trace Memory contains a record of the last 252 cycles of the transfer.

### **CODE C2—VERIFY TARGET WITH FRONT PANEL PROM**

The Code Function C2 compares the Front Panel PROM with the address range you specify. The address range should be specified using the BEG and END registers in the same manner as described for C1.

---

### **CODE C3—LOAD TARGET FROM SERIAL LINK (DOWNLOAD)**

The Code Function C3 transfers hex data from the serial RS-232C input to the target system. The data to be entered must first be converted into the Intel MDS\* format, which is an ASCII-hexadecimal format.

The destination address range is specified by the incoming data and need not be specified in the BEG and END registers. Furthermore, if the data is properly received, the BEG and END registers contain the low and high limits of the loaded data, regardless of the initial register settings. In addition, the limits will always be correct even if non-contiguous data is loaded.

To use the Code Function, connect the RS-232C input to the source of information, start this routine and then enable the source to "download" the appropriate data. During the transfer, note the displays showing the data being loaded. If there are no errors, the end-of-file record completes the transfer and the displays contain CODE C3. The Trace Memory contains a record of the last 252 cycles of the transfer.

**Error Codes**—During the data transfer process, various types of errors can occur. If an error occurs, the Diagnostic Emulator emits three beeps and displays the appropriate error code. The Trace Memory will contain a record of the address and erroneous data. Once an error is detected the transfer process is aborted and may not be resumed. The C3 Code Function Error Codes are listed in Table 7.3-1.

\*MDS is an Intel Trademark.

Table 7.3-1

C3 and C7 Code  
Function Error Codes

CODE	DESCRIPTION
01	Framing Error. The serial data character is not properly framed by start and stop bits. This error may be caused by an incorrect setting of the baud rate selector or by noise on the transmission link.
02	Overrun Error. This error may occur if the processor is operated with an extremely low clock frequency while receiving data at high baud rates.
11	Non-Hexadecimal Character Received. This error indicates that a hex character was expected at some point, but a non-hex character was received.
12	Sum-Check Error. In the Intel MDS format, each record contains an eight-bit check-sum to ensure data integrity. If this sum is incorrect, this error code is given.
16	Non-Zero Record Type. If the record type byte is other than zero (except for the end-of-file record), this error is signaled.
21	Target Memory Write Error. If an attempt is made to load data to an area of memory containing no RAM or faulty RAM, this error occurs. This error is detected by doing a read-back-check of each location as it is stored, and recording both the write cycle and the read cycle in Trace Memory. After the error occurs, register ADDR will contain the data the EM-800 attempted to store.

#### CODE C4—DUMP TARGET TO SERIAL LINK

This Code Function C4 transfers data from a selected area of Target Memory to the serial RS-232C output. The data being output is ASCII-Hexadecimal and is compatible with the Intel MDS format. The use of this function requires you to specify the address range. The BEG register contains the starting address while the END register contains the address of the last location to be output.

To use this Code Function, first specify the address limits, next prepare the receiving device to accept data, then start the transfer by executing CODE C4. During the transfer the display shows the address and data currently being transmitted. When transmission is completed, the displays show CODE C4 and the Trace Memory contains a record of the last 252 cycles.

The rate of transfer can be controlled by the receiving device. If enabled by the Option-switch (with position 3 open), the CTS line (Clear-to-Send) can prevent output if held in the marking (negative) condition. In the spacing (positive) condition, output speed is determined by the baud rate selected.

Each record is followed by a RETURN, line feed and two null characters.

---

### **CODE C5—LOAD OVERLAY RAM FROM TARGET MEMORY**

The Code Function C5 transfers data from a selected area of target memory space to the equivalent area in Overlay Memory. To use this Code Function, the Overlay Memory must first be located at the proper address by rotating the thumb-wheel switch. The Overlay is then enabled by setting the selector switch to the appropriate position. Then the BEG and END registers are set to the range of addresses over which data is to be transferred. The last step is to call the Code Function to execute the transfer. While executing, the displays show the addresses and data. When data transfer is completed, the displays show CODE C5 and the Trace Memory contains a record of the last 252 cycles of the data transfer.

If a non-verify occurs during the transfer, the Diagnostic Emulator emits three beeps and temporarily halts the transfer. The error may be skipped and the transfer resumed by depressing INC, or the operation may be aborted by depressing a mode select keyswitch, such as CODE. While the operation is halted, the address and the data that failed to verify are shown on the display. By depressing and holding the EXAM keyswitch, the correct target data may be displayed.

### **CODE C6—VERIFY RAM OVERLAY WITH TARGET MEMORY**

The Code Function C6 compares data from a selected area of Target Memory to the equivalent area in Overlay Memory.

For information on the operation of this function, see Code C5.

### **CODE C7—VERIFY TARGET WITH SERIAL LINK**

The Code Function C7 is nearly identical to the C3 Code Function. It differs in two respects:

1. Data is not stored to target memory but only verified.
2. A non-verify results in Error 22 and the compare operation is aborted. Register ADDR will contain the address of the non-compare while register DATA will contain the data that was supposed to be in the target memory location.



---

### **CODE C8—FILL MEMORY WITH DATA**

The Code Function C8 is used to fill a block of target memory or RAM Overlay with the same data, usually all one's (FF) or all zeros. To use the Code Function, set the BEG and END registers to the range of target memory or RAM Overlay to be filled, load the DATA Register with the data to be stored, then execute CODE C8. The Display shows the transfer as it takes place. After transfer is completed the display shows CODE C8 and the trace contains a record of the last 252 cycles of the transfer.

If a location fails to store the correct data, the Diagnostic Emulator emits three beeps and temporarily halts the fill operation. The error may be skipped and the transfer resumed by depressing INC, or aborted by depressing a mode select keyswitch such as TRACE. While the operation is halted the address and the data that failed to verify are shown on the Display. By depressing and holding EXAM, the correct data (which was in DATA) may be displayed.

### **CODE C9—VERIFY MEMORY WITH DATA**

The Code Function C9 compares a block of target memory or RAM Overlay with the byte in register DATA. See the explanation of CODE C8 for the operation of the function.

### **CODE CB—BLOCK MOVE**

The Code Function CB is used to move a block of data residing in the target system to a new location in target system RAM. Define the block of data to be moved by entering the address of the first byte of the block in the BEG register and the address of the last byte of the block in the END register. Enter the address of the first byte of the destination block in the ADDR register. Execute Code Function CB to move the data.

This routine is able to move the block of data to a higher address or to a lower address. In addition, the blocks may overlap in any manner and move without loss of data; for example, a block of 2K bytes could be moved up or down by fifteen positions.

The entire destination block must be in writeable memory.

### **CODE CC—DATA OUTPUT TO SERIAL PORT IN HEX AND ASCII FORMAT**

This Code Function provides a formatted dump of a block of memory to the serial port. The memory block is defined by the addresses contained in the BEG and END registers. When the function is executed, data will be output from the serial port in the format shown in Figure 7.3-2.

Figure 7.3-2  
Memory Dump Format

ADDRESS <sup>16</sup>	DATA <sup>16</sup>	ASCII
0000	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	.....
0010	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	.....
0020	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F	!*"#\$%&'()*+,-./
0030	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F	0123456789:;<=>?
0040	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F	@ABCDEFGHIJKLMNO
0050	50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F	PQRSTUVWXYZ[\]_^
0060	60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F	`abcdefshijklmno
0070	70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F	perstuvwxyz{ }~.
0080	80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F	.....
0090	90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F	.....
00A0	A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF	.....
00B0	B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF	.....
00C0	C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF	.....
00D0	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF	.....
00E0	E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF	.....
00F0	F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE 00	.....
0100	07 07 DD 77 00 07 DD 77 01 07 DD 77 02 07 DD 77	...w...w...w...w
0110	03 07 DD 77 04 07 DD 77 05 07 DD 77 06 07 DD 77	...w...w...w...w
0120	07 C9 21 00 10 DD 21 00 30 0E 00 1E 00 41 0C 78	..!...!.0...A.x
0130	A9 A1 28 1F 47 AB 5F 16 86 78 A3 28 02 16 C6 78	..(.G...x.(...x
0140	06 07 0F 38 02 10 FB 36 DD 23 36 CB 23 70 23 72	...8...6.#6.#F#r
0150	23 18 DA 36 E9 21 74 01 DD CB 07 86 DD CB 06 86	#..6.!t.....
0160	DD CB 05 86 DD CB 04 86 DD CB 03 86 DD CB 02 86	.....

NOTE: Memory data formatted into lines of 16 bytes with the address of the first byte at the left margin.

**CODE CE—REPEAT SEGMENT OF DATA OVER BLOCK**

The purpose of Code Function CE is to make replicas of a block of data throughout a larger block. For example, suppose you want to have a block of identical jump instructions throughout memory. Follow these steps:

- Enter the instruction (3 bytes) in the first three locations of the block.
- Enter the length of the block (3) in the DATA register.
- Enter the address of the first location of the block in the BEG register (this is the same address as the first byte of the jump instruction).
- Enter the last address of the block in the END register.

When the function is executed, the jump instruction will be copied as many times as it will fit in the specified block.

---

#### 7.4 GROUP D: MISCELLANEOUS

##### **CODE D0—CLEAR INTERRUPT ENABLE FLIP-FLOP**

The interrupt enable flip-flop of the emulation processor is cleared so that the next time the processor starts running, interrupts will not be allowed.

##### **CODE D1—INTERRUPT ENABLE FLIP-FLOP**

The interrupt enable flip-flop of the emulation processor is set so that the next time the processor starts running, interrupts will be allowed.

##### **CODE D2—DISPLAY CLOCK FREQUENCY**

This Code Function is a routine that determines the clock frequency of the emulation processor by comparing the instruction execution rate of the processor with the EM-800 internal 1.2 KHz reference frequency. The internal reference frequency is derived from the crystal controlled UART clock. The frequency is displayed on the ADDRESS display and is given in kilohertz. For example, an NSC 800 operating with a 4.0 MHz clock will display 4000 (kilohertz) on the ADDRESS displays when CODE D2 is executed. The result is accurate to about  $\pm .01\%$  (the accuracy of the UART crystal).

##### **CODE D3—DISPLAY PROM/ROM SIGNATURE**

The purpose of this Code Function is to provide a convenient way of verifying that all bits in a PROM or ROM are correct. The routine operates by reading each 8-bit byte in a specified range and shifting the bits into a firmware implemented feedback shift register. By this means, the routine calculates a 16-bit check value that is displayed as a 4-digit hexadecimal signature on the ADDRESS display. This signature has a very high probability (.9998) of being unique for any given bit pattern in a ROM.

A PROM or ROM signature is obtained by setting the first address of the ROM in the BEG register and the last address of the ROM in the END register; then executing the routine. The code Function routine will calculate and display the ROM signature. If the correct signature has been obtained previously with a known good ROM, then the ROM under test is good if it has the same signature.

There is a technique that may be used to create PROMs or ROMs whose signatures are zero. For the method to work, the last two locations of the ROMs must be unused. Proceed as follows:

1. Program a PROM with the desired information, making sure that the last two bytes are zeros.
2. Determine the signature of the PROM using the CODE D3 function.
3. Program a new PROM with the desired information, but replace the last two bytes, which previously were zero, with the bit pattern of the signature obtained in Step 2.

---

Now, while the signature of the new PROM is being calculated, the routine will arrive at a point just prior to processing the last two bytes of the PROM and at that time, the shift register will contain the signature of the PROM as calculated in Step 2; entry of the last two bytes, containing the same bit pattern as that already in the shift register, results in the shift register reaching a final value of zero when computation is complete. Thus, the PROM will have a signature of 0000.

**CODE D4—OUTPUT 50 NULLS TO SERIAL PORT**

This Code Function outputs 40 null characters (00<sub>16</sub>) to the serial port for the purpose of providing leader or trailer for users using punched paper tape as a data storage media. There are no parameters for this Code Function.

**CODE D5—CALL USER ROUTINE IN INTERNAL RAM AT 3000<sub>16</sub>.**

**CODE D6—CALL USER ROUTINE IN INTERNAL RAM AT 3003<sub>16</sub>.**

These two Code Functions provide a means for you to transfer control to routines that have been entered into the EM-800 internal scratch pad RAM for various reasons. To make use of this feature, you must understand the requirements of the programs that run in the EM-800 internal environment. See Section 8—User Implemented Code Functions.

**CODE D7—CLEAR TRACE MEMORY**

The EM-800 Trace Memory is cleared when power is applied as part of the power-on-reset operations. Code Function D7 is used to clear the Trace Memory at any other time. This routine does not use any parameters.

**CODE D8—DISASSEMBLE AND OUTPUT ENTIRE CONTENT OF TRACE MEMORY (IF DISASSEMBLER FIRMWARE IS INSTALLED)**

This Code Function outputs the entire content of the Trace Memory to the serial port in the standard disassembler format (Code E2, 72-character lines\*). This routine may be called even if the regular disassembly feature of the EM-800 is disabled. Data output may be suspended for a moment by depressing the EXAM key; when the key is released, data output will continue. See Section 6—Disassembly.

*\*The disassembly format can be changed by executing Code E1 or E3 before executing Code D8.*

---

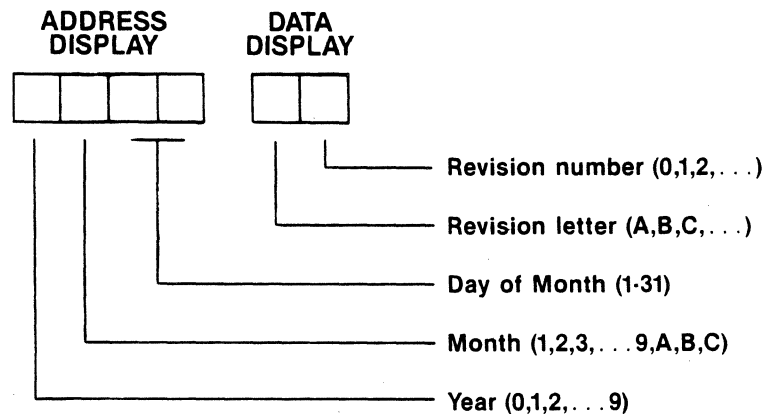
### CODE D9—HALT CPU

Code Function D9 causes the CPU to execute a HALT instruction, thereby halting the CPU. It is recommended that the CPU be halted any time that an EPROM is inserted into or removed from the Front Panel Diagnostic PROM Socket to avoid the possibility of crashing the internal control program of the EM-800. After the CPU has been halted, RESET must be used to resume normal operation. There are no parameters for this function.

### CODE DA—DISPLAY REVISION NUMBER FOR CONTROL PROM

### CODE DB—DISPLAY REVISION NUMBER FOR DISASSEMBLY PROM

These two Code Functions display the data and revision information for the control PROM software and the disassembly PROM software respectively. The format is as follows:



### CODE DC—CALCULATE BRANCH OFFSETS

Code Function DC is intended to simplify the task of calculating branch offsets for NSC800 relative branch instructions. To use this routine:

- Enter the address of the destination of the branch instruction in the BEG register.
- Enter the address of the byte **following** the branch instruction in the ADDR register.
- Execute the function and the Address Display will show two digits that are the required branch offset.

If the required offset is too large to be reached by branch instructions, four digits will be displayed showing the offset.

---

### **CODE DD—SELF TEST OF INTERNAL PROM DATA**

Code Function DD is used to perform a check of the data in the internal PROMS —the Control PROM and the Optional Disassembly PROM. When this function is called, the data display will show 01 while the first 4K (Control PROM) is being tested and 02 while the second 4K (Disassembly PROM) is being tested.

A failure of this test will result in three beeps and the display will show EC 31 if the Control PROM failed or EC 32 if the Disassembly PROM failed.

### **CODE DE—OUTPUT LINE ENDING SEQUENCE TO SERIAL PORT**

This Code Function outputs the line ending sequence to the serial port consisting of a **RETURN**, a line feed, and two null characters. The routine is used to obtain a new line on a CRT or other ASCII display.

### **CODE DF—DISPLAY HOURS, MINUTES, SECONDS**

Code Function DF places the EM-800 in a clock mode that counts hours, minutes and seconds on the Address and Data displays. To set the initial display, enter the desired hours and minutes into the ADDR register and the desired seconds display into the DATA register. Then execute the function to start the clock. If the initial values are set to zero, then the clock will indicate elapsed time from 0000 00 to 1259 50 (13 hours).

## **7.5 GROUP E: CHANGE DEFAULT PARAMETERS**

### **CODE E0—DISABLE DISASSEMBLY (DEFAULT)**

Code Function E0 is used to disable the disassembly software if it is in operation. See Section 6—Disassembly.

### **CODE E1—ENABLE DISASSEMBLY**

Code Function E1 enables the disassembly firmware and configures the firmware to output 80-character lines with one line of register display. See Section 6.

### **CODE E2—ENABLE DISASSEMBLY**

Code Function E2 enables the disassembly firmware and configures the firmware to output 72-character lines with one line of register display. See Section 6.

### **CODE E3—ENABLE DISASSEMBLY**

Code Function E3 enables the disassembly firmware and configures the firmware to output 72-character lines with two lines of register display. This is the only format that displays all the NSC 800 internal registers of general interest. See Section 6, Disassembly.

### **CODE EF—CALL REMOTE CONTROL SOFTWARE**

Code EF disables the EM keyboard, placing it into remote control mode, so that commands received through the serial port control the emulator.

**GROUP F:  
INTROSPECTION MODE****CODE F—SET INTROSPECTION MODE**

Execution of this Code Function sets the EM-800 so that its own internal address space becomes the “target system.” After execution of the CODE F function, memory examine and store operations will be directed to the EM-800 internal address space; programs internal to the EM-800 may be executed in a single-step mode and other internal operations performed. See Section 8, User-Implemented Code Functions.

**CODE F0, F1, . . . F9**

Code Functions F0 through F9 are used to set up the EM-800 to debug user programs residing in the front panel Diagnostic PROM Socket. These functions each:

- Set the emulator into “introspection” mode so that the internal address space is accessible.
- Set the stack pointer to 3060<sub>16</sub> (the top of the internal user RAM area).
- Set the program counter to the starting address of the respective user Code Function.

The EM-800 is then ready to execute a user program in single-step mode or at full speed; breakpoints may be set and registers examined, and other normal debugging activities carried out.





---

# **SECTION 8**

## **USER-IMPLEMENTED CODE FUNCTIONS**

---

- 8.1 OVERVIEW**
  - 8.2 INTERNAL ENVIRONMENT**
  - 8.3 ENTRY TO USER CODE FUNCTIONS**
  - 8.4 INTROSPECTION MODE**
  - 8.5 GETTING TO AND FROM  
THE TARGET SYSTEM**
  - 8.6 USER-ACCESSIBLE SUBROUTINES**
  - 8.7 INTERRUPTS**
  - 8.8 CODE FUNCTION EXAMPLES**
-

---

## 8.1 OVERVIEW

The EM-800 Diagnostic Emulator has a low-insertion-force socket on the front panel that is designed to accept EPROMs similar to the Intel 2716 or 2732 devices. This front panel socket is called the Diagnostic PROM Socket. The purpose of the Diagnostic PROM Socket is to provide a means for you to insert EPROMs programmed with your own diagnostic routines and execute them in a convenient manner from the EM-800 Keyboard. Your routines may perform almost any imaginable function. In most cases, you will probably write special test or diagnostic routines to help test portions of the target system for which no Built-In Code Functions are provided. This discussion provides a view of the internal environment of the EM-800 from a programmer's perspective and is intended to provide the information you need to write and debug your own Code Functions.

You are already familiar with the environment of your own target system. There is a 64K byte address space called the Memory Address Space, and within this address space are various blocks of ROM, RAM and (in some systems) I/O control or data registers. In addition, there is a 256-byte I/O address space which, in most systems, contains the addresses of I/O devices.

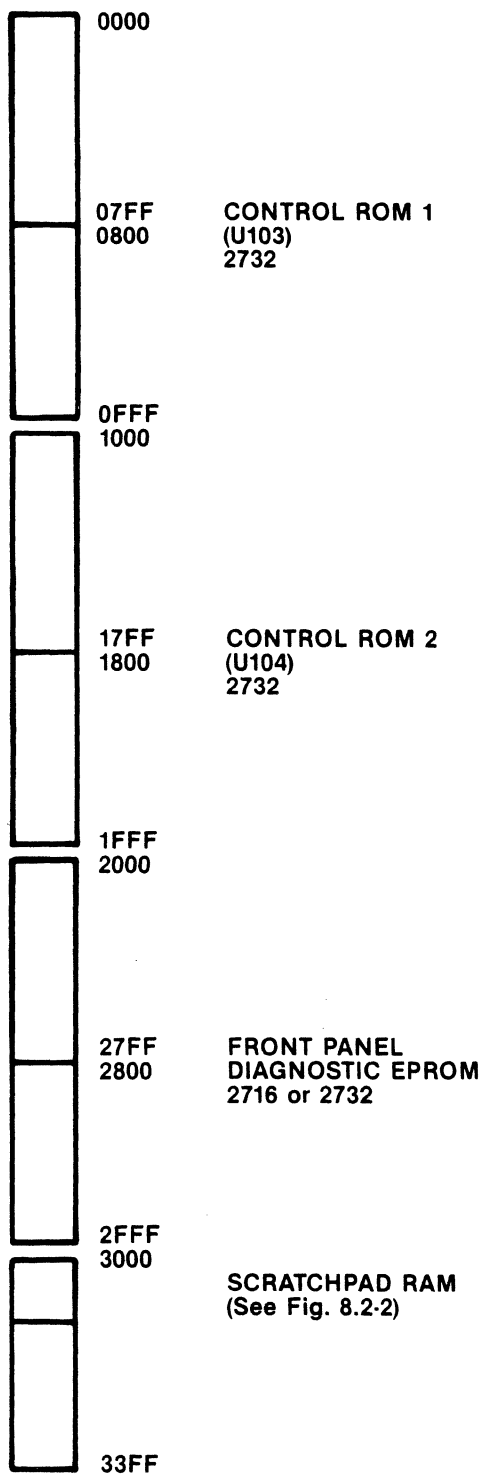
The EM-800 has an internal address space with its own ROM, RAM, and I/O. The EM-800 control program and Built-In Code Functions reside in this address space. Any EPROM plugged into the Diagnostic PROM Socket also appears in this internal address space. It is thus possible for User Code Functions to access all Diagnostic Emulator facilities and to function exactly as if they were factory-programmed.

However, the Code Function programs executing within the internal address space do not have direct access to your system's target address space. If it is necessary for a Code Function to read or write to the external target system, it must do so in cooperation with the Diagnostic Emulator hardware circuits. Consequently, a rigidly defined routine must be executed to perform read or write operations to the target system. The Built-In Code Functions, as well as the EXAMINE and STORE routines, use EM-800 control program subroutines, and you may also use these same subroutines to read and write to the target program address space.

## 8.2 INTERNAL ENVIRONMENT

The internal environment of the EM-800 contains ROM, RAM and I/O. The I/O devices of the EM-800 are memory mapped. Figure 8.2-1 shows an overview of the EM-800 internal address space.

Figure 8.2-1  
EM-800  
Internal Memory Map



---

### 8.2.1 ROM

The EM-800 has two sockets, located on the Keyboard circuit card, that accept EPROMs or ROMs that contain the control program for the unit. The circuit board connections are normally set up for EPROMs or ROMs having the Intel 2732 pinout; a jumper modification of the board allows use of the 2K byte 2716 as well. See Figure 8.2-1.

### 8.2.2 FRONT PANEL EPROM SOCKET

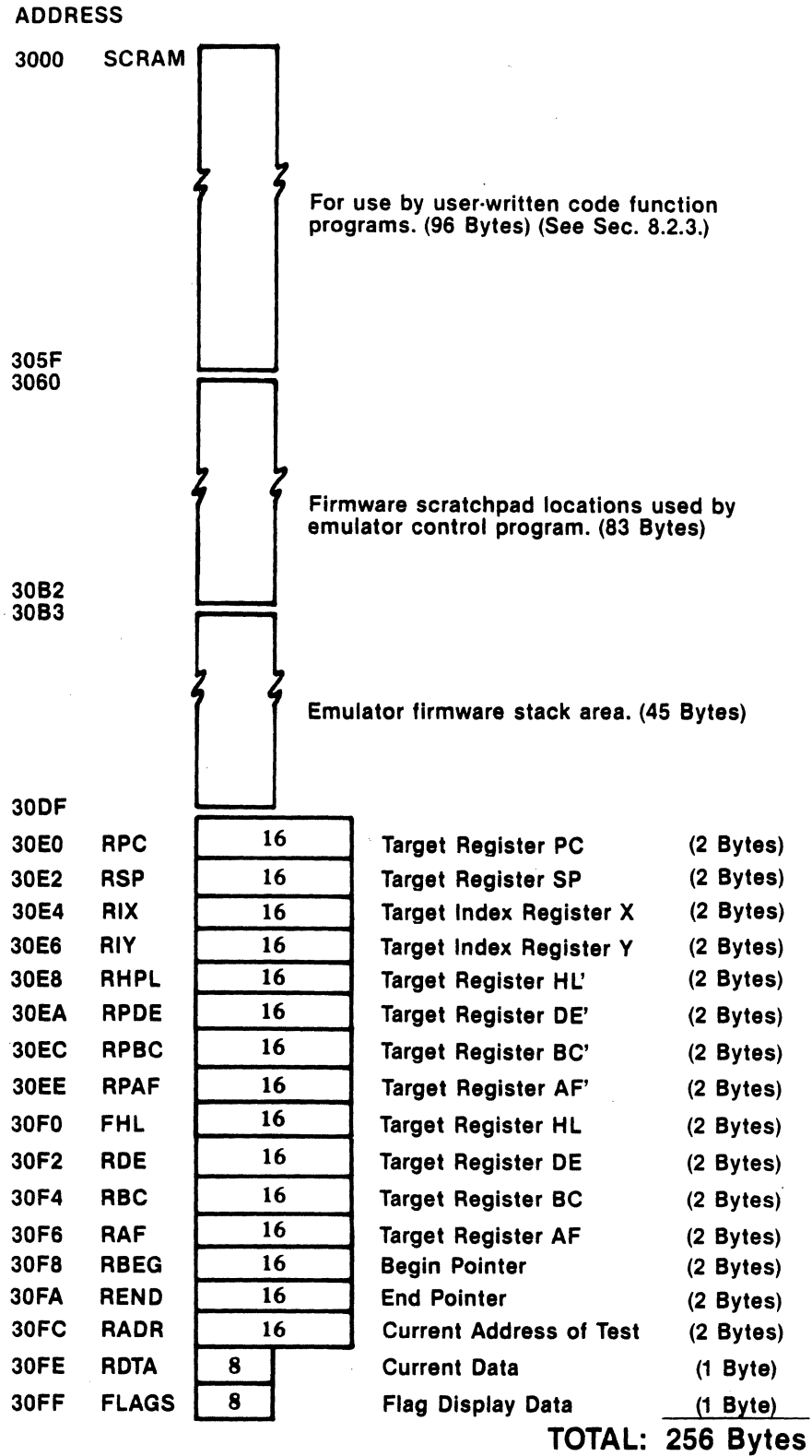
The EM-800 Front Panel EPROM Socket also accepts EPROMs of the 2716 or 2732 variety. A small switch located in the center of the socket selects the appropriate connections for either the 2K byte or 4K byte EPROM types. In the internal address space, the EPROM plugged into the Front Panel Socket will appear in the address range of  $2000_{16}$  to  $27FF_{16}$  (2716) or  $2000_{16}$  to  $2FFF_{16}$  (2732). It is not possible to have this EPROM appear in the external (target) address space. See Figure 8.2-1.

### 8.2.3 SCRATCHPAD RAM

The EM-800 also contains a small amount of Scratchpad RAM in the internal address space that is used by the control program in keeping track of the status of the emulator and the emulation processor. The Scratchpad RAM resides in the internal address space at addresses  $3000_{16}$  to  $33FF_{16}$ . Figure 8.2-2 shows a detail of the Scratchpad RAM. The first 96 bytes of the RAM are available to user-written Code Function programs. Locations  $3100_{16}$  to  $33FF_{16}$  are also available.

The Scratchpad RAM also contains the area where the processor registers are saved each time the emulation processor pauses; also, the saved register values are restored each time the emulation processor begins to run. User-implemented programs may obtain these register values or even alter them if desired. You should carefully avoid altering any of the data contained in the firmware stack area or firmware scratchpad locations to avoid crashing the control program.

Figure 8.2-2  
Map of Internal  
Scratchpad RAM



3100  
↓  
33FF  
Additional Scratch RAM  
(See Sec. 8-2.3.)

---

## 8.2.4 I/O DEVICES

The EM-800 control program, running in the internal address space, has access to various I/O registers associated with different components of the emulator. You may wish to create special programs that control the EM-800 components in a way different than that provided for by the standard software, and for that reason, this detailed information is provided. In most cases, however, you will be able to obtain the desired result by using I/O handler subroutines that are already present in the EM-800 firmware. Section 8.6 provides information on the characteristics and use of the User-Accessible Subroutines.

**KEYBOARD:** The state of the Keyboard keyswitches may be read by the processor at a series of eight addresses from 3400<sub>16</sub> through 3407<sub>16</sub>. Four keyswitches may be read at each of the input addresses as shown in Figure 8.2-3. A key depression causes the corresponding bit to go low as seen in the input data. For example, if Key 9 is depressed, bit 1 of location 3402<sub>16</sub> will be low. Bits 4, 5, 6 and 7 of all eight of the input ports see the same data; bit 4 in all locations will be low if a jumper on the Keyboard called the Option A jumper is installed. Bit 5 in all locations senses the state of pin 61 on the keyboard card connector; this pin is left open in the EM-800. Bit 6 will be low if the most recent system reset was caused by the power-on-reset circuitry; bit 6 will be high if the most recent system reset was caused by the RESET Key or a reset command from the target system. Bit 7 will be high if any of the following keys is depressed: RUN, RUN-BKPT or STEP.

If you write software to directly read the Keyboard, you must be aware that there is no key debouncing or other processing of the key closure done by the hardware. Consequently, you must provide the keystroke debouncing, repeating key features or other special processing in the software that scans the Keyboard. There is a Keyboard scan routine already in the EM-800 that may be accessed and that provides the most commonly needed features. See Section 8.6.

**SERIAL INPUT/OUTPUT PORT:** The EM-800 Diagnostic Emulator contains circuitry that implements a full-duplex (two-way) serial Input/Output port that conforms to RS-232C requirements. The baud-rate, parity and character length of the data transmitted and received is set up by hardware switches. (See Sections 9.2 and 9.3). The nature and format of data transmitted is under the control of software. The software is able to send data to the serial output circuits, read data from the serial input circuits and test the status of the serial port circuitry via three ports as shown in Figure 8.2-4. Data is transferred to and from the serial port by means of a Universal Asynchronous Receiver-Transmitter (UART).

Data to be output through the serial port is written to the UART data write address. The data enters the UART transmit buffer register, and then enters the transmit shift register where it is shifted out in serial form bit by bit. New data may be written to the transmit buffer register as soon as the previous data has entered the transmit shift register and before it has completed the process of shifting out.

Figure 8.2-3  
Keyboard Input  
Locations

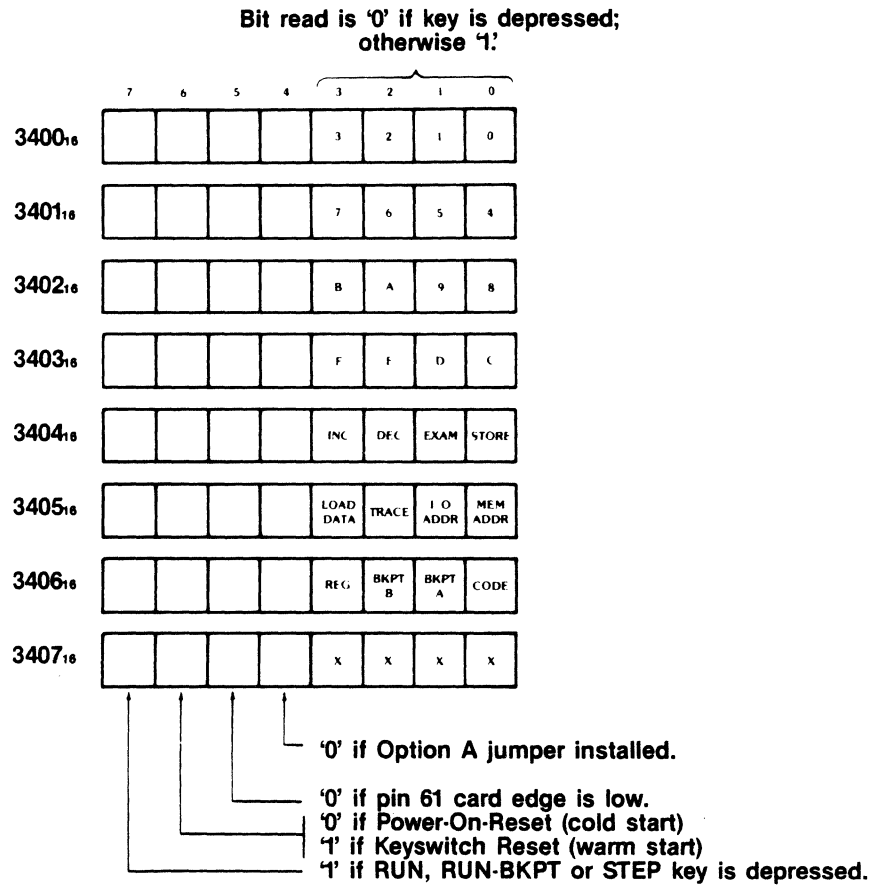
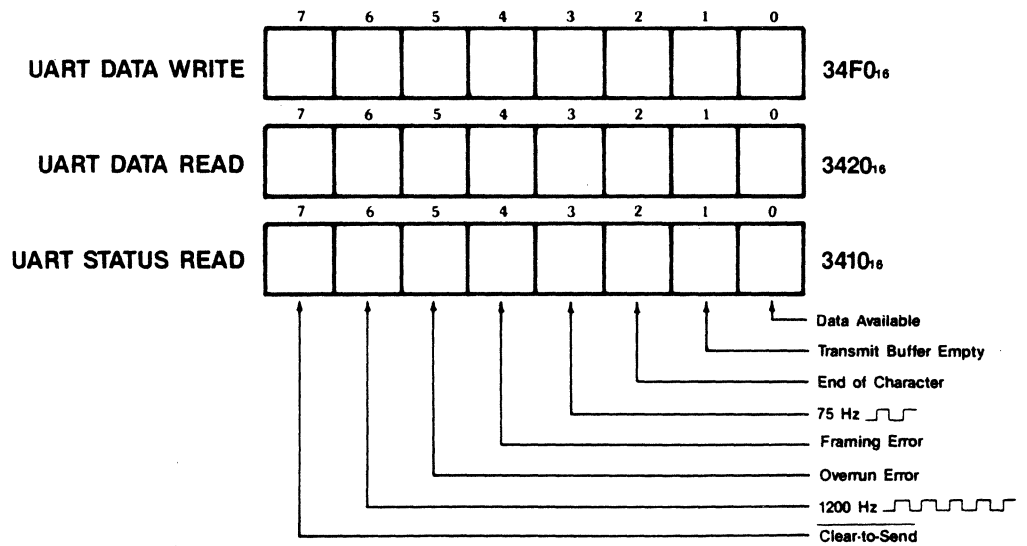


Figure 8.2-4  
Serial Port Data and  
Status Locations



---

The UART Status Register (Figure 8.2-4) contains two bits that inform the software of the status of the transmitter registers as follows:

**Bit 1, Transmit Buffer Empty**, will be read as a '1' when the transmit buffer register may be loaded with another character. A '0' means that the transmit data register contains data that has not yet been moved into the transmit shift register.

**Bit 2, End of Character**, will go to '1' at the time that a character has shifted out of the transmit shift register. If there is another character waiting in the transmit buffer register, then bit 2 will immediately go to '0' as the new character enters the shift register to be transmitted.

Data received by the EM-800 through the serial port is entered into the UART receiver shift register. When an entire character has been received, it is transferred to the receiver holding register and is then available to the software by reading the data at the UART Data Read address. Several status bits in the UART Status Register (Figure 8.2-4) give information about the received data as follows:

**Bit 0, Data Available**, goes to '1' when an entire character has been received and transferred to the receiver holding register. When the software reads the UART Data Read location, this bit is cleared to '0'.

**Bit 4, Framing Error**, goes to '1' if the received character has no stop bit at the expected location. This usually means that the transmitting device is sending characters of different length or baud rate than the EM-800 is set up to receive. Noise may also cause this error.

**Bit 5, Overrun Error**, goes to '1' if a previously received character in the receiver holding register is not read by the CPU before another character is received and transferred into the holding register.

The clear-to-send input (Auxiliary Connector, Pin 5) is visible to the software as Bit 7 of the UART status register. This bit is '0' if clear-to-send is true (high); if clear-to-send is low or disconnected, this bit is '1'. (Note, however, that this bit may be forced to the '0' state by setting Option Switch 3 closed. See Section 9.2.)



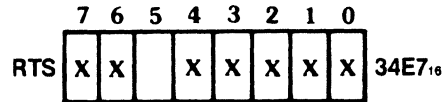
---

Two other bits share the UART Status Register, but are not directly involved in the communications functions.

**Bit 3, 75 Hz,** is a 75 Hz square wave that is derived from the bit-rate-generator crystal oscillator. This bit is seen by the software as alternate '1' and '0' with a 13.33 mSEC full cycle.

**Bit 6, 1200 Hz,** is a 1200 Hz square wave that is derived from the bit-rate-generator crystal oscillator.

One additional output port is associated with the communications interface. This port controls the Request-to-Send (RTS) signal that is output on Pin 4 of the Auxiliary connector.



The RTS output port is located at address 34E7<sub>16</sub>. Writing a '1' to Bit 5 of the port will set the RTS signal to its negative (marking, or OFF) state; writing a '0' sets the RTS signal positive (spacing or ON). All of the remaining bits of the port are "don't care" and have no effect.

**HEXADECIMAL DISPLAYS AND TRACE MEMORY:** The EM-800 Trace Memory is a 252-word by 32-bit memory whose primary function is to record each bus cycle that occurs to the target system. At any given time, a single word of the trace Memory is selected by an 8-bit register called the 'XADDR' (trace index address) register. If, for example, the XADDR register contains a 43, then the next bus cycle that occurs will be written into location 43 of the Trace Memory. Immediately after the data is written, the XADDR register is incremented (by hardware) so that the current Trace Memory address becomes 44. If the emulation processor is executing a target program, each bus cycle is written into the Trace Memory and XADDR is incremented for the next cycle: When XADDR reaches its maximum value FF<sub>16</sub> and is again incremented, it overflows to 00<sub>16</sub> so that the first location of the memory effectively follows the last location. Thus the Trace Memory may be viewed as a ring memory in which each additional bus cycle may be entered in the next position around the ring. Once the Trace Memory is full, each additional bus cycle simply overwrites the oldest bus cycle in the memory.

The Address and Data hexadecimal displays and the eight discrete Machine Cycle indicators are wired directly to the Trace Memory circuitry, so that the current Trace Memory word (the word designated by the XADDR register) is always displayed unless the displays are explicitly blanked.

---

When the EM-800 is in the PAUSE mode, the internal control program has access to the Trace Memory and the displays by a set of five ports. See Figure 8.2-5. The five ports are as follows:

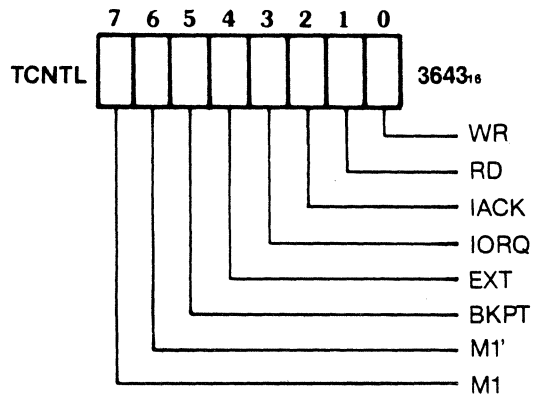
**XADDR (3600<sub>16</sub>)** This port gives access to the Trace Index Address register. The control program may read this location to obtain the current value of the XADDR register, and may store new values in the register. Storing a new value in the XADDR will change the current trace word that is accessed and displayed on the display panel.

**TDATA (3640<sub>16</sub>)** This port gives access to the eight-bit portion of the current Trace Memory word that records the data bus signals of each machine cycle. The current program may read this location to obtain the data portion of the current trace word, or may store new data to the data portion of the trace word.

**TADDL (3641<sub>16</sub>)** This port gives access to the eight-bit portion of the current trace word that records the low-order eight bits of the address bus of each machine cycle. The control program may read or write this location.

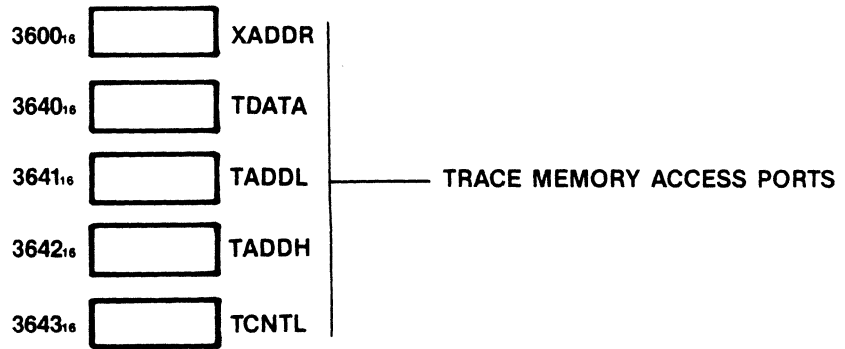
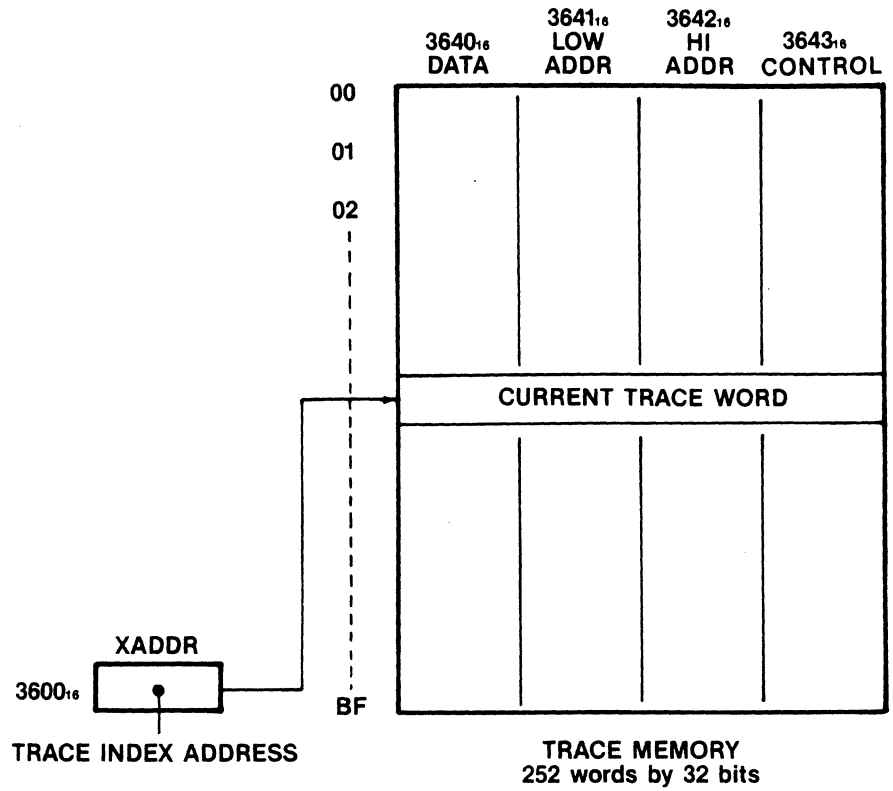
**TADDH (3642<sub>16</sub>)** This port gives access to the eight-bit portion of the current trace word that records the high-order eight bits of the address bus. The control program may read or write this location.

**TCNTL (3643<sub>16</sub>)** This port gives access to the eight-bit portion of the current trace word that records the control bits each machine cycle. The control bits are arranged in the port as shown below:



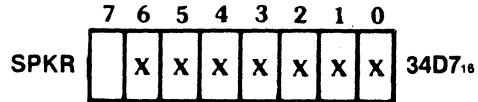
Any time that the control program writes new data into the Trace Memory, the data stored will immediately be seen on the appropriate displays.

Figure 8.2-5  
Trace Memory Format



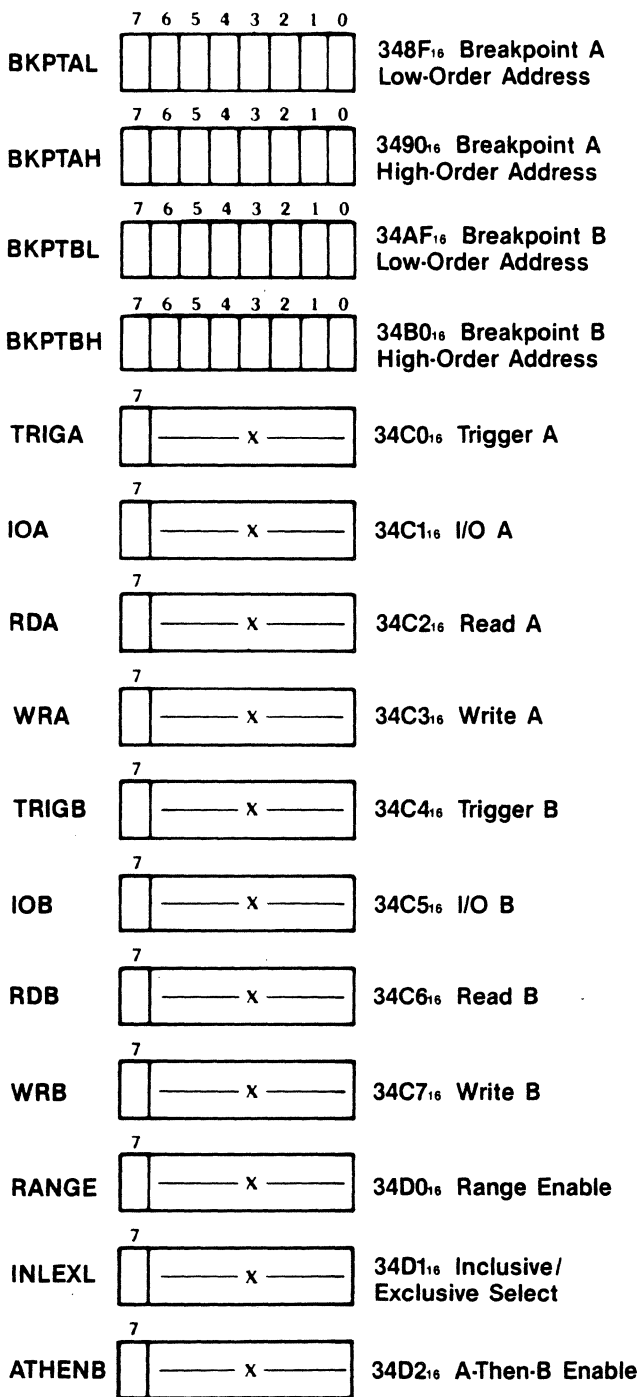
---

**SPEAKER:** The EM-800 incorporates a very small dynamic speaker located on the Keyboard printed circuit board. A port controls the current to the speaker to generate tones or other sounds under software control. It is necessary for the software to generate the actual waveform to be output by the speaker; there is no tone generation hardware in the EM-800. The speaker output port is diagrammed below:



Writing a '1' to Bit 7 switches DC current to the speaker ON; writing '0' to Bit 7 switches the current OFF. Bits 0 through 6 of the port are "don't care" bits and have no effect.

**BREAKPOINT COMPARATORS:** A series of output ports is used to set up the Breakpoint Comparator address values and to control the desired operating mode. These ports are detailed below:



---

The ports BKTAL and BKPTAH are used to set up the A Breakpoint address. The software should store the appropriate low-and high-order address bits to these ports. In the case of an I/O address which only has eight bits, the low-order breakpoint address should be loaded; the high-order breakpoint address port will have no effect.

The ports BKPTBL and BKPTBH are used to set up the B Breakpoint address in a manner analogous to that described above.

Two ports, TRIGA and TRIGB, are used to output pulses to the BKPT A and BKPT B output pins of the auxiliary connector (pins 12 and 13; see Figure 9.1-1). Storing a '1' to the bit 7 position of the ports results in a high level at the corresponding output pin. To generate an output pulse the software must store a '1' followed by a '0' to the bit 7 position of the port. Bit positions 1-6 of the ports have no effect on the system. Inclusion of these ports enables user-programmed Code Functions to output trigger signals to external equipment, such as oscilloscopes or signature analyzers.

Ports IOA and IOB are used to configure the breakpoint comparators to respond to either memory cycles or I/O cycles. Storing a '1' to the bit 7 position of either port will cause the associated breakpoint comparator to respond to I/O cycles with matching eight-bit low order address values. Storing a '0' to the bit 7 position of either port will cause the associated breakpoint comparator to respond to memory cycles with matching 16-bit address values. Bits 0 through 6 of these ports have no effect.

Ports RDA and RDB are used to enable the breakpoint comparators to respond to read cycles (either memory read cycles or I/O read cycles if a '1' is stored to bit position 7 of the associated port).

Ports WRA and WRB are used to enable the breakpoint comparators to respond to write cycles (either memory write cycles or I/O write cycles by storing a '1' to bit position 7 of the relevant port).

Notice that if '1' bits are stored to both the RDA and WRA ports, the A breakpoint comparator will respond to both read and write cycles. If a '0' is stored to both ports, the comparator will not respond to any cycles and is thus disabled. The B breakpoint comparator may be controlled in a similar manner.

Three ports, RANGE, INLEXL and ATHENB are used to configure the breakpoint circuitry for several special operating modes (see Section 4.1.4). Any address in the range from A to B may be detected by writing a '1' to bit 7 of the INLEXL port. In order to include both end points of the range (address = A and address = B), it is recommended that both the A and B comparators also be enabled for read and write cycles. Note that it is not possible to break on a range, only on the occurrence of a read or write cycle.

With a '1' stored to the INLEXL port, the inclusive range (everything from A to B) is detected. With a '0' stored to the INLEXL port, the exclusive range (every address outside of the range A to B) is detected. Again, the A and B comparators must also be set up to detect read and write cycles to ensure that both end points are included in the range.

The ATHENB port is used to set up sequential operation of the comparators so that a breakpoint stop signal is generated only when the B address is encountered after the A address. To set up this mode, store a '1' to the bit 7 position of the ATHENB port. Also, be sure the RANGE circuitry is disabled ('0' to RANGE port).

### 8.3 ENTRY TO USER CODE FUNCTIONS

The Code Functions that are built into the EM-800 are called with keystroke sequences that begin with a letter key, such as CODE A1, CODE C4 and CODE D2. The Code Functions that use the decimal digit keys (0-9) are reserved for calling user-programmed Code Functions. The keystroke sequences used to transfer control to user Code Functions are as follows:

Key Sequence	Transfer Address
CODE 0	2000 <sub>16</sub>
CODE 1	2003 <sub>16</sub>
CODE 2	2006 <sub>16</sub>
CODE 3	2009 <sub>16</sub>
CODE 4	200C <sub>16</sub>
CODE 5	200F <sub>16</sub>
CODE 6	2012 <sub>16</sub>
CODE 7	2015 <sub>16</sub>
CODE 8	2018 <sub>16</sub>
CODE 9	201B <sub>16</sub>

---

Thus, each of the key sequences has associated with it an entry address in the address space assigned to the Diagnostic Prom socket. It is your responsibility to properly code the EPROM so that the desired actions occur for each entry address. The first instruction of every Code Function must be a jump instruction (op-code C3<sub>16</sub>) because the control software examines the Diagnostic Prom for the presence of this data before transferring control to it. See the examples given in Section 8.8.

## 8.4 INTROSPECTION MODE

The EM-800 Diagnostic Emulator has been designed with a special feature that is mainly intended as an aid to testing and debugging Code Function programs programmed into EPROMs and plugged into the Diagnostic Prom socket. This special feature is the "Introspection Mode" in which the EM-800 turns its attention to its own internal address space. In this way, you may examine and store to the internal address space and, with certain limitations, may single-step programs that execute in the internal address space.

### 8.4.1 CODE F

The introspection mode is entered by the key sequence:



After entering the CODE F mode, you may examine or alter the internal memory space, step or run programs in the internal memory space, and review the contents of the trace memory after program execution. Breakpoints may also be used to halt program execution at appropriate internal addresses. The RESET key returns the EM-800 to normal operation.

## 8.5 GETTING TO AND FROM THE TARGET SYSTEM

The EM-800 Control Program, together with the built-in diagnostic routines and any user-programmed code functions, executes within the EM-800 internal "protected" address space. As a consequence, programs in this internal address space do not have direct access to the target address space, but must make use of special hardware in the EM-800 logic to make the target address space accessible. Code Function Programs may have a requirement from time to time to do one of the following things:

1. Read from or write to a location in the target address space.
2. Read from or write to an I/O port address in the target address space.
3. Go to and begin executing a program residing in the target address space.
4. Return from running a program in the target address space to a program (user code function routine) in the internal address space.

The following sections give detailed information on these functions.



---

## 8.5.1 EXAMINE AND STORE

The simplest method of reading and writing data to the target system is to use subroutines that exist in the EM-800 control program. Four subroutines are provided, as follows:

- STM** Store the data contained in the accumulator to the target address specified in register-pair HL.
- EXM** Load the accumulator from the target address specified in the register pair HL.
- STIO** Store the data contained in the accumulator to the target I/O port specified by register C.
- EXIO** Load the accumulator from the target I/O port specified by register C.

The entry addresses of these subroutines (and other useful subroutines) are given in Section 8.6, User-Accessible Subroutines.

The four subroutines shown above operate by performing a read or a write operation to the specified address after commanding the EM-800 hardware to make the target address space accessible during the transfer interval. The machine code listing of the EXM subroutine is shown below.

```

;
;Subroutine to read data from the target address
;specified by HL. Data is returned in A. The
;target read cycle is recorded in the trace memory.
EXM      PUSH BC
          LD A, OEOH      ;Delay value
          LD (DELAY), A   ;Command to hardware
          LD B (HL)       ;Read from target
;
;The data read during the target memory read
;cycle above is automatically recorded in the
;trace memory. The trace memory bookkeeping
;values must be updated.
          LD A, (XADDR)   ;Get trace index
          LD (XBASE), A   ;Save
          DEC A
          LE (TRACE), A
;
;Put data in place and return to caller.
          LD A, B
          POP BC
          RET
```

---

### 8.5.2 PAUSE to RUN

The EM-800 Control Program (firmware) is normally in control of the emulator when in the PAUSE mode. Depressing the RUN or RUN BKPT Key causes the control program to execute a sequence of operations that will load the processor registers with the values that had previously been saved (when the EM-800 last entered the PAUSE mode) and then does a coordinated jump to the target system program. The EM-800 hardware will switch to the target address space at the correct time for execution of the first instruction. It is possible for a user-written Code Function program to jump to a target system program in the same manner. This facility allows a user Code Function to place programs into the target address space (either user RAM or Overlay RAM) and then transfer control to that program. The target system program will then proceed to execute from within the target address space in a normal manner.

The best way for you to transfer control to a target system program is to use the existing EM-800 internal routine. The following steps are suggested:

1. Set up the user register save locations (addresses 30E0<sub>16</sub> to 30FF<sub>16</sub> in the scratchpad RAM) as desired. In particular, be sure to set the target program counter (RPC) location to the correct starting address. Other registers may be used, if desired, as a means of passing parameters to the target program.
2. If the target program needs parameters or data from the internal program, then transfer the data to the target RAM using the STORE subroutine as appropriate.
3. Perform the transfer of control to the target program by jumping to the RUN routine at address 00A5<sub>16</sub>.

The RUN routine will load the processor registers, do the required coordination of the EM-800 hardware, and start the target program running with the desired register values initialized.

### 8.5.3 RUN to PAUSE

When a program is executing in the target address space and you want to transfer control into the internal software, there are only three ways available to do this:

1. Reset the system.
2. Press the STEP key.
3. Cause a breakpoint to occur, either with one of the breakpoint comparators or by means of the external breakpoint input connection.

All three methods may be used during manual operation. The third method may also be used for a sort of automatic operation where a Code function sets up the conditions to enable a target program to get back to the internal environment when needed.

---

The following steps are suggested as a method that will allow a program executing in the target system address space to re-enter the internal address space:

1. A Code Function program sets up one of the breakpoint comparators to monitor a prearranged address in the target memory space.
2. The Code Function program sets up the re-entry jump address so that when the prearranged address is encountered and the breakpoint occurs, control will be given back to the Code Function program instead of the Keyboard Scan routine. See Section 8.5.4 below.
3. The Code Function program gives control to the target system program which then begins running.
4. At this point, depress the RUN BKPT Key to arm the breakpoint system.
5. When the target system program is ready to return control to the Code Function program, it accesses the prearranged address and causes the breakpoint to occur. After the EM-800 software has saved the processor registers, it will jump to the address specified in Step 2 (above) and the Code Function program may proceed.

#### 8.5.4 RE-ENTRY JUMP

Some applications require that the EM-800 control software transfer control to a user program each time emulation of the target program is halted. An example would be a "soft shutdown" program that prevents damage to the target system when execution is halted. (See Section 9.8, Soft Shutdown.) The EM-800 has the flexibility required to give control to a user-written subroutine each time the RUN to PAUSE sequence of the emulator is executed. Normally, this subroutine would be programmed into an EPROM and inserted into the front panel socket of the EM-800.

In normal operation, the EM-800 executes an internal RUN to PAUSE routine each time the target program is halted. This routine first saves the processor registers in the scratchpad RAM save area, then sets up the display to show the correct data, and finally goes to the keyboard input routine to determine the next action required. Before going to the keyboard routine, however, the RUN to PAUSE routine examines location  $30A0_{16}$  to see if it contains a jump instruction op-code ( $C3_{16}$ ). If it does, the EM-800 will regard the jump instruction as the first instruction of a user-supplied subroutine, and will call the subroutine. (The EM-800 calls the address of the jump instruction, which then jumps to the main body of the subroutine.)

---

A user-supplied subroutine will usually be located in the front panel EPROM, but may also be located in the user portion of the internal scratchpad RAM as the following example illustrates.

The following small program may be entered from the keyboard of the EM-800. It causes the EM-800 to beep each time it transfers from RUN to PAUSE. Enter the program with the following steps:

1. Reset the EM-800, then execute CODE F to place the emulator in "introspection" mode.

2. Enter the jump instruction:

at 30A0<sub>16</sub> enter C3<sub>16</sub>

at 30A1<sub>16</sub> enter 00<sub>16</sub>

at 30A2<sub>16</sub> enter 30<sub>16</sub>

These three bytes constitute a jump instruction to location 3000<sub>16</sub> in the internal address space. Location 3000<sub>16</sub> is the first location of the user portion of the scratchpad RAM.

3. At memory address 3000<sub>16</sub>, enter the following four-byte program:

at 3000<sub>16</sub> enter CD<sub>16</sub>

at 3001<sub>16</sub> enter D5<sub>16</sub>

at 3002<sub>16</sub> enter 00<sub>16</sub>

at 3003<sub>16</sub> enter C9<sub>16</sub>

4. Reset the emulator to exit the "introspection" mode and proceed to operate the emulator. Note that each time the emulator transfers from RUN to PAUSE, the beeper will sound.

In most practical cases, a user subroutine will be located in the front panel EPROM instead of RAM as was done in this example. Also, the jump instruction may be easily written into addresses 30A0 through 30A2 by a Code Function program also residing in the EPROM. Executing the Code Function will enable the user subroutine and a second Code Function could be written to disable the subroutine by changing the jump instruction op-code to 00<sub>16</sub> (or any other code except C3<sub>16</sub>).

## 8.6 USER-ACCESSIBLE SUBROUTINES

The EM-800 Control Program contains handlers and subroutines that you may use in constructing your own Code Functions. The entry addresses and functions of the routines are summarized in the following table.

Table 8.6-1  
User-Accessible  
Subroutines

ADDRESS	NAME	DESCRIPTION
1000 <sub>16</sub>	STM	Store the data contained in the accumulator to the target address specified in register-pair HL. (Flags, register F, are altered.)
1003 <sub>16</sub>	EXM	Load the accumulator from the target address specified in register-pair HL. (Flags and accumulator altered.)
106D <sub>16</sub>	CODEFN	Executes the built-in Code Function designated by the contents of the accumulator. For example, if the accumulator is loaded with A8 <sub>16</sub> and this subroutine is called then the EM-800 will execute Code Function A8; if the Code Function completes successfully, this subroutine will return to the calling program.
1006 <sub>16</sub>	STIO	Store the data contained in the accumulator to the target I/O port specified by register C. (Flags may be altered.)
1009 <sub>16</sub>	EXIO	Load the accumulator from the target I/O port specified by register C. (Flags and accumulator altered.)
101C <sub>16</sub>	LWLMIT	Compares HL to the BEGIN address; the subroutine returns with CY = if HL = BEGIN. (Registers A and F may be altered.)
101F <sub>16</sub>	HILMIT	Compares HL to the END address; the subroutine returns with CY = if HL = END. (Registers A and F may be altered.)
1076 <sub>16</sub>	SIA	ASCII Serial Input. Serial Data entered at the serial port is returned in the accumulator. Bit seven is always zero. Registers A and F may be altered. When this routine is called, the RTS line (request-to-send) automatically goes high. The RTS line goes low again whenever the XECUTE routine is entered.
102E <sub>16</sub>	SIB	Binary Serial Input. Serial Data entered at the serial port is returned in the accumulator. All eight bits are returned to the user without alteration. Registers A and F may be altered. When this routine is called, the RTS line (request-to-send) automatically goes high. The RTS line goes low again whenever the XECUTE routine is entered.

---

ADDRESS	NAME	DESCRIPTION
1031 <sub>16</sub>	RHB	Read-Hex-Byte. This subroutine obtains two ASCII characters from the serial port. It checks to see if the characters represent valid hexadecimal digits. If they do, they are then converted to an eight-bit binary number and the subroutine returns to the calling program with this result in the accumulator. If any characters received are not valid hexadecimal characters, an error code (EC 11) is displayed and the Diagnostic Emulator must be reset to proceed. This subroutine alters the accumulator, flags and D register. The D register is used to accumulate a sum check of the data received. (All the eight-bit binary values are added into register D, with overflows ignored.)
1034 <sub>16</sub>	CNVHEX	Convert ASCII character to binary value. This subroutine expects an ASCII character in the accumulator that represents one of the digits, 0, 1, 2, . . . 9 or one of the letters A, B, C, D, E, F. The character is converted into a numeric value corresponding to the ASCII-hex character in the accumulator. The routine returns with the value in the accumulator. If the ASCII character originally in the accumulator does not represent one of the hexadecimal digits, an error code will be displayed and the EM-800 must be reset to proceed.
1016 <sub>16</sub>	CRLF	Output a line ending sequence to the RX-232C port that consists of a RETURN (0D <sub>16</sub> ), line feed (0A <sub>16</sub> ), and two null characters (00 <sub>16</sub> ).
1013 <sub>16</sub>	TAB	This subroutine outputs ASCII space characters (20 <sub>16</sub> ) until the total number of printable characters output since the last RETURN character (0D <sub>16</sub> ) is equal to the number specified in the byte following the call to this routine. This routine is used to format output displays to CRT or printer terminals. No registers are altered.
1070 <sub>16</sub>	NIB	This subroutine converts the low-order 4 bits of the accumulator to one ASCII character representing the value in hexadecimal and outputs the character to the serial port.
1019 <sub>16</sub>	BYTE1	Converts the contents of the accumulator to two ASCII characters representing the value in hexadecimal, then outputs these characters to the RS-232 port. No registers are altered.

---

---

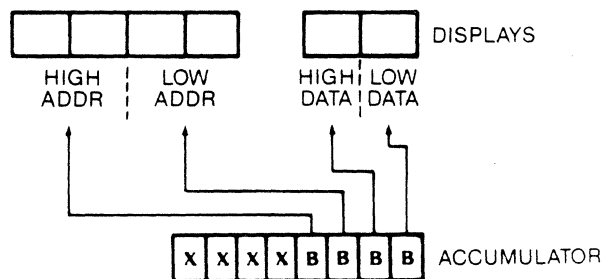
ADDRESS	NAME	DESCRIPTION
1037 <sub>16</sub>	HEXOUT	This subroutine is the same as the BYTE routine except that the contents of the accumulator are added to the D register for computing a checksum.
103A <sub>16</sub>	SO50NL	Output 50 ASCII null characters (00 <sub>16</sub> ) to the serial interface. This routine is primarily useful to produce blank tape leader on a paper tape punch interfaced to the EM-800.
1040 <sub>16</sub>	ERROR1	Subroutine to report an error on the display. This routine shows the characters "EC" on the address display (meaning Error Code), displays the contents of the accumulator in the data display, and emits three beeps. The routine then waits for you to depress some mode selection key. This subroutine does not return to the calling program.
1025 <sub>16</sub>	XECUTE	This routine returns program control to the standard Diagnostic Emulator firmware and readies it for keyboard input. If you have changed the display, it will return to its former state. This routine does not return to the calling program.
1043 <sub>16</sub>	KCN	User's keyboard scan routine. Data representing the keyswitch depressed is returned in the accumulator. In addition, if the keyswitch depressed is one of the hexadecimal numeric keys, the carry bit (CY) is set to one (true) when the subroutine returns. This routine ignores the RUN, BUN BKPT and STEP keyswitches. The table shows the data returned in the accumulator for each keyswitch depression.

KEYSWITCH	DATA	KEYSWITCH	DATA
RESET	*	0	00 <sub>16</sub>
RUN		1	01 <sub>16</sub>
RUN BKPT	**	2	02 <sub>16</sub>
STEP		3	03 <sub>16</sub>
ODE	18 <sub>16</sub>	4	04 <sub>16</sub>
BKPT A	19 <sub>16</sub>	5	05 <sub>16</sub>
BKPT B	1A <sub>16</sub>	6	06 <sub>16</sub>
REG	1B <sub>16</sub>	7	07 <sub>16</sub>
MEM ADDR	14 <sub>16</sub>	8	08 <sub>16</sub>
I/O ADDR	15 <sub>16</sub>	9	09 <sub>16</sub>
TRACE	16 <sub>16</sub>	A	0A <sub>16</sub>
LOAD DATA	17 <sub>16</sub>	B	0B <sub>16</sub>
STORE	10 <sub>16</sub>	C	0C <sub>16</sub>
EXAM	11 <sub>16</sub>	D	0D <sub>16</sub>
DEC	12 <sub>16</sub>	E	0E <sub>16</sub>
INC	13 <sub>16</sub>	F	0F <sub>16</sub>

\* When RESET is depressed, the User's Code Function is aborted and the EM-800 reinitialized.

\*\* Keyswitches RUN, RUN BKPT and STEP are ignored by this routine.

ADDRESS	NAME	DESCRIPTION
102B <sub>16</sub>	DSPCTL	Display Control. This subroutine uses the low-order four bits of the accumulator to control the blanking of the display digits. This is shown in the example below:



If a bit of the accumulator is a one when this subroutine is called, then the display digit or pair of digits corresponding to that bit illuminates. The accumulator and flags may be altered by this routine.



---

1073 <sub>16</sub>	BEEP	A subroutine to beep the speaker located on the Keyboard printed circuit card. No registers are altered.
0028 <sub>16</sub>	SOUT	Data in the accumulator is sent to the serial port. In addition, this subroutine affects a one byte counting location that is reset to zero each time a RETURN code is output (0D <sub>16</sub> ) and is incremented by one each time a printable ASCII character is output. This counting location is also affected and examined by the TAB routine as well as other output operations. This subroutine may also be called with an RST 5 instruction.
0030 <sub>16</sub>	IMSO	The data byte following the subroutine call or restart code is sent to the serial port. No registers are altered. this subroutine may also be called with an RST 6 instruction.

## 8.7 INTERRUPTS

Target system interrupts are invisible to programs executing in the EM-800 internal environment. For this reason, it is not possible to write Code Function programs that directly work or test your interrupt system. Nevertheless, it is possible for a Code Function program to test or work with interrupts as follows:

1. The internal Code Function program, when it begins executing, first copies the interrupt portion of the routine to target system RAM. (If no RAM is available in the target system, the RAM Overlay may be used.)
2. The Code Function program sets up one of the breakpoint comparators to facilitate re-entry into the internal environment.
3. The Code Function program sets up the re-entry jump address in order to gain control after the breakpoint occurs (See Section 8.5.4).
4. The Code Function program transfers control to the program copied to the target system.
5. When the breakpoint occurs, the internal program may read results left in RAM by the target system routine and take whatever additional action is desired. See Section 8.5.3.

## 8.8 CODE FUNCTION EXAMPLE

Two examples of Code Function programs are given in this section. The first example is a very simple routine that writes a range of target system memory to zeros.

---

**EXAMPLE 1:**

```

                                ORG 2000H
                                JP CODE0      ;CODE 0 ENTRY
                                JP XECUTE     ;CODE 1
                                JP XECUTE     ;CODE 2
                                JP XECUTE     ;CODE 3
                                JP XECUTE     ;CODE 4
                                JP XECUTE     ;CODE 5
                                JP XECUTE     ;CODE 6
                                JP XECUTE     ;CODE 7
                                JP XECUTE     ;CODE 8
                                JP XECUTE     ;CODE 9
;
;INITIALIZE—
CODE0      LD HL, 6000H      ;INITIALIZE MEMORY POINTER
           LD B, 00H        ;INITIALIZE BYTE COUNT
;
;NOW LOOP TO CLEAR EACH TARGET MEMORY LOCATION
;FROM 6000H to 60FFH
C1         SUB A            ;CLEAR ACCUMULATOR
           CALL STM         ;STORE TO TARGET SYSTEM
           INC HL          ;INCREMENT TO NEXT ADDRESS
           DEC B           ;DECREMENT BYTE COUNT
           JP NZ, C1       ;LOOP UNTIL COUNT EQUALS ZERO
;
EXIT TO CONTROL PROGRAM
           JP XECUTE
;
;DEFINE SUBROUTINE ADDRESSES
STM        EQU 1000H
XECUTE     EQU 1025H
           END
```

This example illustrates the following points:

1. The program originates at location 2000<sub>16</sub> because this is the start of the address range allocated for the Diagnostic PROM.
2. The first instruction tells the program to jump to the actual starting point of the CODE 0 program. This jump instruction provides room for the other entry points, each having its own jump instruction. In this simple example, only one Code Function is implemented; consequently a full set of jump instructions, as shown, is really not needed. Notice the CODE 0 entry point has a jump instruction to the program; all other entry points jump to a routine labeled XECUTE. The XECUTE routine is one of several ways to exit a Code Function, giving control back to the Diagnostic Emulator firmware. A RET instruction could also be used.
3. The Code Function program is written in standard NSC800 assembly language.

4. Whenever the Code Function program wishes to access the target system memory space, it may most easily do so by using subroutines already present in the Diagnostic Emulator firmware. In this example, the STM subroutine is called to perform the write operation to the target system.
5. When the Code Function program has finished executing, it returns control to the Diagnostic Emulator firmware by jumping to the XECUTE routine.
6. The EQU statements inform the assembler of the addresses of routines within the Diagnostic Emulator firmware—in this case, the addresses of the STM and XECUTE entry points.

#### EXAMPLE 2:

The second example is a scope loop program. This program rotates a bit through all eight positions of an output port. The port address used is selected as is usual for other built-in functions. The routine outputs a synchronizing pulse to the BKPT A output pin of the auxiliary connector just before rotating the bit and outputs a pulse to the BKPT B output pin after it has moved the bit through all eight positions of the output port. This program loops indefinitely and must be terminated with the RESET Key.

```

                                ORG 2000H
                                JP CODE0
CODE 0      LD SP, 03060H ;SET UP STACK
;
;OUTPUT START PULSE TO BKPT A OUTPUT—
C1         LD A, 0FFH
           LD (034C0H), A ;TURN BKPT A ON
           SUB A
           LE (034C0H), A ;TURN BKPT A OFF
;
;ROTATE BIT THROUGH OUTPUT PORT
           LD A, 030FDH ;PICK UP PORT ADDRESS
           LD C, A
           LD A, 01H ;SET ACCUMULATOR BIT
C2         CALL STIO ;OUTPUT TO TARGET PORT
           ADD A, A ;SHIFT BIT LEFT
           JP NZ, C2
;
;FLOW HERE WHEN THE BIT HAS SHIFTED OUT
;OF THE ACCUMULATOR. NOW OUTPUT THE STOP
;PULSE TO THE BKPT B OUTPUT.
           LD A, 0FFH
           LD (034C4H), A ;TURN BKPT B ON
           SUB A
           LE (034C4H), A ;TURN BKPT B OFF
;
;NOW REPEAT
           JP C1
;
;DEFINE SUBROUTINE ADDRESS
STIO       EQU 1006H
END

```



---

# **SECTION 9**

## **SUPPLEMENTARY INFORMATION**

---

- 9.1      AUXILIARY CONNECTOR**
  - 9.2      OPTION SWITCHES**
  - 9.3      SERIAL INTERFACE**
  - 9.4      UPLOAD/DOWNLOAD PROTOCOL**
  - 9.5      EXTERNAL BREAKPOINT**
  - 9.6      TRACE HOLD**
  - 9.7      SIGNATURE ANALYSIS**
  - 9.8      SOFT SHUTDOWN**
-

- 
- Pin 7**     **Signal Ground:** Connected in the EM-800 to the system logic ground, which is isolated from the protective ground (Pin 1). Note, however, that this ground is connected to the emulator probe ground pin; then when the EM-800 is connected to the target equipment, the target system logic ground and the EM-800 logic ground are connected together and to the ground system of the equipment plugged into the Auxiliary Connector.
- Pin 10**    **External Break-In:** A TTL level input with an internal 3.3K pull-up resistor. If this input is pulled low, the Diagnostic Emulator stops executing the target program as though STEP were depressed or an Internal Breakpoint were detected. (If the Diagnostic Emulator is already in PAUSE, this has no effect). This input stops execution even when the breakpoints are not enabled.
- Pin 11**    **Trace Hold (In):** A TTL level input with an internal 3.3K pull-up resistor. If the Diagnostic Emulator is executing a target program and this input is pulled low, further updating of the Trace Memory stops, although the program continues to execute. The contents of the Trace Memory are effectively frozen, and can be reviewed later after program execution has been halted.
- Pin 12**    **BKPT A and SA START (Out):** A TTL level output providing a high-going pulse at the time breakpoint conditions are satisfied for the Breakpoint A Comparator. This signal can be used to trigger an oscilloscope at a particular point of program execution. It can also be used as the START signal for a signature analyzer. This signal may be set high or low under software control when the Diagnostic Emulator is in PAUSE. This permits diagnostic routines to generate sync pulses or signature analyzer START signals under direct program control.
- Pin 13**    **BKPT B and SA STOP (Out):** A TTL level output associated with the Breakpoint B Comparator. It is functionally identical with the BKPT A signal described above.
- Pin 20**    **DATA TERMINAL READY:** This signal is driven to a nominal +12 volts to indicate that the EM-800 is ready to send data. Its signal state does not change.
- Pin 22**    **RUN (Out):** A TTL level output that is active (low) if the EM-800 is executing the target program or accessing the target address space.
- Pin 23**    **+5 VOLTS (Out):** Loading should not exceed .5 amp.
- Pin 24**    **GROUND:** This is the return line for the +5 volts available on Pin 23. This line is internally connected to the signal ground (Pin 7).
- Pin 25**    **SIGNATURE CLOCK (Out):** A TTL level output signal (RD) from the CPU. It is primarily used as a clock for signature analysis testing of equipment for which the EM-800 provides the stimulus.
-

## 9.2.1 OPTION JUMPERS

When you are using interrupting devices during your tests, you will have to set the option jumpers that are located in the pod. The option jumpers allow devices that use mode 2 interrupts and require an RETI response (such as PIO and SIO chips) to function properly. Set the option jumpers as follows:

JP2	JP3	DRIVE RAM OVERLAY DATA TO TARGET SYSTEM ON READ CYCLES
2-3	2-3	No
2-3	1-2	Yes, M1 cycles only*
1-2	2-3	No
1-2	1-2	Yes, all READ cycles*

\*Allows the instruction located in RAM overlay in the emulator to be sent to the target system.

The following figure shows the location of JP2 and JP3. To set the jumpers, simply insert a shorting jumper as needed for your application. You can, in effect, store the jumper on the 2-3 connection when mode 2 interrupts are not in use.

Figure 9.2-1  
Option Jumpers

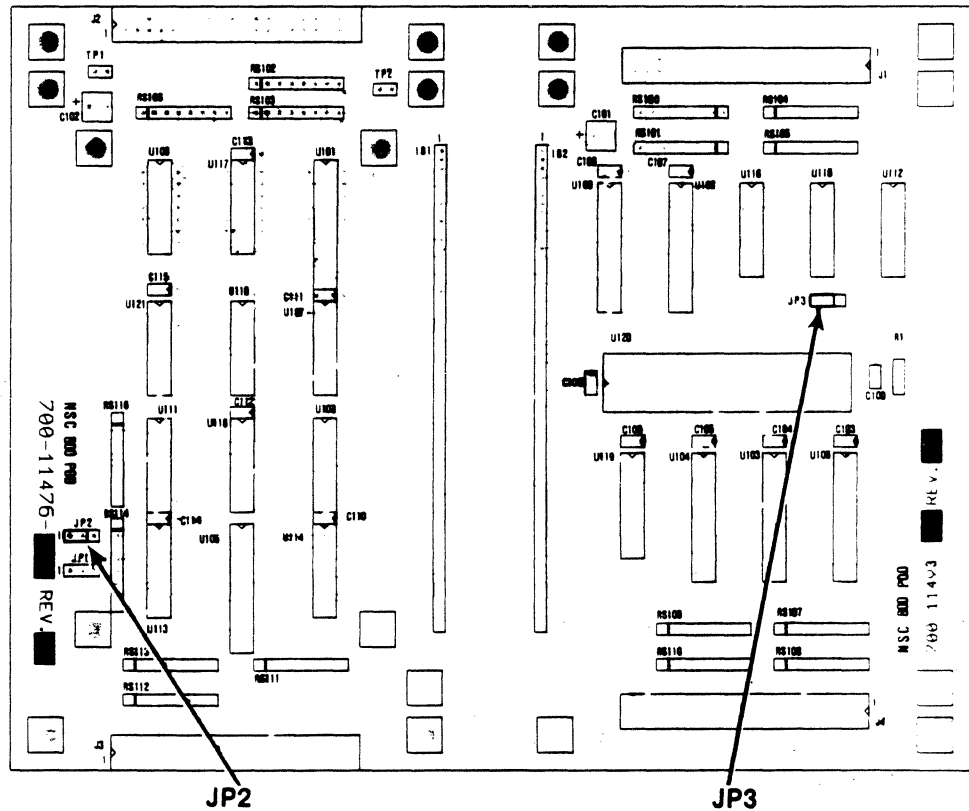
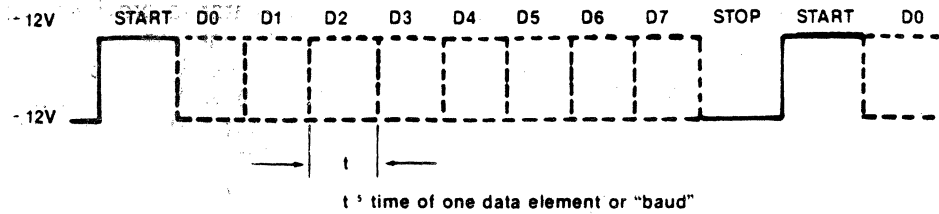


Figure 9.3-1  
Serial Word Format

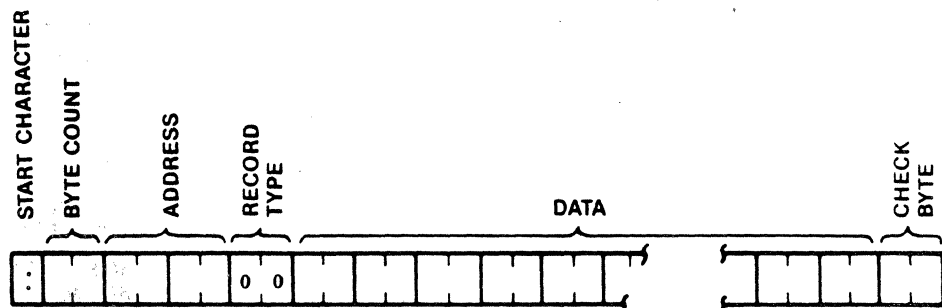


Switch Position	Baud Rate	t	
D	50	20	mSEC
C	75	13.33	
0	110	9.09	
B	134.5	7.43	
1	150	6.67	
A	200	5	
2	300	3.33	
9	600	1.67	
4	1,200	833	uSEC
5	1,800	556	
3,8	2,400	417	
6	4,800	208	
7	9,600	104	
E,F	19,200	52	

#### 9.4 UPLOAD/DOWNLOAD PROTOCOL

The EM-800 routines CODE C3 and CODE C4 initiate routines to load the target memory space with data from the serial link or dump data from the target address space to the serial link. The format used to transfer the data is compatible with the Intel family of development systems.

#### DATA RECORD



#### START CHARACTER

As ASCII colon is used to signal the start of a record.

#### BYTE COUNT

Two ASCII characters representing hexadecimal digits giving the number of data bytes in the record.



## 9.5 EXTERNAL BREAKPOINT

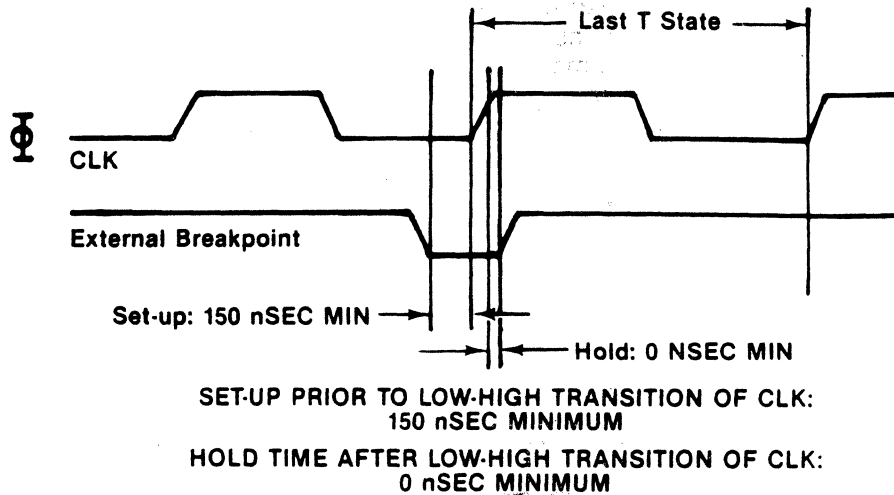
The EM-800 Diagnostic Emulator is provided with an input that permits an external signal to halt the execution of the target program when the EM-800 is in the RUN mode. Pin 10 of the back panel Auxiliary Connector (J3) is the input connection. External Breakpoint is a TTL level input with a 3.3K resistor pull up to +5 volts. If this input is in the high state, or if the input is left open, then the EM-800 will run the target program in the normal manner. If this input is pulled low, the target program will halt; if the target program is already halted, the External Breakpoint signal will have no effect.

The EM-800 samples the External Breakpoint input at the low-to-high transition of the clock that begins the final T-state of an instruction. If the signal is low at the sample time, the signal is entered into the Trace Memory, thus marking the cycle during which the signal was detected; circuitry in the EM-800 is also armed to halt program execution after completion of the current instruction. When the target program has been halted, the EM-800 firmware will determine which cycle of the last instruction caused the breakpoint and the Trace Memory will be positioned to display that cycle. Figure 9.1-5 shows the timing relationships of the External Breakpoint signal.

You can also use a logic analyzer as a trigger for the External Breakpoint to expand the breakpoint system. The logic analyzer supplies the trigger based on what it is looking at (address bus, line bus, etc.).

The External Breakpoint system can also be used in a multiprocessor debugging situation, where you are using two or more emulators; if one emulator reaches a breakpoint, it may be configured to breakpoint the other emulator(s).

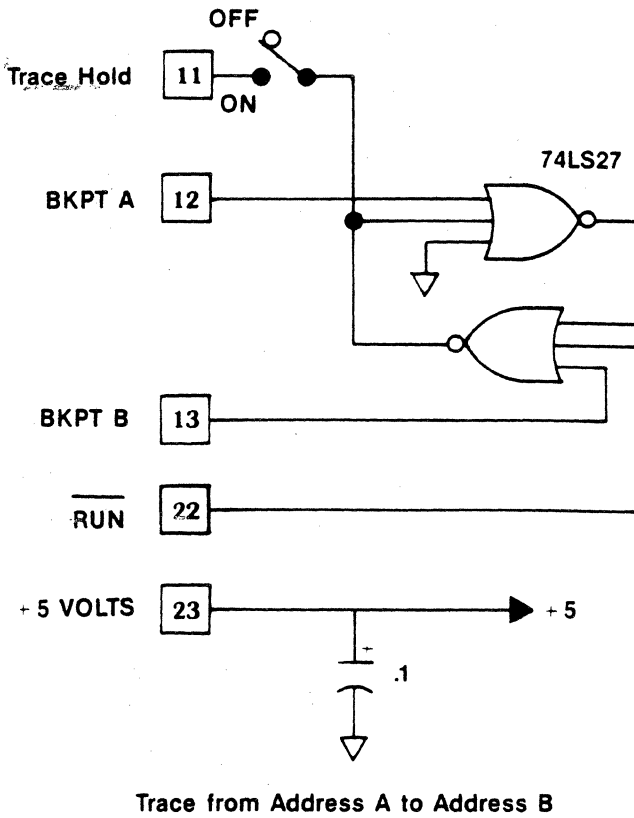
Figure 9.5-1  
Timing Relationships



### 9.6.1 WINDOW MODE

Figure 9.6-2 shows a schematic of a simple external circuit that may be used to implement window mode operation of the Trace Memory. This circuit controls the Trace Hold input of the EM-800 so that only bus activity occurring between the Breakpoint A address and the Breakpoint B address is recorded.

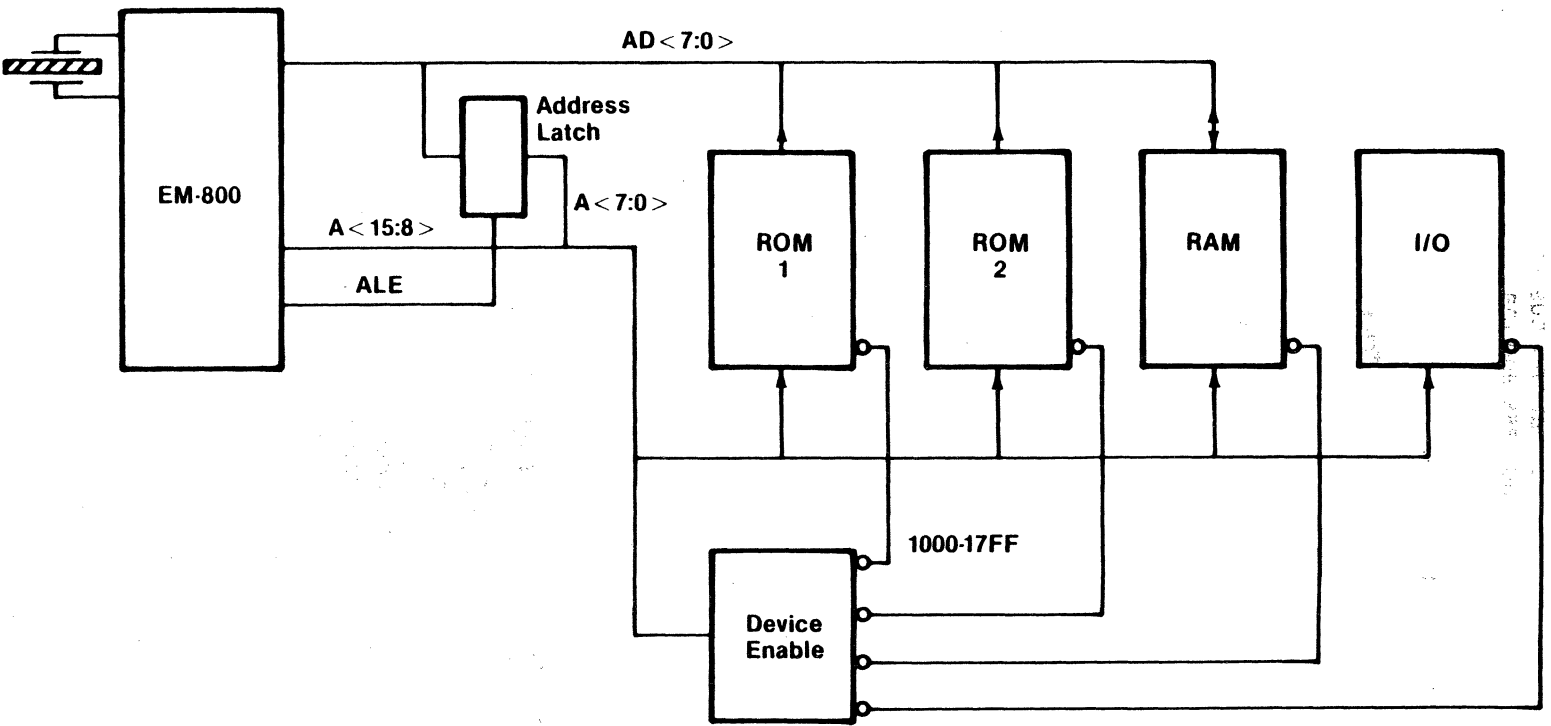
Figure 9.6-2  
Window Mode Circuit



### 9.6.2 SELECTIVE TRACE

External circuitry may be designed for a variety of selective tracing functions. An example is an application in which it is desired to use the capacity of the Trace Memory to capture cycles written to a particular I/O port; the I/O port select signal (port decode) may be routed to the Trace Hold input to permit Trace Memory operation only when the port select signal is active. When execution is halted and the Trace Memory content is reviewed, only bus cycles to the I/O port will be seen.

Figure 9-7-1  
Simplified  
Microprocessor Diagram



---

The EM-800 also has a built-in test function for obtaining a signature of a ROM in a system, and no signature analyzer is needed. The test is set up by entering the first and last address of the ROM into the BEG and END registers of the EM-800 to define the range over which the routine will operate. Then start the routine with the keys for CODE D3. The routine will execute and then display a four-digit hexadecimal signature on the EM-800 front panel. The signature obtained does not have any simple relationship to signatures obtained with the HP 5004A; for one thing, the CODE D3 algorithm operates on all eight data bits of the ROM word simultaneously while the eight signatures obtained by the HP 5004A for a ROM are computed from one "bit slice" of the ROM at a time. In addition, the generating polynomial used by the EM-800 routine differs from that used by the HP 5004A. See Section 7.4 for additional information.

Other routines that are programmed in the EM-800 Diagnostic Emulator may be useful as stimulus routines for signature analysis testing. The CODE B4 routine repetitively stores a data pattern and the complement of that pattern to a selected address. An I/O port, such as the one shown in Figure 9.7-1, may be tested by storing the complementing data to the I/O port and observing the signatures obtained at the output side of the I/O port.

In special cases, you may have to write custom CODE function routines to stimulate a system in a way that useful signatures may be obtained. As an example, consider the problem of obtaining a signature at the outputs of an LSI interface chip such as the Zilog PIO. This device requires that various control registers and data direction bits be set up for the intended application before data transfers are performed. A custom Code Function routine can easily perform the desired set-up and then generate the stimulus for signature analyzer probing.

For additional information on Signature Analysis testing, see the following publications:

1. *"Hexadecimal Signatures Identify Troublespots in Microprocessor Systems,"* Gary Gordon and Hans Nadig. ELECTRONICS, March 3, 1977.
2. Application Note 222, *"A Designer's Guide to Signature Analysis,"* Hewlett Packard Corporation.

---

# **SECTION 10 MAINTENANCE & TROUBLESHOOTING**

---

<b>10.1</b>	<b>MAINTENANCE</b>
10.1.1	Power Supply
10.1.2	Cables
10.1.3	Probe Tip Assembly
<b>10.2</b>	<b>TROUBLESHOOTING</b>
<b>10.3</b>	<b>PARTS LIST</b>

---

---

### 10.1.3 PROBE TIP ASSEMBLY

The Probe Tip Assembly is the small DIP header assembly that plugs into the target system CPU socket. The most obvious area to inspect is the 40-pin adapter as the pins can be broken during insertion or extraction. If one of the pins should be inadvertently broken, you should replace the complete 40-pin adapter.

#### NOTE:

The 40-pin adapter can be protected by installing a CPU socket (male-female) onto the 40-pin adapter. If a pin is then broken on the CPU-socket, it is easier to replace because of its common usage.

You should also inspect the probe tip assembly to see if any of the  $\frac{1}{8}$  watt resistors have been broken.

#### NOTE:

Due to the close physical tolerance surrounding the  $\frac{1}{8}$  watt resistors, we recommend that they be returned to the factory for repair.

## 10.2 TROUBLESHOOTING

Troubleshooting microprocessor-based equipment can be a complex process, due mainly to the complex nature of several peripheral devices, such as the data and address lines. To assist you in identifying the faulty PC card or possibly a component, your emulator is equipped with diagnostic test routines. The diagnostic programs are described in Section 7; if you need to perform any specific test, you should refer to the description in Section 7. Before starting troubleshooting procedures, be sure that interconnect cables are installed properly in a compatible target system, with power applied to *both* the target system and the emulator.

The most common problems encountered are listed in Table 10-2. We recommend that you contact the Technical Services Department of Applied Microsystems Corporation if you experience *any* problems that do not fall within this range of items.

#### NOTE

We do not recommend a component-level repair in the field, unless performed by a qualified service engineer.

SYMPTOM	POSSIBLE CAUSES	SECTION
	5. Broken pin on interconnect cable connector.	.
	6. RAM Overlay switch on but memory not programmed	5.3
	7. No clock in target system	3.3
	8. No power (+ 5 volts) in target system	**
	9. Option switches set improperly	9.2
	10. Emulator and target system not compatible	1.1
	11. RUN key bad	.
	12. Constant target reset	3.3
	13. Target system has one or more DMA devices requesting the bus. (Example: Bus Req line = True)	**

\*Call Applied Microsystems (Technical Services Department)

\*\* Check Target System

### 10.3 PARTS LIST

The following parts are available for you to order:

40-Pin Adapter 210-11410  
Short Cable Set 600-11284  
Long Cable Set 600-10653-0  
Key Switch 510-10128  
Hex Display 370-10009

---

# APPENDIX A. NULL TARGET — A SOFTWARE SIMULATION TOOL

---

## A.1 INTRODUCTION

The Null Target — A Software Simulation Tool comes as a standard feature with all of the EM-Series Diagnostic Emulators with the exception of the 8080, 6800, 6802, and 6808. The Null Target allows software to be developed in parallel with hardware. This feature is possible because the Null Target supplies all of the signals needed to operate the emulator in a stand alone mode.

The Null Target also allows easy program debugging. By downloading software to the RAM Overlay Memory, programs can be debugged using the emulator's breakpoint system. Subroutines can also be tested and finalized, decreasing the number of bugs that need to be fixed when hardware and software are integrated.

## A.2 COMPONENTS

The Null Target consists of the box itself and a five-pin strip that attaches to the emulator probe.

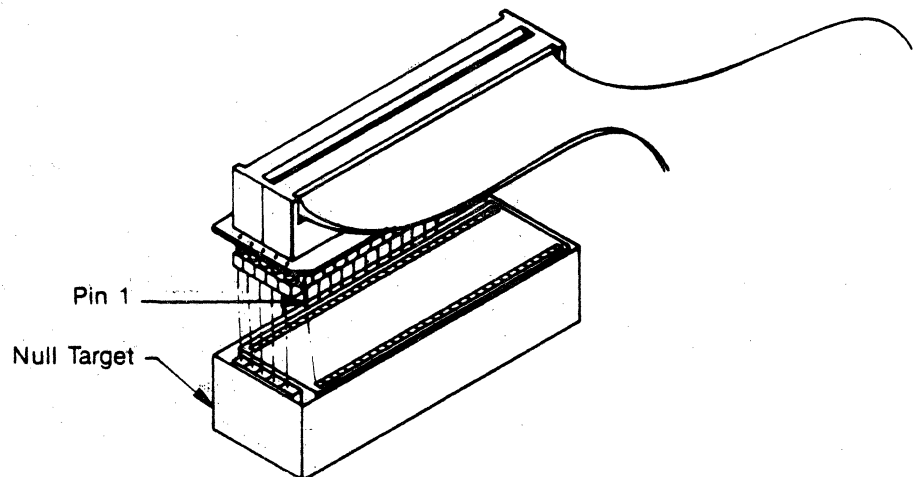
## A.3 INSTALLATION

Installation of your Null Target system is a simple process. First, attach the five-pin strip to the emulator probe and then insert the other end of the five-pin strip into the socket on the Null Target as shown in Figure A-1.

### NOTE

To operate the EM-800, the target microprocessor chip must be installed in the pod socket.

Figure A-1  
Installing the Null Target





---

# APPENDIX B. SYSTEM ERROR CODES

---

ERROR CODE	DESCRIPTION
1	Framing Error
2	Overflow Error
11	Non-Hex Character Received
12	Sum-Check Error
13	Non-Zero Record Type
21	Target Memory Write Error
22	Target Memory Non-Compare

---

# INDEX

---

## A

Auxiliary connector, 2-5, 9-2

## B

Back panel controls, 2-5  
Baud rate selector switch, 2-5  
Breakpoint system, 4-6  
Breakpoint comparators, 8-13

## C

Cables, 1-3, 2-3, 10-2  
Check byte, 9-8  
Chips, LSI interface, 4-15, 9-5  
Code functions, 7-2  
Connection to target equipment, 3-2  
Communications, 9-2  
CPU registers, 4-11

## D

Data transmission, 9-2  
Default parameters, 7-18  
Diagnostic EPROM socket, 2-2  
Diagnostic functions, 7-2  
Disassembler, 2-5  
Disassembly format, 6-3  
Display panel, 2-3  
Displays, front panel, 2-3, 4-4  
Downloading  
    Protocols, 9-7  
    Protocol selection, 9-4  
    RAM overlay from front panel EPROM, 5-7  
    RAM overlay from serial link, 5-8  
    RAM overlay from target memory, 5-8

## E

Emulator probe, 2-2  
Entry to user code functions, 8-15  
EPROM socket, 8-4  
Error Codes, 7-11, B-1  
Examine and store, 8-17  
Examination and alteration  
    CPU registers, 4-11  
    Memory locations, 4-13  
    I/O ports, 4-15  
External breakpoint, 4-15  
Execution and control, 4-2

## F

Flags, 6-5  
Format definition, disassembly, 6-3  
Functions, EM-800, 4-2

## G

Getting started, 3-2

## H

Hexadecimal displays and trace memory, 8-9

## I

I/O devices, 8-6  
I/O ports, 4-15  
Installing RAM overlay, 5-4  
Interface chips, 9-5  
Introspection mode, 7-19, 8-16  
Internal environment of EM-800, 8-3  
Interrupts, 8-6

## J

Jump, re-entry, 8-19, 9-14

## K

Keyboard, 2-2, 8-7

## L

LSI interface chips, 4-15, 9-5

## M

Main power switch, 2-5  
Maintenance, 10-2  
Memory  
    Map, 8-3  
    Overlay, 8-2  
    Trace, 4-10  
Memory tests, 7-4

## N

Null target, A-1

## O

Operator's station, 2-3  
Option jumpers, 9-5

---