M68HC05E1EVS/D1

April 1991

M68HC05E1EVS

EVALUATION SYSTEM

USER'S MANUAL

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unathorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

IBM-PC is a registered trademark of International Business Machines Corp.

The computer program stored in the Read Only Memory of the device contains material copyrighted by Motorola Inc., first published 1991, and may be used only under a license such as the License For Computer Programs (Article 14) contained in Motorola's Terms and Conditions of Sale, Rev. 1/79.

First Edition

© MOTOROLA, INC., 1991

PREFACE

Unless otherwise specified, all address references of this manual are in hexadecimal format.

TABLE OF CONTENTS

PAGE

CHAPTER 3 OPERATING INSTRUCTIONS

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

PAGE

FIGURE

LIST OF TABLES

PAGE

TABLE

CHAPTER 1

GENERAL INFORMATION

1.1 INTRODUCTION

This manual provides general information, hardware preparation, installation instructions, operating instructions, functional description, and support information for the M68HC05E1EVS Evaluation System (hereafter referred to as EVS). Appendix A contains EVS downloading S-record information.

The EVS consists of two printed circuit boards (PCBs):

- M68HC05E1EM Emulator Board (EM)
- M68HC05PFB Platform Board (PFB)

1.2 FEATURES

EVS features include:

- An economical means of evaluating target systems incorporating MC68HC05E1 HCMOS microcontroller unit (MCU) devices.
- Monitor/debugger firmware
- One-line assembler/disassembler
- Host computer download capability
- Dual monitor and user memory maps
- RS-232C terminal input/output (I/O) port
- MCU extension I/O port
- Logic analyzer connector

1.3 SPECIFICATIONS

Table 1-1 lists EVS specifications.

TABLE 1-1. EVS Specifications

CHARACTERISTICS	SPECIFICATIONS
Internal Clock (EM)	Up to 2 MHz, controlled by on-chip phase lock loop (PLL).
External Clock (EM)	32.768 KHz
Memory size	
Monitor EPROM	8K bytes
Pseudo ROM	8K bytes
MCU extension I/O port	HCMOS compatible
Terminal I/O port	RS-232 compatible
Temperature	
Operating	+25° C
Storage	-40° to +85° C
Relative humidity	0 to 90% (non-condensing)
Power requirements	+5 Vdc @ 1.0 A (max)
Dimensions	
EM	7.2 X 4.5 in. (18.3 X 11.4 cm)
Platform Board	10 X 7.5 in. (25.4 X 19.1 cm)

1.4 GENERAL DESCRIPTION

The EVS is an economical tool for designing, debugging, and evaluating MC68HC05E1 MCU-based target systems. The EVS is shipped with a resident MC68HC05E1 MCU device. By providing the essential MCU timing and I/O circuitry, the EVS simplifies user evaluation of prototype hardware/software products. The EVS requires a user-supplied power supply and terminal or host computer.

An RS-232C compatible terminal or a personal computer is required for communication with the EVS. The RS-232C communication baud rate is fixed at 9600. The personal computer must have a terminal emulation package, such as PCKERMIT.EXE or PROCOMM.

There are two methods of generating MCU code:

1. Using the resident one-line assembler/disassembler.

2. Downloading assembled code from an external source to user program RAM (pseudo ROM) through the RS-232C terminal I/O port.

The user enters data and debugs MCU code by means of the EVS-resident debug monitor. MCU device ROM is simulated by write-protecting user program RAM during program execution.

As mentioned, the EVS consists of two printed circuit boards, the M68HC05E1EM Emulator Module (EM), and the M68HC05PFB Platform Board (PFB).

The MCU extension I/O port (EM connector P4) facilitates connection of the EVS to the target system. A logic analyzer may be connected to EM connector P1. The 64-pin expansion header connectors interconnect the PFB and the EM.

The RS-232C terminal I/O port (PFB connector P2) facilitates connection with the external terminal. This I/O port also enables the user to download programs (via Motorola S-records) directly from a personal computer to the EVS. Downloading is performed using the debug monitor commands.

The EVS includes jumper-selectable options such as clock source selection, pseudo ROM compatibility selection, and write-protection selection. Switches allow user control of the reset and abort functions.

1.5 EQUIPMENT REQUIRED

Table 1-2 lists external equipment requirements for EVS operation.

TABLE 1-2. External Equipment Requirements

EXTERNAL EQUIPMENT
+5 Vdc power supply(1)
Terminal (RS-232C compatible)
Host computer (RS-232C compatible)(2)
Terminal/host computer - EVS RS-232C cable assembly(1)
Target system - EVS emulator cable assembly(3)

Notes:

(1) Refer to Chapter 2 for details.

(2) Optional - not required for basic operation.

(3) DIP cable assemblies fabricated by user.

CHAPTER 2

HARDWARE PREPARATION AND INSTALLATION INSTRUCTIONS

2.1 INTRODUCTION

This chapter provides unpacking instructions, hardware preparation information, and installation instructions for the EVS.

2.2 UNPACKING INSTRUCTIONS

NOTE

Should the product arrive damaged, save all packing material, and contact the carrier's agent.

Unpack EVS from its shipping carton. Refer to the packing list and verify that all items are present. Save packing material for storing and shipping the EVS.

2.3 HARDWARE PREPARATION

The user should inspect and prepare the EM and PFB PCBs before connecting the EVS to a target system. This portion of the text explains how to do this, making sure that the EVS is properly configured for target system operation.

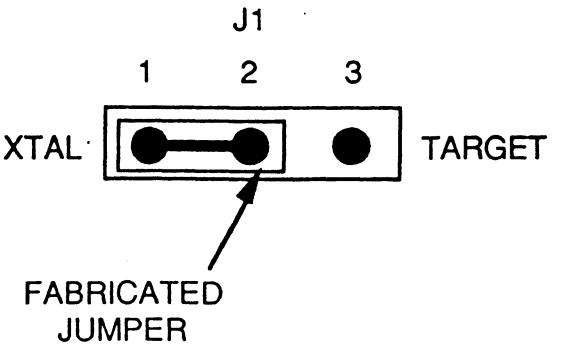
Figure 2-1 shows EM installation on the PFB, via EM expansion header connectors P2 and P3, and PFB expansion header connectors P3 and P4.

The EVS has been factory-tested and is shipped with factory-installed jumpers. Figures 2-1 through 2-3 illustrate EVS configuration and jumper header, connector, and switch locations.

2.3.1 EM Configuration

Figure 2-2 shows jumper header and connector locations of the emulator module (EM). Jumper header J1 lets the user select the EVS clock source. Connector P4 is the MCU extension I/O port, for connection to the target system. Logic analyzer connection, available through connector P1, facilitates target-system hardware and software debugging. Expansion header connectors P2 and P3 facilitate interconnection of the EM and PFB. Refer to Chapter 5 for connector pin assignments.

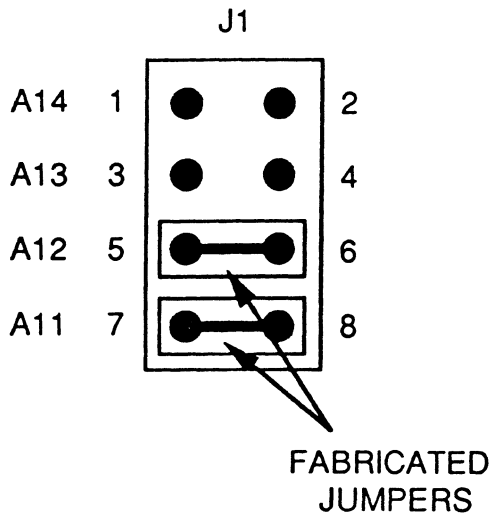
2.3.1.1 Clock Source Select Header (J1). The factory configuration of jumper header J1 is shown below. The fabricated jumper installed between pins 1 and 2 selects the EVS MCU crystal (XTAL) clock source. To use a clock source originating from the target system, reposition the jumper between pins 2 and 3, and connect the target-system clock signal to the OSC1 pin (pin 6 of EM connector P4).



2.3.2 PFB Configuration

Figure 2-3 shows jumper header and connector locations of the platform board (PFB). The five PFB jumper headers determine pseudo ROM configuration, port selection, and write-protection. Connector P1 is for system power. Connector P2 is the terminal I/O port. Expansion header connectors P3 and P4 facilitate interconnection of the PFB and EM. Switches SW1, SW2, and SW3 are the abort, user reset, and master reset switches, respectively. Refer to Chapter 5 for connector pin assignments.

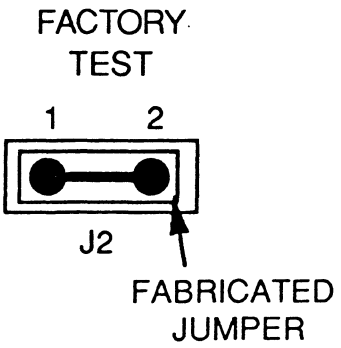
2.3.2.1 Pseudo ROM Configuration Header (J1). The factory configuration of jumper header J1 is shown below. This configures pseudo ROM for compatibility with M68HC05E1 MCU target systems.



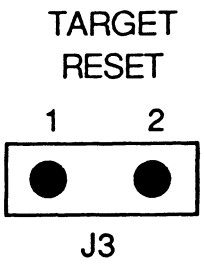
NOTE

The PFB is common to all M68HC05 type emulator modules. Jumper header J1 reconfigures pseudo ROM of the EM in use with the PFB. To adapt the PFB for use with a different EM, refer to the appropriate EVS user's manual for applicable jumper installation.

2.3.2.2 Factory Test Header (J2). The factory configuration of jumper header J2 is shown below. The fabricated jumper should not be removed during normal EVS operations. (Jumper header J2 has significance only for factory testing; only factory test personnel should remove the jumper from this header.)



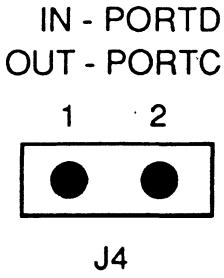
2.3.2.3 Target Reset Header (J3). The factory configuration of jumper header J3 is shown below. This is correct for the E1 EVS. (The PFB is common to several EVS products, some of which may use a jumper in header J3).



NOTE

For normal E1 EVS operation, there should not be a jumper installed in header J3.

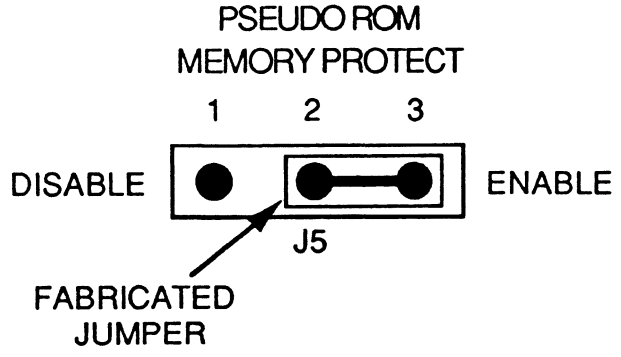
2.3.2.4 Port Select Header (J4). The factory configuration of jumper header J4 is shown below. This establishes the correct MC68HC26 port replacement unit configuration for E1 operation. (The PFB is common to several EVS products, some of which do use a jumper in header J4.)



NOTE

For normal E1 EVS operation, there should not be a jumper installed in header J4.

2.3.2.5 Write-Protect Select Header (J5). The factory configuration of jumper header J5 is shown below. The fabricated jumper installed between pins 2 and 3 write-protects user program space (pseudo ROM) during program execution. To disable write protection, reposition the fabricated jumper between pins 1 and 2.



2.4 INSTALLATION INSTRUCTIONS

The EVS is designed for table-top operation. A user-supplied power supply and RS-232C compatible terminal are required. An RS-232C compatible host computer may be connected to the EVS, but is not required for basic EVS operation.

The EVS is shipped with the EM installed on the PFB. EM expansion header connectors P2 and P3, and PFB expansion header connectors P3 and P4, interconnect the two PCBs.

The following paragraphs explain the remaining EVS connections.

2.4.1 Power Supply - EVS Connection (PFB P1)

The EVS requires a +5 Vdc @ 1.0 Amp power supply for operation.

Use PFB connector P1 to connect power to the EVS. Contact 1 is GND; black lever. Contact 2 is VDD (+5 Vdc); red lever. Use 20 or 22 AWG wire for power connections. For each wire, trim back the insulation 1/4 in. (.635 cm), lift the appropriate lever of P1 to release tension on the contacts, then insert the bare wire into P1 and close the lever. (Contact 3 is for VPP, which is not required for E1 operation.)

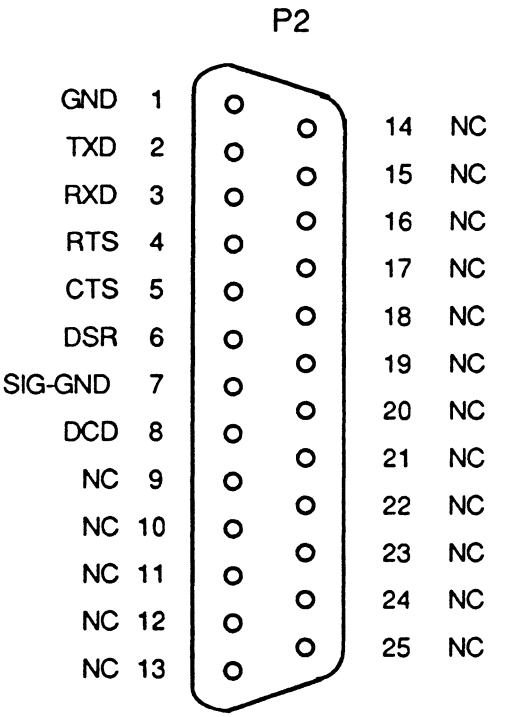
CAUTION

Do not use wire larger than 20 AWG in connector P1. Such wire could damage the connector.

Turn off PFB power when installing the EM or removing the EM from the PFB. Sudden power surges could damage EVS integrated circuits.

2.4.2 Terminal - EVS Connection (PFB P2)

Connection of an RS-232C compatible terminal to the EVS requires a user-supplied 20- or 25-conductor flat ribbon cable assembly, as shown in Figure 2-4. One end of the cable assembly connects to the RS-232C terminal I/O port (PFB connector P2), shown below. The other end of the cable assembly connects to the user-supplied terminal. For connector pin *assignments and* signal descriptions of PFB connector P2, refer to Chapter 5.



RS-232C Terminal I/O Port Connector

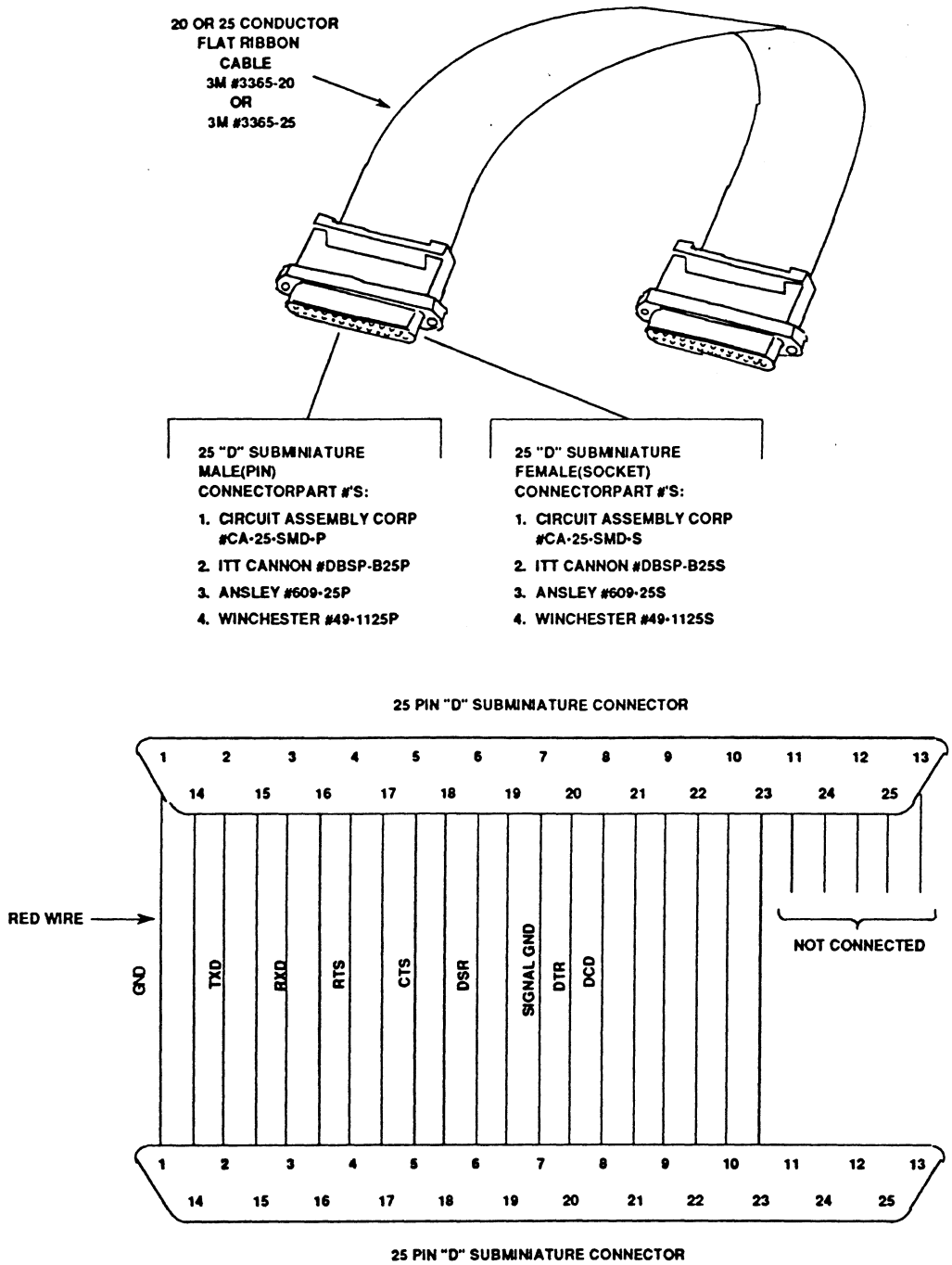


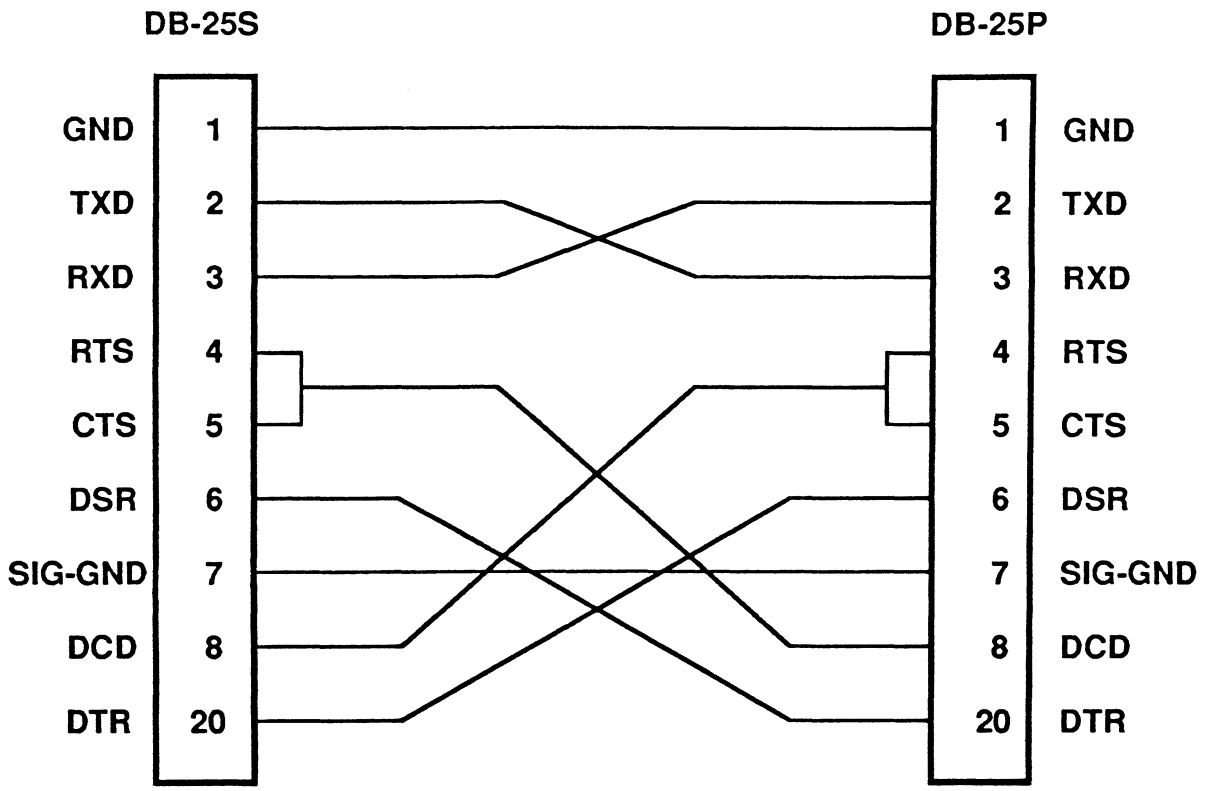
Figure 2-4. Terminal/Host Computer Cable Assembly Diagram

A Hayes compatible modem cable, purchased from a local computer store, can be used to connect the EVS to the host computer.

The EVS is wired as data communication equipment (DCE) whereas a terminal and most serial modem ports on host computers are wired as data terminal equipment (DTE). This lets a straight-through cable be used for most setups.

If a different type of cable is used to connect the EVS to a host computer, a null modem adapter (shown below) may be required to match the cable to the EVS terminal port connector.

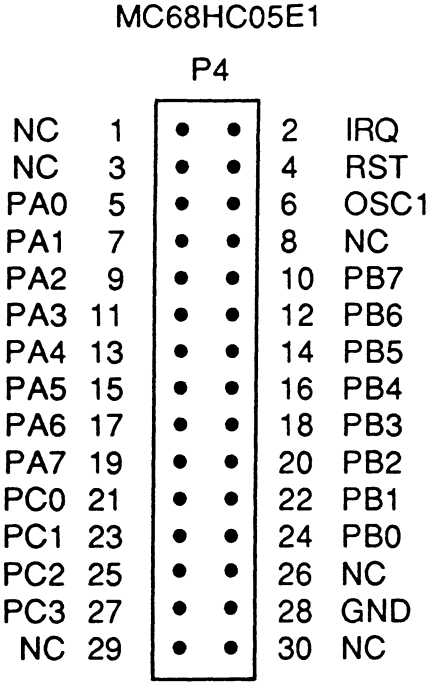
A null modem adapter reverses the roles of various data and control signals to make a DTE device appear as a DCE device, or vice versa.



Null Modem Adapter

2.4.3 Target System - EVS 30-Pin Connection (EM P4)

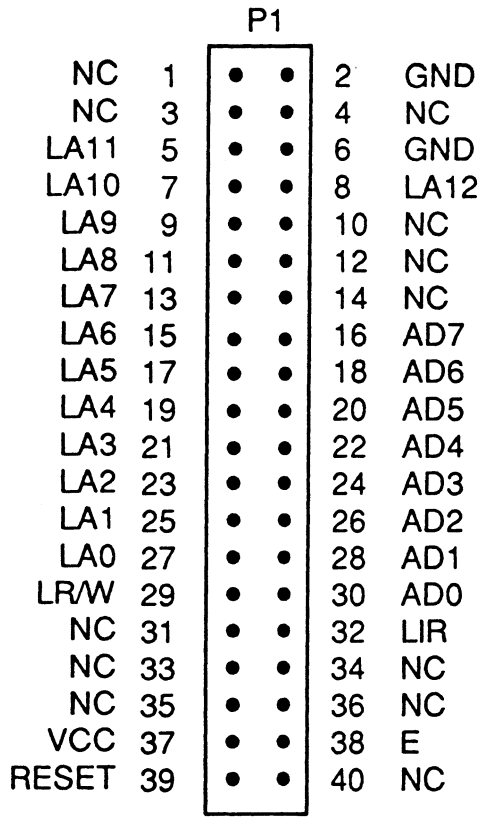
Use the MCU extension I/O port (EM connector P4) to connect the EVS to a target system. EM connector P4 is a 30-pin header (shown below), that requires a user-supplied 28-pin dual-in-line package (DIP) cable assembly, shown in Figure 2-5. For connector pin assignments and signal descriptions of EM connector P4, refer to Chapter 5.



30-Pin MCU I/O Port Connector

2.4.4 Logic Analyzer - EVS Connection (EM P1)

A logic analyzer can be connected to EM conector P1, as an aid to debugging target system hardware and software. Logic analyzer pin-outs are shown below. For connector P1 signal descriptions, refer to Chapter 5.



Logic Analyzer Connector

CHAPTER 3

OPERATING INSTRUCTIONS

3.1 INTRODUCTION

This chapter provides the information necessary to initialize and operate the EVS in a target system environment. MCU assembling/disassembling and EVS downloading procedures are also provided. This information is in this order:

- Control switch descriptions
- Limitations
- Operating procedures
- Command line format
- EVSbug commands
- Assembling/disassembling procedures
- Downloading procedures

The EVS is factory tested for target system evaluation.

The monitor is the resident firmware (EVSbug) for the EVS, which provides a self contained operating environment. The monitor interacts with the user through commands entered at a terminal. These commands perform functions such as memory display or modification, MCU internal register display or modification, program execution under various levels of control, and control access to various I/O peripherals connected to the EVS.

3.2 CONTROL SWITCHES

The three EVS switches (PFB switches SW1 through SW3) control the reset and abort functions. Table 3-1 identifies these switches by name, description, and function.

TABLE 3-1. EVS Control Switches

NAME	DESCRIPTION AND FUNCTION
ABORT Switch (PFB SW1)	Momentary action pushbutton switch that returns EVS operation to the monitor map from the user map assuming proper operation of user code. (If MCU gets lost in the user map, the abort is useless and a master reset must be issued.) The abort function has no effect when operating in the monitor map.
USER RESET Switch (PFB SW2)	Momentary action pushbutton switch that resets EVS MCU and user I/O, and enables map switching to the user map. This switch is used as a map switch for execution of user code from the user reset vector.
MASTER RESET Switch (PFB SW3)	Momentary action pushbutton switch that resets EVS MCU and user I/O, places EVS operation in the monitor map, and enables the EVS prompt to be displayed on terminal.

NOTE

Press the abort switch when debugging code, rather than the master reset switch. A master reset leaves SWI opcode $83 at all breakpoint addresses.

3.3 LIMITATIONS

The user may not trace a clear interrupt mask (CLI) or return from interrupt (RTI) instruction with an interrupt enabled and pending, due to MCU interrupt handling. Attempting such a trace causes an interrupt in the monitor map; this forces a software reset of the EVS. User breakpoints remain in the user map, but the monitor cannot recognize them. The user stack pointer also reflects the occurrence of an interrupt.

Do not trace a branching instruction (e.g., BRA) that branches to itself. Because the monitor places an SWI instruction on the object of the branch, the instruction would never be executed. However, it would appear to the user that the instruction had executed. The user may enter a G command while the PC points to this type of instruction as long as the instruction is not a breakpoint address.

Whenever possible, avoid mixing interrupt requests (IRQs) and user software interrupts (SWIs). This prevents a possible IRQ-SWI EVS timing problem: a concurrent hardware interrupt and SWI could cause an EVS failure which could stop program execution. Activating the master reset switch (SW3) restores the EVS to proper operation. Statistically, such a failure occurs very infrequently.

When emulating a device with a computer operating properly (COP) watchdog timer, user software must enable the COP by writing a $04 to the test register (location $1F).

The EVS does not provide protection to limit user programs to the exact amount of MCU ROM available. The user must be aware of the memory map of the MCU being simulated and must be sure to use only valid ROM locations.

To run the EVS at the 2.097 MHz speed, set AUTO = OFF and BWC = 0 in the phase lock loop control (PLCC) register (location $07). To do so, write a $4E to the PLCC register. This prevents noise injection in the VDDSYN signal, as well as oscillator jitter.

3.4 OPERATING PROCEDURE

A power on reset (POR) occurs when the user applies power to the EVS. This POR resets the MCU and user I/O port circuitry, and actuates the monitor. All user registers are in an unknown state during monitor power-up.

The input serial format for the EVS terminal I/O port must be configured for 8 data bits, 1 stop bit, no parity, and 9600 baud. The baud rate is fixed at 9600.

The terminal displays the following message after the user presses the terminal keyboard carriage return (<CR>) key:

```
EVSbug-HC05 X.X
S=FF     P=XXXX     A=XX     X=XX     C=E8          111.I...
>
```

where:

X is a revision of the software or an unknown register state.

Condition code register (CCR) HINZC bits are:

1	=	Fixed bit, set to logic 1
1	=	Fixed bit, set to logic 1
1	=	Fixed bit, set to logic 1
H	=	Half carry bit
I	=	Interrupt mask bit
N	=	Negative bit
Z	=	Zero bit
C	=	Carry/borrow bit

The display format of the CCR bits shows their status:

When all CCR bits are set (logic 1), bits appear as:

```
S=XX     P=XXXX     A=XX     X=XX     C=FF          111HINZC
```

When all CCR bits are cleared (logic 0), bits appear as:

```
S=XX     P=XXXX     A=XX     X=XX     C=E0          111.....
```

When a specific bit is set (I), bits appear as:

```
S=XX     P=XXXX     A=XX     X=XX     C=E8          111.I...
```

When specific bits are set (H, I, and C), bits appear as:

```
S=XX     P=XXXX     A=XX     X=XX     C=F9          111HI..C
```

After initialization or return of control to the monitor, the terminal displays the prompt " > " and waits for a response. If the user enters an invalid response, the terminal displays "ILLEGAL/INSUFFICIENT ENTRY" followed by the prompt " > ".

The EVS waits for a command line input from the user terminal. When the user enters a proper command, the operation continues in one of two basic modes. If the command causes execution of a user program, the monitor may or may not be reentered, depending upon the desire of the user. For the alternate case, the command is executed under the control of the monitor, and the system returns to a waiting condition after the command is completed. During command execution, additional user input may be required, depending on the command function.

The user can use any of the commands supported by the monitor. A standard input routine controls the EVS operation while the user types a command line. Command processing begins when the user presses the keyboard carriage return (<CR>) key, at the end of the command line.

3.5 COMMAND LINE FORMAT

The command line format is:

><command> [<parameters>]<CR>

where:

>	EVSbug monitor prompt.
<command>	Command mnemonic.
<parameters>	Expression or address.
<CR>	Carriage return keyboard key - pressed to enter command.

NOTES

(1) Command line format definitions use special characters that have these syntactical meanings:

< >	Enclose syntactical variable
[ ]	Enclose optional fields

(The user does not enter these characters, which are for definition purposes only.)

(2) A single space separates fields.

(3) All input numbers are interpreted as hexadecimal. A dollar sign ($) may precede any number input, but is not required.

(4) All input commands can be entered either upper or lower case. The monitor automatically converts input commands to upper case; the only exceptions are downloading commands sent to the host computer.

(5) A maximum of 30 characters may be entered on a command line. After the 30th character, the monitor automatically terminates the command entry and processes the command line.

(6) Parameters are interpreted to be the last two or three characters in the parameter file. Backspace to correct parameter errors.

3.6 EVSbug COMMANDS

Table 3-2 lists the monitor (EVSbug) commands alphabetically by mnemonic. Detailed descriptions of each command follow the table.

Additional terminal keyboard functions are:

(BREAK)	Abort command
(CTRL)S	Freeze screen
(CTRL)X	Cancel command line
<CR>	Enter command

NOTE

When using the control key for a specialized command such as (CTRL)X, press and hold the (CTRL) key, then press the X key, then release both keys.

During memory display output to terminal, (CTRL)S delays the output until another character is entered.

Command line input examples in this chapter follow this format:

Boldface type denotes entries the user makes via the terminal keyboard.

Command line input is entered when the user presses carriage return (<CR>).

A typical example of this explanation is:

```
>MD 100 21F
```

TABLE 3-2. EVSbug Commands

COMMAND	DESCRIPTION
ASM <start addr>	Assembler/disassembler (interactive)
BF <start addr> <end addr> <data>	Block fill memory with data
BR [<addr1 - addr5>]	Breakpoint set
FAST	2MHz phase lock loop (PLL) rate
G [<start addr>]	Go (execute program)
HELP	Help (display commands)
LOAD T	Load (S-records) from I/O port(1)
MD <start addr> [<end addr>]	Memory display
MM <address>	Memory modify (interactive)
NOBR [<addr1 - addr5>]	Remove breakpoint
P [<count>]	Proceed (through breakpoint)
RD	Register display
RM	Register modify (interactive)
SLOW	32KHz External oscillator rate
T [<count>]	Trace
TURBO	Previous PLL setup

1. See Appendix A for S-record information.

3.6.1 Assembler/Disassembler

ASM <start addr>

where <start addr> is the starting address for the assembler operation.

The assembler/disassembler is an interactive assembler/editor for the source program. As the user enters each line of source code, the assembler/disassembler converts the line into machine language and stores it in memory. The assembler/disassembler disassembles the machine code of each instruction, in order to display the instruction mnemonic and operands. All valid opcodes are converted to assembly language mnemonic. All invalid opcodes return a form constant byte (FCB).

The ASM command lets the user create, modify, and debug MC68HC05 MCU code. The command does not make provision for line numbers or labels.

Assembler input must have exactly one space between the mnemonic and the operand. There must be no space between the operand and the index specification (,X) except in the case of indexed no offset. A carriage return (<CR>) must terminate assembler input. No comments are allowed after the instruction input. Examples are:

a. `>LDA 0,X`

b. `>STA 10,X`

c. `>ASRA`

d. `>COMX`

e. `>CMP 200`

After each new assembler input line, the new line is disassembled for the user before stepping to the new instruction. The new line may assemble to a different number of bytes than the previous one.

During disassembly of the Branch if High or Same (BHS) or Branch if Carry Clear (BCC) mnemonics, the screen displays the BCC mnemonic. During disassembly of the Branch if Lower (BLO) or Branch if Carry Set (BCS) mnemonics, the screen displays the BCS mnemonic.

The assembler automatically calculates branch address offsets, so the user should input the addresses, not offset values, as operands.

Terminate the assembler by entering a period (.) followed by a carriage return (<CR>) as the only entry on the command input line. Entering a <CR> alone on an input line steps to the next instruction.

Entering (CTRL)X cancels an input line. The monitor remains in the assembler mode.

EXAMPLES **DESCRIPTION**

```
>ASM 1000<CR>
1000 9D       NOP          >LDA #$55<CR>   Immediate mode addressing,
1000 A6 55    LDA  #$55                    requires # before operand.

1002 C1 00 9D CMP  $009D   >STA $60<CR>    Direct mode addressing, may
1002 B7 60    STA  $60                     have $ but not necessary.

1004 9D       NOP          >LDA 0,X<CR>    Index mode, if not offset
1004 E6 00    LDA  $00,X                   (,X) will be accepted.

1006 FB       ADD  ,X      >BRA $1000<CR>  Branch offsets calculated
1006 20 F8    BRA  $1000                   automatically, address
                                           required as conditional
                                           branch operand.

1008 FF       STX  ,X       >.<CR>         Assembler operation
                                           terminated.
>
```

Refer to the end of this chapter for additional operational information about using the assembler/disassembler.

3.6.2 Block Fill

BF <start addr> <end addr> <data>

where:

<start addr>	Lower limit for fill operation.
<end addr>	Upper limit for fill operation.
<data>	Fill pattern hexadecimal value.

The BF command repeats a specific pattern throughout a specifed user memory range.

The user should be very careful when modifying locations internal to the MCU device, to prevent inadvertent overwriting of port addresses, timer registers, and other such values.

EXAMPLES	DESCRIPTION
>BF 80 FF FF<CR>	Fill each byte of memory, $0080 -- $00FF, with data pattern FF.
>BF 1200 1200 0<CR>	Assign value 0 to location $1200.

3.6.3 Breakpoint Set

BR [<addr1 - addr5>]

The BR command sets an address into the breakpoint address table. During execution of the user program, a debug halt occurs immediately before the execution of any instruction address in the breakpoint table.

Do not place a breakpoint on a software interrupt (SWI) instruction because the monitor uses this instruction to breakpoint/single step a user program. However, the user may use the SWI instruction in the user program.

A maximum of five breakpoints may be set. After the user sets breakpoints, the screen shows the current breakpoint addresses, if any. Enter all multiple breakpoints on the same line.

COMMAND FORMATS	DESCRIPTION
BR	Display all current breakpoints.
BR <address>	Set breakpoint.

EXAMPLES	DESCRIPTION
>BR 1324<CR> Brkpts=1324 >	Set breakpoint at location $1324.
>BR 1324 1212 1100<CR> Brkpts=1324 1212 1100 >	Set three breakpoints. Breakpoints at same location set only one breakpoint.
>BR<CR> Brkpts=1324 1212 1100 >	Display all current breakpoints.

3.6.4 2MHz Clock Rate

FAST

The FAST command switches the EVS to a 2MHz clock rate, using the phase locked loop (PLL) on the E1. A value of $4E is stored in the PLL control register ($07).

EXAMPLE	DESCRIPTION
>**FAST**	Change bus speed to 2 MHz.
>	

3.6.5 Go

G [<start addr>]

where <start addr> is the optional starting address for program execution.

The G command starts execution of a user program (free run in real time). Execution starts at the current program counter (PC) address, or at a starting address the user specifies. Program execution continues until it reaches a breakpoint, until the user presses the abort switch (PFB SW1), or until the user presses the master reset switch (PFB SW3).

EXAMPLES	DESCRIPTION
>G<CR>	Go to user map and begin program execution at current PC address location.
>G 1000<CR>	Go to user map and begin program execution at PC address location $1000.
>G 1000<CR> Abort S=FF P=1004 A=55 X=FF C=E8 111.I...	Transfer of EVSbug monitor control. ABORT switch S1 is used to restore EVSbug monitor control if no breakpoints were preset prior to the G command entry.

3.6.6 Help

HELP

The HELP command displays EVS command information for quick reference.

EXAMPLE

```
>HELP<CR>

BREAK = Abort command,
CTRL-S = Freeze screen, CTRL-X = Cancel command line
ASM <START ADDR>- Assembler/disassembler
BF <START ADDR> <END ADDR> <DATA>- Block fill memory
BR [<ADDR1 - ADDR5>]- Set 1 to 5 breakpoints
CHCK [<START ADDR> <END ADDR>] <device>- Blank check MCU
COPY [<START ADDR> <END ADDR>] <device>- Copy MCU to memory
FAST- 2MHz phase lock loop (PLL) rate
G [<START ADDR>]- Execute user program
LOAD T - Download from port to memory
MD <START ADDR> [<END ADDR>]- Display memory
MM <ADDRESS>- Modify memory
NOBR [<ADDR1 - ADDR5>]- Remove breakpoints
P [<COUNT>]- Proceed 1-FF times through a breakpoint
PROG [<START ADDR> <END ADDR>] <device>- Program MCU from memory
RD- Register display
RM- Register modify
SLOW- 32KHz external oscilator rate
T [<COUNT>]- Trace 1-FF instructions
TURBO- Previousl PLL setup
VERF [<START ADDR> <END ADDR>] <device>- Verify MCU to memory
>
```

NOTE

The CHCK, COPY, PROG, and VERF commands of the multi-product EVSbug monitor pertain to programming OTPROMS and EPROMs. These commands do not apply to the E1 EVS.

3.6.7 Load T

LOAD T

The LOAD T command downloads object data in S-record format (see Appendix A) from an external host computer to EVS user pseudo ROM.

The S-record starting address must be a valid pseudo ROM memory location.

EXAMPLES	DESCRIPTION
>LOAD T<CR>	LOAD command entered to download data from host computer (e.g., IBM-PC) to EVS via terminal port.

NOTE

Object data for the LOAD T command must be in S-record format. Otherwise, the EVS will disard the data as quickly as the data arrives. Furthermore, the LOAD T command will not terminate until it receives a termination S-record.

Refer to the downloading procedures at the end of this chapter for additional information about the LOAD T command.

3.6.8 Memory Display

MD <start addr> [<end addr>]

where:

<start addr>	Beginning address of the memory to be displayed.
<end addr>	Optional ending address of the memory to be displayed.

The MD command displays a section of user memory from the starting address through the ending address. If the user does not enter an ending address, the display shows the 16 bytes beginning at the starting address. If the monitor prompt appears instead of memory values, it means that the starting address is greater than the ending address.

EXAMPLES

```
>MD 80 A0<CR>
0080    AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA   ................
0090    AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA   ................
00A0    AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA   ................
>


>MD 1122<CR>
1122    AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA   ................
>
```

3.6.9 Memory Modify

MM <address>

Where <address> is the starting memory location for display/modify.

The MM command lets the user interactively examine and modify contents of specified user memory locations. After entering this command, the user modifies and verifies data by entering these terminators:

[<data>]<CR>	Update location and sequence forward.
[<data>]^<CR>	Update location and sequence backward.
[<data>]=<CR>	Update location and reopen same location.
[<data>].<CR>	Update location and terminate.

An entry of only <CR> terminates the memory modify interactive operation. (CTRL)X cancels any input line without leaving the MM command.

The address must be a valid pseudo ROM memory address. If it is not, the invalid address and the message BAD MEMORY appear on the terminal screen.

EXAMPLES	DESCRIPTION
>MM 1000<CR>	Display memory location $1000.
1000=00>**AA=<CR>**	Change data at $1000 and reexamine location.
1000=AA>	
1001=11>**44^<CR>**	Change data at $1001 and back up one location.
1000=AA>	
1001=44>**33<CR>**	Change data at $1001 and terminate MM operation.
>	
>MM 1002<CR>	Display memory location $1002.
1002=22>**55<CR>**	Do not change data at $1003.
1003=AA>.**<CR>**	Terminate operation.
>	

3.6.10 Remove Breakpoint

NOBR [<addr1 - addr5>]

The NOBR command removes one or more breakpoints from the internal breakpoint table. After removing the breakpoint, the monitor displays any remaining breakpoint addresses.

COMMAND FORMATS	DESCRIPTION
NOBR	Removes all current breakpoints.
NOBR <address>	Removes breakpoint.

EXAMPLES **DESCRIPTION**

```
>NOBR 1200<CR>                    Remove breakpoint at location $1200.
Brkpts=1321    0080    1420
>

>NOBR 1321 80<CR>                 Remove breakpoints at locations
Brkpts=1420                       $1321 and $0080.
>

>NOBR<CR>                         Removes all breakpoints.
Brkpts=
>
```

3.6.11 Proceed

P [<count>]

The P command resumes execution from the current breakpoint, passing the current breakpoint <count> number of times (in hexadecimal, $FF max.) before returning control to the monitor. All other breakpoints are ignored during this command.

This command is ideal for applications where registers must be examined after a given number of passes within a software loop.

EXAMPLE	DESCRIPTION
>P 5<CR>	Current breakpoint location is passed five times before breakpoint returns control to the monitor.

3.6.12 Register Display

RD

The RD command displays the MCU register contents.

COMMAND FORMAT	DESCRIPTION
>RD	Contents of the following registers are displayed: S = Stack pointer P = Program counter A = Accumulator A X = Index register C = Condition codes

Condition code register (CCR) 111HINZC bits are displayed as follows:

All CCR bits set (logic 1)	C=FF	111HINZC
All CCR bits cleared (logic 0)	C=E0	111.....
Specific CCR bit set (I)	C=E8	111.I...
Specific CCR bits set (H, I, and Z)	C=FA	111HI.Z.

EXAMPLE

```
>RD<CR>
Regs
S=FF    P=1002    A=31    X=FF    C=F9        111HI..C
>
```

3.6.13 Register Modify

RM

The RM command modifies the contents of the MCU registers. The RM command requires no parameters. It begins by displaying the S (stack pointer) register contents and allowing changes to be made. Register contents appear in this order:

S	(stack pointer)
P	(program counter)
A	(accumulator A)
X	(index register)
C	(condition code)

After entering this command, the user modifies and verifies data by entering these terminators:

[<data>]<CR>	Update register and sequence forward.
[<data>]^<CR>	Update register and sequence backward.
[<data>]=<CR>	Update register and reopen same location.
[<data>].<CR>	Update register and terminate.

An entry of only <CR> terminates the register modify interactive operation. (CTRL)X cancels any input line without leaving the RM command. The user may not modify the stack pointer.

EXAMPLE	DESCRIPTION
>**RM<CR>**	Register modify command entered.
S=FF	
P=1002>**1000<CR>**	Change P register and go to A register.
A=31>**AA<CR>**	Change A register.
X=FF>	Examine X register.
C=F9>.**<CR>**	Examine C register and terminate command.
>	

3.6.14 32KHz Clock Rate

SLOW

The SLOW command switches the EVS clock rate to the 32KHz clock source. For this clock rate, the bus speed of the E1 is 16KHz.

The EVS enters the SLOW mode during power-up and when the user presses the master reset switch (PFB SW3).

EXAMPLE	DESCRIPTION
>**SLOW** >	Change bus speed to 16 KHz.

3.6.15 Trace

T [<count>]

<count> is the number of instructions (in hexadecimal, $FF max.) to execute.

The T command lets the user monitor program execution on an instruction-by-instruction basis. The user may execute several instructions at a time by entering an optional count value (up to $FF). Execution starts at the current program counter (PC). The PC displayed with the event message points to the next instruction to be executed. During the tracing operation, breakpoints are active: program execution stops when the PC reaches a breakpoint.

Do not trace a branching instruction (e.g., BRA) that branches to itself. The monitor places an SWI instruction on the object of the branch, so the instruction would never be executed. However, it would appear to the user that the instruction had executed. As long as the instruction is not a breakpoint address, the user may enter a G command while the PC points to this type of instruction. The user should not trace a clear interrupt mask (CLI) with interrupts enabled and pending.

If the user attempts to trace at an address that contains an invalid opcode, the message ILLEGAL/INSUFFICIENT ENTRY appears on the terminal screen.

SINGLE TRACE EXAMPLE

```
>T<CR>
1001  4C          INCA
S=FF     P=1001     A=00     X=FF     C=EA          111.I.Z.
>
```

MULTIPLE TRACE EXAMPLE

```
>T 2<CR>
1002  9D          NOP
S=FF     P=1002     A=01     X=FF     C=E8          111.I...
1003  20 FC       BRA    $1001
S=FF     P=1003     A=01     X=FF     C=E8          111.I...
>
```

3.6.16 Previous PLL Setup

TURBO

The TURBO command configures the phase locked loop (PLL) to the previous setup. To use TURBO, the user must:

1. Execute the SLOW command,
2. Execute the memory modify (MM) command, to adjust the PLL synthesizer speed select to the desired rate, then
3. Execute TURBO.

EXAMPLE	DESCRIPTION
>SLOW	Change bus speed to 16 KHz.
>MM $7 2D	Set 1 MHz phase lock loop (PLL) speed (bus continues at 16 KHz).
>TURBO	Enable PLL.
>	

3.7 ASSEMBLING/DISASSEMBLING PROCEDURES

The assembler/disassembler is an interactive assembler/editor for the source program. As the user enters each line of source code, the assembler/disassembler converts the line into machine language and stores it in memory. The assembler/disassembler disassembles the machine code of each instruction, in order to display the instruction mnemonic and operands. All valid opcodes are converted to assembly language mnemonic. All invalid opcodes return a form constant byte (FCB).

Use the ASM command to create, modify, or debug MC68HC05 MCU code. Assembler input must have exactly one space between the mnemonic and the operand. There must be no space between the operand and the index specification (,X) except in the case of indexed no offset. A carriage return must terminate assembler input. No comments may follow the instruction input.

After each new assembler input line, the new line is disassembled for the user before stepping to the new instruction. The new line may assemble to a different number of bytes than the previous one.

During disassembly of the Branch if High or Same (BHS) or Branch if Carry Clear (BCC) mnemonics, the screen displays the BCC mnemonic. During disassembly of the Branch if Lower (BLO) or Branch if Carry Set (BCS) mnemonics, the screen displays the BCS mnemonic.

The assembler automatically calculates branch address offsets, so the user should input the addresses, not offset values, as operands.

Terminate the assembler by entering a period (.) followed by a carriage return as the only entry on the command input line. Entering a carriage return alone on an input line steps to the next instruction.

The following pages show how to operate the assembler/disassembler by creating a typical program loop, and debugging the program using EVS monitor commands. The first example shows assembly of a typical program loop. Subsequent examples illustrate how to set a breakpoint, proceed from a breakpoint, display and modify registers, and initiate user program execution.

The program loop assembly is:

EXAMPLE PROGRAM					PROGRAM DESCRIPTION
>ASM 1000<CR>					*Enter assembler mode.*
1000	B7 B4	STA	$B4	**>CLRA<CR>**	*Clear inner-loop counter.*
1000	4F	CLRA			
1001	B4 24	AND	$24	**>CLRX<CR>**	*Clear outer-loop counter.*
1001	5F	CLRX			
1002	24 97	BCC	$009B	**>INCA<CR>**	*Increment inner-loop counter.*
1002	4C	INCA			
1003	97	TAX		**>BNE 1002<CR>**	*Wait for counter overflow.*
1003	26 FD	BNE	$1002		
1005	9F	TXA		**>INCX<CR>**	*Increment outer-loop counter.*
1005	5C	INCX			
1006	AF	FCB	$AF	**>BNE 1002<CR>**	*Wait for counter overflow.*
1006	26 FA	BNE	$1002		
1008	DF 00 80	STX	$0080,X	**>BRA 1000<CR>**	*Go to program start.*
1008	20 F6	BRA	$1000		
100A	80	RTI		**>.<CR>**	*Exit assembler mode.*

Typical routines performed on the preceding program loop are:

TERMINAL CRT/KEYBOARD	ROUTINE DESCRIPTION
>RD<CR>	*Register display user machine state.*
S=FF P=0FF8 A=60 X=FF C=E8 111.I...	
>	
>RM<CR>	*Modify program counter register.*
S=FF	
P=0FF8**>1000=<CR>**	
P=1000>**<CR>**	
A=60>.**<CR>**	
>BR 1003 1006<CR>	*Set breakpoints.*
Brkpts=1003 1006	
>G<CR>	*Begin execution of program.*
Brkpt	
S=FF P=1003 A=01 X=00 C=E8 111.I...	
>	
>P 45<CR>	*Proceed 45 times within loop.*
Brkpt	
S=FF P=1003 A=46 X=00 C=E8 111.I...	
>	
>NOBR 1003<CR>	*Remove breakpoint.*
Brkpts=1006	
>G<CR>	*Continue program execution.*
Brkpts	
S=FF P=1006 A=00 X=01 C=E8 111.I...	
>	
>T 2<CR>	*Monitor program execution.*
1002 4C INCA	
S=FF P=1002 A=00 X=01 C=E8 111.I...	
1003 26 FD BNE $1002	
S=FF P=1003 A=01 X=01 C=E8 111.I...	
>	

3.8 DOWNLOADING PROCEDURES

Downloading transfers information from a host computer to the EVS via the LOAD T command. The procedure described below lets the user download with an IBM personal computer (PC) host computer.

The LOAD T command moves data in S-record format (see Appendix A) from an external host computer to the EVS user pseudo ROM.

Stop an active I/O port downloading operation by pressing any alphanumeric key on the terminal keyboard. An incorrect keyboard entry may cause a terminal lockup condition if made during downloading. To correct such a lockup, press any alphanumeric keyboard key.

Before downloading from any IBM-PC, make sure that both IBM-PC and EVS baud rates are 9600, and that the IBM-PC asynchronous port is configured for terminal operation mode. If the asynchronous port is hard-wired for host operation mode and cannot be reconfigured for a terminal operation mode, a null modem (cross-coupled transmit, receive, and associated handshake lines) is required.

NOTE

For downloading, use a single RS-232C cable assembly to connect the IBM-PC to the EVS. Connect the cable to EVS terminal I/O port connector P2.

Follow this example to download from the IBM-PC to the EVS:

EXAMPLE	DESCRIPTION
C>**KERMIT<CR>**	IBM-PC prompt. Enter Kermit program.
IBM-PC Kermit-MS VX.XX	
Type ? for help	
Kermit-MS>**SET BAUD 9600<CR>**	Set IBM-PC baud rate.
Kermit-MS>**CONNECT<CR>**	Connect IBM-PC to EVS.
[Connecting to host, type Control-] C to return to PC]	
<CR>	
>LOAD T<CR>	EVS download command (via terminal
(CTRL)]C<CR>	port) entered.
Kermit-MS>**PUSH<CR>**	
The IBM Personal Computer DOS	
Version X.XX (C)Copyright IBM Corp 1981, 1982, 1983	
C>**TYPE (File Name) > COM1<CR>**	Motorola S-record file name.
C>**EXIT<CR>**	S-record downloading completed.
Kermit-MS>**CONNECT<CR>**	Return to EVS monitor program.
>(CTRL)]C<CR>	
Kermit-MS>**EXIT<CR>**	Exit Kermit program.

CHAPTER 4

FUNCTIONAL DESCRIPTION

4.1 INTRODUCTION

This chapter is an overall description of the EVS. This description includes a block diagram (Figure 4-1) of the EVS circuits and I/O ports. This chapter also includes a memory map diagram (Figure 4-2).

4.2 EVS DESCRIPTION

The EVS uses its resident MC68HC05E1 MCU to evaluate a target system based on an MC68HC05E1 MCU. The EVS contains switchable monitor and user memory maps that allow modification of user memory and execution of user programs. Monitor ROM firmware controls data transfer within the EVS; an external RS-232C compatible terminal controls the monitor ROM firmware.

Figure 4-1 is a block diagram of EVS functionality. The EVS consists of an emulator module (EM) and a platform board (PFB) that have these functional circuits:

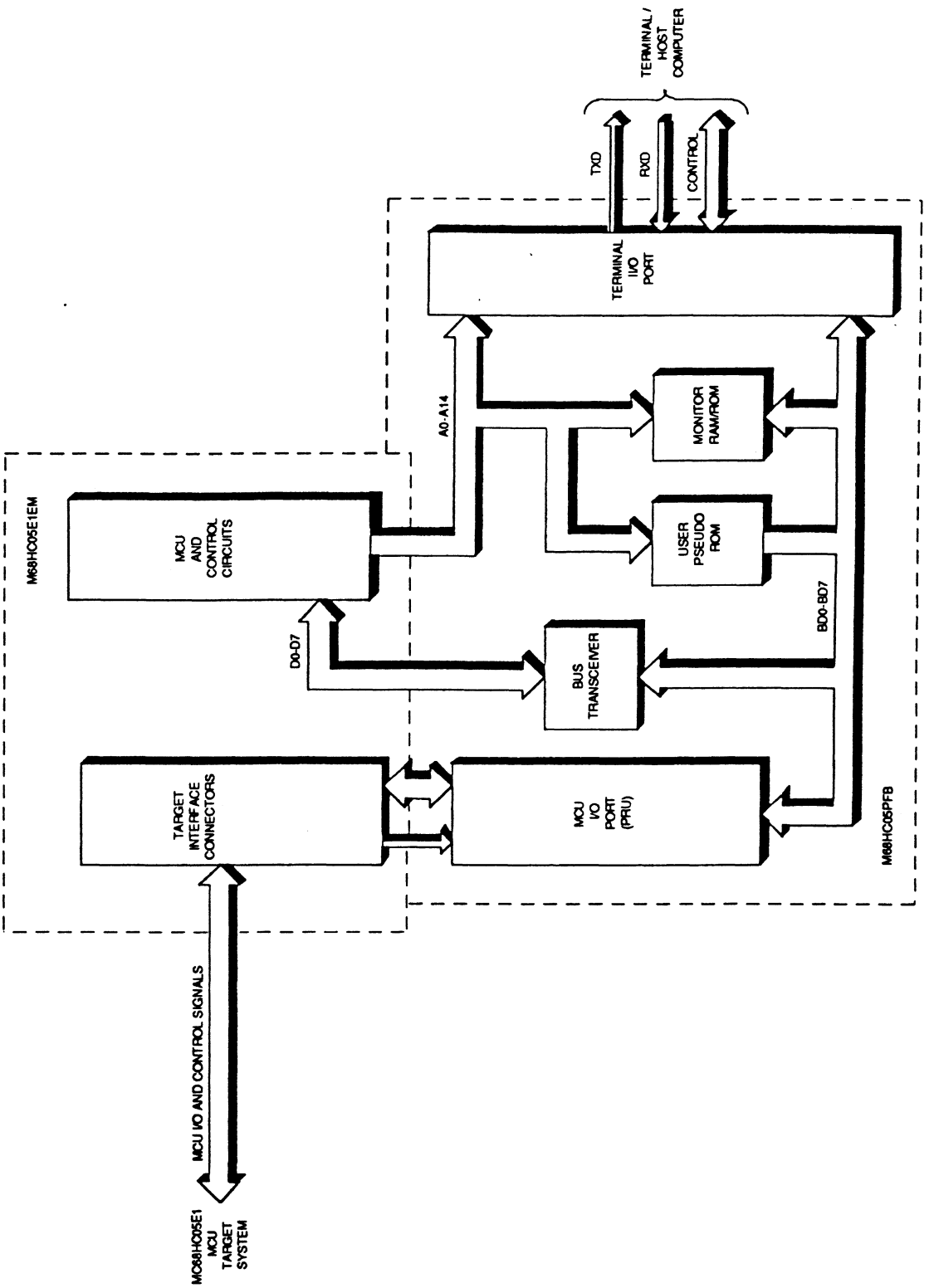
- MCU and control
- Monitor and user memory
- Terminal I/O port
- MCU extension I/O port

4.2.1 MCU and Control Circuits

The EVS contains a resident MC68HC05E1 MCU and associated control circuits, which provide the basic capabilities for target system use.

Figure 4-1 shows the specific control circuits implemented into the EVS:

- Map switching
- Abort
- Address decoding


TERMINAL / HOST COMPUTER
TXD
RXD
CONTROL
TERMINAL I/O PORT
A0-A14
MONITOR RAM/ROM
USER PSEUDO ROM
M68HC05E1EM
MCU AND CONTROL CIRCUITS
D0-D7
BUS TRANSCEIVER
BD0-BD7
TARGET INTERFACE CONNECTORS
MCU I/O PORT (PRIU)
M68HC05PFB
MCU I/O AND CONTROL SIGNALS
M68HC05E1 MCU TARGET SYSTEM

4.2.1.1 Map Switching. The EVS operates in either of two memory maps, monitor or user, as Figure 4-2 illustrates. There are two types of memory map switching: temporary switching for modification of user memory, and permanent switching for execution of user programs.

Modifying user pseudo ROM (RAM) requires temporary map switching. The EVS fetches the opcode and operand fom the monitor map, switches memory maps for one cycle, and executes the read or write cycle in user space. On the next cycle, the EVS automatically switches back to the monitor map.

Permanent map switching is initiated by a command from the monitor, or by the user pressing the user reset switch (PFB SW2). The EVS fetches the return from interrupt (RTI) opcode from the monitor map, switches memory maps, and fetches user register contents from the user stack. Then the EVS fetches program counter (PC) contents from the user stack and execution proceeds from the current PC value.

When the user presses PFB SW2, the EVS resets the MCU and user I/O ports, and switches from the monitor to the user memory map. The MCU fetches the reset vector from the user map. Breakpoints are ignored during this operation.

Execution of user code continues until a software interrupt (SWI) is decoded on the data bus during a load instruction register (LIR) cycle. SWI occurs when either a breakpoint is detected, or when the user presses the abort switch (PFB SW1). After saving user register contents on the user stack, the EVS switches back to the monitor map. An SWI that occurs when no breakpoint is set, or when the abort switch is not activated, does not cause memory map switching to take place. This allows user SWIs to be executed in real time.

The abort switch (PFB SW1), when activated, forces an SWI on the data bus during the LIR cycle. The EVS saves user register contents on the user stack, and switches to the monitor map. The abort switch has no effect on the monitor map. Abort switch activation while in the user map re-enters the monitor map (assuming the MCU is operating properly in the user map).

NOTE

This memory map switching operation assumes proper operation in the user map. If the MCU is not operating properly in the user map (i.e., an illegal opcode or stop instruction executed) the abort switch may not cause a map switch.

4.2.1.2 Abort. The abort circuitry generates internal abort signals when the user presses the abort switch (PFB SW1). These signals place a software interrupt opcode on the MCU data bus synchronously with the LIR signal, resulting in a switch to the monitor map.

4.2.1.3 Address Decoding. For address decoding, the EVS relies on a resident C22V10 field programmable logic array (FPLA) device. The FPLA provides the necessary chip select signals for memory and peripheral device circuitry that are memory mapped in the EVS.

4.2.2 Monitor and User Memory

As explained on the preceding page, the EVS operates in either the monitor or the user memory maps. Figure 4-2 shows these maps.

4.2.2.1 Monitor Map Area. The 8K monitor map area contains the MCU I/O ports and internal registers, DACIA (terminal), map switch register, monitor scratch pad RAM, monitor stack capture register, and 8K bytes monitor ROM. The EVSbug monitor EPROM contents, in the monitor ROM, are not available to the user.

All monitor operations are controlled via the monitor I/O (terminal DACIA). The DACIA is available only in the monitor map. User programs in the user map cannot access these peripheral ports.

The monitor uses the scratch pad RAM and stack capture register for such general operations as temporary data storage, command entries, and downloading.

The monitor map switch register is at address $0050. It is used for temporary and permanent map switching operations, user memory protection when in the monitor map, and breakpoint/trace/abort monitor flagging. Only the monitor controls the monitor map switch register.

4.2.2.2 User Map Area. Figure 4-2 shows the user map areas, which consist of the MCU I/O ports and internal registers, user RAM, and user pseudo ROM.

User program space (user pseudo ROM) is RAM. This RAM is write-protected via PFB jumper header J5 during user program execution. Paragraph 2.3.2.5 provides additional information about write-protect header J5. This feature requires all programs to be ROMable and protects against program errors that otherwise would overwrite the program space.

NOTE

The entire 8K-byte user map (pseudo ROM) is available to the user, although all M68HC05 MCU family devices do not have user ROM throughout the entire 8K-byte memory map. Refer to the specific device data sheet for valid program space locations.

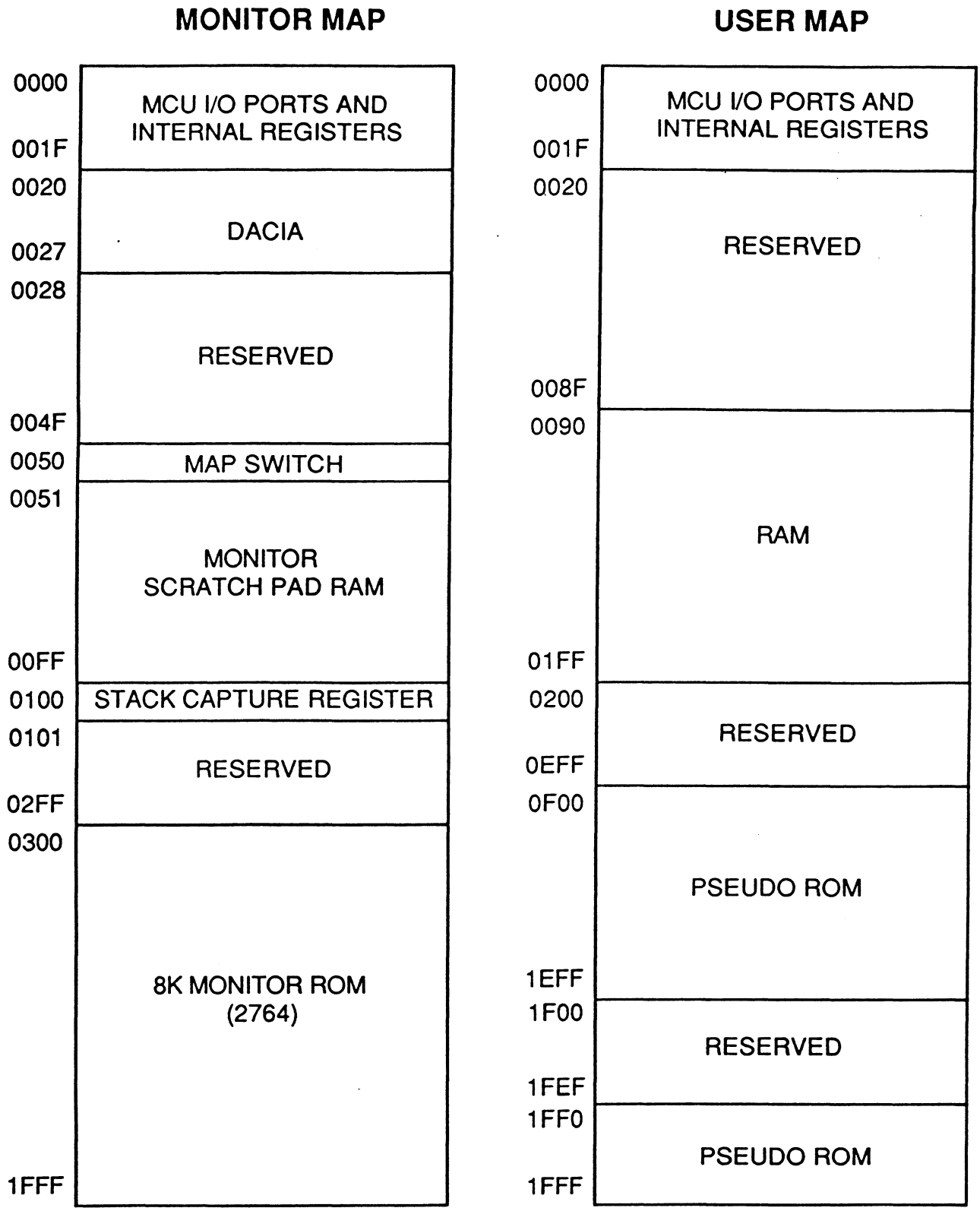


FIGURE 4-2. EVS (E1) Memory Map

4.2.3 Terminal I/O Port

The terminal I/O port (PFB connector P2) uses an R65C52 dual asynchronous communications interface adapter (DACIA) high speed device. This device contains an independent full duplex channel with buffered receiver and transmitter. The internal baud rate is set at 9600 on the EVS. Channel control and monitoring are also provided on the DACIA.

The EVS terminal I/O port communicates with an RS-232C compatible terminal via a user-supplied cable assembly. The R65C52 DACIA terminal interface circuitry provides communication and data transfer operations for the EVS and user terminal. Because the DACIA does not have internal timing logic, auto baud rate capabilities are not implemented on the EVS. Therefore the baud rate is fixed at 9600 for the terminal port. RS-232C drivers/receivers are also implemented for this port. For connector P2 pin assignments and signal descriptions, refer to Chapter 5.

Files may be downloaded through the terminal I/O port via the LOAD T command.

4.2.4 MCU Extension I/O Port

The EVS has a user HCMOS-compatible MCU extension I/O port (EM connector P4) for target-system evaluation of MC68HC05E1 HCMOS MCU devices. To connect a target system to the EVS, use a user-supplied 28-pin dual-in-line plastic (DIP) cable assembly. For connector P4 pin assignments and signal descriptions, refer to Chapter 5.

FIGURE 4-1. EVS Block Diagram

CHAPTER 5

SUPPORT INFORMATION

5.1 INTRODUCTION

The tables of this chapter describe EVS connector signals. This chapter also explains how to obtain EVS schematic diagrams and parts lists.

5.2 CONNECTOR SIGNAL DESCRIPTIONS

The EVS platform board (PFB) has an RS-232C I/O port (PFB connector P2) to connect the EVS to an RS-232C compatible terminal or host computer.

The MCU extension I/O port connector (EM connector P4) connects the EVS to target system equipment. This connector permits MC68HC05E1 MCU device evaluation.

EM connector P1 connects a logic analyzer to the EVS. PFB connector P1 connects an external power supply to the EVS.

Tables 5-1 through 5-4 list pin assignments for these connectors:

Table 5-1. EM P1 logic analyzer connector

Table 5-2. EM P4 30-pin MCU extension I/O port connector

Table 5-3. PFB P1 input power connector

Table 5-4. PFB P2 RS-232C I/O port connector

Connector signals are identified by pin number, signal mnemonic, and signal name and description.

TABLE 5-1. EM P1 Logic Analyzer Connector Pin Assignments

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
1, 3, 4, 10, 12, 14, 31, 33 - 36, 40	-----	No connection
2, 6	GND	GROUND
5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27	LA11 - LA0	LATCHED ADDRESSES (bits 11 - 0) - MCU latched output address bus.
8	LA12	LATCHED ADDRESSES (bit 12) - MCU latched output address bus.
16, 18, 20, 22, 24, 26, 28, 30	AD7 - AD0	DATA BUS (bits 7 - 0) - MCU multiplexed I/O data bus.
29	LR/W	LATCHED READ/WRITE - Active-high output signal that indicates the direction of data transferred on the bus.
32	LIR	LOAD INSTRUCTION REGISTER - Open-drain, active-low output signal that indicates an instruction is starting.
37	VCC	+5 VDC POWER - Input voltage (+5 Vdc @ 1.0 A) used by the EVS logic circuits.
38	E	EXTERNAL CLOCK - Internally generated output clock signal used as a timing reference. The frequency of E clock is 1/2 the input frequency of the signal on the OSC1 pin.
39	RESET	RESET - Active-low bidirectional signal for starting an EVS reset.

TABLE 5-2. EM P4 30-Pin MCU Extension I/O Port Pin Assignments

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
1, 3, 8, 26, 29, 30	-----	No connection
2	IRQ	TARGET INTERRUPT REQUEST - Active-low input signal from the target that asynchronously applies an MCU interrupt.
4	RST	TARGET RESET - Active-low input signal from the target system that starts a system reset.
5, 7, 9, 11, 13, 15, 17, 19	PA0 - PA7	PORT A (bits 0 - 7) - General purpose I/O lines controlled by software via data direction and data registers.
6	OSC1	CLOCK OSCILATOR - External clock input signal.
10, 12, 14, 16, 18, 20, 22, 24	PB7 - PB0	PORT B (bits 7 - 0) - General purpose I/O lines controlled by software via data direction and data registers.
21, 23, 25, 27	PC0 - PC3	PORT C (bits 0 - 3) - General purpose I/O lines controlled by software via data direction and data registers.
28	GND	GROUND

TABLE 5-3. PFB P1 Input Power Connector Pin Assignments

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
1	GND	GROUND
2	+5V	+5 VDC POWER - Input voltage (+5 Vdc @ 1.0 A) used by the EVS logic circuits.
3	VPP	Not applicable to this EVS.

TABLE 5-4. PFB P2 RS-232C I/O Port Connector Pin Assignments

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
1	GND	GROUND
2	TXD	TRANSMIT DATA - Serial data input line.
3	RXD	RECEIVE DATA - Serial data output line.
4	RTS	REQUEST TO SEND - Input signal that requests permission to transfer data.
5	CTS	CLEAR TO SEND - Output signal that indicates a ready-to-transfer data status.
6	DSR	DATA SET READY - Output signal (held high) that indicates on-line/in-service/active status.
7	SIG-GND	SIGNAL GROUND - Signal ground or common return connection between the EVS and RS-232C compatible terminal.
8	DCD	DATA CARRIER DETECT - Output signal (held high) that indicates detection of an acceptable carrier signal.
9-25	------	No connection

5.3 SCHEMATICS AND PARTS LISTS

The EVS is shipped with current schematic diagrams and parts lists for both the EM and the PFB printed circuit boards (PCBs). A user who requires schematic diagrams or a parts list for another revision of either PCB should fill out and mail the form on page 5-7.

Alternatively, the user may order schematic diagrams and parts lists by calling 512-891-EVMS. (The user should have EVS (EM and PFB) PCB revision levels and serial numbers available.)

PARTS LIST
and
SCHEMATIC DIAGRAM

To provide you with the parts list and schematic diagram for this low-cost evaluation system, please fill out the applicable information and return this letter to the following address:

Motorola Inc.
Microcontroller Division
6501 Wm. Cannon Drive West
Austin, Texas,78735-8598

Attention: EVB/EVM/EVS Products
Maildrop OE319

Please type or print the following information:

(Cut along dotted line.)

NAME: _____

TITLE: _____

COMPANY: _____

DIVISION: _____

ADDRESS: _____

MAILDROP: _____

CITY: _____

STATE: _____

ZIP: _____

EVS MODEL NUMBER: _____

EVS REVISION LETTER: _____

EVS SERIAL NUMBER: _____

EVS model number, revision letter, and serial number must be supplied in order for this request form to be processed.

APPENDIX A

S-RECORD INFORMATION

INTRODUCTION

The S-record format for output modules encodes programs or data files in a printable format for transportation between computer systems. This facilitates S-record editing and permits visual monitoring of the transportation process.

S-RECORD CONTENT

S-records are character strings of several fields which identify the record type, length, memory address, code/data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first character represents the high-order 4 bits, and the second the low-order 4 bits of the byte.

The 5 fields of an S-record are:

Field compositions are:

FIELD	PRINTABLE CHARACTERS	CONTENTS
Type	2	S-record type - S0, S1, etc.
Record length	2	The count of the character pairs in the record, excluding the type and record length.
Address	4, 6, or 8	The 2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
Code/data	0-2n	From 0 to n bytes of executable code, memory loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
Checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

There are three possible terminators for an S-record: CR, LF, and NULL. Additionally, an S-record may have an optional initial field to accommodate such other data as line numbers generated by some time-sharing systems.

The record length (byte count) and checksum fields ensure accuracy of transmission.

S-RECORD TYPES

There are eight types of S-records, to accommodate the various needs of the encoding, transportation, and decoding functions. The various Motorola upload, download, and other record transportation control programs, as well as cross assemblers, linkers, and other file-creating or debugging programs, use only the S-record types that serve the purpose of the program. For specific information on which S-records a particular program supports, consullt the user manual for that program.

NOTE

The EVS monitor supports only S0, S1, and S9 record types. All data before the first S1 record is ignored. All subsequent records must be S1 type, except the S9 record, which terminates data transfer.

An S-record format module may contain S-records of the following types:

S0	The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. The address field is normally zeroes.
S1	A record containing code/data and the 2-byte address at which the code/data is to reside.
S2-S8	Not applicable to EVS.
S9	A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. If such an address is not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

There is only one termination record for each block of S-records. Normally, there is only one header record, although multiple header records are possible.

S-RECORD CREATION

Several dump utilities, debuggers, cross assemblers, or cross linkers may produce S-record format programs. Several programs are available for downloading a file in S-record format from a host system to an 8-bit or 16-bit microprocessor-based system.

S-RECORD EXAMPLE

This example shows how a typical S-record format module is printed or displayed:

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

This example module consists of an S0 header record, four S1 code/data records, and an S9 termination record.

The S0 header record consists of these character pairs:

S0	S-record type S0, indicating a header record.
06	Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
00 00	Four-character 2-byte address field, zeroes.
48 44 52	ASCII H, D, and R - "HDR".
1B	Checksum of S0 record.

The exlanation of the first S1 code/data record is:

S1	S-record type S1, indicating a code/data record to be loaded/verified at a 2-byte address.
13	Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.
00	Four-character 2-byte address field; hexadecimal address 0000, indicates location where the following data is to be loaded.

The next 16 character pairs are the ASCII bytes of the actual program code/data. In this assembly language example, the hexadecimal opcodes of the program are written in sequence in the code/data fields of the S1 records:

OPCODE	INSTRUCTION	
28 5F	BHCC	$0161
24 5F	BCC	$0163
22 12	BHI	$0118
22 6A	BHI	$0172
00 04 24	BRSET	0,$04,$012F
29 00	BHCS	$010D
08 23 7C	BRSET	4,$23,$018C

. (Balance of this code is continued in the code/data fields of the
. remaining S1 records, and stored in memory location 0010, etc..)
.

2A Checksum of the first S1 record.

The second and third S1 code/data records each also contain $13 (19) character pairs and are ended with checksums 13 and 52, respectively. The fourth S1 code/data record contains 07 character pairs and has a checksum of 92.

The S9 termination record is explained as follows:

S9 S-record type S9, indicating a termination record.

03 Hexadecimal 03, indicating three character pairs (3 bytes) follow.

00 Four-character 2-byte address field, zeroes.
00

FC Checksum of S9 record.

Each printable character in an S-record is encoded in hexadecimal (ASCII in this example) representation of the binary bits which are actually transmitted. For example, the first S1 record above is sent as shown below.

TYPE				LENGTH				ADDRESS								CODE/DATA										CHECKSUM			
S		1		1		3		0		0		0		0		2		8		5		F		---		2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	---		3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	---		0011	0010	0100	0001

M68HC05E1
CPU BOARD

NOTES, UNLESS OTHERWISE SPECIFIED

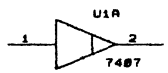
1. VCC PIN LOCATIONS :

   VCC IS APPLIED TO PIN 8 OF ALL 8-PIN IC's, PIN 14 OF ALL 14-PIN IC's, PIN 16 OF ALL 16-PIN IC's, PIN 20 OF ALL 20-PIN IC's, ETC.

2. GROUND PIN LOCATIONS :

   GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.

3. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR OF GATES ARE SHOWN AS FOLLOWS :



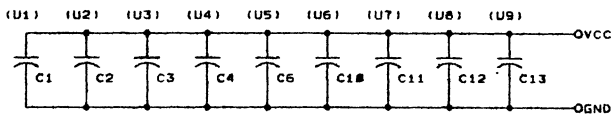
7407	= DEVICE TYPE
1 AND 2	= PIN NUMBERS
U1A	= REFERENCE DESIGNATORS

4. RESISTANCE VALUES ARE IN OHMS.
5. RESISTORS ARE 1/4 WATT, 5%.
6. CAPACITANCE VALUES ARE IN MICROFARADS.

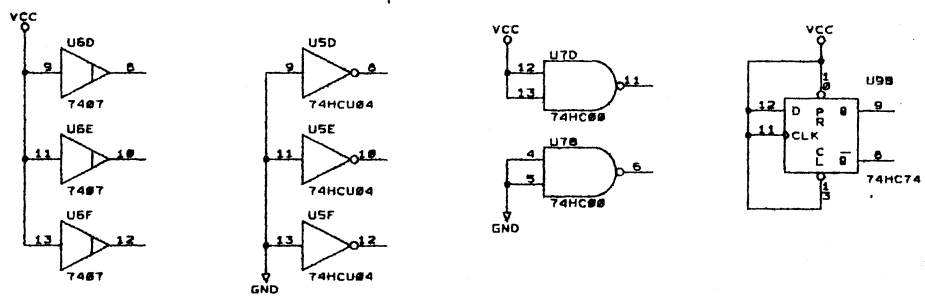
REVISIONS

ECN #	SCH REV	DESCRIPTION	DATE
15	2	p2 - tap from oscillator for osc1 wrong p2 - cap for XFC connected wrong, PLLTST to GND p2 - added r/c filter to vddsync R1B0,C1B0 p3 - decode table wrong for MRAM p4 - added cap to fix t-irq problem, C1B1	2/14/91

Decouple Caps for ICs as labeled.
All caps are 0.1 uF @ 50 V



Spare Gates

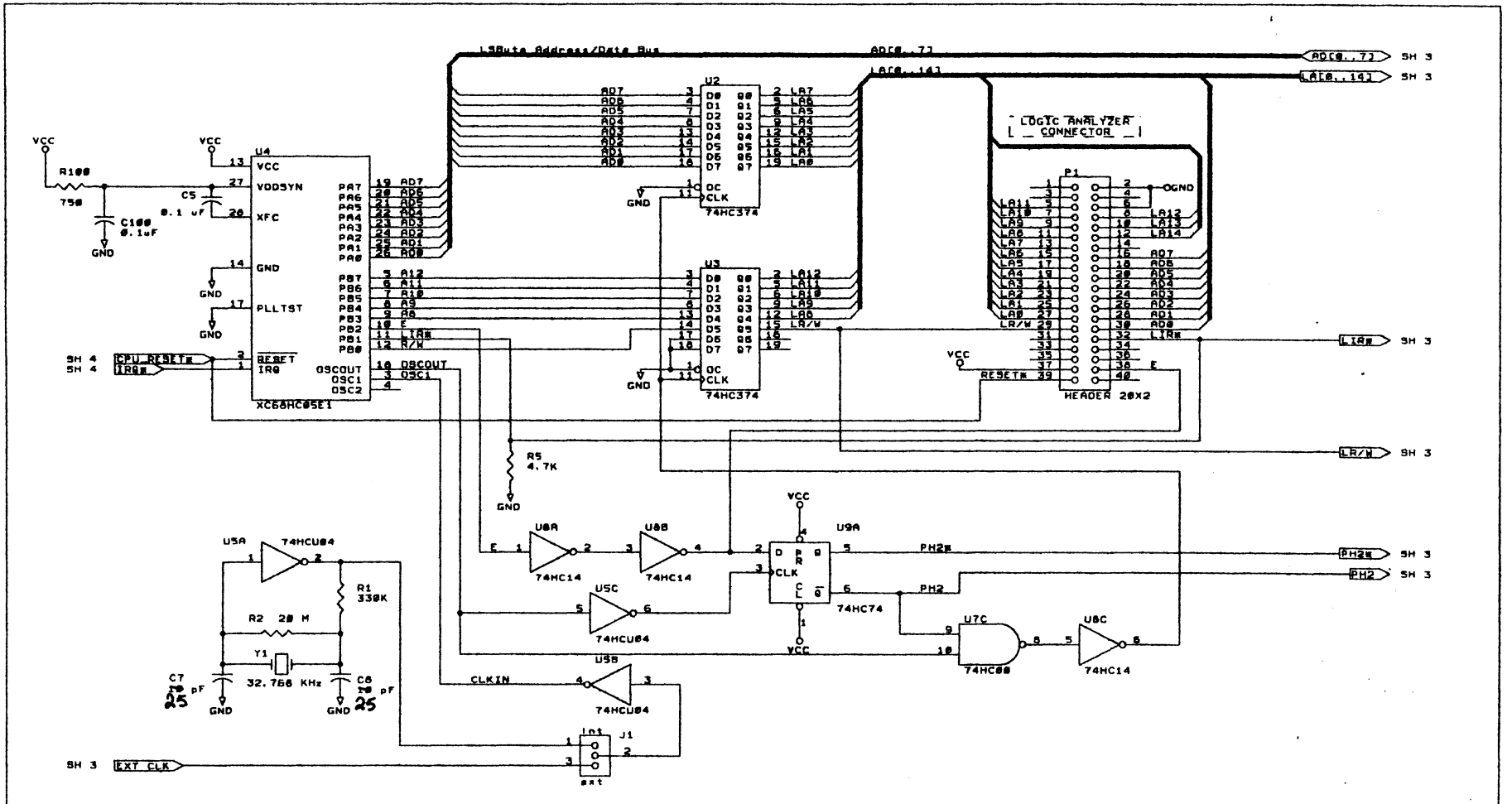


ORCAD IV FLAT FILES

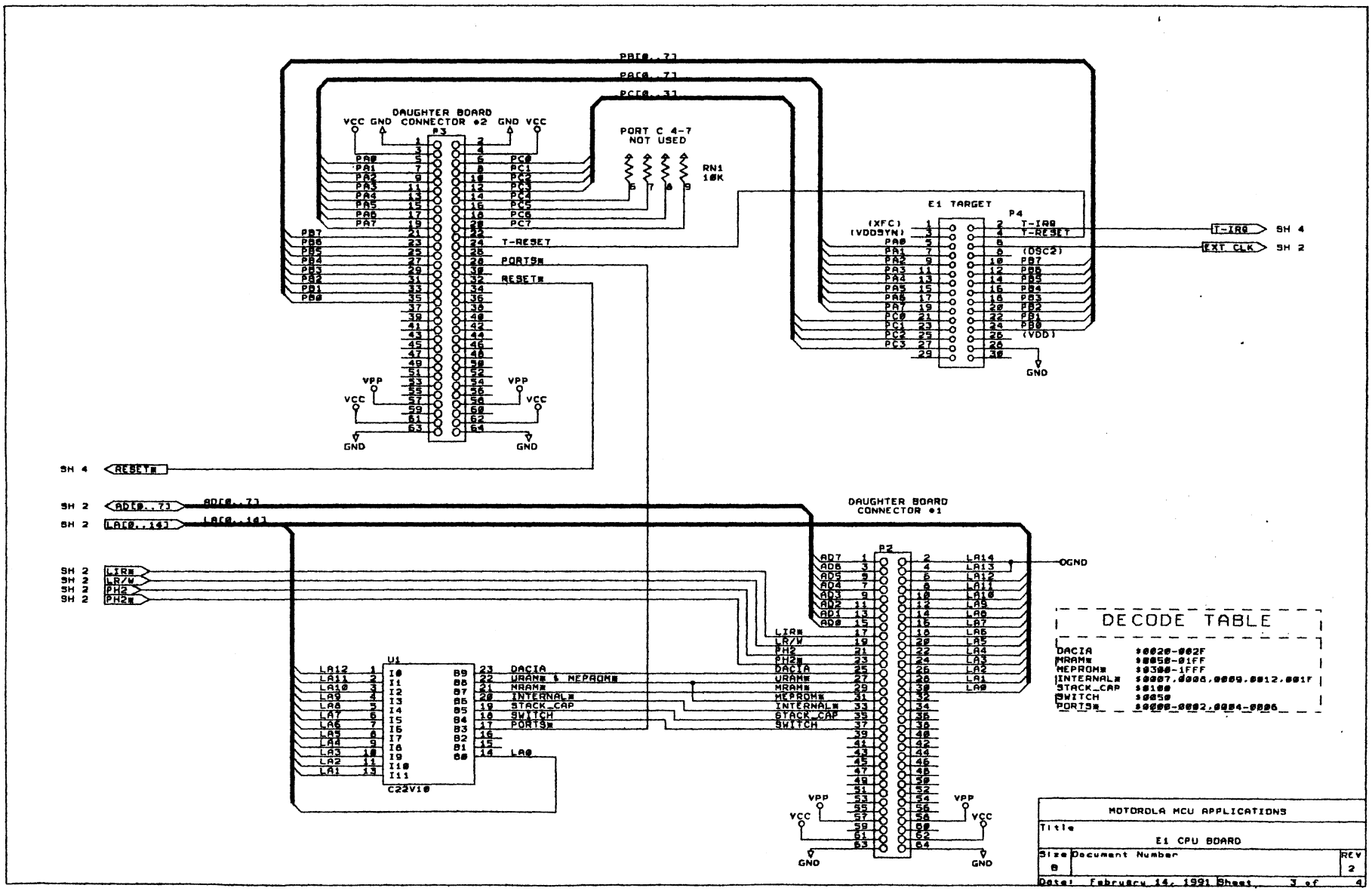
ILINK
I 05E1R252.SCH
I 05E1R253.SCH
I 05E1R254.SCH

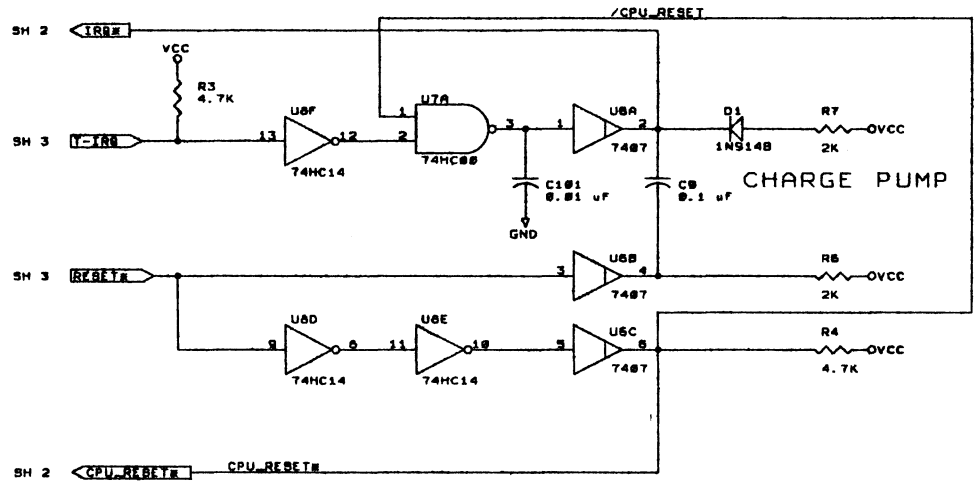
COMPUTER GENERATED DRAWING : DO NOT REVISE MANUALLY

MOTOROLA MCU APPLICATIONS		
Title: E1 CPU BOARD		
Size: B	Document Number:	REV: 2
Date: February 14, 1991	Sheet: 1 of 4	



MOTOROLA MCU APPLICATIONS		
Title: E1 CPU BOARD		
Size: B	Document Number	REV: 2
Date: February 14, 1991	Sheet 2 of 4	


PB[0..7]
PA[0..7]
PC[0..3]
DAUGHTER BOARD
VCC GND CONNECTOR #2 GND VCC
P3
PORT C 4-7
NOT USED
RN1
10K
E1 TARGET
P4
(XFC)
(VDDSYN)
(OSC2)
(VDD)
T-IRQ
T-RESET
EXT CLK
SH 4
SH 2
T-RESET
PORTS#
RESET#
VPP
VCC
GND
SH 4 RESET#
SH 2 AD[0..7]
SH 2 LA[0..14]
SH 2 LIR#
SH 2 LR/W
SH 2 PH2
SH 2 PH2#
DAUGHTER BOARD
CONNECTOR #1
P2
OGND
U1
C22V10
DACIA
VRAM# & MEPROM#
MRAM#
INTERNAL#
STACK_CAP
SWITCH
PORTS#
LA0
DECODE TABLE
DACIA $0020-002F
MRAM# $0050-01FF
MEPROM# $0300-1FFF
INTERNAL# $0007,0008,0009,0012,001F
STACK_CAP $0100
SWITCH $0050
PORTS# $0000-0002,0004-0006
MOTOROLA MCU APPLICATIONS
Title
E1 CPU BOARD
Size B
Document Number
REV 2
Date: February 14, 1991 Sheet 3 of 4



MOTOROLA MCU APPLICATIONS		
Title: E1 CPU BOARD		
Size: B	Document Number:	REV: 2
Date: February 14, 1991	Sheet 4 of 4	