

EASE64168

User's Manual

M6416x Series microcontrollers
Program Development Support System

◆ *This manual contains important information pertaining to the safe use of the above product. Before using the product, read these safety notes thoroughly and then keep this manual handy for immediate reference.*

SECOND EDITION
ISSUE DATE: Dec., 1999

NOTICE

1. The information contained herein can change without notice owing to product and/or technical improvements. Before using the product, please make sure that the information being referred to is up-to-date.
2. The outline of action and examples for application circuits described herein have been chosen as an explanation for the standard action and performance of the product. When planning to use the product, please ensure that the external conditions are reflected in the actual circuit and assembly designs.
3. When developing and evaluating your product, please use our product below the specified maximum ratings and within the specified operating ranges including, but not limited to, operating voltage, power dissipation, and operating temperature.
4. **OKI assumes no responsibility or liability whatsoever for any failure or unusual or unexpected operation resulting from misuse, neglect, improper installation, repair, alteration or accident, improper handling, or unusual physical or electrical stress including, but not limited to, exposure to parameters beyond the specified maximum ratings or operation outside the specified operating range.**
5. Neither indemnity against nor license of a third party's industrial and intellectual property right, etc. is granted by us in connection with the use of product and/or the information and drawings contained herein. No responsibility is assumed by us for any infringement of a third party's right which may result from the use thereof.
6. The products listed in this document are intended only for use in development and evaluation of control programs for equipment and systems. These products are not authorized for other use (as an embedded device and a peripheral device).
7. Certain products in this document may need government approval before they can be exported to particular countries. The purchaser assumes the responsibility of determining the legality of export of these products and will take appropriate and necessary steps at their own expense for these.
8. No part of the contents contained herein may be reprinted or reproduced without our prior permission.
9. MS-DOS is a registered trademark of Microsoft Corporation.

PREFACE

This manual explains the operation of the EASE64168 in-circuit emulator for the M6416x series microcontrollers (MSM64162D, MSM64162, MSM64162A, MSM64164C, ML64168) built on Oki Electric's nX-4/20 and nX-430 CMOS 4-bit core.

The following are related manuals.

MSM64162A User's Manual

- MSM64162A hardware description

MSM64162 User's Manual

- MSM64162 hardware description

MSM64162D User's Manual

- MSM64162D hardware description

MSM64164C User's Manual

- MSM64164C hardware description

ML64168 User's Manual

- ML64168 hardware description

nX-4/20, 4/30 Core Instruction Manual

- OLMS 64K series instruction set description

Structured Assembler SASM64K User's Manual

- SASM64K assembler operation description

- SASM64K assembly language description

MASK162A User's Manual

- MASK162A (ML64162A mask option generator) operation description

MASK162 User's Manual

- MASK162 (MSM64162 and MSM64162D mask option generator) operation description

MASK164 User's Manual

- MASK164 (MSM64164 mask option generator) operation description

MASK168 User's Manual

- MASK168 (ML64168 mask option generator) operation description

Table of Contents

Preface	0-1
1. Product Inquiries	0-2
2. Using this Product Safely and Properly	0-3
2.1 Icons	0-3
2.2 Important Safety Notes	0-4
3. Notation	0-7
4. Manual Organization	0-8
5. Package Contents	0-9
5.1 Verify Shipping Contents	0-9
Chapter 1. Before Starting	1-1
1. Confirm Shipping Contents	1-3
2. Confirm Floppy Disk Contents	1-8
2.1 EASE64X Debugger Operating Environment	1-8
2.2 Operating System	1-8
Chapter2. Overview	2-1
1. EASE64168 Emulator Configuration	2-2
1.1 EASE64168 Emulation Kit	2-2
1.2 SASM64K Structured Assembler	2-3
1.3 EASE64X Debugger	2-4
1.4 Mask option generator (MASK162/162A/164/168)	2-4
1.5 System Configuration	2-5
2. Program Development With EASE64168	2-6
2.1 General Program Development and EASE64168	2-6
2.2 From Source File To Object File	2-7
2.3 Generating Mask Option Files	2-8
2.4 Files Usable with the EASE64168 Emulator	2-9
Chapter3. EASE64168 Emulator	3-1
1. EASE64168 Functions	3-3
1.1 Overview	3-3
1.2 Changing Target Chips	3-5
1.3 Emulation Functions	3-6
1.4 Realtime Trace Functions	3-7
1.5 Break Functions	3-9
1.6 Performance/Coverage Functions	3-12
1.7 EPROM Programmer	3-13
1.8 Indicators	3-14

2. EASE64168 Emulator Initialization-----	3-15
2.1 Setting Operating Frequency -----	3-15
2.2 EASE64168 Switch Settings -----	3-20
2.3 Connecting The ADC POD -----	3-21
2.4 Confirming EASE64168 Power Supply Voltage-----	3-23
2.5 Starting the EASE64168 Emulator-----	3-24
3. EASE64X Debugger Commands-----	3-30
3.1 Debugger Command Syntax-----	3-30
3.1.1 Character Set-----	3-31
3.1.2 Command Format -----	3-31
3.1.3 Command Summary -----	3-33
3.2 History Functions -----	3-50
3.3 Special Keys For Raising Command Input Efficiency-----	3-52
3.4 Command Details -----	3-56
Call OS Shell-----	3-56
Line Assembler Command-----	3-57
Batch Processing -----	3-59
Change contents of target chip registers-----	3-61
Change Cycle Counter-----	3-65
Display/Change Clock Mode-----	3-66
Change Code Memory-----	3-67
Change Data Memory -----	3-69
Change Display Register-----	3-71
Set Target Chip-----	3-73
Display/Change Interface Power Supply -----	3-74
Change Trace Data Memory -----	3-75
Change Trace Object -----	3-76
Display contents of target chip registers-----	3-78
Disassemble Command -----	3-81
Display Break Condition Register-----	3-83
Display Break Point Bits-----	3-84
Display Break Status-----	3-86
Display Cycle Counter -----	3-87
Display Code Memory -----	3-88
Display Cycle Counter Trigger -----	3-90
Display Data Memory -----	3-91
Display Display Register-----	3-93
Display Instruction Executed Bits -----	3-94
Display Trace Data Memory -----	3-96
Display Trace Memory-----	3-97

Table of Contents

Display Trace Object-----	3-101
Display Trace Pointer-----	3-102
Display Trace Enable Bits -----	3-103
Display Trace Trigger-----	3-105
Enable Break Point Bits -----	3-106
Enable Trace Enable Bits -----	3-107
Terminate the Debugger and Exit to OS-----	3-108
Expand Code Memory-----	3-109
Fill Break Point Bits -----	3-111
Fill Code Memory-----	3-112
Fill Data Memory-----	3-113
Fill Trace Enable Bits-----	3-114
Realtime Emulation (continuous execution) -----	3-115
Listing (Redirect the Console output to Disk file) -----	3-121
Load Disk file program into Code Memory-----	3-122
Load Disk file Mask Option into memory-----	3-123
No Listing (Cancel the Console output Redirection) -----	3-124
Pause Command Input-----	3-125
Program EPROM-----	3-126
Program Mask Option Data into EPROM-----	3-128
Reset Break Point Bits -----	3-129
Reset Cycle Counter Trigger-----	3-130
Reset Instruction Executed Bits -----	3-131
Reset the System-----	3-132
Reset the Evaluation chip -----	3-133
Reset Trace Pointer-----	3-134
Reset Trace Enable Bits -----	3-135
Reset Trace Trigger -----	3-136
Save Code Memory into Disk file -----	3-138
Set Break Condition Register-----	3-139
Set Cycle Counter Trigger -----	3-141
Step Execution -----	3-143
Set Trace Trigger -----	3-146
Transfer EPROM into Code Memory-----	3-149
Transfer EPROM into System Memory-----	3-151
Set EPROM type-----	3-152
Set User Reset Terminal (on user connector)-----	3-153
Verify Disk file with Code Memory-----	3-154
Verify Disk file Mask Option into System memory -----	3-156

Verify EPROM with Code Memory ----- 3-157
 Verify Disk file with System Memory ----- 3-159

Chapter4. Debugging Notes----- 4-1

1. Debugging Notes ----- 4-2
 1.1 Ports----- 4-2
 1.2 LCD Drivers----- 4-5
 1.3 Stack Pointer -----4-10
 1.4 HALT Pin -----4-11
 1.5 XT and OSC1 Pins -----4-11
 1.6 ADC POD -----4-12
 1.7 DASM Command-----4-12
 1.8 Breaks -----4-13
 1.9 MSM64162D -----4-13
 1.10 Battery Check Circuit-----4-14
 1.11 Power Supply and Connections-----4-15
 1.12 A/D converter counter A registers -----4-18
 1.13 Warning messages (Mask Options) -----4-19

Appendix -----A-1

1. EASE64168 External Views ----- A-2
 2. User Cable Configuration ----- A-5
 3. Pin Layout of User Connector ----- A-6
 4. RS232C Cable Configuration ----- A-11
 5. Emulator RS232C Interface Circuit ----- A-12
 6. If EASE64168 Won't Start----- A-13
 7. Mounting EPROMs ----- A-15
 8. Error Messages ----- A-17
 9. Hardware Specifications ----- A-19
 10. Command Summary ----- A-20

Preface

1. Product Inquiries



Thank you for purchasing the EASE64168 Development Support System. Please direct any comments or questions that you may have about this product to your nearest Oki Electric Industry representative.

2. Using this Product Safely and Properly

2.1 Icons

This User's Guide uses various labels and icons that serve as your guides to operating this product safely and properly so as to prevent death, personal injury, and property damage. The following table lists these labels and their definitions.

Labels

 Warning	This label indicates precautions that, if ignored or otherwise not completely followed, could lead to death or serious personal injury.
 Caution	This label indicates precautions that, if ignored or otherwise not completely followed, could lead to personal injury or property damage.

Icons



A triangular icon draws your attention to the presence of a hazard. The illustration inside the triangular frame indicates the nature of the hazard—in this example, an electrical shock hazard.









A circular icon with a solid background illustrates an action to be performed. The illustration inside this circle indicates this action—in this example, unplugging the power cord.









A circular icon with a crossbar indicates a prohibition. The illustration inside this circle indicates the prohibited action—in this example, disassembly.

2.2 Important Safety Notes






Please read this page before using the product.

 Warning	
<p>Use only the specified voltage.</p> <p>Using the wrong voltage risks fire and electrical shock.</p>	
<p>At the first signs of smoke, an unusual smell, or other problems, unplug the emulator and disconnect all external power cords.</p> <p>Continued use risks fire and electrical shock.</p>	
<p>Do not use the product in an environment exposing it to moisture or high humidity.</p> <p>Such exposure risks fire and electrical shock.</p>	
<p>Do not pile objects on top of the product.</p> <p>Such pressure risks fire and electrical shock.</p>	
<p>At the first signs of breakdown, immediately stop using the product, unplug the emulator, and disconnect all external power cords.</p> <p>Continued use risks fire and electrical shock.</p>	

Please read this page before using the product.

 Caution	
<p>Do not use this product on an unstable or inclined base as it can fall or overturn, producing injury.</p>	
<p>Do not use this product in an environment exposing it to excessive vibration, strong magnetic fields, or corrosive gases.</p> <p>Such factors can loosen or even disconnect cable connectors, producing a breakdown.</p>	
<p>Do not use this product in an environment exposing it to temperatures outside the specified range, direct sunlight, or excessive dust.</p> <p>Such factors risk fire and breakdown.</p>	
<p>Use only the cables and other accessories provided.</p> <p>Using non-compatible parts risks fire and breakdown.</p>	
<p>Do not use the cables and other accessories provided with other systems.</p> <p>Such improper usage risks fire.</p>	

Please read this page before using the product.

 Caution	
<p>Do not exceed the rated input voltage for the user cable VDD pins.</p> <p>Doing so risks fire and breakdown.</p>	
<p>Always observe the specified order for turning equipment on and off.</p> <p>Using the incorrect order risks fire and breakdown.</p>	
<p>Always cut the power to the emulator before altering connections.</p> <p>Connection or disconnection with the power on risks fire and breakdown.</p>	
<p>Always cut the power to the emulator and the user application system before altering connections between the two.</p> <p>Connection or disconnection with the power on risks fire and breakdown.</p>	

3. Notation

This User's Guide uses the following notational conventions.

Type	Notation	Meaning
Numerals	xxh, xxH	Hexadecimal number
	xxb	Binary number
Units	W (word)	1 word = 2 bytes = 4 nibbles = 16 bits
	B (byte)	1 byte = 2 nibbles = 8 bits
	N (nibble)	1 nibble = 4 bits
	M (mega-)	10^6
	K (kilo-)	1024 — only in KB (kilobytes) and KW (kilowords)
	k (kilo-)	$10^3 = 1000$
	m (milli-)	10^{-3}
	μ (micro-)	10^{-6}
n (nano-)	10^{-9}	
	s	second(s)
Terms	"H" level	High signal level — that is, the VDD voltage level.
	"L" level	Low signal level — that is, the VSS voltage level.
Cross References	<p>■ Reference ■</p> <p>(See Note n)</p>	<p>This notation gives a cross-reference to related material elsewhere in this manual.</p> <p>This notation refers the reader to a numbered note providing supplementary information later in the same Section.</p>
	<p>■ Note n ■</p>	<p>This notation introduces a numbered note providing supplementary information.</p>

4. Manual Organization

This manual consists of the following four chapters.

Chapter 1. Before Starting

This chapter describes procedures to follow after receiving delivery of an EASE64168.

Chapter 2. Overview

This chapter introduces the emulator and its parts.

Chapter 3. EASE64168 Emulator

This chapter describes the functions of the emulator.

Chapter 4. Debugging Notes

This chapter contains important usage notes. Be sure to read it before using the emulator.

Appendices

5. Package Contents

5.1 Verify Shipping Contents

When you receive your EASE64168 development support system, check the package contents against the EASE64168 packing list.

Oki Electric has every confidence that the contents are both complete and undamaged. Should a component be damaged or missing, however, please contact your nearest Oki Electric representative.

Chapter 1. Before Starting

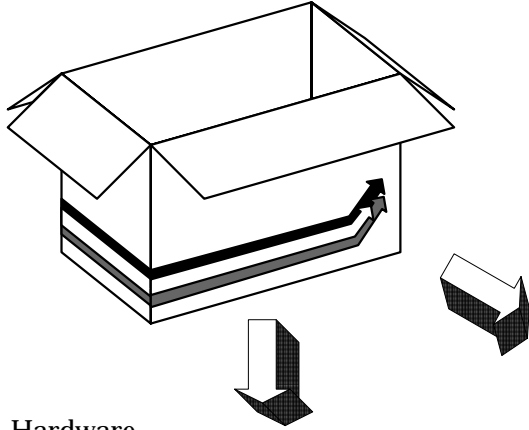
This chapter describes procedures to follow after receiving delivery of an EASE64168 program development support system. It is recommended that this chapter be read before supplying power to the emulator.

Thank you for buying Oki Electric's EASE64168 program development support system. When your system was shipped we made every effort to ensure that it would not be damaged or mispacked, but we recommend that you confirm once more that this did not occur following the explanations in this chapter.

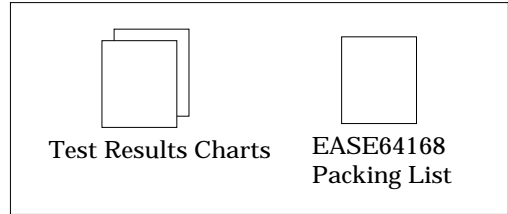
The RS232C cable, floppy disks, or other items may differ depending on the model of host computer that you will use. Use with a different model could cause damage to the hardware, so please take particular care to avoid this. If the system shipped to you was damaged, if any components were missing, or if your host computer model is different, then please contact the dealer from whom you purchased the system or Oki Electric's sales department.

1. Confirm Shipping Contents

EASE64168 Contents



Documentation



Software

3 Floppy Disks

SASM64K



EASE64x

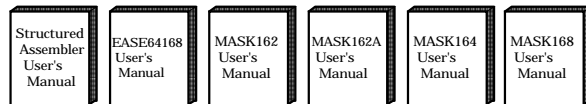


MASK162/
162A/164/168

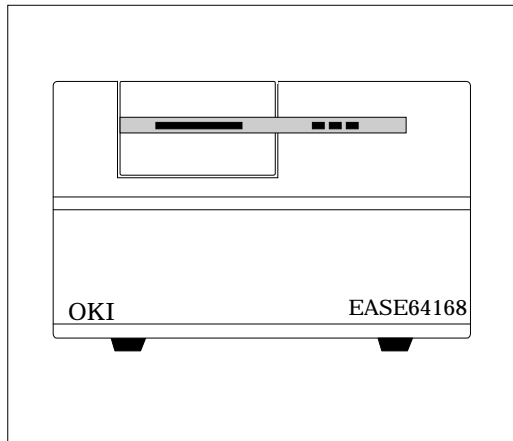


6 Manuals

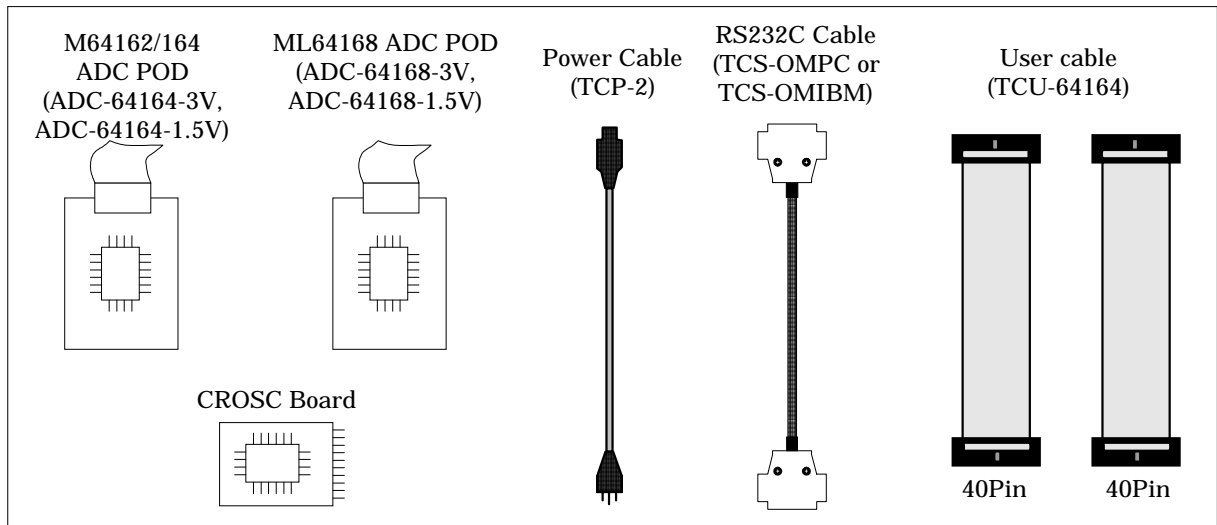
- Structured Assembler User's Manual
- EASE64168 User's Manual
- MASK162 User's Manual
- MASK162A User's Manual
- MASK164 User's Manual
- MASK168 User's Manual



Hardware

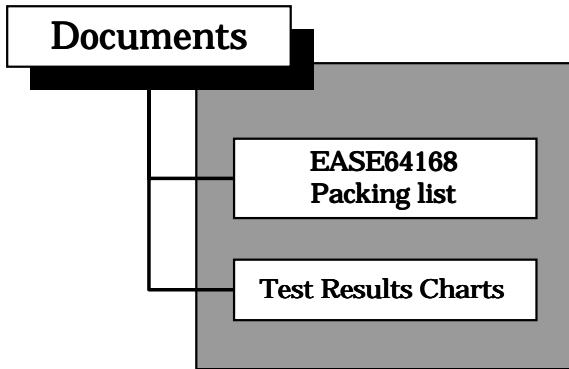


Accessories



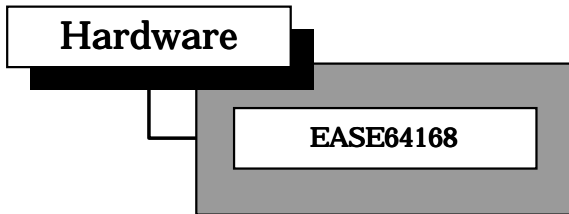
Your purchase of the EASE64168 will be followed by delivery of the necessary hardware, software, and manuals in the shipping box illustrated in the upper left of page 1-3. After taking delivery, open the box and confirm that it contains all the contents illustrated on page 1-3.

Each component is described below. Note that those marked with (note1) will differ depending on the model of host computer.

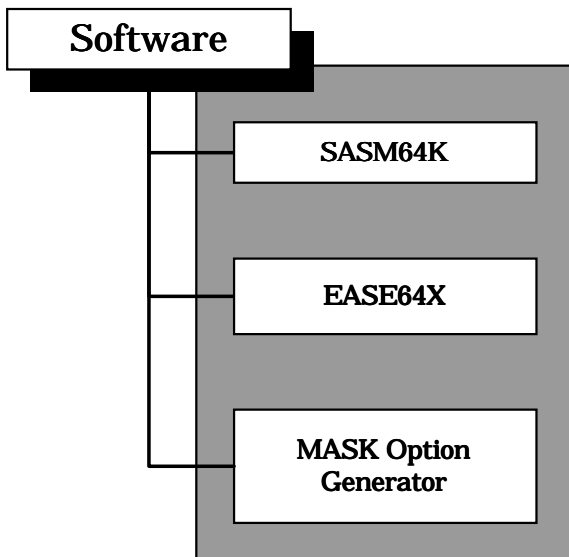


This is a list of the items shipped.

This chart shows that the EASE64168 passed all tests before shipping.



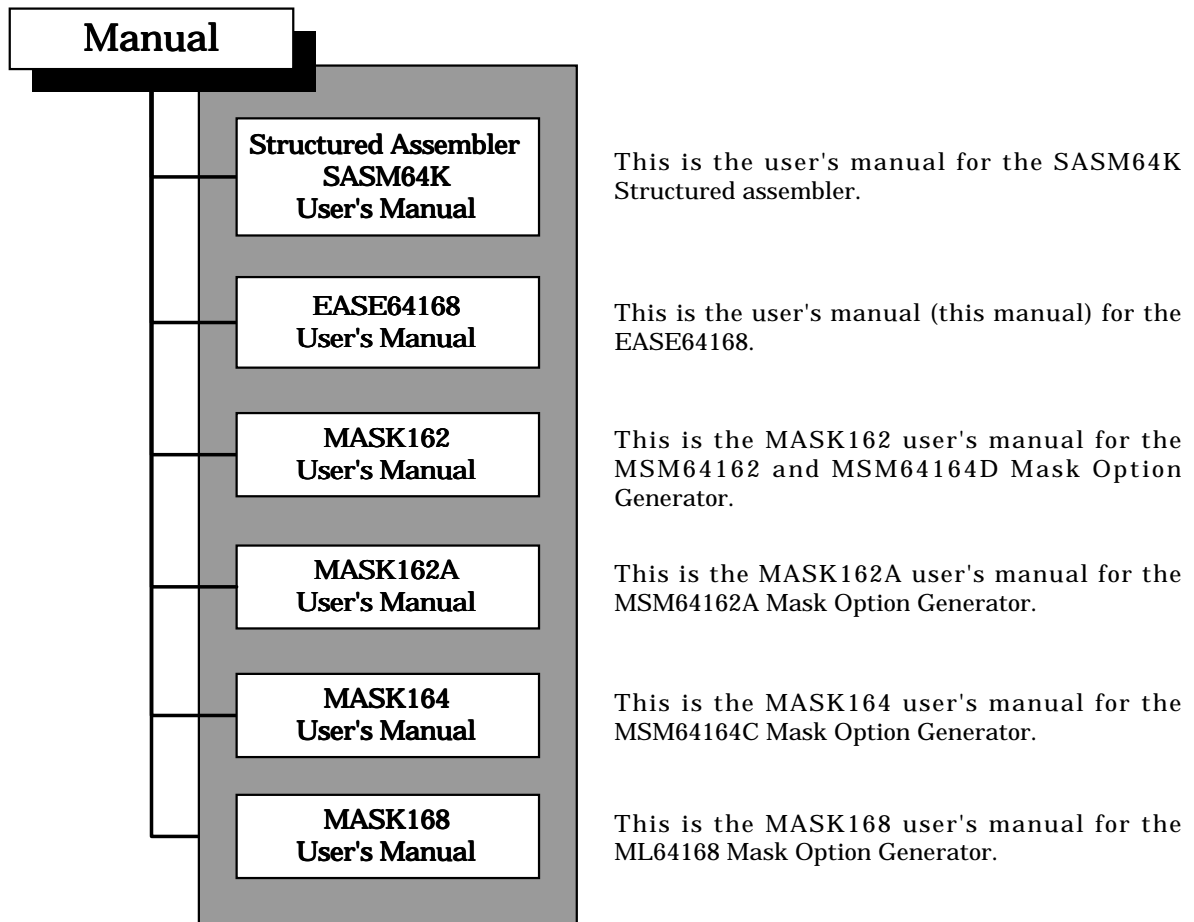
This is the EASE64168 emulation kit. It contains hardware for host computer communications, EPROM programming, and emulation of M6416x series microcontrollers operation.

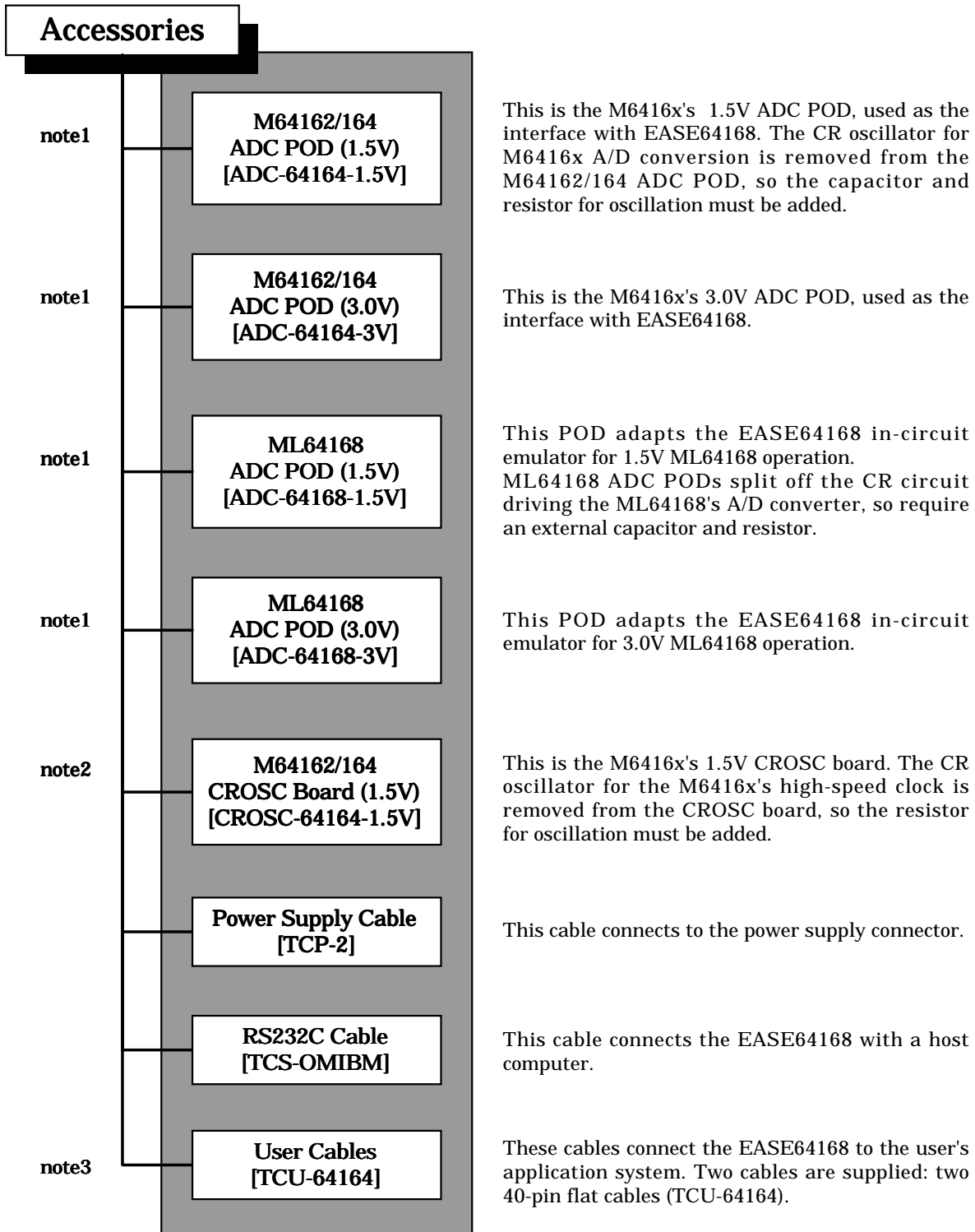


This disk contains the SASM64K executable files.

This disk contains the executable files for EASE64X.

This disk contains the executable files for MASK162, MASK162A, MASK164 and MASK168.





■ Note1 ■

The analog-to-digital converter CR circuit's oscillation characteristics differ between the ML64162/162A/162D/164C and the ML64168. They also differ between target chip operating voltages (1.5 and 3.0V). The EASE64168 in-circuit emulator therefore covers this differences with four ADC PODs: ADC-64164-3V, ADC-64164-1.5V, ADC-64168-3V, and ADC-64168-1.5V. Choose the one appropriate for the target chip and operating voltage. The ADC POD model number appears on the label on the board.

■ Note2 ■

The oscillation characteristics of the CR oscillator for the high-speed clock differ for the 1.5V and 3.0V versions of M6416x. Select the CROSC board that matches your version. A label on the board identifies the CROSC board as 1.5V or 3.0V. The 3.0V CROSC board is mounted in the EASE64168 when it is shipped.

■ Note3 ■

The user cables are used for the following applications.

TCU-64164 (40-pin flat cable)

The cable connects the user application system with the EASE64168's USER connector. The voltage level of the USER connector's interface power supply is set by the CIPS command to an internal voltage (5V) or an external voltage (3V~5V).

■ Reference ■

CIPS command

TCU-64164 (40-pin flat cable)

The cable connects the user application system with the EASE64168 LCD connector or LED connector.

The LCD and LED connectors correspond to pins L0~L33 of the M6416x. LCD drive signals (0V~4.5V) are output from the LCD connector. LED drive signals (0V~5V) are output from the LED connector.

2. Confirm Floppy Disk Contents

2.1 EASE64X Debugger Operating Environment

The environment for running the EASE64X debugger is as follows.

Host computer

IBM PC/AT and compatibles

Operating system

MS-DOS Ver.3.3 or later

Main memory

At least 640K bytes

2.2 Operating System

The operating system of computers should be MS-DOS version 3.3 or later.

Chapter2. Overview

This chapter provides an overview of EASE64168 system configuration, describes the program development procedure with the EASE64168 system.

1. EASE64168 Emulator Configuration

1.1 EASE64168 Emulation Kit

The EASE64168 is a general-purpose control system for in-circuit emulators for Oki Electric's M6416x CMOS 4-bit microcontrollers.

The internal configuration of the EASE64168 is as follows.

* System controller	MC68000
(note1) * Code memory	8192 × 8 bits
(note2) * Data memory	512 × 4 bits
(note1) * Trace memory	8192 steps × 64 bits
(note1) * Attribute memory	8192 × 8 bits
(note1) * Instruction executed bit memory	8192 × 1 bit
* EPROM programmer	For 2764/128/256/512
* RS232C ports	1 channel
(note3) * Evaluation board	For MSM64162/162A/162D/164C, ML64168
* System power supplies	1

■ Note1 ■

The maximum address of code memory, attribute memory, and instruction executed memory is depending on the microcontroller. But the maximum address can be extended up to 3FFFH with the EXPAND command.

■ Note2 ■

Data memory size is depending on the microcontroller.

■ Note3 ■

The evaluation board emulates the functions of the MSM64162/162A/162D/164C and ML64168. It is internal to the EASE64168.

The evaluation board consists of an MSM64E900 evaluation chip with an nX-4/20,30 core that matches the MSM64162/162A/162D/164C and ML64168 CPU core, hardware that matches the I/O portion of the MSM64162/162A/162D/164C and ML64168 (excluding the CR oscillator for A/D conversion), and hardware that matches the MSM64162/162A/162D/164C and ML64168's LCD drivers.

The I/O hardware is constructed from ordinary discrete components, so the electrical characteristics of the ports will differ from those of the MSM64162/162A/162D/164C and ML64168.

The emulator uses special hardware to allocate the mask option registers of the LCD drivers, so display timing will differ from the MSM64162/162A/162D/164C and ML64168.

The CR oscillator for the MSM64162/162A/162D/164C and ML64168 A/D converter is added to the optional ADC PODs. The MSM64164 and ML64168 are mounted in the M64162/164 ADC POD and ML64168 ADC POD, so CR oscillation will be with the same electrical characteristics as the MSM64164 and ML64168. The CR oscillation clock is input to the EASE64168 and performs A/D conversions.

1.2 SASM64K Structured Assembler

SASM64K is a Structured assembler developed for the OLMS-64K series. It is stored on a floppy disk that comes with the purchase of an EASE64168 development support system.

Source files constructed from OLMS-64K series instruction mnemonics and directives are converted to Intel HEX formal object files with SASM64K. Object files (machine language files) generated this way are read and executed by EASE64X, explained in the next section.

SASM64K can be used with host computers that satisfy the following conditions.

- * The operating system is MS-DOS version 3.3 or higher.
- * There is a free area of at least 128K bytes in main memory.

For details about SASM64K, refer to the Structured Assembler SASM64K User's Manual.

■ Note1 ■

The DCL file for SASM64K includes the following definitions for M6416x series microcontrollers.

- a. SFR (Special Function Register) address and access attributes
 - b. Code memory (program memory) address range
 - c. Data memory address range
-

The following DCL files are provided for the M6416x series microcontrollers. (The floppy disk contains DCL files for all the devices supported by SASM64K.)

M64162.DCL : For MSM64162 and MSM64162A
M64162D.DCL : For MSM64162D
M64164.DCL : For MSM64164C
M64168.DCL : For ML64168

1.3 EASE64X Debugger

The EASE64X debugger is software that supports debugging.

EASE64X is stored on a floppy disk that comes with the purchase of an EASE64168 development support system.

EASE64X can be used with host computers that satisfy the following conditions.

- * The operating system is MS-DOS version 3.3 or higher.
- * There is a free area of at least 100K bytes in main memory.
- * A channel for an RS232C interface.

1.4 Mask option generator (MASK162/162A/164/168)

The mask option generator (MASK162/162A/164/168) allow an operator to input the MSM64162/162A/162D/164C and ML64168 mask option settings shown below, and convert the input data to mask option files in Intel HEX format.

- LCD driver duty value
- Assignment of segment pins (L0~L33) to ports, commons, and segments
- Assignment of segment pins to display registers
- Operating power supply voltage
- Presence of capacitor for Crystal oscillator

The mask option files generated by the mask option generator (MASK162/162A/164/168) are used to create the mask needed to manufacture the MSM64162/162A/162D/164C and ML64168.

The EASE64X debugger reads the mask option files so the EASE64168 can determine the above settings (except for operating power supply voltage and presence of capacitor for Crystal oscillator).

1.5 System Configuration

The system is used the following configuration.

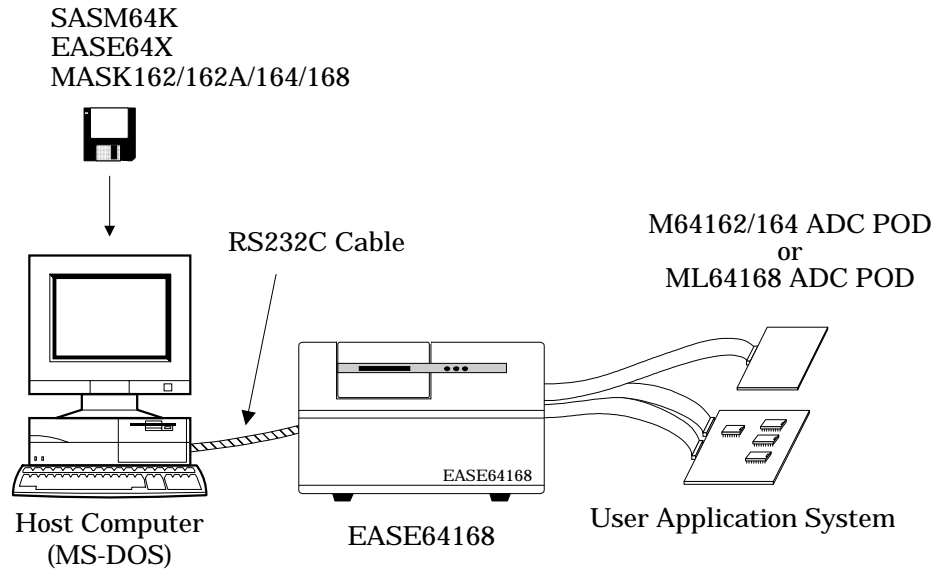


Figure 2-1 System Configuration Diagram

2. Program Development With EASE64168

2.1 General Program Development and EASE64168

Figure 2-2 shows the general flow of program development (note1).

First, one decides on the functions of the product to be developed, and evaluates which hardware and software should be designed to implement them. Specific considerations include which MPU to use, how to allocate MPU interrupts, how much ROM and RAM to add, etc. This is called the functional design process.

Next is the specification design process. Here the functions to be implemented are evaluated in detail, and the methods to use those functions in the final product are decided. Specifically, commands are decided upon and a command input specification is written. The specification generated by this process is usually called the functional specification.

The process of creating a program based on the functional specification is called the program design process. Algorithms, flowcharts, and a program specification are created. This process can also include coding (source program creation) and assembly. In other words, SASM64K is used in this process. This process also includes the creation of a mask option file with the mask option generator (MASK162/162A/164/168).

Next is the debug process. This is the process in which the EASE64168 especially excels (note2). An object file and a mask option file created in the program design process is downloaded to the EASE64168, and by using the various functions of the EASE64168 emulator, program bug analysis, fixing, and testing are performed.

The last position of the overall program development process is occupied by the testing process. The complete program from the debug process is operated in the actual product, and operation according to the functional specification is verified with test programs, etc. If there are bugs in the operation, then the flow from the program design process on is repeated until there are no more bugs.

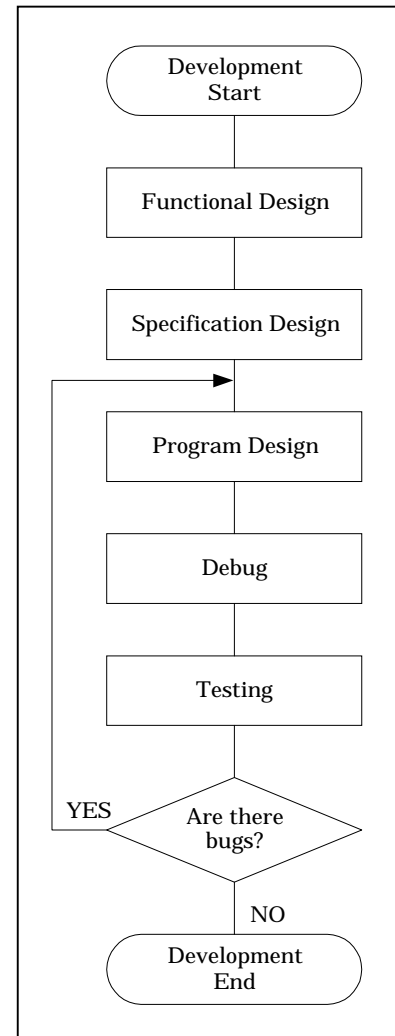


Figure 2-2 General Flow of Program Development

■ Note1 ■

The general flow and terminology given here are generally used, but other documents and manuals may have different expressions.

■ Note2 ■

Refer to Chapter3, "EASE64168 Emulator," for details about the various function of the EASE64168 emulator.

2.2 From Source File To Object File

In order to perform debugging with the EASE64168 emulator, an object file for downloaded to the EASE64168 must be generated (note1, 2).

Figure 2-3 shows the process of generating an object file from a source program file coded in assembly language (hereafter called a source file).

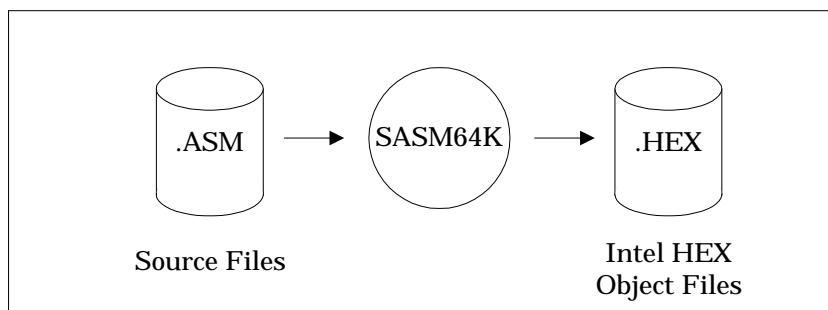


Figure 2-3 Process of Generating Object Files From Source Files

In the above figure, circles indicate operation of the SASM64K Structured assembler program, while cylinders indicate files generated by programs.

Object files that the EASE64168 emulator can handle are Intel HEX format object files that include symbol information, as shown in Figure 2-3.

■ Note1 ■

Downloading means storing the contents of an object file in EASE64168 code memory with the EASE64X LOD command.

■ Note2 ■

Object files in this document refer to Intel HEX format object files that not include symbol information which the EASE64168 emulator can handle.

2.3 Generating Mask Option Files

To perform debugging with the EASE64168 emulator, the mask option files must be created in addition to the object file described in the previous sections.

Figure 2-4 shows the process for generating mask option files.

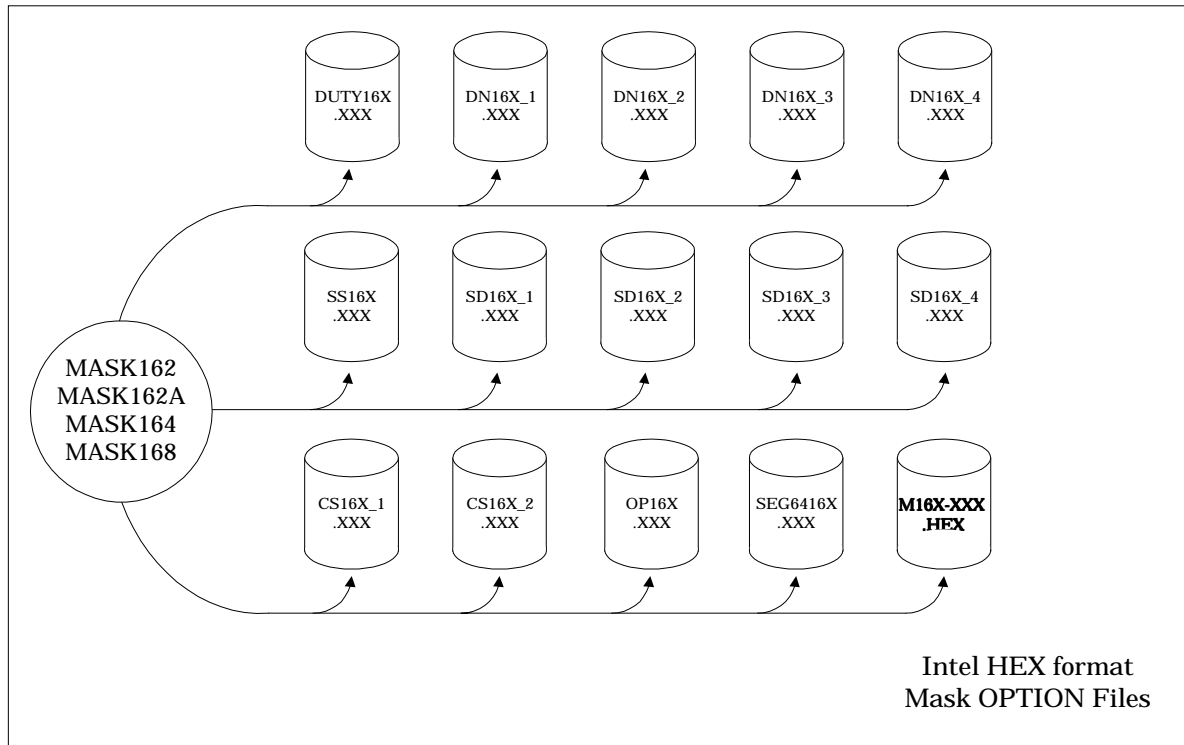


Figure 2-4 Process For Generating Mask Option Files

In the above figure, circles indicate operation of the mask option generator (MASK162/162A/164/168) programs, while cylinders indicate files generated by the programs.

Based on mask option settings input by the operator, mask option generator (MASK162/162A/164/ 168) outputs the fifteen files shown in Figure 2-4 The EASE64168 emulator can handle mask option files in Intel HEX format.

2.4 Files Usable with the EASE64168 Emulator

The files usable with the EASE64168 emulator are described in the sections about files created by SASM64K and the mask option generator (MASK162/162A/164/168). As described in those sections, there are two types of files that can be handled by the EASE64168 emulator. These are explained.

(1) Intel HEX files generated by SASM64K

These are object files generated by SASM64K from source files that consist of OLMS-64K mnemonics and various directives. Object files are read into code memory using the LOD command.

(2) Intel HEX files generated by the mask option generator (MASK162/162A/164/168)

These are mask option files generated by the mask option generator (MASK162/162A/164/168) from MSM64162/162A/162D/164C and ML64168 mask option settings. Mask option files are read into the system memory of the EASE64168's MC68000 system controller using the LODM command.

■ **Note1** ■

If the "/S" option is added when the SASM64K assembler is executed, then the generated object file will include symbol information. However, EASE64168 cannot handle object files that include symbol information.

Chapter3. EASE64168 Emulator

This chapter explains the actual use of the EASE64168 emulation kit and the EASE64X debugger in detail.

In this chapter ...

Section 1 gives an overview of each group of functions that can be used with the EASE64168 emulation and the EASE64X debugger

Section 2 explains how to start the EASE64168. EASE64168 dipswitch settings (to set the communications mode with the host computer, etc.) are also explained in this section.

Section 3 explains in detail the actual use of EASE64X debugger commands with the EASE64168.

Section 3.1 describes the general input format of debugger commands and lists all debugger commands by function. This list also gives a reference page for each command, so it is convenient for use as a command index.

Section 3.3 and 3.4 explain the history function and special-purpose keys respectively. These are provided to support efficient input of debugger commands.

Section 4 explains each debugger command in detail.

1. EASE64168 Functions

1.1 Overview

Chapter 2. Section 2. explained the program development process with the M6416x series microcontrollers. This section gives an overview of the actual emulator functions used to debug prototype programs created by that process.

The most basic function of the emulator is to read and execute a user-created program (an Intel HEX format file generated by SASM64K). Here "execute" means to execute a program at the same speed (realtime) as the volume-production M6416x series microcontrollers with internal mask ROM. This is known as emulation, as distinguished from program simulation with large computers.

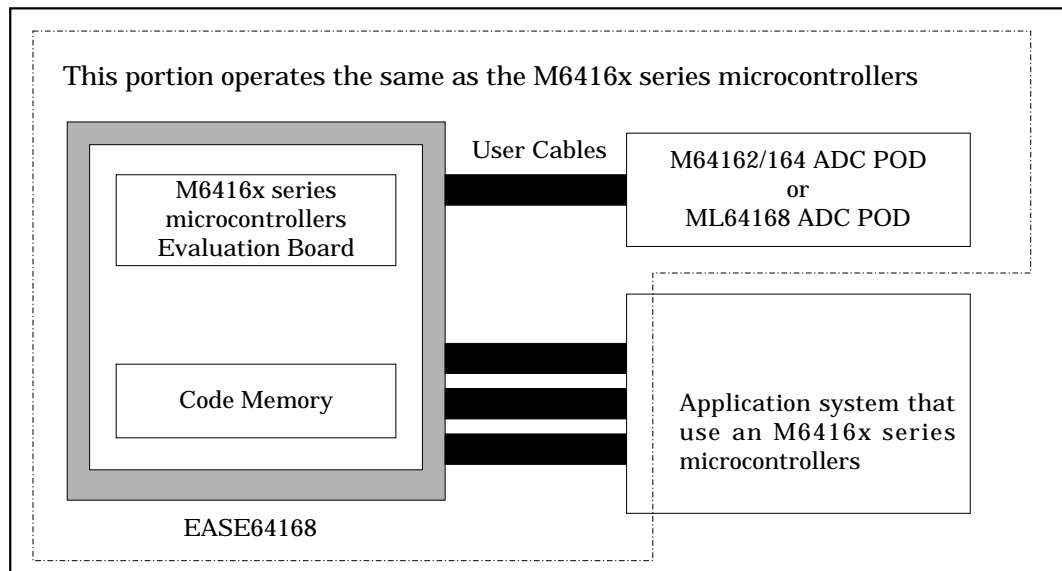


Figure 3-1 System configuration

The volume-production M6416x series microcontrollers has mask ROM on-chip, but once mask ROM has been written it cannot be changed. However, program during the development stage is difficult to debug unless it is stored in rewritable memory (RAM).

Thus the EASE64168 has in internal 16K bytes program storage RAM. This RAM is called code memory. Refer to Figure 3-1 on the previous page.

EASE64168 executes programs in this code memory instead of mask ROM. When the user application system is being produced in volume, it will be mounted with an M6416x series microcontrollers, but at the debug stage it is replaced with a connector in the user application system. This connector is attached to an EASE64168 user cable (Refer to Figure 3-1).

Within the EASE64168 is an internal evaluation board for emulating the functions of the M6416x series

microcontrollers. This evaluation board has the same CPU circuit and the same external pins as the M6416x series microcontrollers. However, the CR oscillator for the A/D converter is implemented in the M64162/164 ADC POD, ML64168 ADC POD (note1).

The main feature of the evaluation board is that it has no internal mask ROM, but it does have some special control circuitry and control pins. These additional circuits and pins are used to control execution of programs and reading of internal memory, registers, and flags.

Also, the contents of code memory instead of mask ROM are read and executed.

The evaluation board's external pins include the same pins as the volume-production chip. These are connected to the corresponding pins in the user application system through the user cables and pins for the CR oscillator of the A/D converter are provided on the M64162/164 ADC POD, ML64168 ADC POD.

As a result, when viewed from the user application system, the pins on the user cable and M64162/164 ADC POD, ML64168 ADC POD appear identical to the M6416x series microcontrollers (Refer to Figure 3-1).

■ Note1 ■

Because the evaluation board uses regular discrete components for the CPU circuits, pin electrical characteristics differ between the MSM64162/162A/162D/164C and the ML64168. There are therefore separate PODs for the CR oscillator circuit driving the analog-to-digital converter. The M64162/164 ADC PODs contain an MSM64164; the ML64168 ADC PODs, an ML64168. The PODs thus provide CR oscillations with the same electrical characteristics as the MSM64164 and ML64168, respectively. They supply the EASE64168 in-circuit emulator with the CR oscillation clock for the analog-to-digital converter.

That the basic function of the emulator is to read and execute programs was already explained, but effective debugging is not possible with just simple execution. For example, one must be able to start and stop program execution at specified addresses. One needs to display and change the states of data memory (internal RAM), registers, and flags after execution. Furthermore, instead of just stopping execution at a specified address, one needs the ability to set complex conditions for stopping after a specified time has elapsed or some address has been passed a specified number of times (pass count). To meet these needs, EASE64168 has many functions beyond its basic one. These features are explained one by one in the following sections.

1.2 Changing Target Chips

The EASE64168 is an emulation kit designed for the M6416x series microcontrollers. It operates in ML64168 mode by default when started, but the target chip can be changed with the CHIP command (note1).

Setting chip mode with the CHIP command

One of the EASE64X debugger commands, the CHIP command, changes the target chip. The EASE64168 chip mode can be changed with this command.

■ Reference ■

CHIP command

The chip modes specified by the CHIP command set the EASE64168 as follows (note2).

Item	MSM64162 Mode	MSM64164C Mode	ML64168 Mode
Code memory addresses	000~7DFH	000~0FDFH	000~1FDFH
Attribute memory addresses	000~7DFH	000~0FDFH	000~1FDFH
Instruction executed memory addresses	000~7DFH	000~0FDFH	000~1FDFH
Data memory	128 x 4 bits	256 x 4 bits	512 x 4 bits
Port 4 data register (P4D)	Invalid	Valid	Valid
Port 40 control register (P40CON)	Invalid	Valid	Valid
Port 41 control register (P41CON)	Invalid	Valid	Valid
Port 42 control register (P42CON)	Invalid	Valid	Valid
Port 43 control register (P43CON)	Invalid	Valid	Valid
Serial port control register (SCON)	Invalid	Valid	Valid
Serial port buffer register (SBUF)	Invalid	Valid	Valid
Backup control register (BUPCON)	Bits 0~2 valid	Bit 0 valid	Bits 0~2 valid
Buzzer frequency control register (BFCON)	Bit 0 valid	Bits 0~3 valid	Bits 0~3 valid
Interrupt enable register 0 (IE0)	Bits 0, 2, 3 valid	Bits 0~3 valid	Bits 0~3 valid
Interrupt request register 0 (IRQ0)	Bits 0, 2, 3 valid	Bits 0~3 valid	Bits 0~3 valid
Time base counter interrupts	1Hz, 4Hz, 16Hz, 32Hz, 256Hz	0.1Hz, 1Hz, 16Hz, 32Hz, 256Hz	0.1Hz, 1Hz, 16Hz, 32Hz, 256Hz
LCD drivers	L0~L23	L0~L33	L0~L33

■ Note1 ■

When evaluating an MSM64162D and MSM64162A with EASE64168, set the chip mode to MSM64162 mode. However, do not use functions that do not exist in the MSM64162D (high-speed clock, A/D converter CROSC1 oscillation mode, IN1 external clock input mode).

■ **Note2** ■

Refer to user's manual of your chip for details about each register.

1.3 Emulation Functions

The EASE64168 has two modes for its emulation functions (program execution functions).

(1) Single-step mode (STP command)

In this mode, program execution stops after each step (one instruction) is executed. After each instruction is executed, the state of the evaluation chip is read and displayed on the CRT. Single-step mode is realized with the STP command.

■ **Reference** ■

STP command

(2) Realtime emulation mode (G command)

In this mode, program execution will continue until some specified break condition is satisfied or an ESC key is input. Realtime emulation mode is realized with the G command.

■ **Reference** ■

G command

Operating Clock

The EASE64168 operates using a clock supplied from an internal oscillation circuit. Its operating frequency is set to 32.768kHz in low-speed mode and 700kHz in high-speed mode. To use other frequencies, replace the crystal on the crystal board or the resistor for CR oscillation on the CROSC board in the emulator.

For details, refer to Section 2.1, "Setting Operating Frequency."

■ **Note1** ■

The allowable operating frequencies for the EASE64168 are 32.768 kHz~800 kHz.
Depending on the manufacturer and frequency of the crystal, the capacitors and resistor for oscillation may also need to be changed.

■ Note2 ■

When the ML64168 mode, the frequency of the internal high-speed clock (700kHz) can not be changed.

If you want to change the frequency of the high-speed clock, input a clock on the user cable OSC1 pin and select the external clock by the CCLK command.

1.4 Realtime Trace Functions

One EASE64168's principal functions is realtime tracing. Realtime tracing occurs during program execution under realtime emulation mode. It stores the executed addresses, the data and addresses in data memory used, and the states of evaluation chip port pins, registers, and flags in memory provided for tracing. The memory provided for tracing is called trace memory.

The EASE64168 has trace memory for 8192 steps. It traces the following items.

Trace Contents
Executed address
State of all ports
A register and B register values
Data memory contents at specified address (note1)
Stack pointer (SP) value
H register and L register values
Carry flag (C) value
Port 2 value
Port 3 value
Port 4 value (note2)
Port 0 value (note2)
Port 1 value (note2)
Bank select register 0 (BSR0) value (note2)
Bank select register 1 (BSR1) value (note2)

■ Note1 ■

The data memory contents at the one address specified by the CTDM command will be traced.

■ Note2 ■

Tracing of port 4, port 0, port 1, bank select register 0, and bank select register 1 is selected by the CTO command.

■ Reference ■

FTR, ETR, RTR, DTR, STT, RTT, DTT, DTM, CTO, DTO, CTDM, DTDM commands

Controlling trace execution

Realtime tracing can always be performed during program execution, but you may want to see trace contents for just a particular part of a program. EASE64168 provides two ways to specify the trace area.

- (1) Specify trace area with trace enable bits.
- (2) Specify a triggers with trace start/stop bits.

Details of each method are explained in the command details section 3. The following are related commands.

■ Reference ■

DTR, ETR, RTR, FTR, STT, RTT, DTT commands

The trace pointer controls the address in trace memory to which data will be written. The trace pointer is actually a 13-bit counter which increments every time an instruction is executed under the conditions described in (1) and (2) above (refer to Figure 3-2).

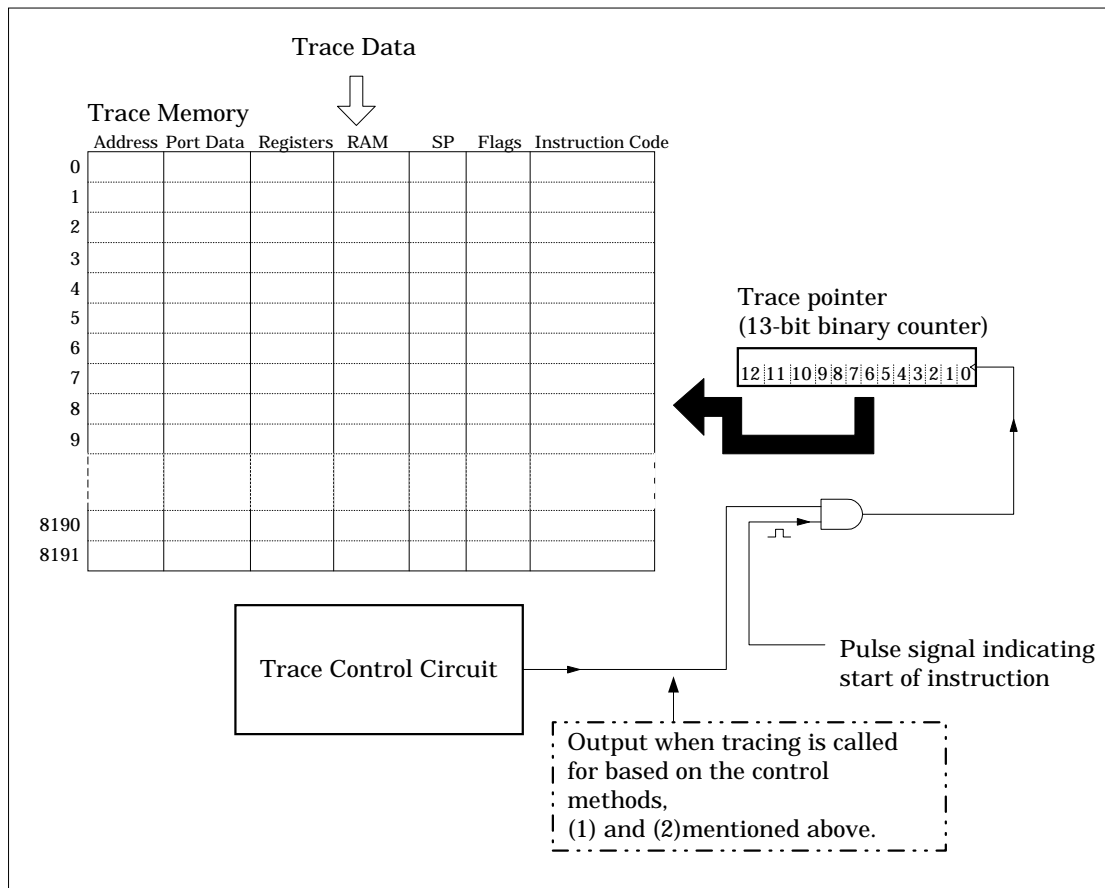


Figure 3-2 Trace Control Conceptual Diagram

The trace pointer's value indicates the address in trace memory to which data will be written. The trace pointer is incremented at the start of each instruction while the conditions of the previously described control methods are satisfied. As a result, while trace conditions are satisfied, the trace memory addresses written are updated one by one as trace data is stored at each.

The trace pointer is a 13-bit counter, so its value will be between 0 and 1FFFH (in decimal, 0 and 8191). When the trace pointer exceeds 1FFFH and the next trace data arrives, the trace pointer overflows and becomes 0. In other words, when traced data exceeds 8192 steps, it will be overwritten in order from the oldest data in trace memory.

1.5 Break Functions

The following methods for breaking program execution are available with the EASE64168.

(a) Breakpoint bit breaks

The EASE64168 has a 1-bit wide memory that corresponds 1-for-1 with the entire program memory address space (0-3FFFH). This memory is called breakpoint bits memory or breakpoint bits.

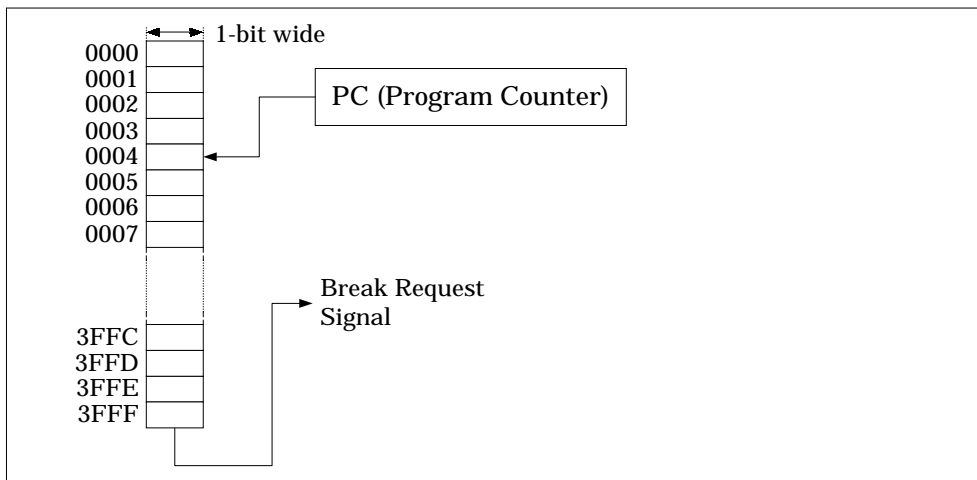


Figure 3-3 Breakpoint Bits Conceptual Diagram

Breakpoint bits can be set to 1 or 0 with the FBP (Fill BreakPoint) command, EBP (Enable BreakPoint) command, and RBP (Reset BreakPoint) command. During emulation execution, the breakpoint bit corresponding to each executed address is referenced, and if “1,” a break request signal is output (refer to Figure 3-3).

By using breakpoint bits, breakpoints can be set throughout the entire address space without a limit to their number. (In this manual breaks generated by breakpoint bits are called breakpoint bit breaks to clearly distinguish them from address breaks, which are generated by break addresses specified as break parameters of the G command.)

■ **Reference** ■

DBP, FBP, EBP, RBP, SBC, DBC commands

(b) Trace pointer overflow breaks

The EASE64168 can cause a break using overflow of the trace pointer. The trace pointer is a 13-bit counter that represents a location in trace memory. When the trace pointer exceeds 1FFFH (8192 steps), it overflows. The overflow of the trace pointer can be used as a break condition.

■ **Reference** ■

DTR, FTR, ETR, RTR, STT, RTT, DTT, SBC, DBC commands

(c) Cycle counter overflow breaks

The EASE64168 has a 32-bit counter that increments every step (called the cycle counter). The overflow of the cycle counter can be used as a break condition.

■ Reference ■

SCT, RCT, DCT, CCC, DCC, SBC, DBC commands

(d) ESC key breaks

Press the host computer's ESC key to forcibly stop G command execution (realtime emulation).

(e) Breaks specified during G command input

- * Break at specified address (with pass count)
- * Break when specified data matches data at a specified address in data memory
- * Break when specified data matches data in A register or B register

■ Reference ■

G, CTDM, DTDM commands

(f) N area access breaks

The EASE64168 will forcibly break when it accesses an area that exceeds the maximum address for its respective chip modes (note1). However, N area access breaks will not occur when code memory is expanded (EXPAND ON mode).

■ Note1 ■

The maximum address of the program memory area differs for each chip mode. In MSM64162 mode it is 7DFH, and in MSM64164C mode it is FDFH, and in ML64168 mode it is 1FDFH.

Break request mask function

The break conditions explained in (a)-(c) above can be masked. As shown in Figure 3-4, each break condition can be selectively and independently masked using a register called the break condition register.

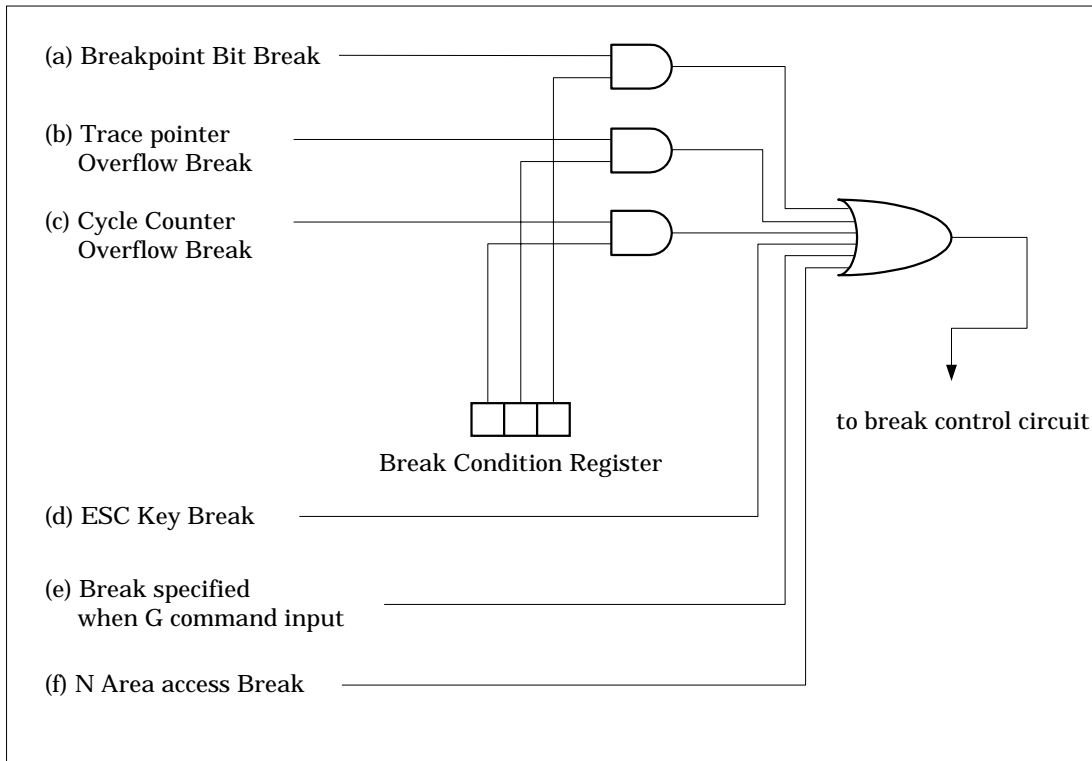


Figure 3-4 Break Masking

■ Note2 ■

The order of bits in the break condition register of Figure 3-4 does not necessarily match the order of bits in the actual register. The purpose of this figure is to show how break conditions are masked, so the break conditions are listed in their order of appearance in the previous section.

1.6 Performance/Coverage Functions

The EASE64168 has the following performance/coverage functions.

- (a) Check for program areas not yet passed

The EASE64168 has a 16384 x 1-bit instruction executed bits memory (or IE bit memory) that corresponds 1-for-1 to code memory's entire address (0H-3FFFH). Whenever an instruction is executed, the contents of IE bit memory at the address corresponding to the instruction will be set to "1." By examining the contents of IE bit memory, one can see which program areas have not been passed (or debugged).

■ Reference ■

RIE, DIE commands

(b) Measuring elapsed time

Elapsed execution time for a specified block can be measured by using the EASE64168 internal 32-bit cycle counter (CC).

■ Reference ■

CCC, DCC, SCT, RCT, DCT commands

1.7 EPROM Programmer

The EASE64168 has an internal EPROM programmer (EPROM writer). By using the EPROM programmer, EPROM contents can be transferred to code memory, and contents of a code memory area can be written to EPROM.

In addition, the EPROM programmer can be used to read mask option data (note1).

The types of EPROM that the EPROM programmer can write are as follows:

2764, 27128, 27256, 27512, 27C64, 27C128, 27C256, 27C512

■ Reference ■

TPR, VPR, PPR, TYPE, TPRM, VPRM commands

Refer to Appendix 7, "Mounting EPROMs," for information about how to mount EPROMs.

■ Note1 ■

DO NOT USE THE EPROM PROGRAMMER FOR PURPOSES OTHER THAN DEBUGGING PROGRAMS. IF RELIABILITY IN WRITE CHARACTERISTICS IS NECESSARY, THEN USE AN EPROM PROGRAMMER DESIGNED FOR THAT PURPOSE.

1.8 Indicators

POWER indicator (red)

This indicator will light after EASE64168 power is turned on and correct operation begins.

RUN indicator (green)

This indicator will dark after EASE64168 power is turned on and correct operation begins. It will also light during while emulation is executing and while EPROM programmer commands are executing.

■ Reference ■

TPR, VPR, PPR, TPRM, VPRM, G, STP commands

PORT5V indicator (green)

This indicator will light when the user connector interface power supply is being supplied from the emulator's internal power supply.

PORT3V indicator (green)

This indicator will light when the user connector interface power supply is being supplied from an external power source (+3V ~ +5V).

■ Reference ■

CIPS command

2. EASE64168 Emulator Initialization

2.1 Setting Operating Frequency

As explained in Section 1.3, the EASE64168 operates with a clock supplied from an internal oscillation circuit (32.768kHz or 700kHz) when shipped. Oki Electric normally recommends that the EASE64168 be used as it is with these settings. Users who do not intend to change this setting can skip this section and proceed to section 2.2.

There are two methods for changing the clock settings.

(a) Oscillation clocks of crystal board and CROSC board in emulator

As explained in section 1.2, the EASE64168 operates with a clock supplied from an internal oscillation circuit (32.768kHz or 700kHz) when shipped. The crystal for the internal oscillation circuit of low-speed mode is mounted on the internal crystal board. The oscillation resistor for the internal oscillation circuit of high-speed mode is mounted on the internal CROSC board. The crystal board and CROSC board can be removed by first taking off the crystal board cover on the EASE64168's right side (see Figures 3-5 and 3-6).

The crystal board can be made to oscillate for use by soldering on a commercial crystal and oscillation resistor and capacitors. The CROSC board can be made to oscillate for use by soldering on a commercial oscillation resistor. The CROSC board is supplied in two versions: 1.5V and 3.0V. Select a resistor that matches the power supply voltage of the target chip that you will use.

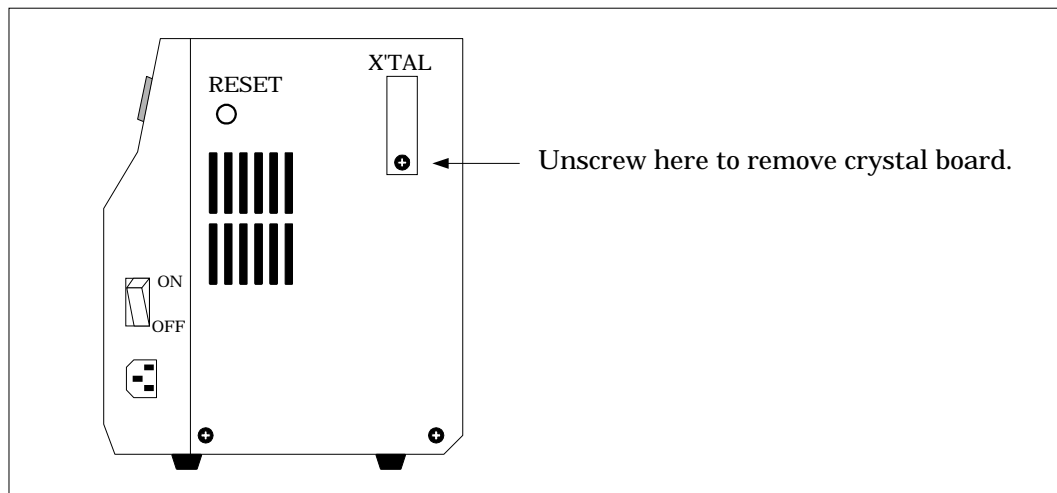


Figure 3-5 Removing Crystal Board (1)

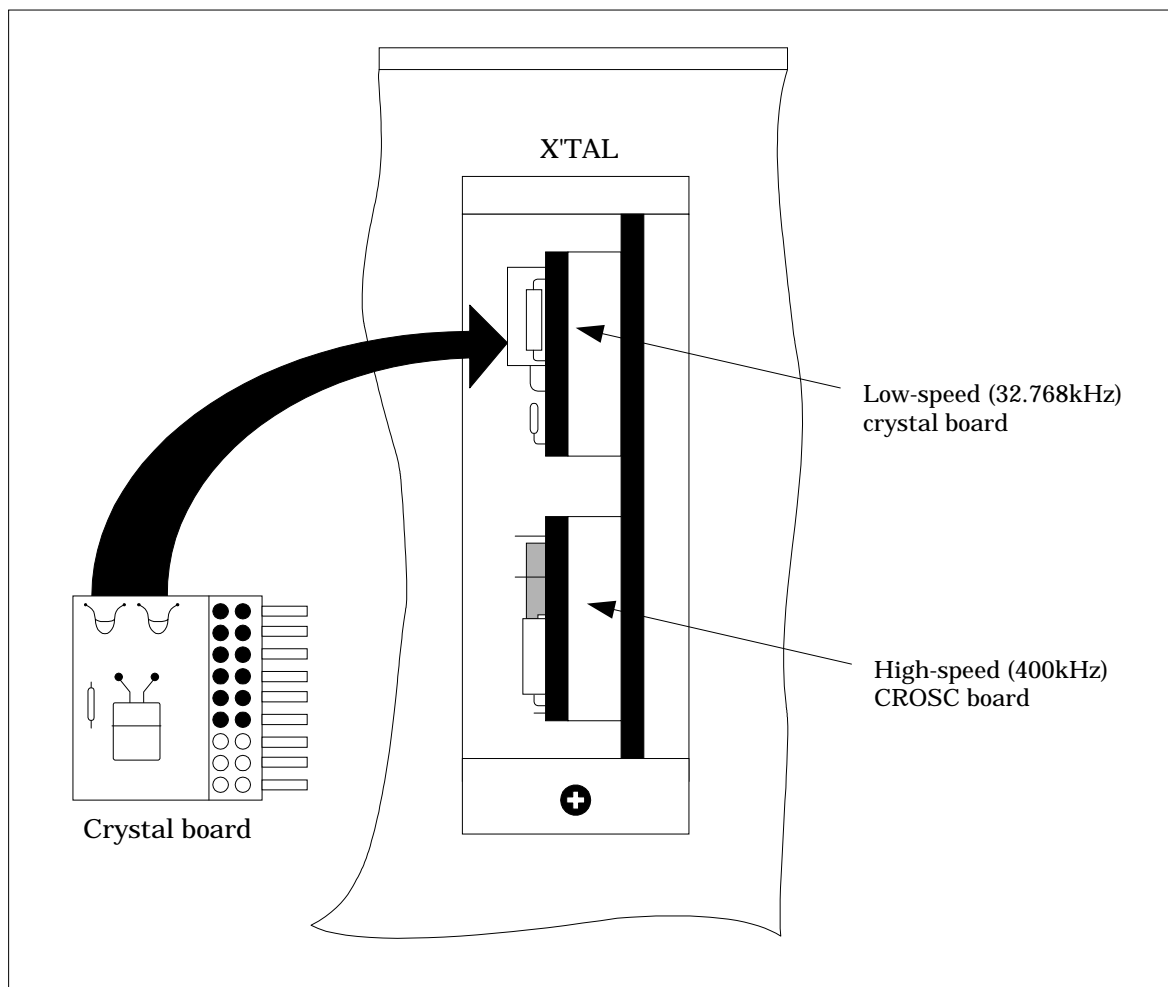


Figure 3-6 Removing Crystal Board (2)

As shown in Figure 3-6, the low-speed (32.768kHz) crystal board and high-speed (400kHz) CROSC board are internal to the EASE64168. The EASE64168 emulator's oscillation circuits are shown in Figures 3-7 and 3-8.

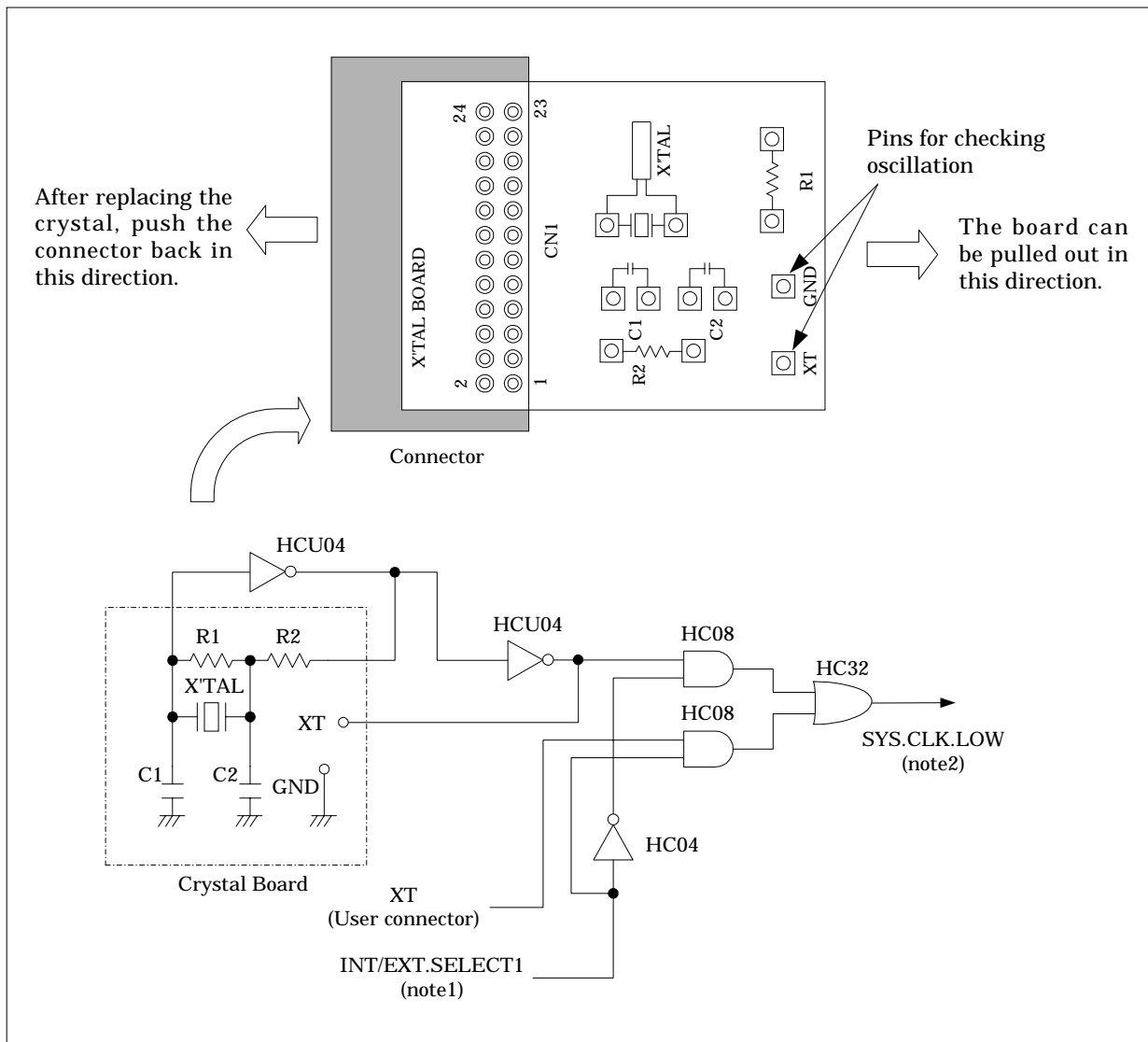


Figure 3-7 Crystal Board And Oscillation Circuit

■ **Note1** ■

The INT/EXT.SELECT1 signal is switched by the CCLK command. It determines whether the oscillation source will be from the internal crystal board or from the user connector XT pin.

■ **Note2** ■

EASE64168 operates with this clock in low-speed mode.

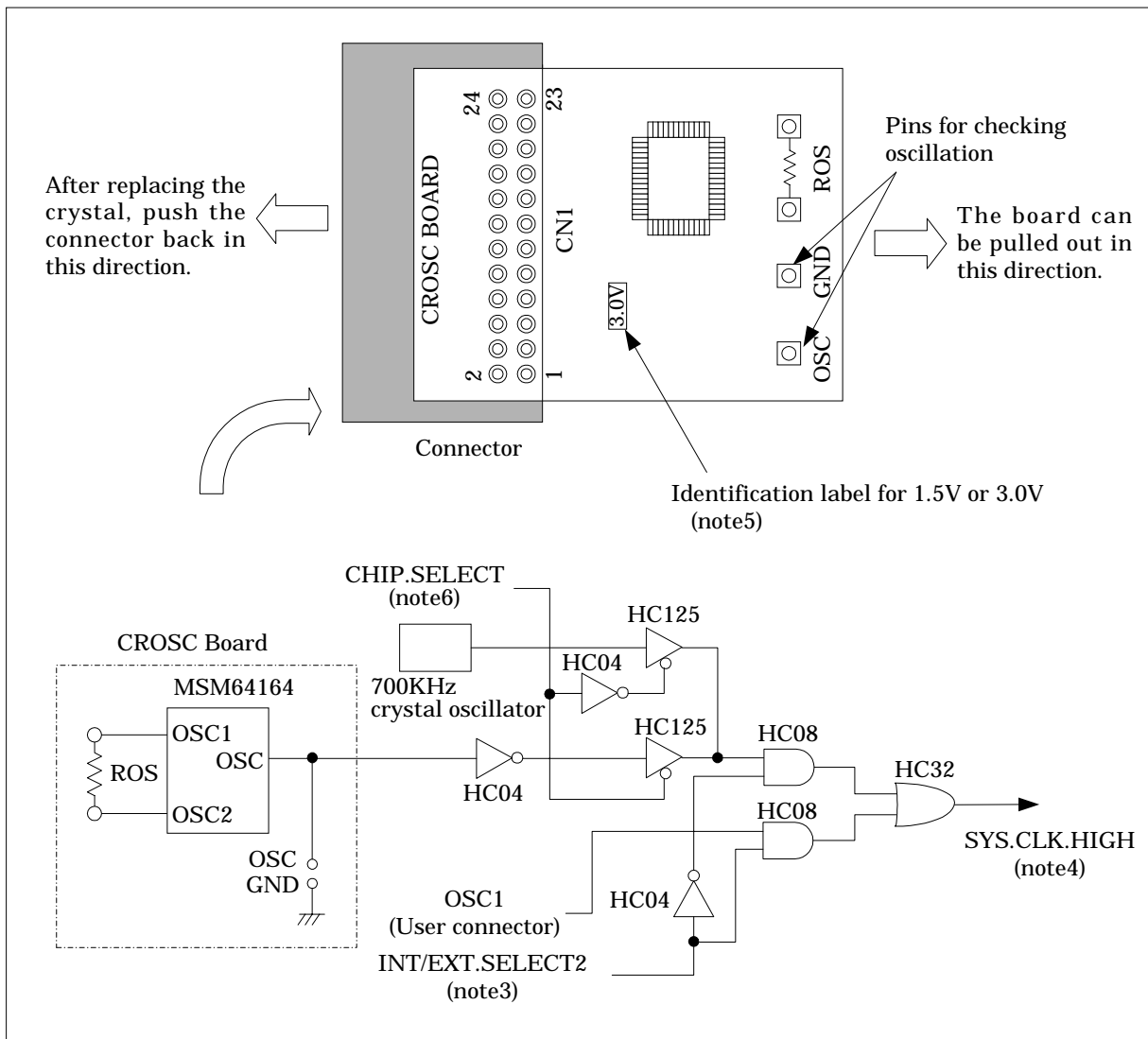


Figure 3-8 CROSC Board And Oscillation Circuit

■ **Note3** ■

The INT/EXT.SELECT2 signal is switched by the CCLK command. It determines whether the oscillation source will be from the internal oscillator board or from the user connector OSC1 pin.

■ **Note4** ■

EASE64168 operates with this clock in high-speed mode.

■ **Note5** ■

The CROSC board is supplied in two versions: 1.5V and 3.0V. Select a resistor that matches the power supply voltage of the target chip that you will use.

Note6

The Chip.select signal is switched by the CHIP command. It determines whether the high-speed oscillation source will be from the internal 700kHz crystal oscillator (ML64168 mode) or from the internal CROSC board (MSM64162/162A/162D/164C mode).

(b) User connector XT pin and OSC1 pin inputs

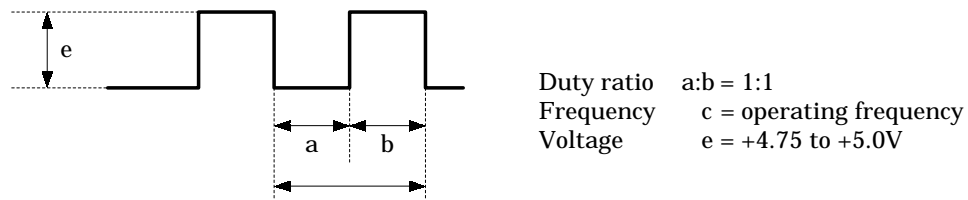
The emulator can be made to operate from clocks input on the user connector XT pin and OSC1 pin.

Reference

CCLK, DCLK commands

Note7

Use a signal like that shown below for clocks input on the user connector XT pin and OSC1 pin.

**Note8**

If you are using the emulator by oscillating from the crystal on the crystal board, then always verify that it oscillates correctly. Depending on the crystal's type and manufacturer, it might not oscillate.

If you have changed the oscillation resistor (ROS) on the CROSC board, then always verify its oscillation. Depending on the resistor's value, it might not oscillate. Refer to the user's manual of each chip for the range of resistor values.

Note9

There is no high-speed clock with the MSM64162D. Please be aware of this if using the EASE64168 in MSM64162 mode to evaluate a MSM64162D.

Note10

When the ML64168 mode, the frequency of the internal high-speed clock (700kHz) can not be changed. So when you have changed the oscillation resistor (ROS) on the CROSC board, the high-speed clock is not changed. If you want to change the frequency of the high-speed clock, input a clock on the user cable OSC1 pin and select the external clock by the CCLK command.

2.2 EASE64168 Switch Settings

There is a 7-bit dipswitch toward the top of the left panel of the EASE64168, labeled BAUD RATE SW (refer to Figure 3-9). The baud rate switch is explained below.

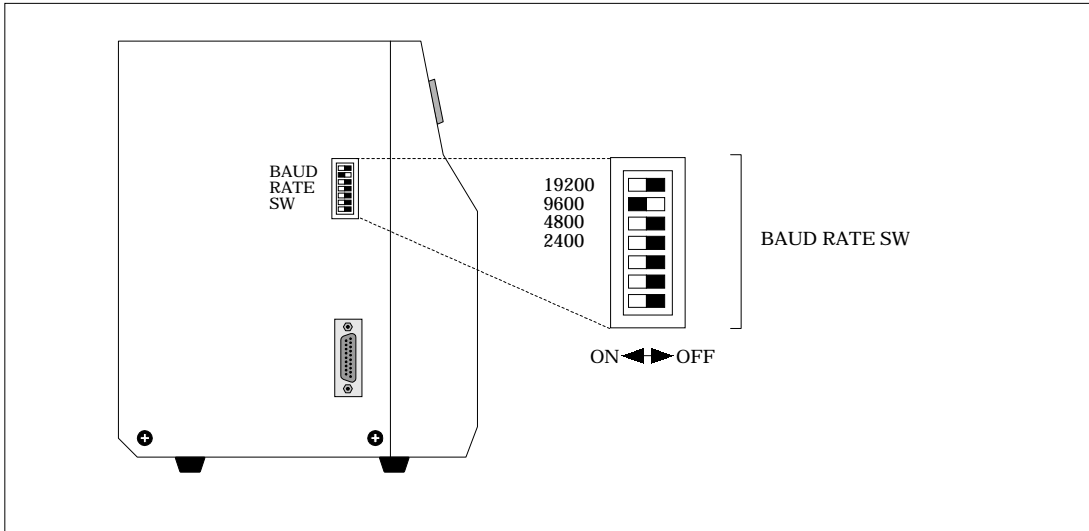


Figure 3-9 EASE64168 Dipswitch

[BAUD RATE SW]

These switches set the baud rate between EASE64168 and the host computer.

19200~2400 baud rate switches

These switches set the baud rate of the RS232C interface. They are used to match the EASE64168 baud rate with that of the host computer.

EASE64168 is set as follows when shipped.

Transfer format	8 bits, 1 stop bit, no parity
Baud rate	9600 bps

The host computer must be set to match all the above EASE64168 parameters except for the baud rate (note2). The baud rate can be set to a value 19200 bps to 2400 bps using the baud rate switches (refer to Table 3-1 below).

Table 3-1 Baud Rate Switch Settings

BAUD RATE SW \ BPS	19200	9600	4800	2400
19200	ON	OFF	OFF	OFF
9600	OFF	ON	OFF	OFF
4800	OFF	OFF	ON	OFF
2400	OFF	OFF	OFF	ON

■ Note1 ■

In Table 3-1:

- ON Flip bit switch to the left.
- OFF Flip bit switch to the right

■ Note2 ■

IBM-PC/AT computers use the INT232C command (described in Section 2.5).
For details, refer to your host computer's user manual.

■ Note3 ■

The EASE64168 settings must match the settings of the host computer connected to the RS232C cable. If the settings do not match, then the EASE64168 cannot operate.

2.3 Connecting The ADC POD

The M64162/164 ADC POD provides the CR oscillator of the MSM6416x series microcontrollers A/D converter. There are two types of M64162/164 ADC POD: 1.5V and 3.0V (note1).

The ML64168 ADC PODs (ADC-64168-3V and ADC-64168-1.5V) provide the CR oscillation clock for driving the ML64168 A/D converter. Note that there is a separate one for each operating voltage. (See Note 1.)

Choose the one appropriate for the target chip and operating voltage. (See Note 4.)

Table 3-2 ADC PODs

Target chip	Power supply voltage	Model number
MSM64162/162A/162D/162C	-3.0V	ADC-64164-3V
MSM64162/162A/162D/162C	-1.5V	ADC-64164-1.5V
ML64168	+3.0V	ADC-64168-3V
ML64168	+1.5V	ADC-64168-1.5V

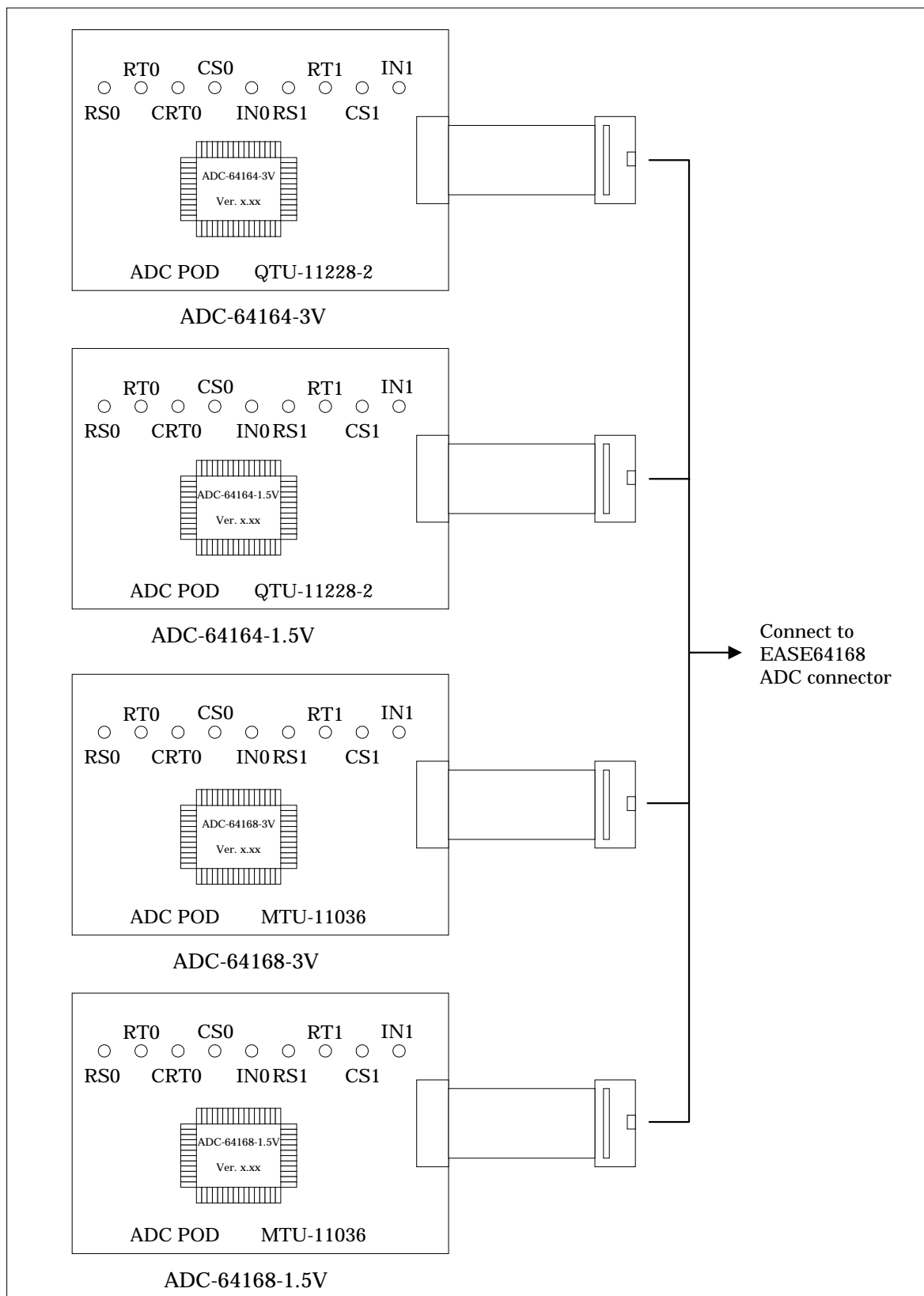


Figure 3-10 External Views Of M64162/164 ADC POD

Each of the pins shown in Figure 3-10 (RS0, RT0, CRT0, CS0, IN0, RS1, RT1, CS1, IN1) is identical to the corresponding M6416x series microcontroller pin. Connect resistors and capacitors that match the oscillation modes. Refer to the user's manual of your target chip for specific interfacing details.

■ **Note1** ■

The CR oscillation characteristics differ for 1.5V and 3.0V. Select values that match the power supply voltage of the target chip.

■ **Note2** ■

Note that the MSM64162D chip does not have the four pins RS1, RT1, CS1, and IN1.

■ **Note3** ■

The four ADC PODs included with the EASE64168 in-circuit emulator are outwardly identical. Check the model number on the label on the board.

 **Warning**

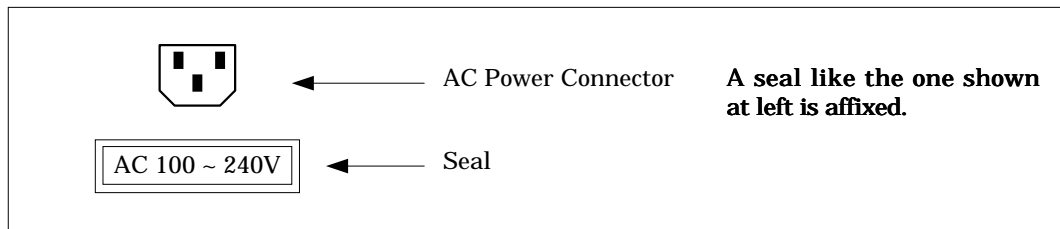
Be sure to connect the ADC POD with the emulator main unit's power supply switched off.



2.4 Confirming EASE64168 Power Supply Voltage

The EASE64168 has an internal power supply circuit that uses normal household power. The rated voltage of the power supply circuit is AC 100~240 V (50/60Hz).

The current voltage range setting is shown on a seal affixed below the AC power supply connector. Be sure that the AC power supply that you will use matches this range.



 **Warning**

ABSOLUTELY DO NOT USE A POWER SUPPLY OTHER THAN AC 100-240 V. DOING SO COULD CAUSE A FIRE.



2.5 Starting the EASE64168 Emulator

The procedure for starting the EASE64168 emulator is as follows.

- (1) Verify that the following EASE64168 emulator (hereafter called the emulation kit) switches are set correctly.

- * Baud rate switches

For details on switch settings, refer to Section 2.2, "EASE64168 Switch Settings."

- (2) Verify that the necessary cable types are connected to the emulation kit.

- * Is the AC power supply connector connected to the AC power supply cable?
- * Is the emulation kit connected to the host computer?
- * Is the user cable connected (when interfacing to the user application system)?
- * Is the M64162/164 ADC POD or ML64168 ADC POD connected?

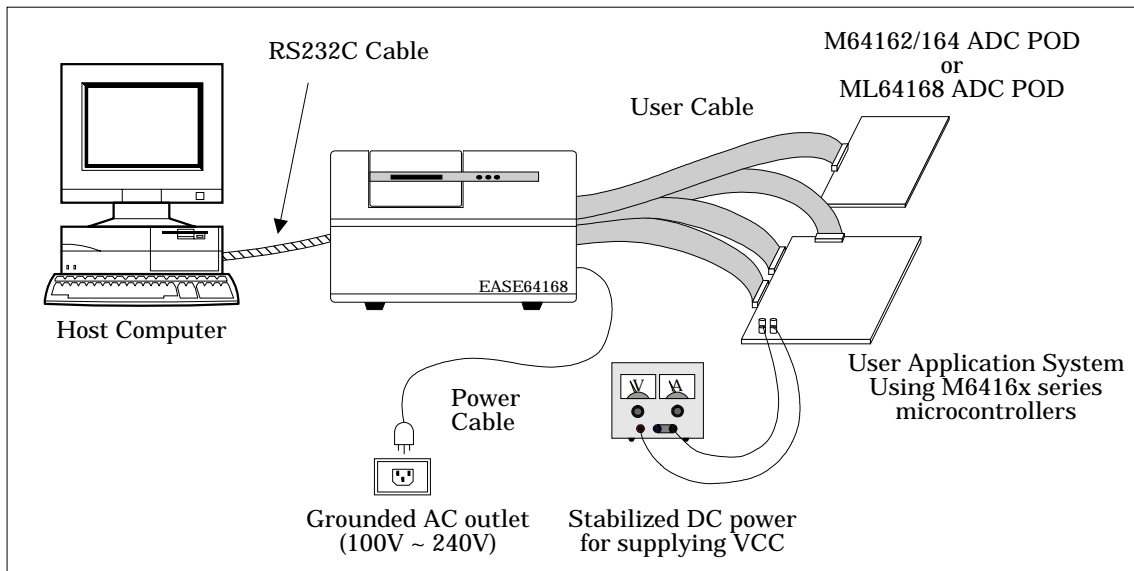


Figure 3-11 Cable Connection Diagram

■ **Note1** ■

The system will start even if the user application system is not connected. In this case, do not connect the user cables.

■ Note2 ■

Vcc is not supplied to the user application system from the user cables (however, GND is connected to the user application system through the user cables). If Vcc must be supplied to the user application system, then supply it from a separate power supply (refer to Figure 3-11).

(3) Turn on the host computer power supply, and start MS-DOS.

■ Note3 ■

Use MS-DOS version 3.3 or later.

(4) Set the host computer's transfer parameters.

When the EASE64168 is shipped, its data transfer parameters are as follows.

Communication method	RS232C interface
Transfer speed	9600 bps
Transfer format	8 bits, 1 stop bit, no parity

For details, refer to the manual of the host computer.

■ Note4 ■

IBM-PC/AT computers use the INT232C program (described in step 5 below).

(5) Invoke INT232C.

INT232C is a TSR (Transient but Stay Resident) program. It sets the RS232C interface operating conditions of the IBM-PC/AT, and simultaneously enables interrupt signals.

Invoking this program once will place it in host computer memory, where it will reside until removed. The method for invoking and removing INT232C is shown below.

```
A> INT232C [<options>[;<baud>,<parity>,<databits>,<stopbits>]]
```

The brackets [] can be omitted. When omitted, the default values of the following explanations apply.

<options>

- X Perform XON/XOFF control (note7).
- M Perform modem control.
- * Do not perform XON/XOFF or modem control.
- R Remove resident INT232C.

<baud>

Specifies the baud rate. Choose one of the following.

2400, 4800, 9600 (default)

<parity>

Specifies whether and what kind of parity checking to perform. Choose one of the following.

- N Do not perform parity checking (default).
- O Perform odd parity checking.
- E Perform even parity checking.

<databits>

Specifies the number of data bits. Choose one of the following.

7, 8 (default)

<stopbits>

Specifies the number of stop bits. Choose one of the following.

1 (default), 2

■ **Note5** ■

EASE64168 does not perform XON/XOFF control. Therefore, only use '*' or 'R' for the INT232C option. With any other settings, the EASE64X will not operate.

Example :

```
A> INT232C *  
(This is the same as: INT232C *;9600,N,8,1 )  
  
A> INT232C *;1200  
  
A> INT232C R
```

List of messages

INT232C outputs the following messages.

- INT232C has been removed from memory.
- INT232C has not been loaded.
- INT232C has already been loaded.
- INT232C has been loaded.

(6) Start the EASE64X debugger.

The debugger executable file EASE64X.EXE can be started from the directory that stores it or from another directory.

(1) Starting from the directory that stores EASE64X.EXE

Input the following after the DOS prompt.

```
A> EASE64X
```

(2) Starting from another directory

If the PATH environment variable includes the directory that contains EASE64X.EXE, then input is the same as in (1). If not specified by PATH, then the EASE64X debugger is invoked as follows.

```
A> pathname¥EASE64X
```

Here pathname is the absolute path name of the directory that contains EASE64X.EXE.

(7) The following message will be displayed on the console, and the system will wait for a reset switch input from emulation kit.

EASE64X Debugger Ver. x.xx
Copyright (C) xxxx. OKI Electric Ind. Co., Ltd.

(8) Turn on the emulation kit power supply switch and the power supply of the user application system. The following message will be displayed on the host computer, and emulator system initialization will end.

(9) A "" prompt will be displayed, and the system will wait for command input.

Low-Power Series Emulator <<EASE64168>> Ver.X.XX

*

(10) Debugger commands can now be input.

■ **Note 6** ■

- (1) For more information on the emulator's RS232C interface, refer to Section 2.2, "EASE64168 Switch Settings."
- (2) The user application system cannot be supplied with Vcc taken from the emulator.
- (3) Before turning on the emulator's power supply, verify that the connected AC power supply voltage is the same as the voltage shown on the AC power supply connector.
- (4) If the emulator does not start, then refer to Appendix 6.
- (5) Table 3-2 shows the various items initialized when EASE64168 is turned on, when the reset switch is pressed, when a RST command is executed, and when a RST E command is executed. A circle indicates that the item is initialized, while a dash indicates that it is not initialized. Also, when the reset switch is pressed, all open files will be closed.

Table 3-3(a) Initialization

Item	Contents Initialized	Power Applied	Reset Switch Pressed	RST Command	RST E Command
Evaluation Board	Initializes to same state as when a reset is input to a microcontroller in the M6416x.	○	○	○	○
Break Conditions	Only breakpoint bits are enabled.	○	-	-	-
Breakpoint Bits	All areas cleared to "0", disabling all breakpoint bit breaks.	○	-	-	-
Break Status	Cleared to state of no breaks generated.	○	○	○	-
Trace Pointer	Cleared to "0"	○	-	-	-

Table 3-3(a) Initialization

Item	Contents Initialized	Power Applied	Reset Switch Pressed	RST Command	RST E Command
Trace Trigger	Trace trigger disabled; set to address tracing.	○	○	○	-
Trace Enable Bits	All areas set to 1, enabling trace enable bit tracing.	○	-	-	-
Cycle Counter	Set to default mode.	○	○	○	-
EPROM Programmer Setting	Set to type 27512	○	-	-	-
Reset Input from User Cable	Prohibited	○	-	-	-
Trace Object Settings	Set to BCF, BSR0, BEF, and BSR1.	○	-	-	-
Memory Expansion	Set to EXPAND OFF state.	○	-	-	-

3. EASE64X Debugger Commands

3.1 Debugger Command Syntax

The explanations of this manual make use of the following symbols.

UPPERCASE Debugger command names are expressed with upper case letters.

Example : DCM, LOD, G

Italics Italicized expressions indicate user-supplied information (changes according to operator input). The following italicized words are used.

<i>parm</i>	This indicates a general parameter that follows after a command name. It includes <i>fname</i> , <i>address</i> , <i>data</i> , <i>number</i> , and <i>mnemonic</i> , explained below.
<i>fname</i>	This indicates a file name, including drive name, path name, primary name, and extension. Except for the extension, a file name is handled with the exact same processing as a DOS file name. Extensions are handled differently depending on the command (when omitted for some commands, default extensions exist).
<i>address</i>	This indicates an address value input.
<i>data</i>	This indicates a data value input.
<i>number count</i>	A number is used to indicate a cycle counter value input, step count, etc. A count indicates input of a pass count value of G command breakpoints. Both are recognized as decimal numbers.
<i>mnemonic</i>	This indicates an optional string input from a set of strings that is determined by the command type.

Special symbols These symbols have the following special meanings for explaining command syntax.

{xxx}	The xxx means an optional string used within an explanation. The xxx enclosed in { } means that it can be omitted.
<u> </u> (underline)	When text displayed automatically by the debugger and operator input are mixed on one line, the underlined portion indicates user input.

■ Note1 ■

Space is a string consisting of one or more spaces (ASCII code 20H) and/or tabs (ASCII code 09H) in any order.

3.1.1 Character Set

EASE64X debugger commands can make use of the following character set.

- | |
|---|
| <p>1. Alphabetic characters (upper and lower case)</p> <p>A B C D E F G H I J K L M
 N O P Q R S T U V W X Y Z
 a b c d e f g h i j k l m
 n o p q r s t u v w x y z</p> <p>2. Digits</p> <p>0 1 2 3 4 5 6 7 8 9</p> <p>3. Delimiters (note2)</p> <p>TAB space CR</p> <p>4. Other special symbols</p> <p>, () - * ></p> |
|---|

■ Note2 ■

TAB is ASCII code 09H; space is ASCII code 20H; CR (carriage return) is ASCII code 0DH.

■ Note3 ■

All characters usable with EASE64X debugger commands are included in this character set. However, any character can be coded in command fields, described later.

3.1.2 Command Format

Debugger Command Format

<pre>command_name parm, parm . . . , parm</pre>
--

Debugger commands consist of a command name followed by several parameters (*parm*). Space always delimits between the command name and *parm*. Commas (,) delimit between *parm* and *parm*. A command line is recognized as ending at the point a carriage return is input.

Comment Input

The entire string following a semicolon (;) is recognized as a comment. It will be ignored during command parsing. For example, the entire input line below is a comment, so the emulator will perform no operation.

Example : * ; ; ; This is an example of whole comment line ; ; ; ;

Command Name Format

Command names are strings consisting of 1-7 alphabetic characters. They express instructions given to the debugger. A command name's function is indicated by its first character. Second and following characters are keywords for memory and internal registers of the evaluation board or emulator.

- D (Display)..... Data display commands
- C (Change)..... Data change commands
- E (Enable) Enable commands
- F (Fill)..... Data fill commands
- R (Reset)..... Reset commands
- S (Set)..... Set commands
- P (Program) Commands for writing data to EPROM
- T (Transfer)..... Commands for reading data from EPROM
- V (Verify) Commands for comparing memory contents
- G (Go) Execute (emulation) commands

3.1.3 Command Summary

This section gives a summary table of all EASE64X commands.

Command Group Name			Page
No.	Name	Function	Reference page
	Syntax		
	Parameters / options		

Detailed explanations of each command are given in 3.4. The table of this section was created with the purpose of first giving a quick overview of the commands, and then in the future serving as a command index.

The table of this section follows the format below.

No.	Sequence number
Name	Name of the command
Syntax	Syntax of the command
Parameters and Options	Describes each of the parameters and options expressed in Command Syntax
Reference Page	The reference page for a explanation in 3.4 "EASE64X command Details".

Evaluations Board Access Commands		Page
1	CHIP	Set target chip
	CHIP [mnemonic]	
	mnemonic : 64162, 64164, 64168	
2	D	Display contents of target chip registers
	D or D <i>mnemonic</i>	
	<i>mnemonic :</i> PC ,P0 ,CAPR1 ,IRQ0 B ,P1D ,CAPCON ,IRQ1 A ,P2D ,TBCR ,IRQ2 HL ,P3D ,DSPCON ,BUPCON XY ,P4D ,CNTA ,MIEF CY ,SBUF ,CNTB SP ,SCON ,ADCON0 BSR0 ,FCON ,ADCON1 BSR1 ,BDCON ,IE0 BCF ,BFCON ,IE1 BEF ,CAPR0 ,IE2	
3	C	Change contents of target chip registers
	C <i>mnemonic data</i>	
	<i>mnemonic : (note1)</i> PC (0 to 7DF or 0 to FDF) B (0 to F) ,CAPR0 (0 to FF) ,TBCR (0 to F) ,P20CON (0 to F) A (0 to F) ,CAPR1 (0 to FF) ,DSPCON (0 to 3) ,P21CON (0 to F) HL (0 to FF) ,CAPCON (0 to 1) ,IE0 (0 to F) ,P22CON (0 to F) XY (0 to FF) ,CNTA (0 to 79999) ,IE1 (0 to F) ,P23CON (0 to F) SP (0 to FF) ,CNTB (0 to 3FFF) ,IE2 (0, 1) ,P30CON (0 to F) BSR0 (0 to F) ,ADCON0 (0 to 3) ,IRQ0 (0 to F) ,P31CON (0 to F) BSR1 (0 to F) ,ADCON1 (0 to F) ,IRQ1 (0 to F) ,P32CON (0 to F) BCF (0, 1) ,SBUF (0 to FF) ,IRQ2 (0 to F) ,P33CON (0 to F) BEF (0, 1) ,SCON (0 to F) ,MIEF (0, 1) ,P40CON (0 to F) P1D (0 to F) ,FCON (0, 1) ,P41CON (0 to F) P2D (0 to F) ,BDCON (0 to F) ,P42CON (0 to F) P3D (0 to F) ,BFCON (0, 1 or 0 to F) ,P43CON (0 to F) P4D (0 to F) ,BUPCON (0 to 3 or 0, 1) ,P01CON (0 to F)	

Evaluations Board Access Commands (cont.)			Page
4	DDSPR	Display Display Register	3-93
	DDSPR		
5	CDSPR	Change Display Register	3-71
	CDSPR <i>mnemonic</i>		
	<i>mnemonic</i> : 0~20 ... MSM64162 mode 0~30 ... MSM64164C and ML64168 mode		

■ **Note1** ■

The numbers in parentheses indicate the input data range for the corresponding *mnemonics*.

The data range of PC is 0H~7DFH in MSM64162 mode and 0H~FDFH in MSM64164C mode and 0~1FDFH in ML64168 mode.

When TBCR is changed, it will be reset to 0 regardless of the specified data.

The change data of CNTA is a decimal value.

In MSM64162 mode, the following mnemonics are invalid.

P4D, SBUF, SCON, P40CON, P41CON, P42CON, P43CON

The data range of BFCON is 0H or 1H in MSM64162 mode and 0H~FH in MSM64164C, ML64168 mode.

The data range of BUPCON is 0H~3H in MSM64162, ML64168 mode and 0H or 1H in MSM64164C mode.

The FCON register does not exist in the MSM64162D chip.

If invalid data (5H, 6H, or 7H) is written to the ADCON1 register when evaluating a MSM64162D, then the emulator may operate incorrectly.

Code Memory Commands		Page
1	DCM	Display Code Memory
	DCM <i>address</i> [, <i>address</i>] or DCM *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : displays entire address range	
2	CCM	Change Code Memory
	CCM <i>address</i>	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
3	FCM	Fill Code Memory
	FCM <i>address</i> , <i>address</i> [, <i>data</i>] or FCM * [, <i>data</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : fills entire address range <i>data</i> : 0 to FF	
4	LOD	Load Disk file program into Code Memory
	LOD <i>fname</i>	
	<i>fname</i> : [Pathname] filename [extension]	
5	SAV	Save Code Memory into Disk file
	SAV <i>fname</i> [<i>address</i> , <i>address</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode <i>fname</i> : [Pathname] filename [extension]	

Code Memory Commands (cont.)		Page
6	VER	Verify Disk file with Code Memory
	VER <i>fname</i> [<i>address</i> , <i>address</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode <i>fname</i> : [<i>Pathname</i>] <i>filename</i> [<i>extension</i>]	
7	ASM	Line Assembler Command
	ASM <i>address</i>	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
8	DASM	Disassemble Command
	DASM <i>address</i> , [<i>address</i>] or DASM *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : displays entire address range	

Data Memory Commands		Page
1	DDM	Display Data Memory
	DDM <i>address</i> [, <i>address</i>] or DDM *	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode * : displays entire address range	
2	CDM	Change Data Memory
	CDM <i>address</i>	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode	
3	FDM	Fill Data Memory
	FDM <i>address, address</i> [, <i>data</i>] or FDM * [, <i>data</i>]	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode * : files entire address range <i>data</i> : 0 to FF	

Emulation Commands		Page
1	STP	Step Execution
	STP [<i>number</i>] [<i>address</i>] or STP *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : executes 65535 steps <i>number</i> : 1 to 65535	
2	G	Realtime Emulation (continuous execution)
	G [<i>address</i>] [, <i>parm</i>]	
	<i>parm</i> : <i>address</i> [, <i>address</i> ..., <i>address</i>] RAM (data-count) BAR (data-count) <i>address</i> (count) <i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	

Break Commands			Page
1	DBC	Display Break Condition Register	3-83
	DBC		
2	SBC	Set Break Condition Register	3-139
	SBC		
3	DBS	Display Break Status	3-86
	DBS		
4	DBP	Display Break Point Bits	3-84
	DBP <i>address</i> [, <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
5	EBP	Enable Break Point Bits	3-106
	EBP <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
6	RBP	Reset Break Point Bits	3-129
	RBP <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
7	FBP	Fill Break Point Bits	3-111
	FBP <i>address, address</i> [, <i>data</i>] or FBP * [, <i>data</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : fills entire address range <i>data</i> : 0,1	

Trace Commands		Page	
1	DTM	Display Trace Memory	3-97
	DTM <i>-number_{step} number_{step}</i> or DTM <i>number_{TP} number_{step}</i> or DTM *		
	<i>number_{step}</i> : number of steps to go back (1~8192) <i>number_{step}</i> : number of steps to display (1~8192) <i>number_{TP}</i> : value of TP at which to start display (0~8191) * : Display the entire area of trace memory		
2	CTO	Change Trace Object	3-76
	CTO		
3	DTO	Display Trace Object	3-101
	DTO		
4	CTDM	Change Trace Data Memory	3-75
	CTDM [<i>address</i>]		
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode		
5	DTDM	Display Trace Data Memory	3-96
	DTDM		
6	STT	Set Trace Trigger	3-146
	STT		

Trace Commands (continued)			Page
7	DTT	Display Trace Trigger	3-105
	DTT		
8	RTT	Reset Trace Trigger	3-136
	RTT		
9	DTR	Display Trace Enable Bits	3-103
	DTR <i>address</i> [, <i>address</i> ..., <i>address</i>] or DTR *		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : displays entire address range		
10	ETR	Enable Trace Enable Bits	3-107
	ETR <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
11	RTR	Reset Trace Enable Bits	3-135
	RTR <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
12	FTR	Fill Trace Enable Bits	3-114
	FTR <i>address</i> , <i>address</i> [, <i>data</i>] or FTR * [, <i>data</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : fills entire address range <i>data</i> : 0,1		

Trace Commands (continued)			Page
13	DTP	Display Trace Pointer	3-102
	DTP		
14	RTP	Reset Trace Pointer	3-134
	RTP		

Reset Commands			Page
1	RST	Reset System and Evaluation Chip	3-132 3-133
	RST	Reset the system	
	RST E	Reset the evaluation chip.	
2	URST	Set User Reset Terminal (on user connector)	3-153
	URST	[<i>mnemonic</i>]	
	<i>mnemonic</i>	: ON, OFF	

Performance/Coverage Commands			Page
1	DCC	Display Cycle Counter	3-87
	DCC		
2	CCC	Change Cycle Counter	3-65
	CCC [-] <i>number</i>		
	<i>number</i> : 0 to 4294967295		
3	SCT	Set Cycle Counter Trigger	3-141
	SCT		
4	DCT	Display Cycle Counter Trigger	3-90
	DCT		
5	RCT	Reset Cycle Counter Trigger	3-130
	RCT		
6	DIE	Display Instruction Executed Bits	3-94
	DIE <i>address</i> [, <i>address</i>] or DIE *		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : displays entire address range		
7	RIE	Reset Instruction Executed Bits	3-131
	RIE		

EPROM Program Commands			Page
1	TYPE	Set EPROM Type	3-152
	TYPE <i>mnemonic</i>		
	<i>mnemonic</i> : 64, 128, 256, 512		
2	PPR	Program EPROM	3-126
	PPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] PPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : programs entire address range <i>address_{EPROM}</i> : EPROM write address		
3	TPR	Transfer EPROM into Code Memory	3-149
	TPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] TPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : programs entire address range <i>address_{EPROM}</i> : EPROM read address		
4	VPR	Verify EPROM with Code Memory	3-157
	VPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] VPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : verifies entire address range <i>address_{EPROM}</i> : EPROM comparison address		

Mask Option File Commands			Page
1	LODM	Load Disk file Mask Option into System memory	3-123
	LODM <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
2	VERM	Verify Disk file with System Memory	3-156
	VERM <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
3	PPRM	Program Mask Option Data into EPROM	3-128
	PPRM		
4	TPRM	Transfer EPROM into System Memory	3-151
	TPRM		
5	VPRM	Verify EPROM with System Memory	3-159
	VPRM		

Commands for Automatic Command Execution			Page
1	BATCH	Batch Processing	3-59
	BATCH <i>fname</i>		
	<i>fname</i> : [<i>Pathname</i>] <i>filename</i> [<i>Extension</i>]		
2	PAUSE	Pause Command Input	3-125
	PAUSE		

Other Commands			Page
1	LIST	Listing (Redirect the Console output to Disk file)	3-121
	LIST <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
2	NLST	No Listing (Cancel the Console output Redirection)	3-124
	NLST		
3	>	Call OS Shell	3-56
	> DOS command		
4	CCLK	Display/Change Clock Mode	3-66
	CCLK [<i>mnemonic</i>]		
	<i>mnemonic</i> : HIN, HOUT, LIN, LOU		
5	CIPS	Display/Change Interface Power Supply	3-74
	CIPS [<i>mnemonic</i>]		
	<i>mnemonic</i> : INT, EXT		
6	EXPAND	Expand Code Memory	3-109
	EXPAND [<i>mnemonic</i>]		
	<i>mnemonic</i> : ON, OFF		
7	EXIT	Terminate the Debugger and Exit to OS	3-108
	EXIT		

3.2 History Functions

EASE64X has a function for saving previous command line input (note1). This function is known as the history function.

When using the debugger, occasionally you will want to input the same command as one several previous, or the same command except with different parameters. This is when the history function is especially powerful.

(1) Current line buffer and history buffer

EASE64X has a current line buffer for editing the current command line input and a history buffer for saving command lines.

The command line buffer is a 72-character buffer for command line input. The history buffer is a 72-character by 20-line buffer for storing command line input in order.

There are two types of history buffers. One is for normal command line input, and one is for command line input during execution of the ASM command.

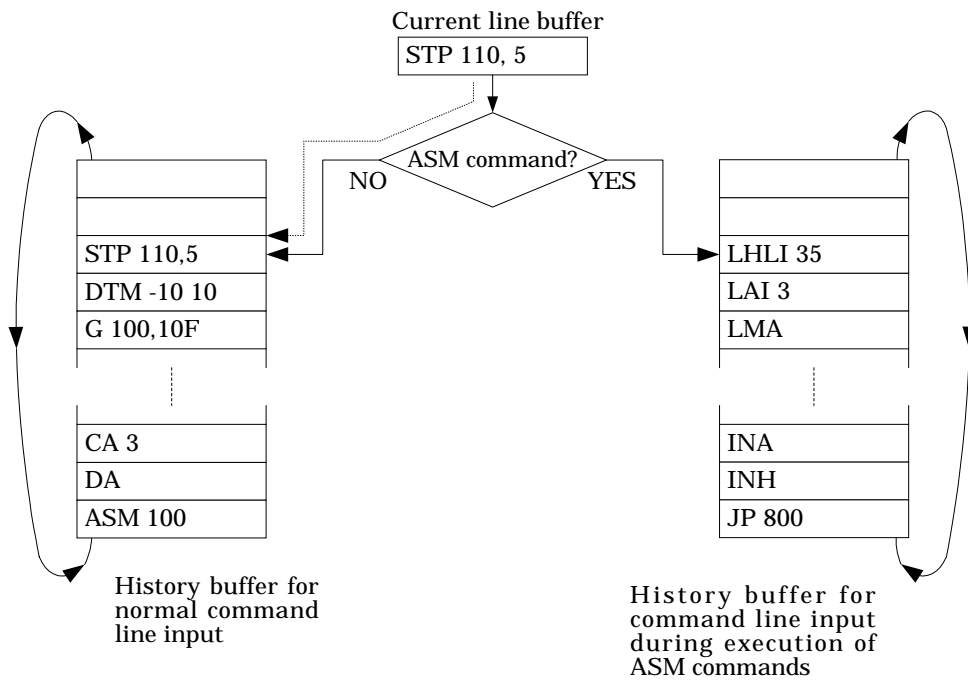


Figure 2-5 Current Line Buffer and History Buffer

A command line input by an operator is first stored in the current line buffer. Simultaneous to the operator pressing a carriage return, the contents of the current line buffer are stored in the history buffer. Each time a command line is input, its contents are stored in order in the history buffer.

The history buffer is configured as a ring. The oldest input line (the command line input 20 lines before the current command line input) is overwritten. As a result, the previous 20 lines of command line input will always be stored.

The operator can read the contents of the history buffer into the current line buffer at any time during command line input.

Note that input from a file called by the BATCH command will not be stored in the history buffer.

Using history functions

This somewhat covers the same material as Section 3.3.3, "Special Keys For Raising Command Input Efficiency," but the history functions are utilized with the ↑ key (or CTRL + K) and the ↓ key (or CTRL + J).

Pressing the ↑ key will read the immediately previous command line input from the history buffer into the command line buffer and display it on the console. Then each time the ↑ key is pressed, the next previous command line input will be read and displayed.

Converse to the ↑ key, the ↓ key reads the command line input immediately afterward from the history buffer and displays it on the console.

After the operator has edited the displayed current line buffer contents with the special keys for command line editing, as explained in the next section, he can enter it as the new command line input by pressing the enter key. At this time, the current line buffer will be executed to its end as the command line input, regardless of the cursor position on the line.

Of course, the contents of the current line buffer can be executed if only the enter key is pressed without any editing.

■ Note1 ■

EASE64X command line input is input from the console after the EASE64X output prompt "*", and during ASM command execution.

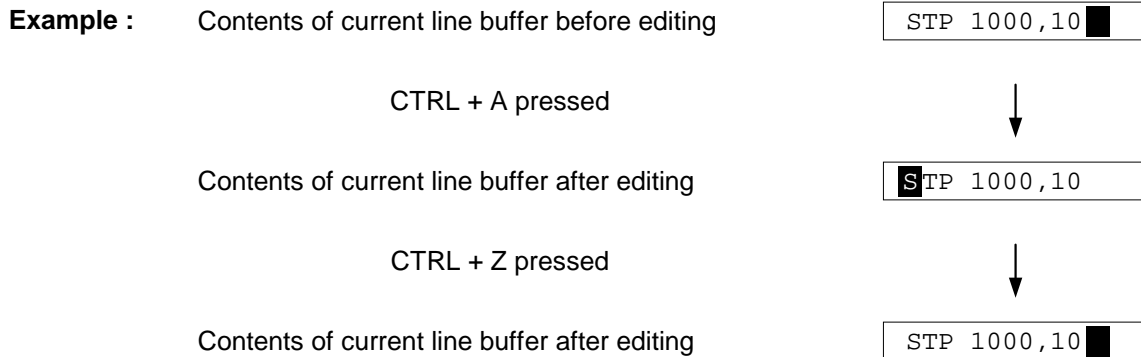
3.3 Special Keys For Raising Command Input Efficiency

EASE64X provides special editing keys, as mentioned in the previous section on the history uncton, for raising efficiency of current line buffer editing. There are a total of 12 special keys. They can effectively create new command line inputs. The special keys and their control functions are explained below.

(1) CTRL+A and CTRL+Z

CTRL+A moves the cursor to the first location of the current line buffer.

CTRL+Z moves the cursor to the last location of the current line buffer.



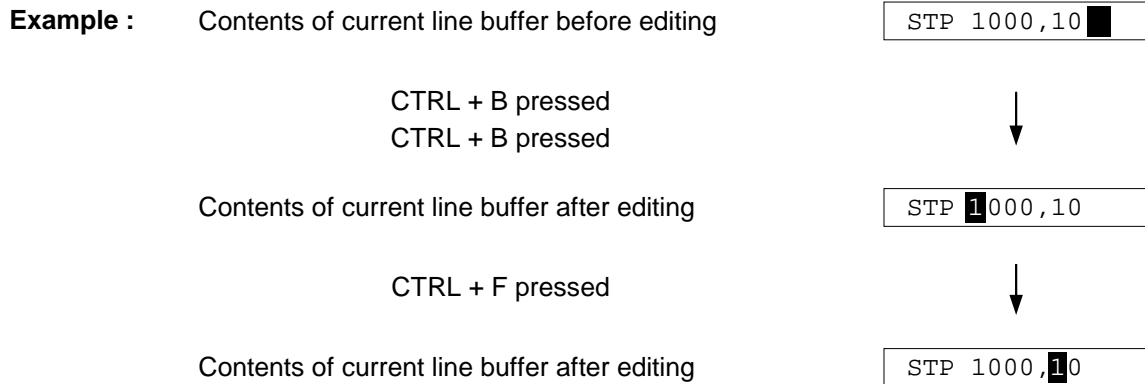
(2) CTRL+B and CTRL+F

CTRL+B searches for a string consisting of letters and digits only from the current cursor location in the current line buffer toward the first location. In other words, it recognizes characters other than letters and digits as string delimiters.

If a string is detected, then the cursor will be moved to its first location. If no string could be detected, then the cursor will be moved to the first location of the current line buffer.

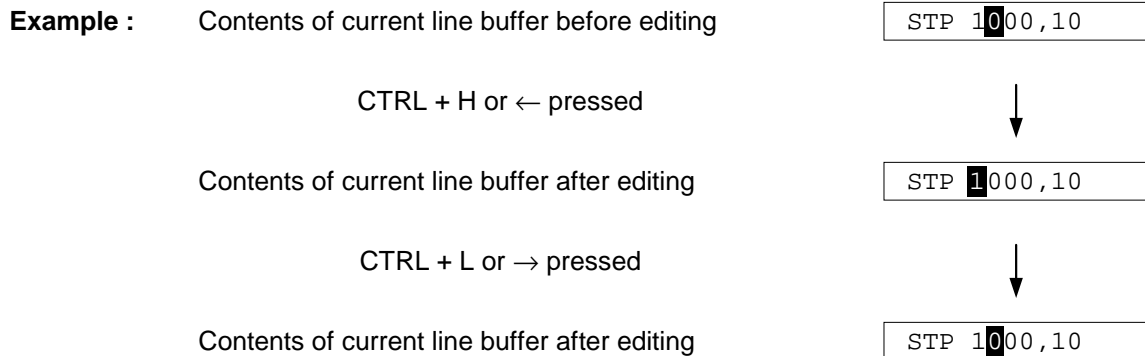
CTRL+F searches for a string consisting of letters and digits only from the current cursor location in the current line buffer toward the last location. In other words, it recognizes characters other than letters and digits as string delimiters.

If a string is detected, then the cursor will be moved to its first location. If no string could be detected, then the cursor will be moved to the last location of the current line buffer.

**(3) CTRL+H (or ←) and CTRL+L (or →)**

CTRL+H moves the cursor one location to the left of its current location in the current line buffer.

CTRL+L moves the cursor one location to the right of its current location in the current line buffer.

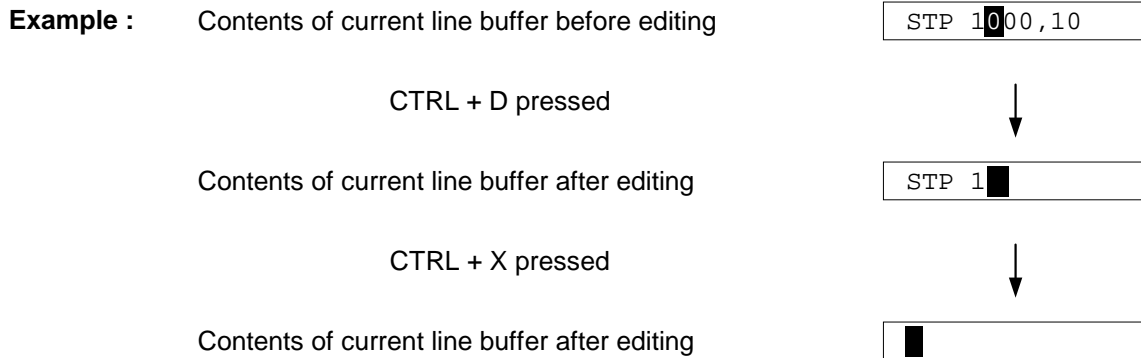
**(4) CTRL+K (or ↑) and CTRL+J (or ↓)**

CTRL+K (or ↑) and CTRL+J (or ↓) read history buffer contents into the current line buffer, as explained in the previous section. For details, refer to the previous Section 3.2, "History Function."

(5) CTRL+D and CTRL+X

CTRL+D deletes current line buffer contents from the current cursor position to the last location, and then moves the cursor to the end of the line.

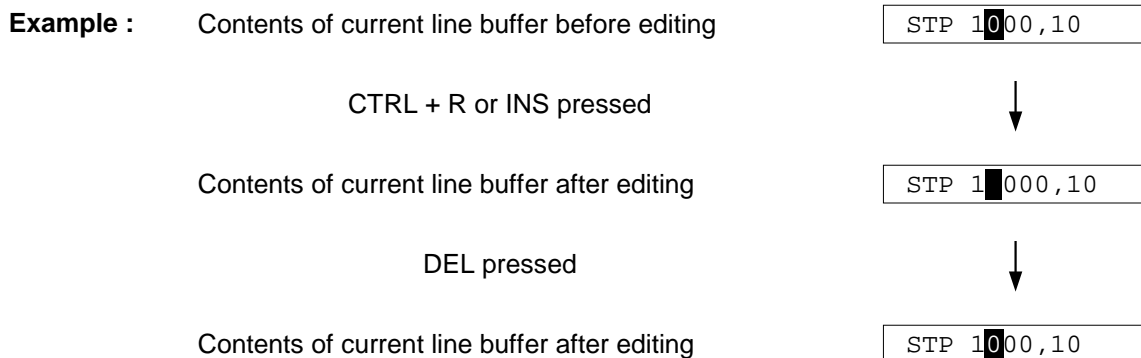
CTRL+X deletes the current line buffer contents, and then moves the cursor to the start of the buffer.



(6) CTRL+R (or INS) and DEL

CTRL+R (or INS) inserts a single blank character at the current cursor position in the current line buffer.

DEL deletes a singles character at the current cursor position in the current line buffer. The cursor position does not change.



■ **Note1** ■

If you will use EASE64X with an IBM PC/AT, then add the appropriate ANSI escape sequence driver from your DOS system disk to CONFIG.SYS. If you forget to do so, then you will not be able to use the special editing keys.

Host Computer	ANSI Escape Sequence Driver Name
IBM PC/AT	ANSI.SYS

■ **Note2** ■

To use the ↑, ↓, ←, →, INS and DEL keys, set your host computer's key table to the same key code settings as in the table on the next page. If the settings do not match, then the danger

exists that a special key function will operate differently.

The table below shows the special editing keys and how they affect the contents of the current line buffer. It also shows the EASE64X internal processing code (in hexadecimal) for each key. Check the settings of your host computer's key table, and if they do not match these settings, then change them to match.

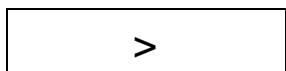
In the table, "line" means the current line buffer.

Editing Key	Control Function	Code
CTRL + A	Moves the cursor to the start of the current line buffer.	01H
CTRL + B	Searches for string of letters and digits only from the current cursor location to the first location, and moves the cursor to the start of the string.	02H
CTRL + D	Deletes all characters from the current cursor location to the last location.	04H
CTRL + F	Searches for string of letters and digits only from the current cursor location to the first location, and moves the cursor to the start of the string.	06H
CTRL + J or ↓	Reads the next command line input from the history buffer into the current line buffer and displays it.	0AH
CTRL + K or ↑	Reads the previous command line input from the history buffer into the current line buffer and displays it.	0BH
CTRL + H or ←	Moves the current cursor position one to the left.	08H
CTRL + L or →	Moves the current cursor position one to the right.	0CH
CTRL + X	Deletes the current line buffer, and moves the cursor to the first location.	18H
CTRL + Z	Moves the cursor to the end of the current line buffer.	1AH
CTRL + R or INS	Inserts a single blank at the current cursor location.	12H
DEL	Deletes a character at the current cursor location.	7FH

3.4 Command Details

This chapter explains the EASE64X commands organized by function.

Call OS Shell



Input Format

>DOS command

Description

The shell function invokes the MS-DOS command processor COMMAND.COM as a child process of the EASE64X debugger. The string input after this command will be passed to COMMAND.COM and executed.

After the MS-DOS command terminates, the EASE64X debugger will again wait for a command to be input.

In order to realize the shell function, the free area of the system being used must have sufficient space for invoked programs. The resident portion of EASE64X.EXE consumes about 90K bytes. Thus, for a program to be invoked after the shell command has been executed, it must have fewer bytes than the original free area less the size of the newly loaded COMMAND.COM.

Execution Example

* >COPY A:SAMP1.LST B:

Line Assembler Command

ASM

Input Format

ASM *address*

address : 0~7DF (MSM64162 mode)

address : 0~FDF (MSM64164C mode)

address : 0~1FDF (ML64168 mode)

Description

The ASM command converts OLMS-64K series instruction statements (directives, mnemonics, and operands) into object code using an assembler based on ASM64K, and then stores that object code in code memory (note).

The address expresses a code memory address. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

When the carriage return is input, the emulator displays the following message and waits for data input.

address * : wait for instruction statement input

At this point the operator can input code that follows the format below.

- (1) The maximum number of characters that can be input on one line is 29.
- (2) After an instruction statement and carriage return are input, the emulator displays the assembled object code and then waits for input at the next address.
- (3) The ASM command terminates with an "END".
- (4) Spaces or tabs can be used as delimiters.
- (5) All OLMS-64K series mnemonics and operands can be used.
- (6) Character constants (such as 'A') and string constants (such as "ABC") cannot be coded in operands.
- (7) A semicolon ";" is used to code a comment.
- (8) The default radix for immediate values used in operands is 10 (decimal values). To use a radix other than 10, input as shown in the following table.

When a hexadecimal constant's first character would normally be a letter (A~F) , a '0' (zero) needs to be inserted as the first character to distinguish in from a symbol.

Radix	Syntax	Example
Binary (radix 2)	Append a 'B' after the number.	01010101B
Octal (radix 8)	Append an 'O' after the number.	777O
Decimal (radix 10)	Append a 'D' or nothing after the number.	10D,10
Hexadecimal (radix 16)	Append a 'H' after the number.	0ABH

(9) The following assembler directives can be used.

Directive Type	Directives Allowed
Address control	ORG
Data definition	DB
Assembly control	END

(10) The history function can be used. The ASM command has a 20-line buffer, separate from the debugger's history buffer, for use as an assembler-only history function. This buffer's contents are preserved even after the ASM command terminates, so when the ASM command is started again, the previously input 20 lines can be easily be brought up for editing using the arrow keys. This feature can simplify input.

■ **Note1** ■

Comments input with the ASM command cannot be displayed with the DASM command.

Execution Example

```
* ASM 100
LOC = 0100 50 C0      * LHLI 0C0H
LOC = 0102 6F         * LMA
LOC = 0200            * ORG
LOC = 0200            * LHLI 03FH
LOC = 0202            * END

* DASM 100,103
LOC = 0100 50C0      LHLI C0
LOC = 0102 6F       LMA
LOC = 0103 00       NOP
```

Batch Processing

BATCH

Input Format

```
BATCH fname
      fname : [Pathname] filename [Extension]
```

Description

The BATCH command automatically executes the contents of the specified *fname* as emulator commands.

The input file name can have a path specification. If the path is omitted, then the file will be taken in the current directory.

If the file extension is omitted, then a default extension (CMD) will be appended. To specify a file without an extension, append a period '.' after the filename.

In addition to emulator commands, the batch file can also contain assembler mnemonics input within the ASM command.

Automatic execution is performed until the end of the file.

■ **Note1** ■

Only one batch file can be open. Therefore, even if a BATCH command is included within a batch file, it will be ignored.

Execution Example

```
* BATCH E4
  Batchfile : E4  opened

* DTR 0,30
      0 1 2 3 4 5 6 7 8 9 A B C D E F
LOC=0000 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
LOC=0010 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
LOC=0020 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
LOC=0030 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

* DTO
  TRACE OBJECT --->  BCF, BSR0, BEF, BSR1

* DCC
```


CYCLE COUNTER STATUS : 000000000

* TYPE

EPROM TYPE ---> 27512

*

* Batchfile : E4 closed

Change contents of target chip registers

C

Input Format

C *mnemonic* [, *data*]
mnemonic : *mnemonic of a register*

Description

The C command changes the contents of the register specified by *mnemonic*. A list of *mnemonics* is shown in Table 3-3. The data differs for each register; Table 3-3 shows the data range of each.

mnemonic : *old-data* OLD ---> (note1)

Here *mnemonic* expresses the mnemonic of the register that is to have its current contents changed. The old-data will be the current contents of the SFR or register. At this point the operator enters new data (*data*) and inputs a carriage return.

mnemonic : *old-data* OLD ---> *data* (note1)

■ Note1 ■

The following mnemonics are write-only registers. If selected, then old-data will not be displayed.

P20CON, P21CON, P22CON, P23CON

P30CON, P31CON, P32CON, P33CON

P40CON, P41CON, P42CON, P43CON

P01CON

■ Note2 ■

The M6416x series microcontrollers mask option specifications can set P5 and P6, but their contents cannot be displayed or changed by EASE64168 with a debugger command.

Table 3-3(a) List of registers mnemonics

Register Name	Mnemonic	Input Data Range in MSM64162 Mode	Input Data Range in MSM64164C Mode	Input Data Range in ML64168 Mode
Program Counter	PC	0 to 7DF	0 to FDF	0 to 1FDF
B Register	B	0 to F	0 to F	0 to F
A Register	A	0 to F	0 to F	0 to F
HL Register	HL	0 to FF	0 to FF	0 to FF
XY Register	XY	0 to FF	0 to FF	0 to FF
Carry Flag	CY	0,1	0,1	0,1
Stack Pointer	SP	80 to FF	0 to FF	0 to FF
Bank Select Register 0	BSR0	0 to 7	0 to 7	0 to 7
Bank Select Register 1	BSR1	0 to 7	0 to 7	0 to 7
Bank Common Flag	BCF	0,1	0,1	0,1
Bank Enable Flag	BEF	0,1	0,1	0,1
note4 Backup Control Register	BUPCON	0 to 3	0,1	0 to 3
note3 Serial Port Buffer Register	SBUF		0 to FF	0 to FF
note3 Serial Control Register	SCON		0 to F	0 to F
note7 Frequency Control Register	FCON	0,1	0,1	0,1
Buzzer Control Register	BDCON	0 to F	0 to F	0 to F
note3 Buzzer Frequency Control Register	BFCON	0,1	0 to F	0 to F
Capture Control Register	CAPCON	0 to F	0 to F	0 to F
note5 Time Base Counter Register	TBCR	0 to F	0 to F	0 to F
Display Control Register	DSPCON	0 to 2	0 to 2	0 to 2
A/D Converter Control Register 0	ADCON0	0 to 3	0 to 3	0 to 3
note8 A/D Converter Control Register 1	ADCON1	0 to F	0 to F	0 to F
note6 Counter A Register	CNTA	0 to 79999	0 to 79999	0 to 79999
Counter B Register	CNTB	0 to 3FFF	0 to 3FFF	0 to 3FFF
Port 1 Register	P1D	0 to F	0 to F	0 to F
Port 2 Register	P2D	0 to F	0 to F	0 to F
Port 3 Register	P3D	0 to F	0 to F	0 to F
note3 Port 4 Register	P4D		0 to F	0 to F
Port 20 Counter Register	P20CON	0 to F	0 to F	0 to F
Port 21 Counter Register	P21CON	0 to F	0 to F	0 to F
Port 22 Counter Register	P22CON	0 to F	0 to F	0 to F
Port 23 Counter Register	P23CON	0 to F	0 to F	0 to F

Table 3-3(b) List of registers mnemonics

Register Name	Mnemonic	Input Data Range in MSM64162 Mode	Input Data Range in MSM64164C Mode	Input Data Range in ML64168 Mode
Port 30 Counter Register	P30CON	0 to F	0 to F	0 to F
Port 31 Counter Register	P31CON	0 to F	0 to F	0 to F
Port 32 Counter Register	P32CON	0 to F	0 to F	0 to F
Port 33 Counter Register	P33CON	0 to F	0 to F	0 to F
note3 Port40 Counter Register	P40CON		0 to F	0 to F
note3 Port41 Counter Register	P41CON		0 to F	0 to F
note3 Port42 Counter Register	P42CON		0 to F	0 to F
note3 Port43 Counter Register	P43CON		0 to F	0 to F
Port01 Counter Register	P01CON	0 to 7	0 to 7	0 to 7
note4 Interrupt Enable Register 0	IE0	0 to F	0 to F	0 to F
Interrupt Enable Register 1	IE1	0 to F	0 to F	0 to F
Interrupt Enable Register 2	IE2	0,1	0,1	0,1
note4 Interrupt Request Register 0	IRQ0	0 to F	0 to F	0 to F
Interrupt Request Register 1	IRQ1	0 to F	0 to F	0 to F
Interrupt Request Register 2	IRQ2	0 to 3	0 to 3	0 to 3
Master Interrupt Enable Flag	MIEF	0,1	0,1	0,1

■ Note3 ■

In MSM64162 mode, the mnemonics *SCON*, *SBUF*, *P4D*, and *P40CON~P43CON* are invalid.

■ Note4 ■

The bit configurations in MSM64162 mode and MSM64164C, ML64168 mode differ. Refer to the chip's user's manuals for details.

■ Note5 ■

When *TBCR* is changed, it will be reset to 0 regardless of the change data specified.

■ Note6 ■

The change data for *CNTA* is a decimal value.

■ Note7 ■

The *FCON* register does not exist in the MSM64162D chip.

■ Note8 ■

If invalid data (5,6 or 7) is written to the *ADCON1* register when evaluating a MSM64162D, then the emulator may operate incorrectly.

Execution Example

```
* CA
  A : 0 OLD ---> 8

* CHL E4

* DHL
  HL : E4
```

Change Cycle Counter

CCC

Input Format

CCC [-] *number*
data : 0-4294967295

Description

The CCC command changes the cycle counter contents to the value indicated by *number*. The *number* should be a decimal value 0 to 4,294,967,295. If a minus sign '-' is input before *number*, then the cycle counter will be set to the value of number subtracted from 4,294,967,295.

Execution Example

* CCC 19
CYCLE COUNTER STATUS : 0000000019

* CCC -1
CYCLE COUNTER STATUS : 4294967294

Display/Change Clock Mode

CCLK

Input Format

CCLK [*mnemonic*]
mnemonic : HIN, HOUT, LIN, LOUT

Description

The CCLK command switches the clock supplied to the evaluation board. One of the following is entered for mnemonic.

HIN : High-speed clock on CROSC board (MSM64162,MSM64164C mode) or 700kHz crystal oscillator (ML64168 mode).
HOUT : High-speed clock from user cable OSC1 pin.
LIN : Low-speed clock on crystal board.
LOUT : Low-speed clock from user cable XT pin.

The EASE64168 clock will be set to internal clocks (HIN, LIN) when power is turned on (note1).

If mnemonic is omitted, then the current setting will be displayed.

■ Note1 ■

Refer to Section 2.1 , “Setting Operating Frequency,” regarding the crystal board, CROSC board, and their peripheral circuits.

■ Note2 ■

The high-speed clock function does not exist in the MSM64162D chip. Please keep this in mind when evaluating the MSM64162D with EASE64168.

Execution Example

```
* CCLK
HIGH CLOCK ---> IN
LOW CLOCK ---> IN

* CCLK HOUT

* CCLK
HIGH CLOCK ---> OUT
LOW CLOCK ---> IN
```

Change Code Memory

CCM

Execution Example

CCM *address*
address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

Description

The CCM command changes the contents of code memory.

The *address* expresses a code memory address. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

When the carriage return is input, the emulator will output the following message and wait for data input.

LOC = *address old-data* OLD --->

Here *address* is the code memory address at which contents are to be changed. The *old-data* is the current contents of code memory. The user inputs new *data* followed by a carriage return. The *data* is the value to change the contents to. It is in the range 0H to FFH.

LOC = *address old-data* OLD ---> *data* NEW
LOC = *address old-data* OLD ---> input data for next parameter

When the carriage return is input, processing will move to the next parameter. If there is no next parameter, then the CCM command will terminate.

When the emulator is waiting for input data for a change, the following three key inputs are valid in addition to data.

Execution Example

```
* CCM 200
LOC=0200 00 OLD ---> 11 NEW
LOC=0201 00 OLD ---> 23 NEW
LOC=0202 00 OLD ---> E4 NEW
LOC=0203 00 OLD ---> A1 NEW
LOC=0204 00 OLD ---> NOT CHANGE
LOC=0205 00 OLD ---> 4 NEW
```


LOC=0206 00 OLD ---> -
LOC=0205 04 OLD ---> 33 NEW
LOC=0206 00 OLD ---> BB NEW
LOC=0207 00 OLD --->

* DCM 200, 20F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0200	11	23	E4	A1	00	33	BB	00	00	00	00	00	00	00	00	00

Change Data Memory

CDM

Input Format

CDM *address*
address : 780~7FF (MSM64162 mode)
 700~7FF (MSM64164C mode)
 600~7FF (ML64168 mode)

Description

The CDM command changes the contents of data memory.

When the carriage return is input, the emulator will output the following message and wait for data input.

LOC = *address old-data* OLD --->

Here *address* is the data memory address at which contents are to be changed. The *old-data* is the current contents of data memory. The user inputs new data followed by a carriage return. The data is the value to change the contents to. It is in the range 0H to FH.

LOC = *address old-data* OLD ---> *data* NEW
 LOC = *address old-data* OLD ---> input next parameter

When the carriage return is input, processing will move to the next parameter. If there is no next parameter, then the CDM command will terminate.

When the emulator is waiting for input data for a change, the following three key inputs are valid in addition to data.

Execution Example

* DDM 750, 75F

	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
LOC=0750	4	2	0	A	0	8	0	0	0	0	1	3	0	0	1	6

* CDM 750

LOC=0750	6	OLD	---	>	0	NEW
LOC=0751	1	OLD	---	>	5	NEW
LOC=0752	0	OLD	---	>	0	NEW
LOC=0753	0	OLD	---	>	-	

LOC=0752 0 OLD ---> E NEW
LOC=0753 0 OLD ---> A NEW
LOC=0754 3 OLD ---> F NEW
LOC=0755 1 OLD ---> 2 NEW
LOC=0756 0 OLD ---> NOT CHANGE
LOC=0757 0 OLD ---> C NEW
LOC=0758 0 OLD ---> 9 NEW
LOC=0759 0 OLD --->

* DDM 750, 75F

	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
LOC=0750	4	2	0	A	0	8	0	9	C	0	2	F	A	E	5	0

Change Display Register

CDSPR

Input Format

CDSPR [*number*]

Description

The CDSPR command changes the values of the display registers (DSPR00~DSPR30). (note1)

The number range differs for MSM64162 mode and MSM64164C, ML64168 mode. In MSM64162 mode, its range is 00 to 20 (decimal). In MSM64164C, ML64168 mode, its range is 00 to 30 (decimal).

DSPR00 = old-data OLD --->

Here old-data is the current value of the corresponding display register. The inputs new data (data) and a carriage return. The data is a value 0H to FH.

DSPR00 = old-data OLD ---> data NEW

DSPR01 = old-data OLD ---> input data for next parameter

When the carriage return is input, processing moves to the next parameter. If there is no next parameter, then the CDSR command terminates.

If number is specified, then changes will begin from the specified display register.

When the emulator is waiting for input data for a change, the following three key inputs are valid in addition to data.

Execution Example

* CDSPR

```

DSPR00 = 0  OLD  --->  8
DSPR01 = 0  OLD  --->  4
DSPR02 = 0  OLD  --->  F
DSPR03 = 0  OLD  --->  NOT CHANGE
DSPR04 = 0  OLD  --->  -
DSPR03 = 0  OLD  --->  1
DSPR04 = 0  OLD  --->

```

* CDSPR 5

DSPR05 = 0 OLD ---> 7
DSPR06 = 0 OLD ---> A
DSPR07 = 0 OLD --->

Set Target Chip

CHIP

Input Format

CHIP [*mnemonic*]
mnemonic : 64162
64164
64168

Description

The EASE64168 can operate in a MSM64162 mode and a MSM64164C mode and a ML64168 mode (note1). The CHIP command sets the EASE64168 operating mode with mnemonic. If mnemonic is omitted, then the current operating chip mode will be displayed.

The EASE64168 resets its evaluation board after the CHIP command is input. The CHIP command will display the following on the CRT.

```
* CHIP 64162  
  
***** EVA BOARD RESET *****
```

■ **Note1** ■

To evaluate an MSM64162D and MSM64162A, set the chip mode to MSM64162 mode.

Execution Example

```
* CHIP  
MSM64164C MODE  
  
* CHIP 64162  
*** EVA BOARD RESET ***
```

Display/Change Interface Power Supply

CIPS

Input Format

CIPS [*mnemonic*]

Description

The CIPS command changes the interface power supply of the user connector. The mnemonic is one of the following.

INT : Supply the user connector interface power supply from the emulator's internal power supply (5V) (note1).

EXT : Supply the user connector interface power supply from the user connector VDD pin (note2).

When the emulator is turned on, the EASE64168 user connector interface power supply will be supplied from the emulator's internal power supply (5V).

If mnemonic is omitted, then the current setting will be displayed.

■ Note1 ■

The EASE64168 "PORT5V" indicator will light up, and the "PORT3V" indicator will go off.

■ Note2 ■

Supply a voltage from 3V to 5V to the user connector VDD pin. The EASE64168 "PORT3V" indicator will light up, and the "PORT5V" indicator will go off.

Execution Example

```
* CIPS
INTERFACE POWER SUPPLY ---> INTERNAL
```

```
* CIPS EXT
```

```
* CIPS
INTERFACE POWER SUPPLY ---> EXTERNAL
```

Change Trace Data Memory

CTDM

Input Format

CTDM [, address]
address : 780~7FF (MSM64162 mode)
 700~7FF (MSM64164C mode)
 600~7FF (ML64168 mode)

Description

The CTDM command sets the address to trace. The address is the address to trace. It is a value 780H to 7FFH when in MSM64162 mode, or 700H to 7FFH when in MSM64164C mode, or 600H to 7FFH when in ML64168 mode. If it is omitted, then the emulator will output the following message and wait for data input.

TRACE DATA MEMORY ADDRESS : old-data OLD --->

Here old-data is the data memory address currently set. The operator inputs a new address and a carriage return. The new address should be a value 780H to 7FFH when in MSM64162 mode, or 700H to 7FFH when in MSM64164C, or 600H to 7FFH when in ML64168 mode. If a carriage return only is input, then the CTDM command will terminate without changing the data address.

■ Note1 ■

If a data memory match break is specified as a break parameter of the G command, then the object of the match will be data memory address specified with the CTDM command.

Execution Example

* CTDM
TRACE DATA MEMORY ADDRESS : 700 OLD ---> 790 NEW

* DTDM
TRACE DATA MEMORY ADDRESS ---> 790

Change Trace Object

CTO

Input Format

CTO

Description

The CTO command selects one of three sets of trace objects traced in 8 bits of trace memory. These trace objects are listed below.

- BCF, BSR0, BEF, BSR1
- P4, P0
- P4, P1

The CTO command sets the trace objects. When the carriage return is input, the emulator outputs the following message and waits for data.

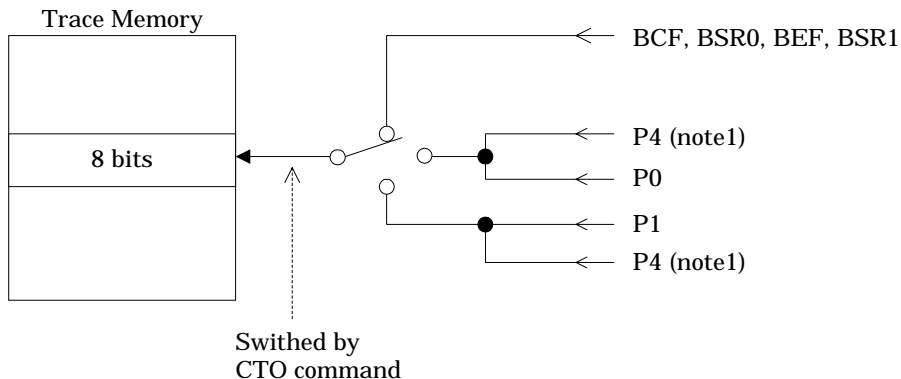
- (1) BCF, BSR0, BEF, BSR1
 - (2) P4, P0
 - (3) P4, P1
- TRACE OBJECT --->

Here the user inputs a 1 or 3, followed by a carriage return.

■ Note1 ■

When the trace objects are changed with the CTO command, the TP (trace pointer) will be reset to '0'.

After a system reset, the trace objects will be set to BCF, BSR0, BEF, and BSR1.



■ **Note2** ■

In MSM64162 mode P4 (port4) data is not traced and is not displayed by the DTO command.
P4 is also not displayed by the CTO command.

Execution Example

* CTO

(1) BCF, BSR0, BEF, BSR1

(2) P4,P0

(3) P4.P1

TRACE OBJECT ---> 2

** RESET TRACE POINTER **

* DTO

TRACE OBJECT ---> P4, P0

Display contents of target chip registers

D

Input Format

D [*mnemonic*]
mnemonic : *mnemonic of a register*

Description

The D command displays the contents of the register specified by *mnemonic*. A register *mnemonic* is one of the mnemonics shown in Table 3-3. If no mnemonic is input, then contents of all registers will be displayed.

■ Note1 ■

The M6416x series microcontrollers mask option specifications can set P5 and P6, but their contents cannot be displayed or changed by EASE64168 with a debugger command.

Table 3-3(a) List of registers mnemonics

	Register Name	Mnemonic
	Program Counter	PC
	B Register	B
	A Register	A
	HL Register	HL
	XY Register	XY
	Carry Flag	CY
	Stack Pointer	SP
	Bank Select Register 0	BSR0
	Bank Select Register 1	BSR1
	Bank Common Flag	BCF
	Bank Enable Flag	BEF
note3	Backup Control Register	BUPCON
note2	Serial Port Buffer Register	SBUF
note2	Serial Control Register	SCON
note5	Frequency Control Register	FCON
	Buzzer Control Register	BDCON
note3	Buzzer Frequency Control Register	BFCON
	Capture Control Register	CAPCON
	Capture Register 0	CAPR0
	Capture Register 1	CAPR1

Table 3-3(b). List of registers mnemonics

	Register Name	Mnemonic
	Time Base Counter Register	TBCR
	Display Control Register	DSPCON
	A/D Converter Control Register 0	ADCON0
	A/D Converter Control Register 1	ADCON1
note4	Counter A Register	CNTA
	Counter B Register	CNTB
	Port 0 Register	P0
	Port 1 Register	P1D
	Port 2 Register	P2D
	Port 3 Register	P3D
note2	Port 4 Register	P4D
note3	Port 01 Counter Register	P01CON
	Interrupt Enable Register 0	IE0
	Interrupt Enable Register 1	IE1
	Interrupt Enable Register 2	IE2
note3	Interrupt Request Register 0	IRQ0
	Interrupt Request Register 1	IRQ1
	Interrupt Request Register 2	IRQ2
	Master Interrupt Enable Flag	MIEF

■ Note2 ■

In MSM64162 mode, the mnemonics *SCON*, *SBUF*, *P4D* are invalid.

■ Note3 ■

The bit configurations in MSM64162 mode and MSM64164C, ML64168 mode differ. Refer to the chip's user's manuals for details.

■ Note4 ■

The display data for *CNTA* is a decimal value.

■ Note5 ■

The *FCON* register does not exist in the MSM64162D chip.

Execution Example

```
* D           MSM64164C mode
PC   : 0000   P0      : F      CNTA   : 80000
A    : 0      P1D     : 0      CNTB   : C000
```

B	: 0	P2D	: F	ADCON0	: C
HL	: 00	P3D	: F	ADCON1	: 0
XY	: 00	FCON	: E	IE0	: 0
CY	: 0	BDCON	: 0	IE1	: 0
SP	: FF	BFCON	: 0	IE2	: E
BSR0	: 0	CAPRO	: 0	IRQ0	: 0
BSR1	: 0	CAPR1	: 0	IRQ1	: 0
BCF	: 0	CAPCON	: 0	IRQ2	: C
BEF	: 0	TBCR	: 0	BUPCON	: E
MIEF	: E	DSPCON	: C		
P4D	: F	SCON	: 0	SBUF	: 00

Disassemble Command

DASM

Input Format

DASM *address* [, *address*]
 or
 DASM *

address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

* : disassemble entire address range

Description

The DASM command disassembles the contents of code memory and displays the results on the console (note1).

The address expresses an address in code memory. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when ML64168 mode.

The DASM command can be forcibly terminated by pressing the ESC key.

Display contents are one of the following, depending on input format.

address : Displays the contents of one address.
address, address : Displays the range from the first address to the second address.
 * : Displays the entire area of code memory.

■ Note1 ■

Comments input with the ASM command cannot be displayed with the DASM command.

■ Note2 ■

The entire area of code memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
* DASM 100
   LOC=0100 50C0    LHLI  C0

* DASM 100,103
```

LOC=0100	50C0	LHLI	C0
LOC=0102	6F	LMA	
LOC=0103	00	NOP	

Display Break Condition Register

DBC

Input Format

DBC

Description

The DBC command displays currently specified break conditions.

ALL BREAK CONDITIONS RESET : All break conditions have been canceled.
BREAK POINT BREAK : Breaks on breakpoint bits are set.
TRACE POINTER OVER-FLOW BREAK : Breaks on trace pointer overflow are set.
CYCLE COUNTER OVER-FLOW BREAK : Breaks on cycle counter overflow are set.

Execution Example

```
* DBC
  BREAK POINT BREAK

* SBC
  BREAK POINT BREAK (Y/N)  N
  CYCLE COUNTER OVER-FLOW BREAK (Y/N)  N
  TRACE POINTER OVER-FLOW BREAK (Y/N)  N

* DBC
  ALL BREAK CONDITION RESET
```


Display Break Point Bits

DBP

Input Format

DBP *address* [, *address*]
or
DBP *
address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDFH (ML64168 mode)
* : display entire address range

Description

The DBP command displays the contents of breakpoint bit memory (note1).

The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

The DBP command can be forcibly terminated by pressing the ESC key.

Breaks will be performed at addresses where the breakpoint bit is '1.' Breaks will not be performed at addresses where the breakpoint bit is '0.'

Display contents are one of the following, depending on input format.

address Displays the contents of one address.
address, address Displays the range from the first address to the second address.
* Displays the entire area of breakpoint bit memory (note2).

■ **Note1** ■

Breakpoint bits correspond one-for-one with addresses in code memory. They are used to cause breaks at specified locations in a user program when executed with the G command.

A breakpoint bit is enabled when the breakpoint bit is '1.' However, the only breakpoint bits that can generate breaks are those corresponding to the address of the first byte of an instruction code in the user program.

Breakpoint bits are enabled as realtime emulation break conditions only when "BREAK POINT BREAK" is set as a break condition.

■ Note2 ■

The entire area of breakpoint bit memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

* FBP

* DBP 40, 70

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0040	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0070	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Display Break Status

DBS

Input Format

DBS

Description

The DBS command displays the break condition from realtime emulation.

ESC KEY BREAK	Break on ESC key input.
BREAK STATUS NOT FOUND	System has just been initialized or no program has been executed.
BREAK POINT BREAK	Break on a breakpoint bit.
TRACE POINTER OVER-FLOW BREAK	Break on trace pointer overflow.
CYCLE COUNTER OVER-FLOW BREAK	Break on cycle counter overflow.
ADDRESS MATCH BREAK	Break on address match.
ADDRESS PASS COUNT BREAK	Break on address pass count.
DATA MEMORY PASS COUNT BREAK	Break on data memory pass count.
BA DATA PASS COUNT BREAK	Break on BA register pass count.
N AREA BREAK	Break when address exceeded 7DFH in MSM64162 mode or FDFH in MSM64614 mode, or 1FDFH in ML64168 mode.

Execution Example

```
* G 0,100
RESET TRACE POINTER
*** EMULATION GO ***
** ADDRESS MATCH BREAK **
[ BREAK PC=0100 NEXT PC=0102 ]
[ NEXT TRACE POINTER=0194 ]

* DBS
** ADDRESS MATCH BREAK **
```

Display Cycle Counter

DCC

Input Format

DCC

Description

The DCC command displays the contents of the cycle counter. When the carriage return is entered, the emulator will output the following message.

CYCLE COUNTER STATUS : *number*

The number is the cycle counter value displayed in decimal.

Execution Example

* DCC

CYCLE COUNTER STATUS : 0000000256

Display Code Memory

DCM

Input Format

DCM *address* [, *address*]
 or
 DCM *

address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

* : display entire address range

Description

The DCM command displays the contents of code memory.

The *address* expresses an address in code memory space for which the contents are to be displayed. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

The DCM command can be forcibly terminated by pressing the ESC key.

Display contents are one of the following, depending on input format.

address Displays the contents of one address.
address, address Displays the range from the first address to the second address.
 * Displays the entire area of code memory (note1).

■ Note1 ■

The entire area of code memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

* DCM 0,2F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
LOC=0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
LOC=0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

* DCM 1A

LOC=001A 00

Display Cycle Counter Trigger

DCT

Input Format

DCT

Description

The DCT command displays the cycle counter start and stop addresses.

The DCT command outputs the following message when its carriage return is input.

START ADDRESS : *start-address*

STOP ADDRESS : *stop-address*

Here start-address will be the address at which to start cycle counter counting, and stop-address will be the address at which to stop cycle counter counting.

■ **Note1** ■

The cycle counter is not incremented in halt mode.

Execution Example

```
* SCT
START ADDRESS 0000 OLD ---> NOT CHANGE
STOP ADDRESS 0100 OLD ---> 100 NEW

* DCT
START ADDRESS 0000
STOP ADDRESS 0100
```

Display Data Memory

DDM

Input Format

DDM *address* [, *address*]
 or
 DDM *

address : 780~7FF (MSM64162 mode)
 700~7FF (MSM64164C mode)
 600~7FF (ML64168 mode)

* : display entire address range

Description

The DDM command displays the contents of data memory.

The address is a value 780H~7FFFH in MSM64162 mode and 700H~7FFH in MSM64164C mode and 600~7FF in ML64168 mode.

The DDM command can be forcibly terminated by pressing the ESC key.

Display contents are one of the following, depending on input format.

address Displays the contents of one address.
address, address Displays the range from the first address to the second address.
 * Displays the entire area of data memory (note1).

■ Note1 ■

The entire area displayed differs for each chip mode. In MSM64162 mode, the area 780H to 7FFH is displayed. In MSM64164C mode, the area 700H to FFFH is displayed. In ML64168 mode, the area 600H to 7FFH is displayed.

Execution Example

* DDM 780, 79F

	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
LOC=0780	0	1	0	B	0	0	2	2	0	0	0	6	0	0	2	0
LOC=0790	0	1	1	9	1	4	4	0	0	3	1	6	4	4	0	8

* DDM 79F

LOC=079F 0

* DDM 790
LOC=0790 8

Display Display Register

DDSPR

Input format

DDSPR

Description

The DDSPR command displays all display register values (DSPR00~DSOR30) (note1).

Execution Example

* DDSPR -----> MSM64162 mode

	0	1	2	3	4	5	6	7	8	9
DSPR0	8	4	F	1	0	7	A	0	0	0
DSPR1	0	0	0	0	0	0	0	0	0	0
DSPR2	0									

* CHIP 64168

*** EVA BOARD RESET ***

* DDSPR -----> ML64168 mode

	0	1	2	3	4	5	6	7	8	9
DSPR0	0	0	0	0	0	0	0	0	0	0
DSPR1	0	0	0	0	0	0	0	0	0	0
DSPR2	0	0	0	0	0	0	0	0	0	0
DSPR3	0									

■ Note1 ■

In MSM64162 mode, display registers DSPR00~DSPR20 are valid. In MSM64164C, ML64168 mode, display registers DSPR00~DSPR30 are valid.

Display Instruction Executed Bits

DIE

Input Format

DIE *address* [, *address*]
or
DIE *
address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)
* : display entire address range

Description

The DIE command displays the contents of instruction executed bit memory.

Instruction executed bits correspond one-for-one with code memory addresses. While realtime emulation of a user program executes, the instruction executed bits at the same addresses as executed instruction codes will be set to '1.' After realtime emulation, using the DIE command to view the contents of instruction executed memory can show ranges which user program executed.

When a start address is specified with the G command, all instruction executed bit memory will be reset to 0.

The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

The DIE command can be forcibly terminated by pressing the ESC key.

Addresses where the instruction executed bits are '1' indicate addresses where the user program was executed. Addresses where the instruction executed bits are '0' indicate addresses where the user program was not executed.

Display contents are one of the following, depending on input format.

address Displays the contents of one address.
address, address Displays the range from the first address to the second address.
* Displays the entire area of instruction executed bit memory (note1).

■ Note1 ■

The entire area of instruction executed bit memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

* RIE

* G 32, 4E

RESET TRACE POINTER

*** EMULATION GO ***

** ADDRESS MATCH BREAK **

[BREAK PC=004E NEXT PC=004F]

[NEXT TRACE POINTER=0029]

* DIE 20, 5F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0030	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0040	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
LOC=0050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Display Trace Data Memory

DTDM

Input Format

DTDM

Description

The DTDM command displays the data memory address being traced. When the carriage return is input, the emulator will output the following message.

```
TRACE DATA MEMORY ADDRESS ---> address
```

The address is the data memory address being traced.

Execution Example

```
* DTDM  
TRACE DATA MEMORY ADDRESS ---> 0700
```

Display Trace Memory

DTM

Input Format

```
DTM parm
  parm : -numberstep , numberstep
        : numberTP , numberstep
        : *
```

Description

The DTM command displays the contents of trace memory as specified by parm. Trace memory is an 8192 x 64-bit RAM area.

The *number_{step}* indicates the number of steps to display as a decimal number 1-8192. The *number_{step}* indicates the number of steps back from the current trace pointer value (called TP below). The *number_{TP}* indicates the TP value at which to start the trace display as a decimal number 0-8191 (note1).

The * indicates that the contents of TP to TP-1 should be displayed if the trace pointer has overflowed, or the contents of 0 to TP-1 should be displayed if it has not.

Trace memory stores various information from realtime emulation. An operator can debug more efficiently by viewing this information.

As shown below, trace memory is configured as a ring, so during realtime emulation trace memory will be overwritten in order from the oldest contents first.

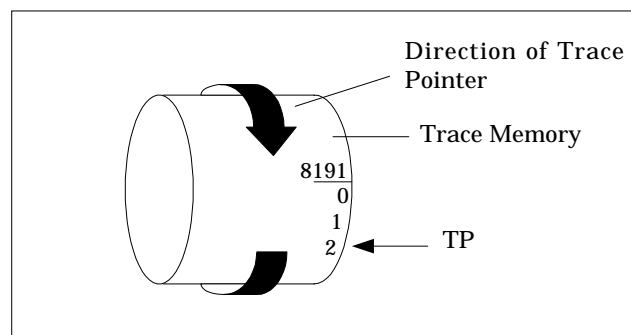
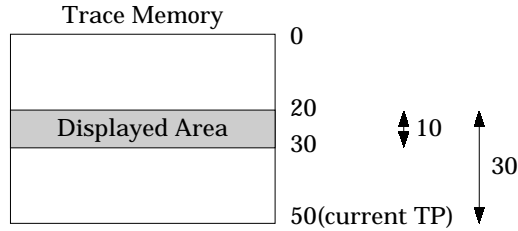


Figure 3-11. Trace Point Example

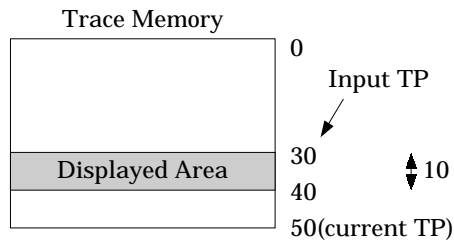
Examples of *-number_{step}* , *number_{step}* input and *number_{TP}* , *number_{step}* input are shown below. Assume that the current TP is 50.

Example

DTM -30 , 10



DTM 30 , 10



After the parameters are correctly input and a carriage return is pressed, a header in the format below will be displayed, followed by the trace memory contents for each trace pointer value.

```
BA HL C SP R(address) P2 P3 BC BE BS01 TP
```

The header is displayed every 10 steps. Trace data is shown as numbers only where it changes. It is displayed as '.' where it has not changed from the previous step. However, the trace data immediately after a header is always displayed as numbers. The address will be displayed as the data memory address specified by the CTDM command (note2).

The above header is the initial display state. It can be changed with the CTO command as shown below.

(1) If BCF, BSR0, BEF, BSR1 are selected

```
BA HL C SP R(address) P2 P3 BC BE BS01 TP
```

(2) If P4, P0 are selected (note3)

```
BA HL C SP R(address) P2 P3 P4 P0 TP
```

(3) If P4, P1 are selected (note3)

BA HL C SP R(address) P2 P3 P4 P1 TP

The trace contents displayed and the corresponding headers are shown below.

B	B register
A	A register
H	H register
L	L register
C	Carry flag
SP	Stack pointer
R(address)	Data memory at address
P2	Port 2
P3	Port 3
BC	BCF flag
BE	BEF flag
BS01	BSR0 register and BSR1 register
P4	Port 4
P0	Port 0
P1	Port 1

Tracing of the BCF flag, BEF flag, BSR0 register, BSR1 register and Port 4, Port0 or Port 4,Port 1 is selected with the CTO command.

■ Note1 ■

Keep in mind the following points when displaying the contents of trace memory.

If trace memory has not overflowed, then trace data will be stored in trace memory from 0 to the current TP. Accordingly, if the input TP or number of back steps is greater than the current TP, then trace memory from 0 will be displayed. If the number of steps input is greater than the number of steps stored in trace memory, then only steps with stored data will be displayed.

If trace memory has overflowed, then trace data will be stored in the entire trace memory (0-8191), regardless of the current TP.

Accordingly, if the number of back steps is greater than the current TP, then data before a TP of 0 (8191, 8190, 8189, ...) will be displayed.

■ Note2 ■

The data memory address to be traced is specified with the CTDM command. Data memory tracing traces write data each time it is written to the specified address in data memory. Usually the value of the upper 4 bits of the data memory trace will be FH, but when data is written with an 8-bit move instruction or 8-bit calculation instruction the full 8 bits of write data will be traced. Data memory tracing will not be performed when the data memory address specified by an

instruction's addressing does not match the address specified by the CTDM command.

■ **Note3** ■

Port 4 will not be displayed if MSM64162 mode is selected.

■ **Note4** ■

When trace data being displayed changes, it is traced with a one-instruction delay.

Execution Example

* CTO

- (1) BCF, BSR0, BEF, BSR1
- (2) P4, P0
- (3) P4, P1

TRACE OBJECT ---> 1

** RESET TRACE POINTER **

* G 0,100

RESET TRACE POINTER

*** EMULATION GO ***

** ADDRESS MATCH BREAK **

[BREAK PC=0100 NEXT PC=0102]

[NEXT TRACE POINTER=0021]

* DTM 0,11

				BA	HL	C	SP	R(700)	P2	P3	BC	BE	BS01	TP
LOC=0000	13	SBC		DA	00	0	EF	DC	F	E	1	0	70	0000
LOC=0001	95	LAI	5	0001
LOC=0002	2D36	LMAD	36	.5	0002
LOC=0004	9A	LAI	A	0003
LOC=0005	2D36	LMAD	36	.A	0004
LOC=0007	2D0F	LMAD	0F	0005
LOC=0009	247C	RMBD	7C,0	0006
LOC=000B	2B31	SMBD	31,1	0007
LOC=000D	287C	SMBD	7C,0	0008
LOC=000F	2809	SMBD	09,0	0009

				BA	HL	C	SP	R(700)	P2	P3	BC	BE	BS01	TP
LOC=0011	A900	JP	0100	DA	00	0	EF	DC	F	F	1	0	70	0010

Display Trace Object

DTO

Input Format

DTO

Description

The DTO command displays the currently set trace objects.

■ Note1 ■

When the trace objects are changed with the CTO command, the TP (trace pointer) will be reset to '0'.

After a system reset, the trace objects will be set to BCR, BSR0, BEF, and BSR1.

Execution Example

```
* DTO  
TRACE OBJECT ---> BCF, BSR0, BEF, BSR1
```

Display Trace Pointer

DTP

Input Format

DTP

Description

The DTP command displays the contents of the current trace pointer (TP). The value is displayed as decimal data.

Execution Example

```
* DTP
TRACE POINTER ---> 5039

* RTP
** RESET TRACE POINTER **

* DTP
TRACE POINTER ---> 0000
```

Display Trace Enable Bits

DTR

Input Format

DTR *address* [, *address*]

or

DTR *

address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

Description

The DTR command displays the contents of trace enable bit memory.

Trace enable bits correspond one-for-one with code memory addresses. When address tracing is selected, the user can control trace execution by manipulating the trace enable bits.

When address tracing is selected and a user program is executed, the emulator examines the trace enable bit at the address of each executed instruction code. If a trace enable bit is '1,' then the trace information at that time will be written to trace memory. Thus, the user can write only the trace information he needs into trace memory by setting the appropriate trace enable bits to '1.'

Only trace enable bits set at the first byte of an instruction code are effective.

The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

The DTR command can be forcibly terminated by pressing the ESC key.

Tracing will be performed at addresses where the trace enable bit is '1.' Tracing will not be performed at addresses where the trace enable bit is '0.'

Display contents are one of the following, depending on input format.

<i>address</i>	Displays the contents of one address.
<i>address, address</i>	Displays the range from the first address to the second address.
*	Displays the entire area of instruction executed bit memory (note1).

■ Note1 ■

The entire area of trace enable bit memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is

0H to 1FDFH.

Display Trace Trigger

DTT

Input Format

DTT

Description

The DTT command displays the trace trigger settings.

The DTT command outputs the following message when its carriage return is input.

```
START ADDRESS : start-address  
STOP ADDRESS  : stop-address
```

Here start-address will be the address at which to start trace execution, and stop-address will be the address at which to stop trace execution.

Execution Example

```
* STT  
START ADDRESS 0000 OLD ---> 100 NEW  
STOP ADDRESS 0200 OLD ---> 200 NEW
```

```
* DTT  
START ADDRESS 0100  
STOP ADDRESS 0200
```

Enable Break Point Bits

EBP

Input Format

EBP address [, address ... address]
address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

Description

The EBP command sets breakpoint bits to '1.'

The address expresses an address to be set to "1." It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. Up to ten address values can be input with one command.

Execution Example

* EBP 60, 68, 6F,73, 79

* DBP 60,7F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0060	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
LOC=0070	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

Enable Trace Enable Bits

ETR

Input Format

ETR address [, address ..., address]

address : 0~7DF (MSM64162 mode)

0~FDF (MSM64164C mode)

0~1FDF (ML64168 mode)

Description

The ETR command sets trace enable bits to '1.'

The address expresses an address to be set to '1.' It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. Up to ten address values can be input with one command.

Execution Example

* ETR 50, 5A, 63, 6F

* DTR 50, 60

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0050	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
LOC=0060	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Terminate the Debugger and Exit to OS

EXIT

Input Format

EXIT

Description

The EXIT command terminates the EASE64X debugger.

If a list file has been opened by the LIST command, then it will be closed before the debugger terminates.

Execution Example

* EXIT

A>

Expand Code Memory

EXPAND

Input Format

EXPAND [*mnemonic*]

Description

The EXPAND command switches the EASE64168 code memory area.

The mnemonic is one of the following.

ON : Make code memory area 16384 bytes (note1).
OFF : Make code memory area 2016 bytes or 4064 bytes or 8192 bytes.

When power is turned on, the EASE64168 code memory area will be set to 8192 bytes (ML64168 mode).

If mnemonic is omitted, then the current setting will be displayed.

■ **Note1** ■

When the code memory area is changed to 16384 bytes, code memory addresses will be expanded to 0H~3FFFH, and command parameter input values will be changed as shown in Table 3-4.

■ **Note2** ■

The setting will be 2016 bytes in MSM64162 mode and 4064 bytes in MSM64164C mode and 8192 bytes in ML64168 mode.

Execution Example

* EXPAND
CODE MEMORY AREA : 4Kbyte

* EXPAND ON
CODE MEMORY AREA : 16Kbyte

* EXPAND
CODE MEMORY AREA : 16Kbyte

Table 3-4 Command Parameter Changes

Command	Parameter
CPC <i>[data]</i>	<i>data</i> : 0H to 3FFFH
DCM <i>address [, address]</i>	<i>address</i> : 0H to 3FFFH
CCM <i>address</i>	<i>address</i> : 0H to 3FFFH
FCM <i>address , address [, data]</i>	<i>address</i> : 0H to 3FFFH
SAV <i>fname [address , address]</i>	<i>address</i> : 0H to 3FFFH
VER <i>fname [address , address]</i>	<i>address</i> : 0H to 3FFFH
ASM <i>address</i>	<i>address</i> : 0H to 3FFFH
DASM <i>address [, address]</i>	<i>address</i> : 0H to 3FFFH
STP <i>[number] [, address]</i>	<i>address</i> : 0H to 3FFFH
G <i>[, address] [, parm]</i>	<i>address</i> : 0H to 3FFFH
DBP <i>address [, address]</i>	<i>address</i> : 0H to 3FFFH
EBP <i>address [, address ..., address]</i>	<i>address</i> : 0H to 3FFFH
FBP <i>address [address [, data]]</i>	<i>address</i> : 0H to 3FFFH
DTR <i>address [, address]</i>	<i>address</i> : 0H to 3FFFH
ETR <i>address [, address ..., address]</i>	<i>address</i> : 0H to 3FFFH
FTR <i>address [, address [, data]]</i>	<i>address</i> : 0H to 3FFFH
DIE <i>address [, address]</i>	<i>address</i> : 0H to 3FFFH
PPR <i>address_{code} , address_{code}, [, address_{EPROM}]</i>	<i>address_{code}</i> : 0H to 3FFFH
TPR <i>address_{code} , address_{code}, [, address_{EPROM}]</i>	<i>address_{code}</i> : 0H to 3FFFH
VPR <i>address_{code} , address_{code}, [, address_{EPROM}]</i>	<i>address_{code}</i> : 0H to 3FFFH
STT START ADDRESS : <i>old-address OLD</i> ---> <i>address</i> STOP ADDRESS : <i>old-address OLD</i> ---> <i>address</i>	<i>address</i> : 0H to 3FFFH
SCT START ADDRESS : <i>old-address OLD</i> ---> <i>address</i> STOP ADDRESS : <i>old-address OLD</i> ---> <i>address</i>	<i>address</i> : 0H to 3FFFH

Fill Break Point Bits

FBP

Input Format

FBP *address* , *address* [, *data*]

or

FBP * [, *data*]

address : 0~7DF (MSM64162 mode)

0~FDF (MSM64164C mode)

0~1FDF (ML64168 mode)

* : display entire address range

data : 0, 1

Description

The FBP command changes the contents of a specified range of breakpoint bit memory.

The address expresses a breakpoint bit memory address. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. The data is a the value of the change data. Its can be '0' or '1.'

The changes are classified by input format as follows.

address, *address*, *data* Fill entire range from first address to second address with the data value.

address, *address* Fill entire range from first address to second address with "0."

* , *data* Fill entire breakpoint bit memory area with the data value.

* Fill entire breakpoint bit memory area with "0." (note1)

■ Note1 ■

The entire area of breakpoint bit memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

* FBP

* DBP 60, 7F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0070	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fill Code Memory

FCM

Input Format

```
FCM address , address [ , data]
or
FCM * [ , data]
address : 0~7DF (MSM64162 mode)
          0~FDF (MSM64164C mode)
          0~1FDF (ML64168 mode)
*       : display entire address range
data    : 0~FF
```

Description

The FCM command changes the contents of code memory.

The *address* expresses a code memory address. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode. The data is a the value of the change data. Its range is 0H to FFH.

The changes are classified by input format as follows.

<i>address, address, data</i>	Fill entire range from first address to second address with the data value.
<i>address, address</i>	Fill entire range from first address to second address with "0."
<i>* , data</i>	Fill entire code memory area with the data value.
<i>*</i>	Fill entire code memory area with "0." (note1)

■ Note1 ■

The entire area of code memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

* FCM 50, 8F, E4

* DCM 50, 6F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0050	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4
LOC=0060	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4	E4

Fill Data Memory

FDM

Input Format

```
FDM  address , address [ , data]
or
FDM  * [ , data]
      address : 780~7FF (MSM64162 mode)
              700~7FF (MSM64164C mode)
              600~7FF (ML64168 mode)
      *       : display entire address range
      data    : 0~F
```

Description

The FDM command changes the contents of data memory.

The *address* expresses a data memory address. The address is a value 780H~7FFFH in MSM64162 mode and 700H~7FFFH in MSM64164C mode and 600H~7FFFH in ML64168 mode. The data is the value of the change data. Its range is 0H to FH.

The changes are classified by input format as follows.

<i>address, address, data</i>	Fill entire range from first address to second address with the data value.
<i>address, address</i>	Fill entire range from first address to second address with "0."
* , <i>data</i>	Fill entire code memory area with the data value.
*	Fill entire code memory area with "0." (note1)

■ Note1 ■

The address is a value 780H~7FFFH in MSM64162 mode and 700H~7FFFH in MSM64164C mode and 600H~7FFFH in ML64168 mode.

Execution Example

```
* FDM 700, 7FF, A
```

```
* DDM 750,76F
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0750	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
LOC=0760	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

Fill Trace Enable Bits

FTR

Input Format

```
FTR address , address [ , data]
or
FTR * [ , data]
  address : 0~7DF (MSM64162 mode)
           0~FDF (MSM64164C mode)
           0~1FDF (ML64168 mode)
  *       : display entire address range
  data    : 0, 1
```

Description

The address expresses a trace enable bit memory address. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. The data is a the value of the change data. Its can be '0' or '1.'

The changes are classified by input format as follows.

```
address, address, data  Fill entire range from first address to second address with the data value.
address, address        Fill entire range from first address to second address with "0."
*, data                 Fill entire trace enable bit memory area with the data value.
*                       Fill entire trace enable bit memory area with "0." (note1)
```

■ Note1 ■

The entire area of trace enable bit memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
* FTR *
```

```
* DTR 30, 60
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0030	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0040	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Realtime Emulation (continuous execution)

G

Input Format

G [, *address*] [, *parm*]
 parm : *address* [, *address* ..., *address*]
 address (*count*)
 : RAM (*data-count*)
 : BAR (*data-count*)
address : 0~7DF (MSM64162 mode)
 (note1) : 0~FDF (MSM64164C mode)
 : 0~1FDF (ML64168 mode)
count : 1~65535
data : 0~FF, X

Description

The G command performs realtime emulation (continuous execution) of a user program in code memory.

The address expresses the first address of the user program at which realtime emulation is to start. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

If the first address is omitted, then realtime emulation will start from the *address* indicated by the current program counter (PC). If a start address is specified for realtime emulation, then both the entire instruction executed memory and the trace pointer will be reset '0.'

The condition that will break realtime emulation is entered in *parm*. There are four break conditions, shown on the next page. If *parm* is omitted, then realtime emulation will continue to execute until a break condition break occurs (note2).

The G command can be forcibly terminated by pressing the ESC key.(note3)

■ Note1 ■

Be sure to input the first address of an instruction within the code memory area for *address*. Breaks will not occur if other addresses are input.

■ Note2 ■

Refer to section , "Break Commands," regarding break conditions.

■ Note3 ■

Breaks by the ESC key are not performed while in halt mode.

(1) Address break (specified as individual addresses)

address [, address , address]

A break will occur when an instruction at any of the addresses specified by address is executed. A maximum of 20 addresses can be entered at one time. The address is a value 0H to 7FDH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. It should be the address of the first byte of an instruction.

(2) Address pass count break

address (count)

A break will occur when the instruction at the address specified by address is executed count times. The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. It should be the address of the first byte of an instruction. The count is a decimal value 1-65535.

(3) Data memory match break

RAM (data-count)

A break will occur when the specified data is written count times to data memory. RAM indicates data memory; the actual data memory address for matching is specified with the CTDM command (note4).

The data is a value 0H to FFH, and can be specified as either 4 bits or 8 bits (note5).

The count is a decimal value 1-65535.

■ **Note4** ■

Refer to Section 3.4.6, "Trace Commands," regarding the CTDM command. If the address specified with the CTDM command does not match the data memory address specified in the addressing of an instruction, then no break will occur.

■ **Note5** ■

The input methods for 4-bit and 8-bit data differ as follows.

8-bit

RAM (3E-16) Break when 3EH is written 16 times to the specified data memory with an 8-bit move instruction or 8-bit calculation instruction.

8-bit (high nibble is FH)

RAM (F5-6) Break when the number of times F5H is written to the specified data memory with an 8-bit move instruction or 8-bit calculation instruction, and the number of times 5H is written with another instruction, totals 6.

4-bit

RAM (X3-10) Break when 3H is written 10 times to the specified data memory. The 'X' indicates a 4-bit input. However, if the data memory address specified with the CTDM command is an odd address, then data writes with 8-bit move instructions or 8-bit calculation instructions will not be counted.

(4) BA register match break

BAR (*data-count*)

A break will occur when the data specified by data is written to the BA register count times. The data is a value 0H to FFH, and can be specified as either 4 bits or 8 bits (note6). The count is a decimal value 1-65535.

■ Note6 ■

The input methods for 4-bit and 8-bit data differ as follows.

8-bit

BAR (1F-5) Break when 1FH is written 5 times to the BA register with an 8-bit move instruction or 8-bit calculation instruction.

4-bit

BAR (X3-3) Break when 3H is written 3 times to the A register. Specification for the B register only is not possible.

When the carriage return is input, the emulator will display the following message.

RESET TRACE POINTER

*** EMULATION GO ***

However, the "RESET TRACE POINTER" message will be output only when a user program start address has been specified.

When this message is output, all instruction executed bits and the trace pointer will be set to '0,' and execution will begin.

There are two ways to break once realtime emulation of a program is begun by a G command. The first is to specify parameters with the command input, as described above. The second is to use

the break condition register. G command break conditions are listed below.

- (1) Break when ESC key is pressed.
- (2) Break when one of the conditions specified by parm in G command input is satisfied.
- (3) Break when the following break conditions are enabled.
 - (a) Break upon execution of an address at which the breakpoint bit is set to '1.'
 - (b) Break when the trace pointer overflows.
 - (c) Break when the cycle counter overflows.
- (4) Break when the execution address exceeds 07DFH in MSM64162 mode or 0FDFH in MSM64164C mode, or 1FDFH in ML64168 mode. If one of the above conditions is satisfied, then the emulator will display the following message after the

***** Break Status *****
[Break PC = *Break-address* Next PC = *Next - address*]

instruction at the address that caused the break condition is executed.

The Break Status is one of the break conditions.

■ **Reference** ■

DBS command

The Break-address is the address of the user program where the realtime emulation break occurred. The Next-address is the first address of the instruction that is to be executed after the Break-address. The Trace-Pointer is the trace pointer value at the point the break occurred.

The Break-address and Next-address are hexadecimal data. The Trace-Pointer is decimal data.

■ **Note7** ■

When a break condition is fulfilled during a skip, the break will be saved. The break will be performed after the skip completes.

However, if an instruction at a break address set as an address break or breakpoint break is skipped, then the break will not be saved. No break will be performed after the skip completes.

■ **Note8** ■

If an interrupt is generated when a break condition is fulfilled, then the break will be saved. The break will be performed after the interrupt transfer cycle completes.

■ **Note9** ■

The time base counter value is preserved after a break occurs until execution begins again. However, while operation of timers and counters synchronized to the microprocessor's internal

clock is guaranteed, operation when synchronized to an external clock is not guaranteed.

■ **Note10** ■

When a break occurs in high-speed clock mode, the time base counter value will not be the same as it would for low-speed clock mode even under the same break conditions because the clocks are asynchronous.

■ **Note11** ■

If a warning message (Warning 2) is displayed after a G command break, then the duty setting of the LCD driver display control register (DSPCON) is different from the duty setting of the mask options previously loaded. You should verify the duty settings. (When power is first applied, the mask option duty setting is 1/4 duty.)

■ **Note12** ■

With the M6416x series microcontrollers, the skip function of an AIS instruction will be disabled in a program where the AIS instruction is executed following either ADCS and ADCS@XY instructions, and SUBCS and SUBCS@XY instructions. With the EASE64168 emulator, however, if a break is set to occur immediately after execution of either ADCS and ADCS@XY instructions, or SUBCS and SUBCS@XY instructions, and execution is set to resume starting with the AIS instruction, then the skip function of the AIS instruction will not be disabled.

Execution Example

```
* G 0,100
  RESET TRACE POINTER
  *** EMULATION GO ***
  ** ADDRESS MATCH BREAK **
  [ BREAK PC=0100  NEXT PC=0102 ]
  [ NEXT TRACE POINTER=0021 ]

* G 0
  RESET TRACE POINTER
  *** EMULATION GO ***
  ** ESC KEY BREAK **
  [ BREAK PC=010B  NEXT PC=010C ]
  [ NEXT TRACE POINTER=2857 ]

* G 0,16F(15)
  RESET TRACE POINTER
  *** EMULATION GO ***
```

```
** ADDRESS PASS COUNT BREAK **  
[ BREAK PC=016F  NEXT PC=0000 ]  
[ NEXT TRACE POINTER=5520 ]
```

```
* G 0,RAM(3C-2)  
RESET TRACE POINTER  
*** EMULATION GO ***  
** DATA MEMORY PASS COUNT BREAK **  
[ BREAK PC=0108  NEXT PC=010A ]  
[ NEXT TRACE POINTER=1311 ]
```

```
* G 0,BAR(XC-4)  
RESET TRACE POINTER  
*** EMULATION GO ***  
** BA DATA PASS COUNT BREAK **  
[ BREAK PC=0108  NEXT PC=0109 ]  
[ NEXT TRACE POINTER=0259 ]
```

Listing (Redirect the Console output to Disk file)

LIST

Input Format

```
LIST fname
      fname : [pathname] filename [Extension]
```

Description

The LIST command stores the contents displayed to the console in the specified file.

The input file name can have a path specification. If the path is omitted, then the file will be created in the current directory. If a file of the same name exists in the specified directory, then that file will be deleted and a new file will be created.

If the file extension is omitted, then a default extension (LST) will be appended.

While a file is being created by a LIST command, another LIST command cannot be used (only one list file can be opened).

■ **Note1** ■

The LIST command becomes valid immediately after it has been input. When any of the following occurs, the LIST command becomes invalid and the list file is closed.

An NLST command is input.

The EASE64X debugger terminates.

The EASE64168 base unit's reset switch is pressed.

Execution Example

```
* LIST SAMP1
```

```
Listfile : FILENAME.LST opened
```

```
* DTR 0, 30
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0020	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0030	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Load Disk file program into Code Memory

LOD

Input Format

LOD *fname*
fname : [*Pathname*] *filename* [*Extension*]

Description

The LOD command loads the contents of an object file output by SASM64K into code memory. For this command, an object file is an Intel HEX format file generated by SASM64K.

If the extension is omitted, then ".HEX" (Intel HEX format file) will be the default.

The input filename can have a path specification. If the path is omitted, then the file in the current directory will be loaded. If the extension is omitted, then the file with the default extension will be loaded.

■ **Note1** ■

The object file generated by the SASM64K structured assembler includes code information obtained by converting the OLMS-64K instruction mnemonics and directives in the source program file, and symbol information obtained from the symbol definitions in the source program file. Do not specify the /S option for assembly that will generate symbol information because EASE64X does not handle symbol information.

Execution Example

```
LOD T1  
  
FILE OPENED NORMALLY. FILE TYPE : INTELLEC HEX  
  
**** LOAD COMPLETED , NEXT ADDRESS = 0300 ****
```

Load Disk file Mask Option into memory

LODM

Input Format

```
LODM fname  
fname : [Pathname] filename [Extension]
```

Description

The LODM command loads the contents of a mask option file output by M6416x mask option generator into the system controller's system memory. A mask option file is an Intel HEX format file generated by M6416x mask option generator (note1).

If the file extension is omitted, then "HEX" (Intel HEX format file) will be the default.

The input file name can have a path specification. If the path is omitted, then the file in the current directory will be loaded. If the extension is omitted, then the file with the default extension appended will be loaded.

■ **Note1** ■

Refer to the MASK162 User's Manual or MASK164 User's Manual or MASK162A User's Manual or MASK168 User's Manual for details about mask option files.

If a program is executed when no mask option file has been loaded into system memory, then LCD driver display will not be performed correctly.

■ **Note2** ■

If mask option data for 1/2 duty is loaded, then a warning message (Warning 1) will be displayed after the load completes. The EASE64168 emulator cannot output a 1/2-bias waveform when 1/2 duty is specified. For details, refer to Chapter 4, "Debugging Notes".

Execution Example

```
* LODM M164_000  
FILE OPENEND NORMALLY. FILE TYPE : INTELLEC HEX  
***** LOAD COMPLETED *****
```


No Listing (Cancel the Console output Redirection)

NLST

Input Format

NLST

Description

The NLST command terminates a previous LIST command. It will close the list file opened by the LIST command.

Contents are stored in the list file until the NLST command.

Execution Example

* NLST

Listfile : SAMP1.LST closed

Pause Command Input

PAUSE

Input Format

PAUSE

Description

The PAUSE command waits for keyboard input when executed. By placing a PAUSE command in a batch file, automatic command execution can be temporarily suspended. The input wait state will be released upon input from the keyboard, or if the emulator reset switch is pressed.

Execution Example

* PAUSE

* PAUSE

Low-Power series Emulator << EASE64168>> Ver 1.00

Program EPROM

PPR

Input Format

```
PPR addresscode, addresscode [ , addressEPROM]  
or  
PPR *  
addresscode : 0~7DF (MSM64162 mode)  
              : 0~FDF (MSM64164C mode)  
              : 0~1FDF (MSM64168 mode)  
*              : writes entire address range  
addressEPROM : EPROM write start address
```

Description

The PPR command writes the contents of the specified code memory area to the EPROM starting at the specified *address_{EPROM}*.

Each *address_{code}* is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168. The *address_{code}*, *address_{code}* specifies the range of code memory to be written. If an '*' is input, then a range of code memory that corresponds to the EPROM type will be set (note1).

The *address_{EPROM}* is the EPROM's starting address for writing. If this address is omitted, then writing will start from EPROM address 0.

Input continues until a carriage return is entered. Then the following message will be output.

```
EPROM TYPE ---> type  
START PROGRAMMING [Y/N] ---> _
```

Here *type* indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to write, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data write will start. If the data write completes normally, then the "RUN" indicator will go off, the PPR command will terminate, and the emulator will wait for another command input.

■ Note1 ■

The range of code memory written to EPROM when '*' is input changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in

ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
* PPR 0, 1FF, 0
  EPROM TYPE ---> 27512
  START PROGRAMMING [Y/N] ---> Y

* TYPE 128

* PPR *, 0
  EPROM TYPE ---> 27128
  START PROGRAMMING [Y/N] ---> Y
```

Program Mask Option Data into EPROM

PPRM

Input Format

PPRM

Description

The PPRM command writes an EPROM with the mask option data in the system controller's system memory. The EPROM address range to be written is always from 0H to BFFH.

When the carriage return is entered after the above input format, the emulator will output the following message.

```
EPROM TYPE ---> type
START PROGRAMMING [Y/N] ---> _
```

Here *type* indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to write, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data write will start. If the data write completes normally, then the "RUN" indicator light will go off, the PPRM command will terminate, and the emulator will wait for another command input.

Execution Example

```
* PPRM
EPROM TYPE ---> 27512
START PROGRAMMING [Y/N] --->Y
```

Reset Break Point Bits

RBP

Input Format

RBP *address* [, *address* ..., *address*]
address : 0~7DF (MSM64162 mode)
 : 0~FDF (MSM64164C mode)
 : 0~1FDF (ML64168 mode)

Description

The RBP command resets breakpoint bits to '0.'

The address expresses an address to be set to "0." It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. Up to ten address values can be input with one command.

Execution Example

* FBP 40, 55, 1

* DBP 40, 7F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0040	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0050	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0070	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* RBP 4A, 50

* DBP 40, 7F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0040	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
LOC=0050	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0070	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset Cycle Counter Trigger

RCT

Input Format

RCT

Description

The RCT command disables the cycle counter start and stop addresses and stops cycle counter counting.

The RCT command cancels the start and stop addresses set by the previous SCT command when its carriage return is input. However, the start address and stop address set by that SCT command will be saved. Later when another SCT command is input, if just a carriage return is input, then cycle counter counting will be performed from the saved start address to the saved stop address.

Execution Example

* RCT

Reset Instruction Executed Bits

RIE

Input Format

RIE

Description

The RIE command resets the entire contents of instruction executed bit memory to '0'.

Execution Example

* RIE

* DIE 10, 3F

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOC=0030	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset the System

RST

Input Format

RST

Description

The RST command resets the EASE64168 as follows.

Cycle counter	Reset to 0.
Break status register	Invalidate all break status.
Trigger trace start address	Disabled.
Trigger trace stop address	Disabled.
Trace mode	Set to address trace mode.
Cycle counter start address	Disabled.
Cycle counter stop address	Disabled.
Instruction executed memory	All reset to 0.
Evaluation board	Reset to same as when M6416x series microcontrollers is reset.

Execution Example

```
* RST
Low-Power Series Emulator << EASE64168 >> Ver 1.00
```

Reset the Evaluation chip

RST E

Input Format

RST E

Description

The RST E command resets the evaluation board.

After this command is executed, the evaluation board will be reset to the same state as when the M6416x series microcontrollers is reset. For details about the state after reset, refer to the user's manual of the chip.

Execution Example

```
* RST E
**** EVA BOARD RESET ****
```

Reset Trace Pointer

RTP

Input Format

RTP

Description

The RTP command clears the trace pointer value to 0. The trace pointer is also initialized to 0 when power is turned on, when a start address is specified with a G command, or when the trace objects are changed with the CTO command.

Execution Example

```
* RTP
**  RESET TRACE POINTER  **

* DTP
TRACE POINTER ---> 0000
```

Reset Trace Enable Bits

RTR

Input Format

```
RTR address [ , address ..., address]
address : 0~7DF (MSM64162 mode)
         : 0~FDF (MSM64164C mode)
         : 0~1FDF (ML64168 mode)
```

Description

The RTR command resets trace enable bits to '0.'

The address expresses an address to be set to '0.' It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. Up to ten address values can be input with one command.

Execution Example

```
* FTR 40, 5F
```

```
* DTR 40, 60
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0040	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0050	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
* RTR 40, 5A
```

```
* DTR 40, 60
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LOC=0040	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
LOC=0050	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
LOC=0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset Trace Trigger

RTT

Input Format

RTT

Description

There are two conditions for executing a trace.

(1) Address tracing

With address tracing, tracing is performed upon execution of addresses where the trace enable bit is set to '1.' The trace enable bit must be set at the address of the first byte of each instruction to be traced (note2).

(2) Trigger tracing

When the STT command sets a trace start address and trace stop address, the trace start bit and trace stop bit at those respective addresses will be set to '1.' With trigger tracing, tracing starts when an address with the trace start bit set to '1' is passed. Tracing then stops when an address with the trace stop bit set to '1' is passed (note3).

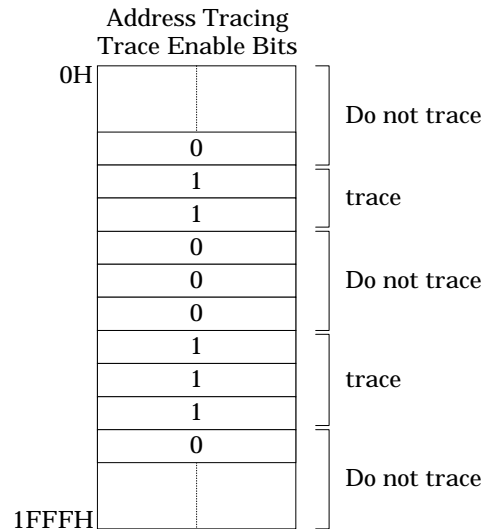
The RTT command cancels trigger tracing, and enables address tracing.

Selection of address tracing or trigger tracing is performed with the STT and RTT commands.

Select address tracing --- Execute RTT command, disabling trigger tracing.

Select trigger tracing --- Execute STT command, enabling trigger tracing.

The concept of address tracing is shown below.



■ **Note1** ■

Address tracing will be selected when the system is reset.

■ **Note2** ■

Trace enable bits correspond one-for-one with addresses in code memory. They enable tracing when address tracing is being executed. Trace enable bits need to be set to '1' at the address of the first byte of each instruction to be traced.

■ **Note3** ■

Trace start bits and trace stop bits both correspond one-for-one with addresses in code memory. They set the start and stop addresses for tracing when trigger tracing is being executed. Trace start bits and trace stop bits need to be set to '1' at the address of the first byte of their respective instructions. The contents of the address where the trace start bit is '1' will be traced, but the contents of the address where the trace stop bit is '1' will not be traced. If the start address set with the G command is identical to the trace start address, then tracing will start when the trace start address is passed the second time.

The RTT command cancels trigger tracing and enables address tracing when its carriage return is input. However, if an STT command has been executed before the RTT command input, then the start address and stop address set by that STT command will be saved. Later when another STT command is input, if just a carriage return is input, then trigger tracing will be enabled and tracing will execute from the saved start address to the saved stop address.

Extraction Example

* RTT

Save Code Memory into Disk file

SAV

Input Format

```
SAV  fname [address, address]
      fname  : [Pathname] filename [Extension]
```

Description

The SAV command saves the contents of the specified range of code memory to a disk file.

The input filename can have a path specification. If the path is omitted, then a file in the current directory will be saved. If the extension is omitted, then the default extension (HEX) will be appended to the file.

The address, address represents the area of code memory to be saved. If omitted, then the entire code memory area will be saved (note1).

■ **Note1** ■

The entire area of code memory changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
SAV T1CH 0,2FF
***** SAVE COMPLETED *****
```

Set Break Condition Register

SBC

Input Format

SBC

Description

The SBC command sets break conditions.

When the carriage return is input, input mode will be entered for each break condition.

BREAK POINT BREAK (Y/N)_

The operator sets or cancels each break condition by entering a 'Y' or 'N' at the underscore.

BREAK POINT BREAK (Y/N) Y
 CYCLE COUNTER OVER-FLOW BREAK (Y/N) N
 TRACE POINTER OVER-FLOW BREAK (Y/N) --- Input next parameter

When each carriage return is input, processing moves to the next parameter. If there is no next parameter, then the SBC command will terminate.

When the emulator is waiting for input data for a change, the following two key inputs are valid.

Execution Example

```
* SBC
  BREAK POINT BREAK (Y/N) Y
  CYCLE COUNTER OVER-FLOW BREAK (Y/N) Y
  TRACE POINTER OVER-FLOW BREAK (Y/N) Y
```

```
* DBC
  BREAK POINT BREAK
  CYCLE COUNTER OVER-FLOW BREAK
  TRACE POINTER OVER-FLOW BREAK
```

```
* SBC
  BREAK POINT BREAK (Y/N) N
  CYCLE COUNTER OVER-FLOW BREAK (Y/N) N
  TRACE POINTER OVER-FLOW BREAK (Y/N) N
```


* DBC
ALL BREAK CONDITION RESET

Set Cycle Counter Trigger

SCT

Input Format

SCT

Description

The SCT command specifies the addresses where cycle counter counting is to start and stop. This command allows program execution time to be measured by incrementing the cycle counter during G command execution (note1).

The cycle counter is a 32-bit binary counter, so it can count up to a maximum of 4,294,967,295. Program execution breaks on cycle counter overflow are also possible.

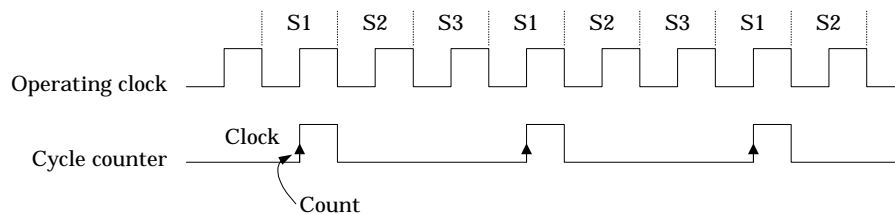
When the SCT command sets a cycle counter start address and cycle counter stop address, the cycle counter start bit and cycle counter stop bit at those respective addresses will be set to '1.' After G command program execution starts, cycle counting starts when an address with the cycle counter start bit set to '1' is passed. Cycle counting then stops when an address with the cycle counter stop bit set to '1' is passed (note2).

■ **Note1** ■

The cycle counter is incremented each machine cycle. Program execution time can be calculated with the following formula.

Program execution time = $1/\text{frequency} \times 3 \times \text{cycle counter value}$

Below is a timing diagram of cycle counter counting.



■ **Note2** ■

Cycle counter start bits and cycle counter stop bits both correspond one-for-one with addresses in code memory. They set the start and stop addresses for cycle counter counting. Cycle counter start bits and cycle counter stop bits need to be set to '1' at the address of the first byte of their respective instructions. The cycle counter will not be incremented at the address where

the cycle counter stop bit is set to '1.'

■ **Note3** ■

The cycle counter is not incremented in halt mode.

When the carriage return of an SCT command is input, the emulator will output the following message and wait for input.

START ADDRESS : old-address OLD --->

Here old-address is the currently set cycle counter start address. The operator inputs the new address at which cycle counter counting is to start, followed by a carriage return. The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. When the carriage return is input, the emulator moves to input mode for the next parameter.

After input of the start address is complete, the emulator outputs the following message and waits for data input.

STOP ADDRESS : old-address OLD --->

Here old-address is the currently set cycle counter stop address. The operator inputs the new address at which cycle counter counting is to stop, followed by a carriage return. The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

When the SCT command terminates, the cycle counter start bit will be set to '1' at the address set as the start address, the cycle counter stop bit will be set to '1' at the address set as the stop address, and then the emulator will wait for the next command to be input. Both the start address and stop address need to be set at the first byte of an instruction code.

After execution of an SCT command, the cycle counter settings will remain valid until an RCT command is executed.

Execution Example

```
* SCT
  START ADDRESS 0000 OLD ---> NOT CHANGE
  STOP ADDRESS 0000 OLD ---> 100 NEW

* DCT
  START ADDRESS 0000
  STOP ADDRESS 0100
```

Step Execution

STP

Input Format

STP [*count*] [, *address*]
or
STP *

address : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)

count : 1~65535

* : execute 65535 steps

Description

The STP command executes a user program in code memory one instruction at a time.

The address expresses the first address of the user program at which step execution is to start. It is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

If address is omitted, then step execution will start from the address indicated by the current program counter (PC). If "*" is input, then 65535 steps will be executed from the current program counter (PC).

The count is a decimal value from 1 to 65535. It indicates the number of steps to be executed. If count is omitted, then step execution will be performed for just one instruction and the command will terminate.

The STP command stops user program execution after each instruction. At each stop, it displays the address and mnemonic of the executed instruction, and then displays the states of the registers and ports after execution. The STP command does not display instructions skipped by the skip instructions.

When the condition for skipping an instruction is met (multiply-accumulate instruction, increment instruction, etc.), the step ends after skipping the next instruction.

The STP command can be forcibly terminated by pressing the ESC key (note1).

■ Note1 ■

Termination by the ESC key cannot be performed while in halt mode.

When the carriage return is input, the emulator will display the following header, followed by register and port values for each step.

B A H L X Y C P0 P1D P2D P3D P4D SP (note2)

The header is displayed every 10 steps. Register and port data are shown as numbers only where they change. They are displayed as '.' where they have not changed from the previous step. However, the data immediately after a header is always displayed as numbers (note3).

The register and port contents displayed and the corresponding headers are shown below.

B	B register
A	A register
H	H register
L	L register
X	X register
Y	Y register
C	Carry flag
P0	Port 0 register
P1D	Port 1 register
P2D	Port 2 register
P3D	Port 3 register
P4D	Port 4 register
SP	Stack pointer

■ **Note2** ■

P4D is not displayed in MSM64162 mode.

■ **Note3** ■

Values are displayed for registers and ports after each instruction is executed.

■ **Note4** ■

When an AIS instruction is executed after an ADCS or SUBCS instruction, the skip function of the AIS instruction is not allowed. However, when executed as a single instruction with the STP command, the skip function of the AIS instruction will no longer be disallowed.

■ **Note5** ■

The time base counter value is preserved between instructions even with the STP command. However, while operation of timers and counters synchronized to the microprocessor's internal clock is guaranteed, operation when synchronized to an external clock is not guaranteed.

Execution Example

* STP 3,0

				B	A	H	L	X	Y	C	P0	P1D	P2D	P3D	P4D	SP
LOC=0000	13	SBC		0	1	0	2	0	0	0	F	0	F	F	F	FF
LOC=0001	2D0F	LMAD	0F00
LOC=0003	95	LAI	5	.	5

* STP 5

				B	A	H	L	X	Y	C	P0	P1D	P2D	P3D	P4D	SP
LOC=0004	2D36	LMAD	36	0	5	0	2	0	0	0	F	0	F	F	F	FF
LOC=0006	9A	LAI	A	.	A
LOC=0007	2D36	LMAD	36
LOC=0009	247C	RMBD	7C,0
LOC=000B	2B31	SMBD	31,3

Set Trace Trigger

STT

Input Format

STT

Description

The STT command sets the trace start address and trace stop address for trigger tracing.

There are two conditions for executing a trace.

(1) Address tracing

With address tracing, tracing is performed upon execution of addresses where the trace enable bit is set to '1.' The trace enable bit must be set at the address of the first byte of each instruction to be traced (note1).

(2) Trigger tracing

When the STT command sets a trace start address and trace stop address, the trace start bit and trace stop bit at those respective addresses will be set to '1.'

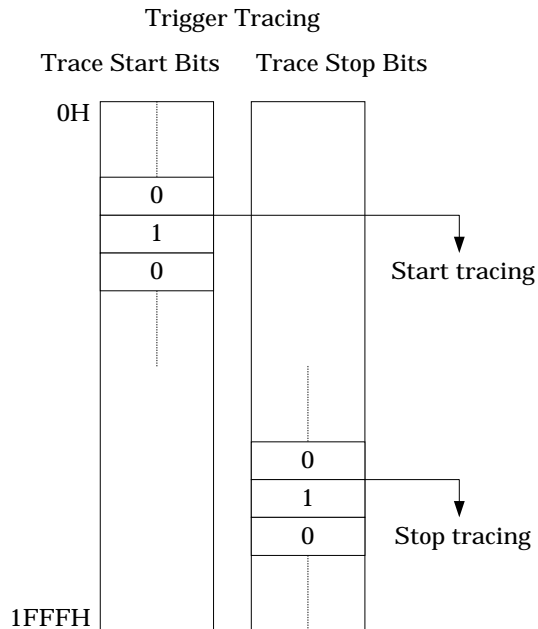
With trigger tracing, tracing starts when an address with the trace start bit set to '1' is passed. Tracing then stops when an address with the trace stop bit set to '1' is passed (note2).

Selection of address tracing or trigger tracing is performed with the STT and RTT commands.

Select address tracing --- Execute RTT command, disabling trigger tracing.

Select trigger tracing --- Execute STT command, enabling trigger tracing.

The concept of trigger tracing is shown below.



■ **Note1** ■

Address tracing will be selected when the system is reset.

■ **Note2** ■

Trace enable bits correspond one-for-one with addresses in code memory. They enable tracing when address tracing is being executed. Trace enable bits need to be set to '1' at the address of the first byte of each instruction to be traced.

■ **Note3** ■

Trace start bits and trace stop bits both correspond one-for-one with addresses in code memory. They set the start and stop addresses for tracing when trigger tracing is being executed. Trace start bits and trace stop bits need to be set to '1' at the address of the first byte of their respective instructions. The contents of the address where the trace start bit is '1' will be traced, but the contents of the address where the trace stop bit is '1' will not be traced. If the start address set with the G command is identical to the trace start address, then tracing will start when the trace start address is passed the second time.

When the carriage return of an STT command is input, the emulator will output the following message and wait for input.

START ADDRESS : *old-address* OLD --->

Here *old-address* is the currently set trace start address. The operator inputs the new address at which trace execution is to start, followed by a carriage return.

The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. When the carriage return is input, the emulator moves to input mode for the next parameter.

After input of the start address is complete, the emulator outputs the following message and waits for data input.

```
STOP ADDRESS : old-address OLD --->
```

Here *old-address* is the currently set trace stop address. The operator inputs the new address at which trace execution is to stop, followed by a carriage return.

The address is a value 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode.

When the STT command terminates, the trace start bit will be set to '1' at the address set as the start address, the trace stop bit will be set to '1' at the address set as the stop address, and then the emulator will wait for the next command to be input. Both the start address and stop address need to be set at the first byte of an instruction code.

After execution of an STT command, the trace triggers will remain valid until an RTT command is executed.

Execution Example

```
* STT
  START ADDRESS 0000 OLD ---> 100 NEW
  STOP ADDRESS  0100 OLD ---> 200 NEW

* DTT
  START ADDRESS 0100
  STOP ADDRESS  0200
```

Transfer EPROM into Code Memory

TPR

Input Format

```
TPR  addresscode , addresscode [ , addressEPROM ]
or
TPR  *
```

address_{EPROM} : EPROM address
address_{code} : 0~7DF (MSM64162 mode)
 0~FDF (MSM64164C mode)
 0~1FDF (ML64168 mode)
 * : transferred entire address range

Description

The TPR command reads EPROM contents in the specified range and transfers them to the specified code memory area.

Each address_{code} represents a code memory address. It is a value 0H to 7DFH in MSM64162 mode or 0H to FDFH in MSM64164C mode, or 0H to 1FDFH in ML64168 mode (note1). The address_{code} , address_{code} specifies the range of code memory to be transferred.

The address_{EPROM} is the starting address in EPROM to be read. If this address is omitted, then reading will start from EPROM address 0. If an '*' is input, then the entire area of the EPROM from address 0 will be transferred to code memory (note2).

Input continues until a carriage return is entered. Then the following message will be output.

```
EPROM TYPE ---> type
START READING [Y/N] ---> _
```

Here type indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to read, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data transfer will start. If the data transfer completes normally, then the "RUN" indicator will go off, the TPR command will terminate, and the emulator will wait for another command input.

■ **Note1** ■

The valid address range for each EPROM type is shown below.

EPROM Type	address range
2764	0~1FFF
27128	0~3FFF
27256	0~7FFF
27512	0~FFFF

■ **Note2** ■

The range of code memory transferred from EPROM when '*' is input changes with the mode. When in MSM64162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
* TPR 0,2FF,0
EPROM TYPE ---> 27512
START READING [Y/N] ---> Y
```

```
* TPR *
EPROM TYPE ---> 27512
START READING [Y/N] ---> Y
```

Transfer EPROM into System Memory

TPRM

Input Format

TPRM

Description

The TPRM command transfers mask option data on an EPROM into the system controller's system memory. The EPROM address range to be transferred is always from 0H to BFFH.

When the carriage return is entered after the above input format, the emulator will output the following message.

```
EPROM TYPE ---> type
START READING [Y/N] ---> _
```

Here *type* indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to write, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data transfer will start. If the data transfer completes normally, then the "RUN" indicator light will go off, the TPRM command will terminate, and the emulator will wait for another command input.

■ **Note1** ■

If mask option data for 1/2 duty is transferred, then a warning message (Warning1) will be displayed after the load completes. The EASE64168 emulator cannot output a 1/2-bias waveform when 1/2 duty is specified. For details, refer to Chapter 4, "Debugging Notes".

Execution Example

```
* PPRM
EPROM TYPE ---> 27512
START READING [Y/N] ---> Y
```

Set EPROM type

TYPE

Input Format

```
TYPE [ , parm]  
parm : mnemonic
```

Description

The TYPE command sets the type of EPROM that will be used in the EPROM programmer. The mnemonic indicates the EPROM type.

Usable EPROM types are one of the following.

Intel products and other EPROMs that are written at high speed with the Intelligent Programming method.

The following can be input for mnemonic.

EPROM Type	mnemonic
2764	64
27128	128
27256	256
27512	512

If mnemonic is omitted, then the currently set EPROM type will be displayed. The setting will be "27512" after power is turned on.

Execution Example

```
* TYPE  
EPROM TYPE ---> 27512
```

```
* TYPE 256
```

```
* TYPE  
EPROM TYPE ---> 27256
```

Set User Reset Terminal (on user connector)

URST

Input Format

URST [mnemonic]

Description

The URST command sets whether the $\overline{\text{RESET}}$ pin input of the user connector is enabled or not.

One of the following parameters is entered for mnemonic.

ON : Inputs from $\overline{\text{RESET}}$ pin during realtime emulation are enabled.

OFF : Inputs from $\overline{\text{RESET}}$ pin are disabled.

If mnemonic is omitted, then the current setting will be displayed. After a system reset, inputs from the $\overline{\text{RESET}}$ pin are disabled.

Execution Example

```
* URST
  USER RESET DISABLE
```

```
* URST ON
```

```
* URST
  USER RESET ENABLE
```

```
* URST OFF
```

```
* URST
  USER RESET DISABLE
```

Verify Disk file with Code Memory

VER

Input Format

```
VER fname [address, address]
  fname : [pathname] filename [Extension]
```

Description

The VER command compares the contents of the specified disk file with the contents of code memory. When a difference is found, the address and the contents of the disk file and of code memory will be displayed as shown below.

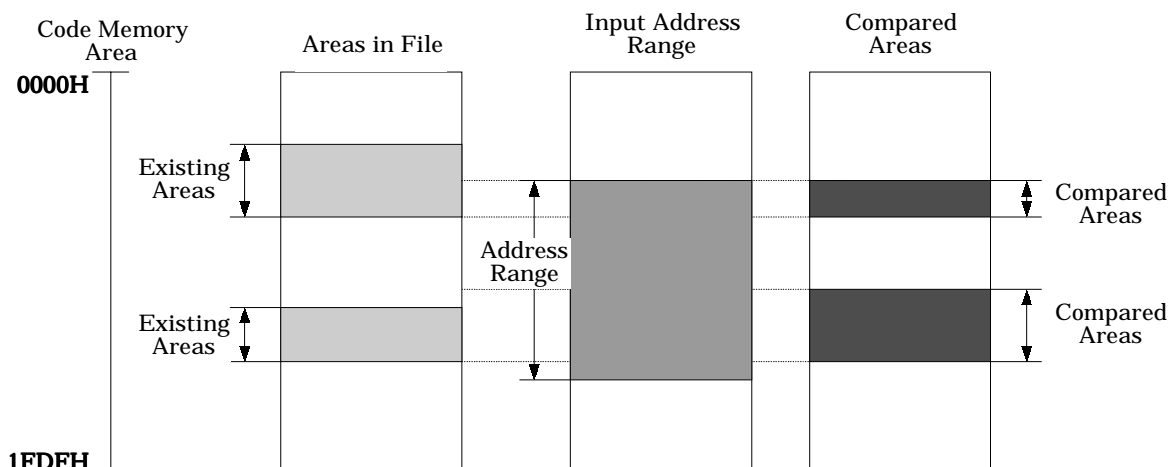
LOC =	XXXX	DISK [XXXX]	CM [XXXX]
	Address	Disk File contents	Code memory contents

The input filename can have a path specification. If the path is omitted, then a file in the current directory will be verified. If the extension is omitted, then the default extension (HEX) will be appended to the file.

The address, address represents the area of disk file and of code memory to be compared. When in MSM64162 mode, the address range is 0H to 7DFH. When in MSM64164C mode, the address range is 0H to FDFH. When in ML64168 mode, the address range is 0H to 1FDFH. If address, address is omitted, then the entire code memory area will be compared (note1).

■ Note1 ■

As shown below, comparison between the disk file and code memory will be performed on the overlap of disk file areas containing data and the “address, address” address range specified with the VER command.



Execution Example

```
* LOD T1
FILE OPENED NORMALLY. FILE TYPE : INTELLEC HEX
**** LOAD COMPLETED , NEXT ADDRESS = 0300 ****
```

```
* VER T1
**** VERIFY COMPLETED ****
```

```
* CCM 100
LOC=0100 BE OLD ---> 12 NEW
LOC=0101 A7 OLD ---> 34 NEW
LOC=0102 11 OLD ---> 45 NEW
LOC=0103 90 OLD ---> 56 NEW
LOC=0104 36 OLD ---> 78 NEW
LOC=0105 00 OLD ---> A1 NEW
LOC=0106 01 OLD ---> 22 NEW
LOC=0107 C4 OLD --->
```

```
* VER T1 0,7FF
LOC=0100 DISK [BE] CM [12]
LOC=0101 DISK [A7] CM [34]
LOC=0102 DISK [11] CM [45]
LOC=0103 DISK [90] CM [56]
LOC=0104 DISK [36] CM [78]
LOC=0105 DISK [00] CM [A1]
LOC=0106 DISK [01] CM [22]
**** VERIFY COMPLETED ****
```

```
* SAV T1CH 0,2FF
**** SAVE COMPLETED ****
```


Verify Disk file Mask Option into System memory

VERM

Input Format

```
VERM fname  
fname : [pathname] filename [Extension]
```

Description

The VERM command compares the contents of the specified mask option file with mask option data stored the system controller's system memory. If a mismatch in contents is found, then the address of the mismatch and the contents of both the mask option file and system memory will be displayed as follows.

LOC =	XXXX	DISK [XX]	SM [XX]
	Address	Contents of mask option file	Contents of system memory

The input file name can have a path specification. If the path is omitted, then the file in the current directory will be compared.

If the extension is omitted, then the file with the default extension (HEX) appended will be compared.

Execution Example

```
* VERM M164_000  
***** VERIFY COMPLETED *****
```

Verify EPROM with Code Memory

VPR

Input Format

```
VPR address_code, address_code [ , address_EPROM ]
or
VPR *
  address_code      : 0~7DF (MSM64162 mode)
                    : 0~FDF (MSM64164C mode)
                    : 0~1FDF (ML64168 mode)
  *                  : compared entire address range
  address_EPROM    : EPROM comparison start address
```

Description

The VPR command compares the contents of the specified range of code memory with the contents of the EPROM starting at the specified address, and displays any differences on the console.

Each *address_code* is a code memory address 0H to 7DFH when in MSM64162 mode, or 0H to FDFH when in MSM64164C mode, or 0H to 1FDFH when in ML64168 mode. The *address_code* , *address_code* specifies the range of code memory to be compared. If an '*' is input, then a range of code memory that corresponds to the EXPAND mode will be set (note1).

The *address_EPROM* is the EPROM's starting address for comparison. If this address is omitted, then comparison will start from EPROM address 0.

Input continues until a carriage return is entered. Then the following message will be output.

```
EPROM TYPE ---> type
START READING [Y/N] ---> _
```

Here *type* indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to compare, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data comparison will start. If the data comparison completes normally, then the "RUN" indicator will go off, the VPR command will terminate, and the emulator will wait for another command input.

When compare errors are encountered, they will be displayed on the console in the following format.

U/M	CM = XXXX	XX	PR = XXXX	XX
Mismatch display marker	Code memory address	Code memory data	EPROM address	EPROM data

■ **Note1** ■

The range of code memory compared with EPROM when '*' is input changes with the mode. When in MSM6162 mode, it is 0H to 7DFH. When in MSM64164C mode, it is 0H to FDFH. When in ML64168 mode, it is 0H to 1FDFH.

Execution Example

```
* TPR *
EPROM TYPE ---> 27512
START READING [Y/N] --->Y

* CCM 100
LOC=0100 E4 OLD ---> 23 NEW
LOC=0101 E4 OLD ---> 65 NEW
LOC=0102 E4 OLD --->

* VPR 0, FDF, 0
EPROM TYPE ---> 27512
START READING [Y/N] ---> Y

U/M   CM = 0100 23   PR = 0100 E4
U/M   CM = 0101 65   PR = 0101 E4
```

Verify Disk file with System Memory

VPRM

Input Format

VPRM

Description

The VPRM command compares an EPROM with the mask option data in the system controller's system memory. The EPROM address range to be compared is always from 0H to BFFH.

When the carriage return is entered after the above input format, the emulator will output the following message.

```
EPROM TYPE ---> type
START READING [Y/N] ---> _
```

Here *type* indicates the currently set EPROM type. If the EPROM type displayed is the same as the EPROM type that the user wants to write, then enter "Y" at the underscore. If they are different, then input "N" and set the EPROM type again with the TYPE command.

When "Y" is input at the underscore, the EASE64168 "RUN" indicator will light, and the data comparison will start. If the data comparison completes normally, then the "RUN" indicator light will go off, the VPRM command will terminate, and the emulator will wait for another command input.

If a comparison error is found, then the emulator will display the following on the console.

U/M	PR =	XXXX	XX	SM	XX
Mismatch display marker		EPROM address	EPROM data		System memory data

Execution Example

```
* LODM M164_000
```

```
FILE OPENED NOMALLY. FILE TYPE : INTELLEC HEX
```

```
***** LOAD COMPLETED *****
```

* PPRM

EPROM TYPE ---> 27512

START PROGRAMMING [Y/N] ---> Y

* VPRM

EPROM TYPE ---> 27512

START READING [Y/N] ----> Y

*TPRM

EPROM TYPE ---> 27512

START READING [Y/N] ----> Y

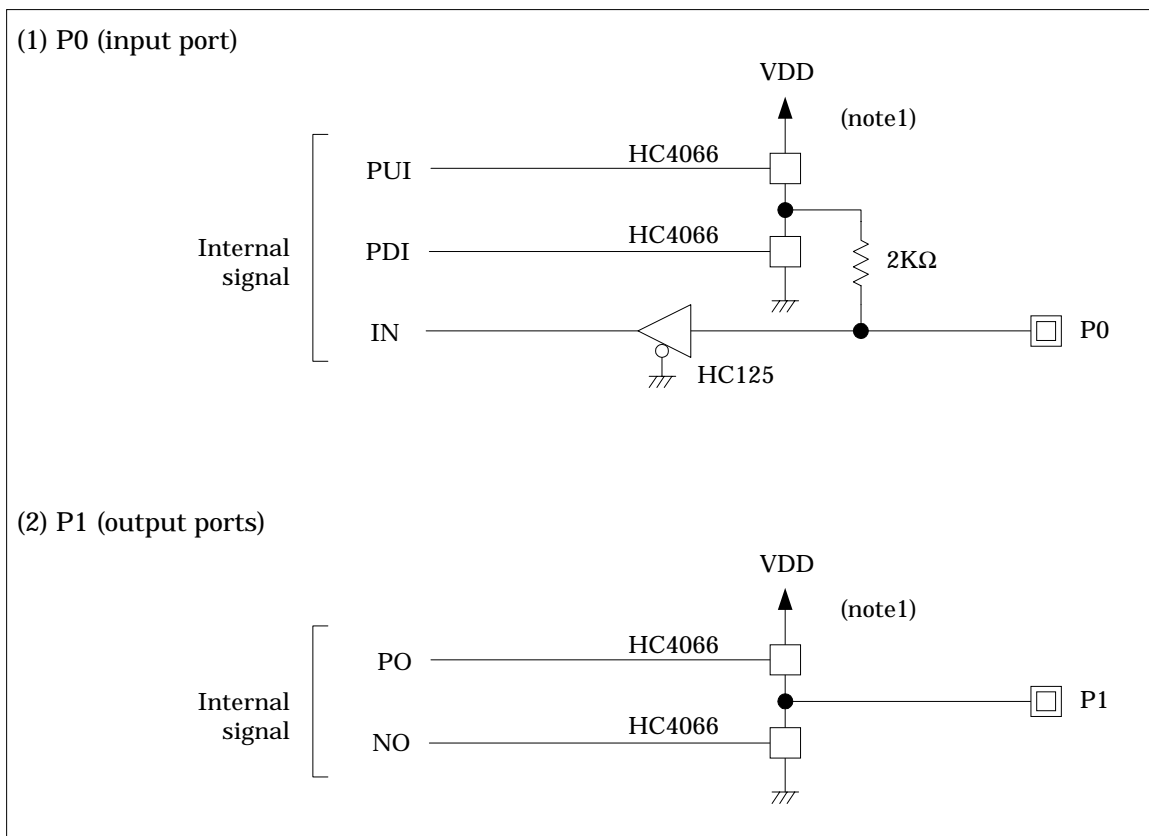
Chapter4. Debugging Notes

This chapter provides some notes about debugging with the EASE64168 system.

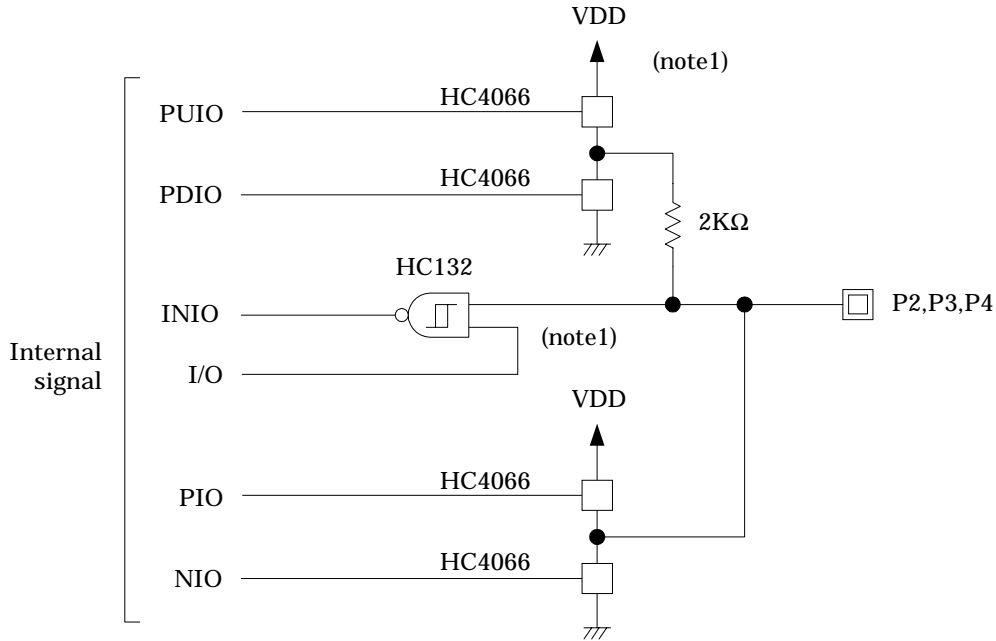
1. Debugging Notes

1.1 Ports

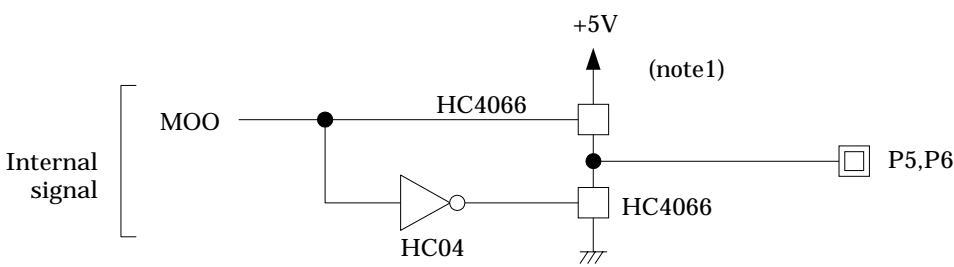
The input/output of the port pins, BD pin, and $\overline{\text{RESET}}$ pin are as shown below. Their input/output characteristics differ from those of the M6416x series microcontrollers.

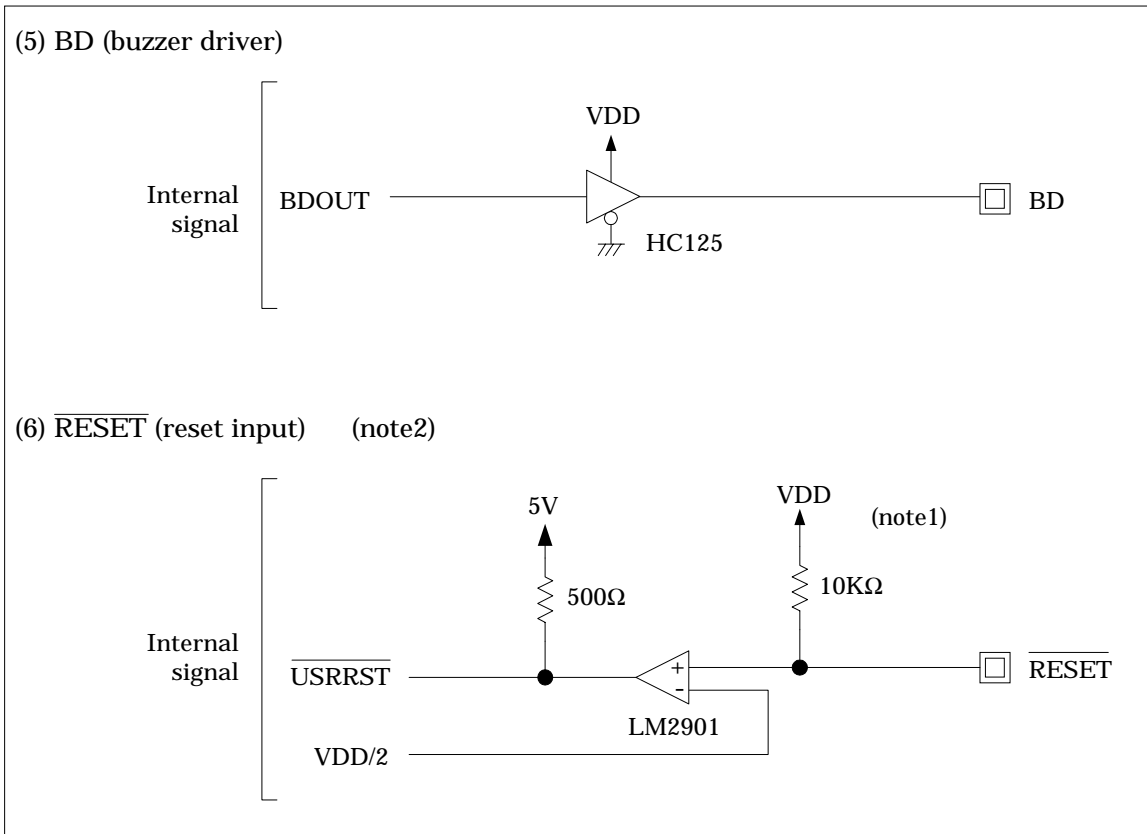


(3) P2, P3, P4 (input/output ports)



(4) P5, P6 (mask option output ports)





■ **Note1** ■

From 3V to 5V is applied to VDD. The VDD supply is switched by the CIPS command: it will be the emulation kit internal 5V supply if CIPS INT is input, or it will be the VDD supply of user connector pins 36 and 37 if CIPS EXT is input. A voltage 3V to 5V can be input to the VDD pin.

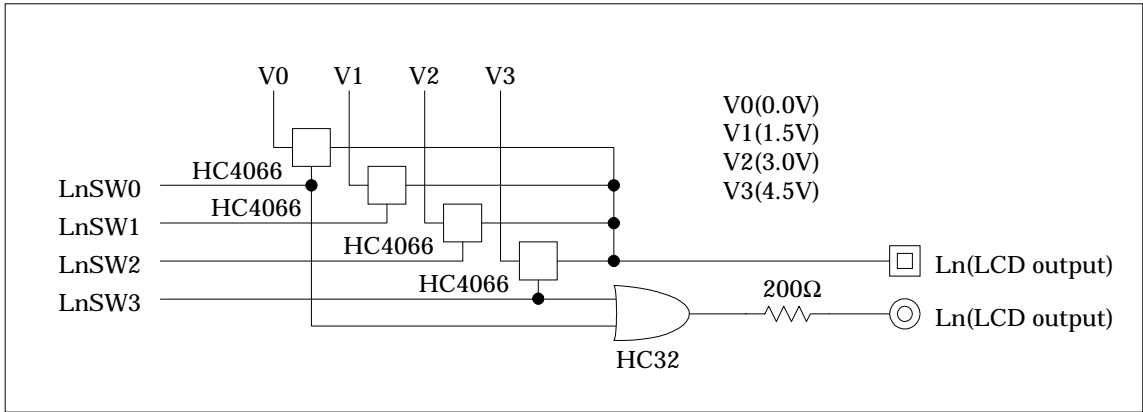
■ **Note2** ■

The $\overline{\text{RESET}}$ pin is effective when specified to be on by the URST command. If an “L” level is input on this pin during real-time emulation, then the evaluation board will reset.

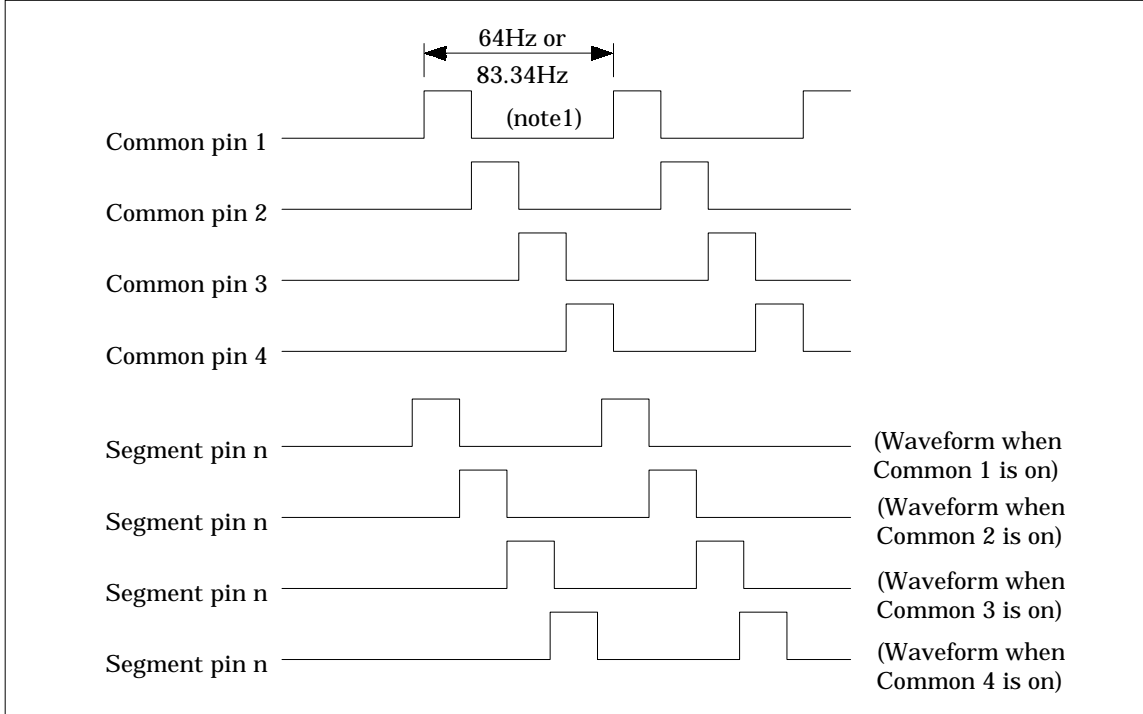
1.2 LCD Drivers

(1) Output Characteristics

The LCD driver outputs are configured as shown below. Their input/output characteristics differ from those of the M6416x series microcontrollers.



The LED output above is used to implement LEDs (light emitting diodes) to evaluate the LCD portion. It outputs the following signals.



To evaluate the timing of an LED turning on, build a circuit like the following.

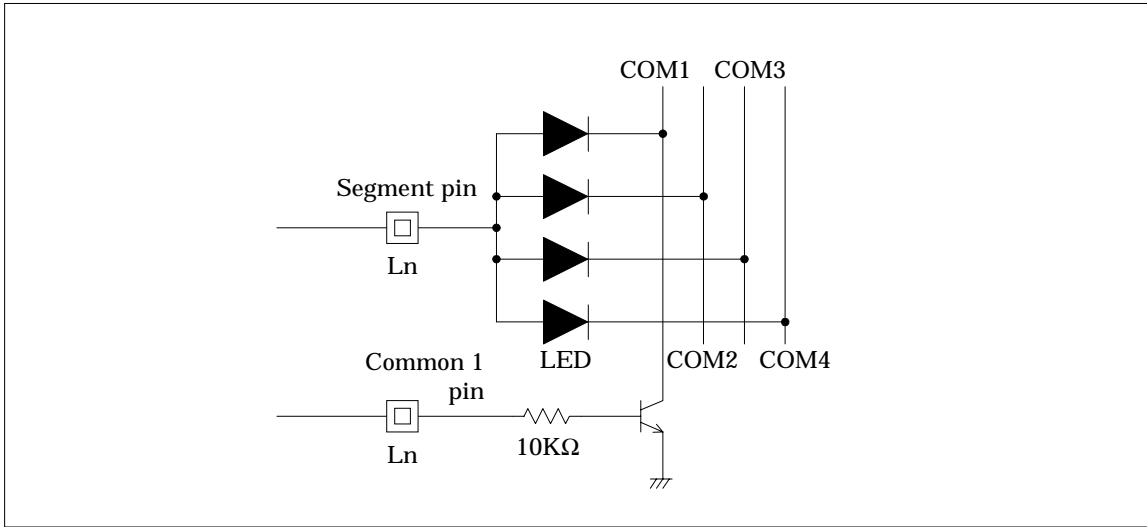


Figure 4-1 Connection Example For LED Timing Evaluation

If the current flowing through in one LED is assumed to be 1.25mA with this circuit, then the collector current of the common pin transistor will be up to 37.5mA (at 1/4 duty), so a transistor that can drive high current is necessary.

■ **Note1** ■

Frequency will be 64Hz when 1/4duty or 1/2duty is selected, or 83.34Hz when 1/3duty is selected.

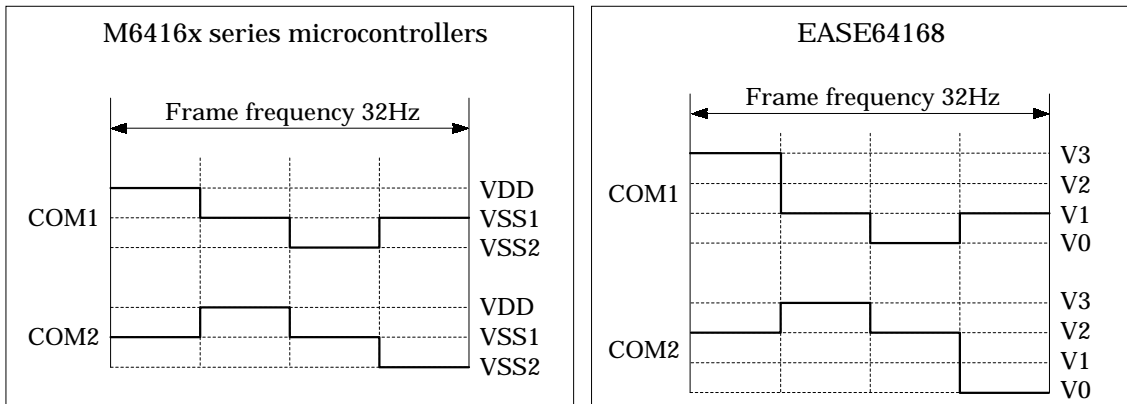


Figure 4-2 1/2-Duty Common Drive Waveform

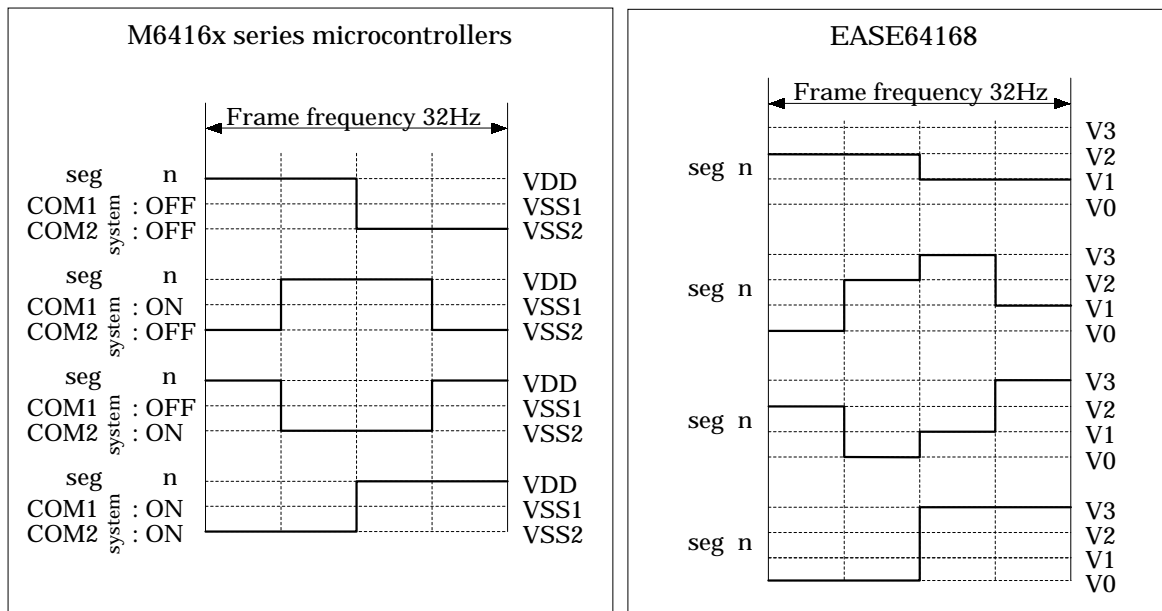


Figure 4-3 1/2-Duty Segment Drive Waveform

(2) Display registers for LCD drivers

In the M6416x series microcontrollers, the display registers and bits that are not specified as either segment or common ports by LCD driver mask option data (note2) cannot be read or written. However, in the EASE64168 emulator, all bits of display registers can be read and written, regardless of mask option data.

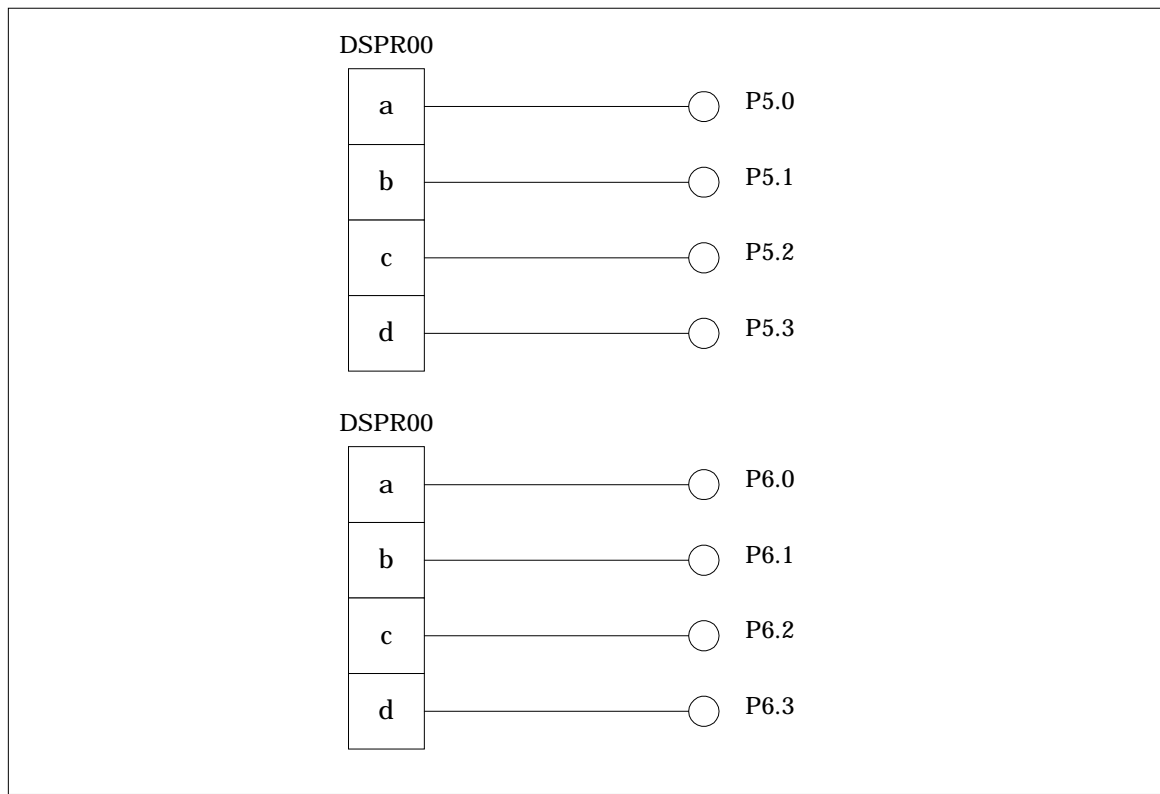
Therefore, an application program that utilizes the unused display register bits as a RAM area will not work with the M6416x series microcontrollers. Also, the M6416x series microcontrollers will always read these bits as 1, but the EASE64168 will read them as undefined (but 0 after reset).

(3) Clearing display registers by user reset

The EASE64168 can initialize the evaluation board with a user reset (note3), but the display registers for LCD drivers are not initialized by user resets.

(4) Output port selection by mask option

Eight pins of LCD driver outputs (L16~L23 for MSM64162, MSM64162A and MSM64162D, L26~L33 for MSM64164C and ML64168) can be set as output ports by mask option. In the these 8 pins can match up in any way with the 8 bits of Display Register 0 and Display Register 1 (DSPR00, DSPR01), but in the EASE64168 emulator the matching is fixed to output to the user connector as shown below.



■ **Note2** ■

Mask option data is created by the mask option generators (MASK162/162A/164/168).

■ **Note3** ■

A user reset initializes the evaluation board by the input of an "L" level on the user connector RESET pin during real-time emulation.

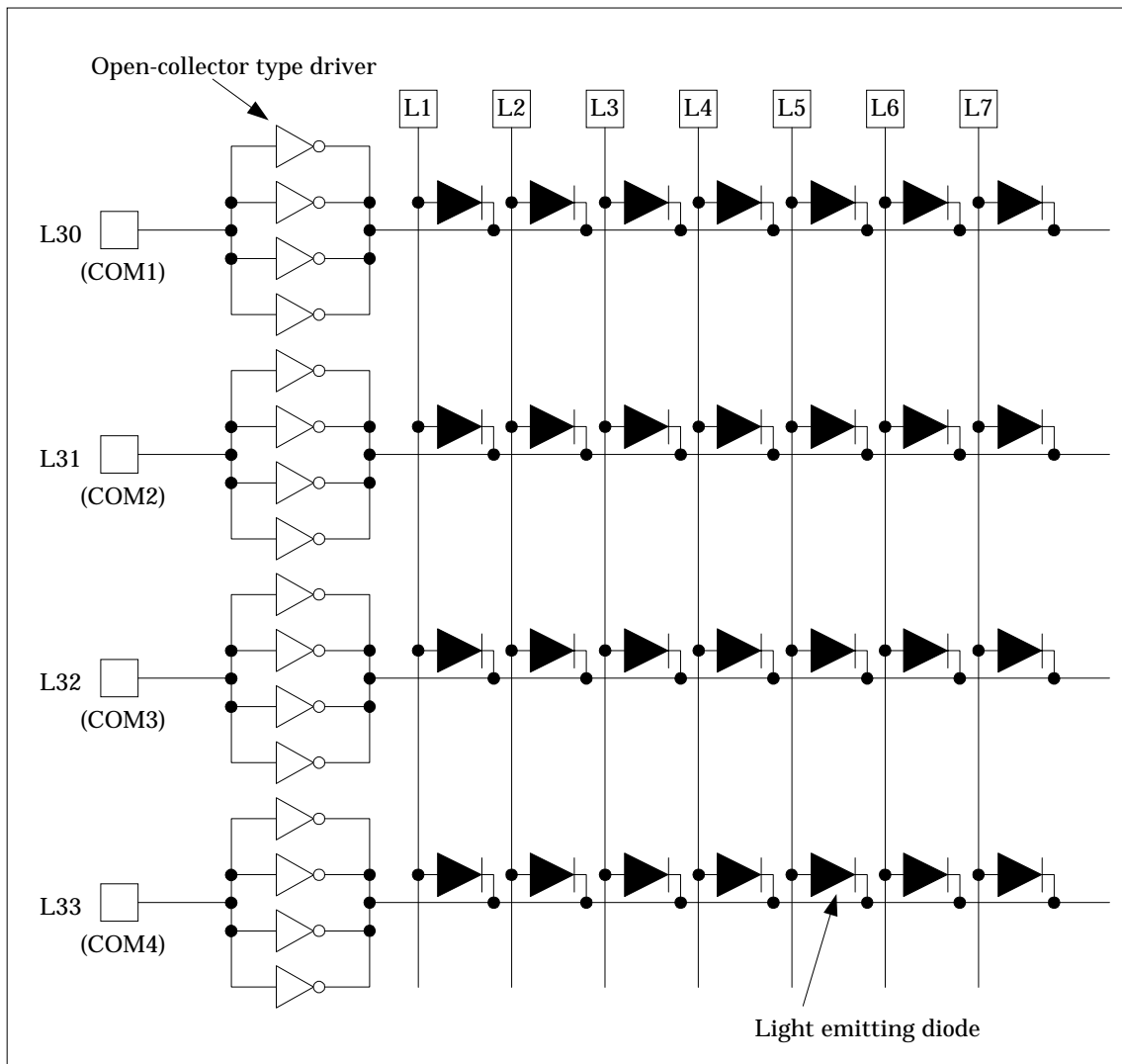


Figure 4-4 Circuit Example For LED Timing Evaluation

(5) Multiple segments assignment

With the M6416x series microcontrollers it is available to assign any one bit of the display register(DSPR) to multiple segment drivers. But with EASE64168 emulator the one bit data of display register can be assigned to only one segment driver.

Example : The table below shows the assignment information in case that the mask option is applied to assign the d-bit of DSPR2 to the common 4 of L28 and also to the common 4 of L29.

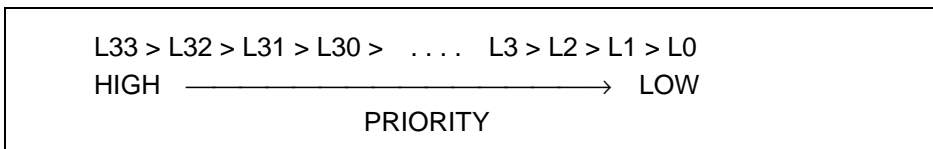
Table 4-1 An appointed example of mask option

SEG	SIGNAL	C/S/P	COM1		COM2		COM3		COM4	
			DATA	DSPR	DATA	DSPR	DATA	DSPR	DATA	DSPR
L28	1a / 1b / 1c / DP1	S	a	2	b	2	c	2	d	2
L29	2b / / 2c / DP2	S	d	0			a	11	d	2

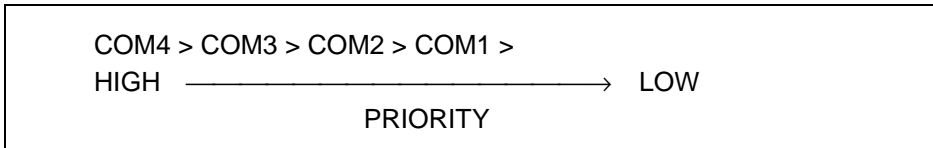
Using EASE64168 emulator on the assignment shown above, d-bit data of the display register DSPR2 appear only on the COM4 pin of L29, but does not appear on the COM4 pin of the L28.

The following figures show the precedence of the LCD driver pins and common pins selection when multiple assignment is specified with the EASE64168 emulator.

* Precedence for multiple segments assignment



* Precedence for multiple LCD drivers assignment

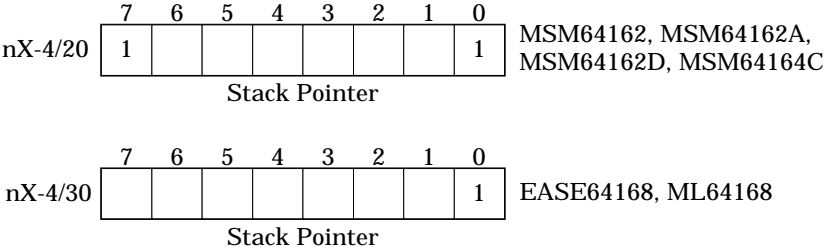


1.3 Stack Pointer

The EASE64168's length of SP is different from MSM64162/162A/162D/164C.

The SP's significant bit length is 6 bits for the nX-4/20 core (MSM64162/162A/162D/164C) and 7 bits for the nX-4/30 core (ML64168). The nX-4/20 core's the highest of the SP is always fixed to "1". But EASE64168's SP always works same as the nX-4/30 core.

When you are debugging the MSM64162/162A/162D/164C mode, please take notice about this difference.



1.4 HALT Pin

The user connector HALT pin is a monitoring pin that outputs an “H” level in halt mode. The peripheral circuitry of the HALT pin is shown below.

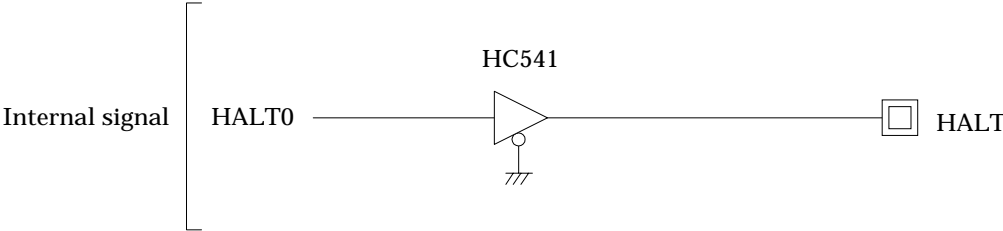
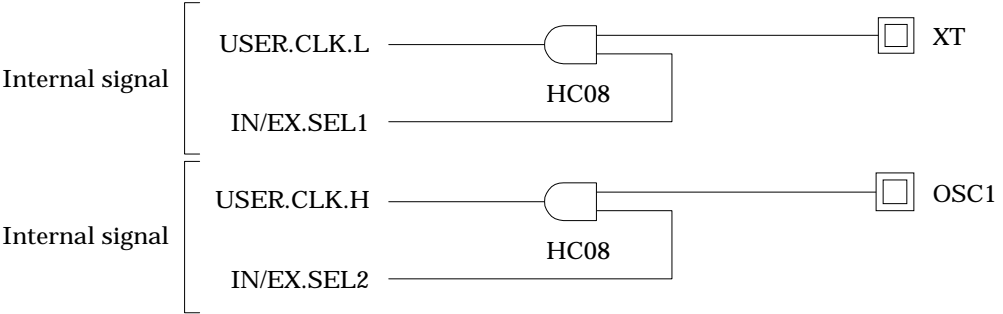


Figure 4-5 HALT Pin Peripheral Circuit

1.5 XT and OSC1 Pins

The user connector XT pin and OSC1 pin are used respectively for input of a low-speed and high-speed clock. The interface power supply voltage must be 5V. The peripheral circuitry of the XT pin and OSC1 pin is shown below.



1.6 ADC POD

(1) Connecting to emulator base unite

Be sure that the emulator power supply is off when connecting the ADC POD to the emulator base unit. If the power supply is on, then even when the ADC POD is connected, A/D conversion will not be performed.

(2) Matching chip mode to ADC POD

The EASE64168 in-circuit emulator chip mode setting specifies the target chip. Never use the MSM64162/164 setting with an ML64168 ADC POD (ADC-64168-3V or ADC-64168-1.5V) or the ML64168 setting with an M64162/164 ADC POD (ADC-64164-3V or ADC-64164-1.5V). Note that the EASE64168 in-circuit emulator provides no warnings of such incompatible mixtures.

(3) CR oscillation clock

The CR oscillation clock for the ADC POD is supplied by the emulator's internal evaluation board, except when the SFR A/D conversion run/stop select bit (EADC) is 1. Therefore the evaluation board's internal counter B (CNTB0~3) will count regardless of whether emulation is executing or stopped.

■ Note1 ■

Refer to Chapter 3. section 2.3, "Connecting the ADC POD."

■ Note2 ■

Refer to Appendix 1, "EASE64168 External Views."

1.7 DASM Command

(1) Mnemonic

The pairs of instructions shown below result in identical instruction codes. When the debugger's DASM, DTM and STP command encounters one of these codes, it will display the mnemonic shown on the left.

NOP	and	AIS 0	(both result in code 0H)
INA	and	AIS 1	(both result in code 1H)
LAM	and	LAMM 0	(both result in code 70H)
XSM	and	XAMM 0	(both result in code 71H)

(2) LJP, LCA instruction

When the code memory address is over 2000H (EXPAND ON mode), the destination address of LJP and LCAL instruction is displayed "(?)". But the instruction is executed normally.

```
* DASM 1FFC, 2007
LOC=1FFC DB0000 LCAL 0000
LOC=1FFF BC1000 LJP 1000
LOC=2002 DB0000 LCAL ( ? )
LOC=2005 BC1000 LJP ( ? )
```

1.8 Breaks

- (1) If a break condition is fulfilled during a skip, then the break will be saved until after the skip operation completes. (operation is the same even with the STP command.) However, if a break address instruction is skipped at an address break or breakpoint break, then the break will not be saved, and no break will occur when the skip completes.
- (2) If a break condition is fulfilled during an interrupt transfer cycle, then the break will occur after the interrupt transfer cycle completes. The break PC will be the interrupt vector address.
- (3) The value of the time base counter when a break occurs in high-speed clock mode will not always be the same even under the same conditions because the high-speed clock and low-speed clock are asynchronous. Furthermore, when EASE64168 is operated with the high-speed clock, interrupt timing may differ between break (emulation) operation and step command execution.
- (4) With the M6416x series microcontrollers the skip function of an AIS instruction will be disabled in a program where the AIS instruction is executed following either ADCS and ADCS@XY instructions, or SUBCS and SUBCS@XY instructions. With the EASE64168 emulator, however, if a break is set to occur immediately after execution of either ADCS and ADCS@XY instructions, or SUBCS and SUBCS@XY instructions, and execution is set to resume starting with the AIS instruction, then the skip function of the AIS instruction will not be disabled. Likewise, the skip function of the AIS instruction will not be disabled with a STP command.

1.9 MSM64162D

The MSM64162D, when evaluated with EASE64168, will be evaluated in MSM64162 mode, but be aware that you can still use functions that do not exist in the MSM64162D chip (high-speed clock, A/D converter CROSC oscillation mode, IN1 external clock input mode). If those functions are used when evaluating a MSM64162D, then chip operation will not be guaranteed.

1.10 Battery Check Circuit

The battery check circuit operating characteristics of the EASE64168 emulator, indicated in Figure 4-5, differ from those of the MSM64162, MSM64162A, MSM64162D and the ML64168.

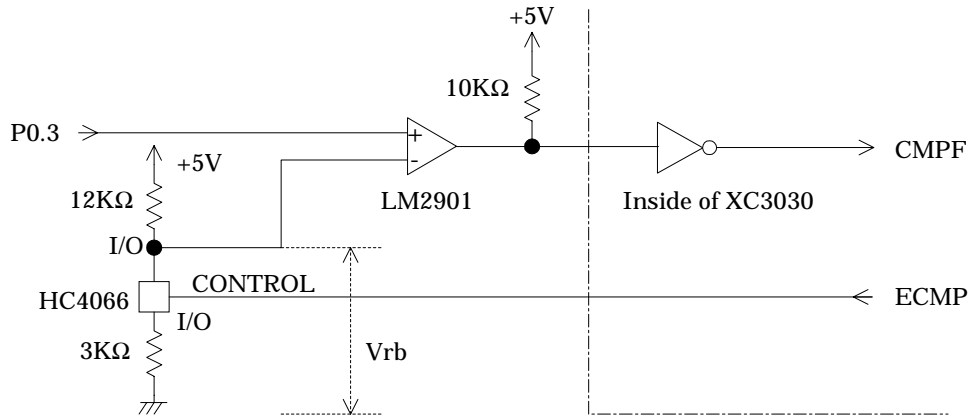


Figure 4-6 EASE64168 Battery Check Circuit

The reference Voltage (Vrb) for detecting the level of the power supply voltage is approximately +1.0V.

The battery check frag (CMPF) of the backup control register (BUPCON) of SFR is configured as a logical 1 when the input voltage level to P0.3 is lower than the reference voltage (Vrb). It is configured as a logical 0 when the input voltage level is higher than the reference voltage (Vrb).

■ **Note1** ■

VDD is the interface power supply voltages of the USER Connector (Pin 36 and Pin 37), to which the voltages ranging from +3V to +5V are applied.

The source of the voltages can be switched by the debugger command CIPS.

In the care of "CIPS INT", +5V is supplied from the inside of emulator.

In the care of "CIPS EXT", voltages ranging from +3V to +5V can be input to VDD.

They are supplied from the USER Connector (Pins 36 and Pin 37)

1.11 Power Supply and Connections

Internal circuit of the EASE64168 emulator, consist of discrete components. Their polarity is opposite to the power supply voltage of the MSM64162, MSM64162A, MSM64162D, and MSM64164C and equal to that of the MSM64P164, ML64P168 (OTP version) and ML64168. Figure 4-6 illustrates the connections between the USER application system and the external power supply and between the ADC-POD board and the power supply.

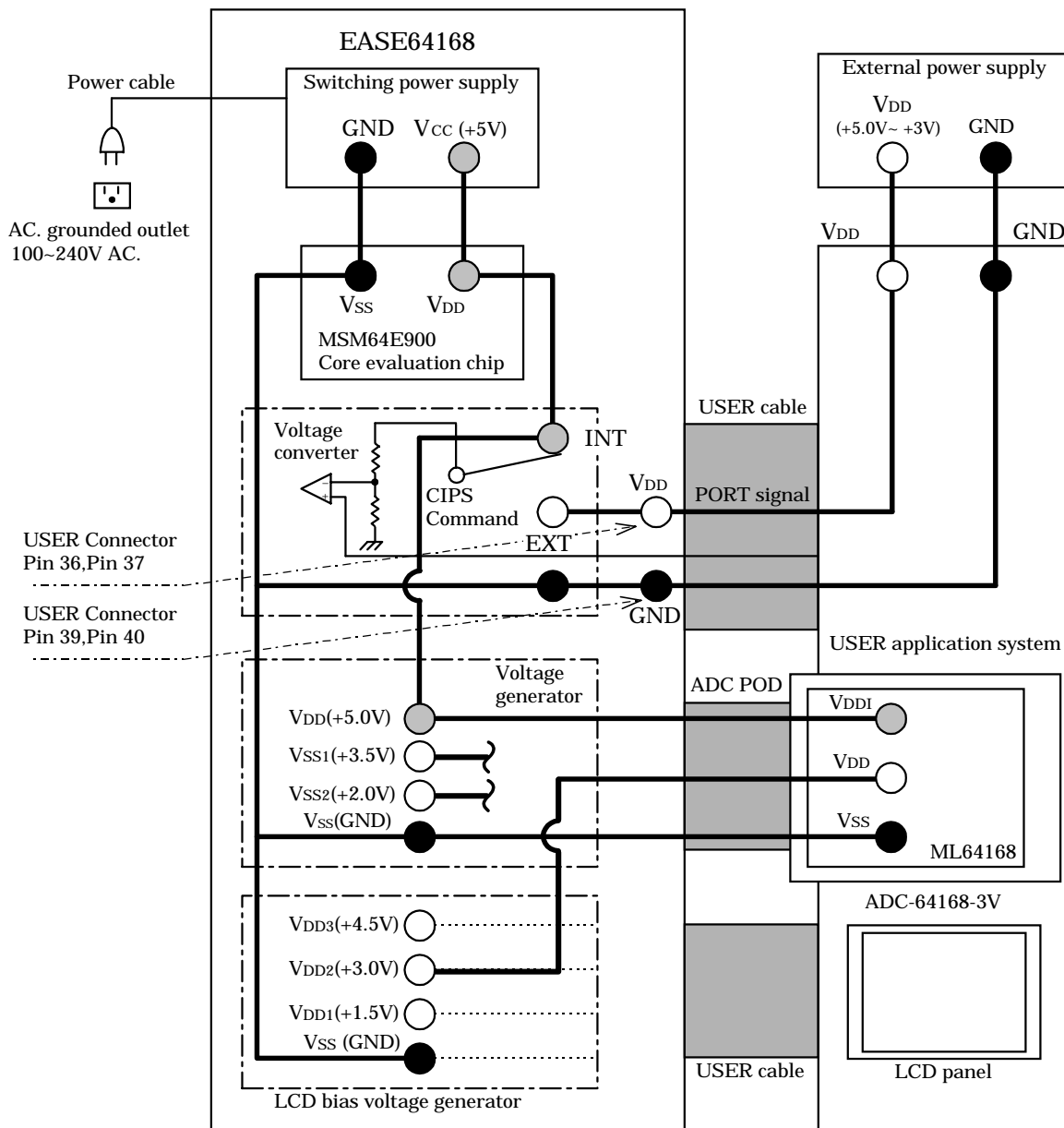


Figure 4-7 Power Supply Connections for 3.0V ML64168

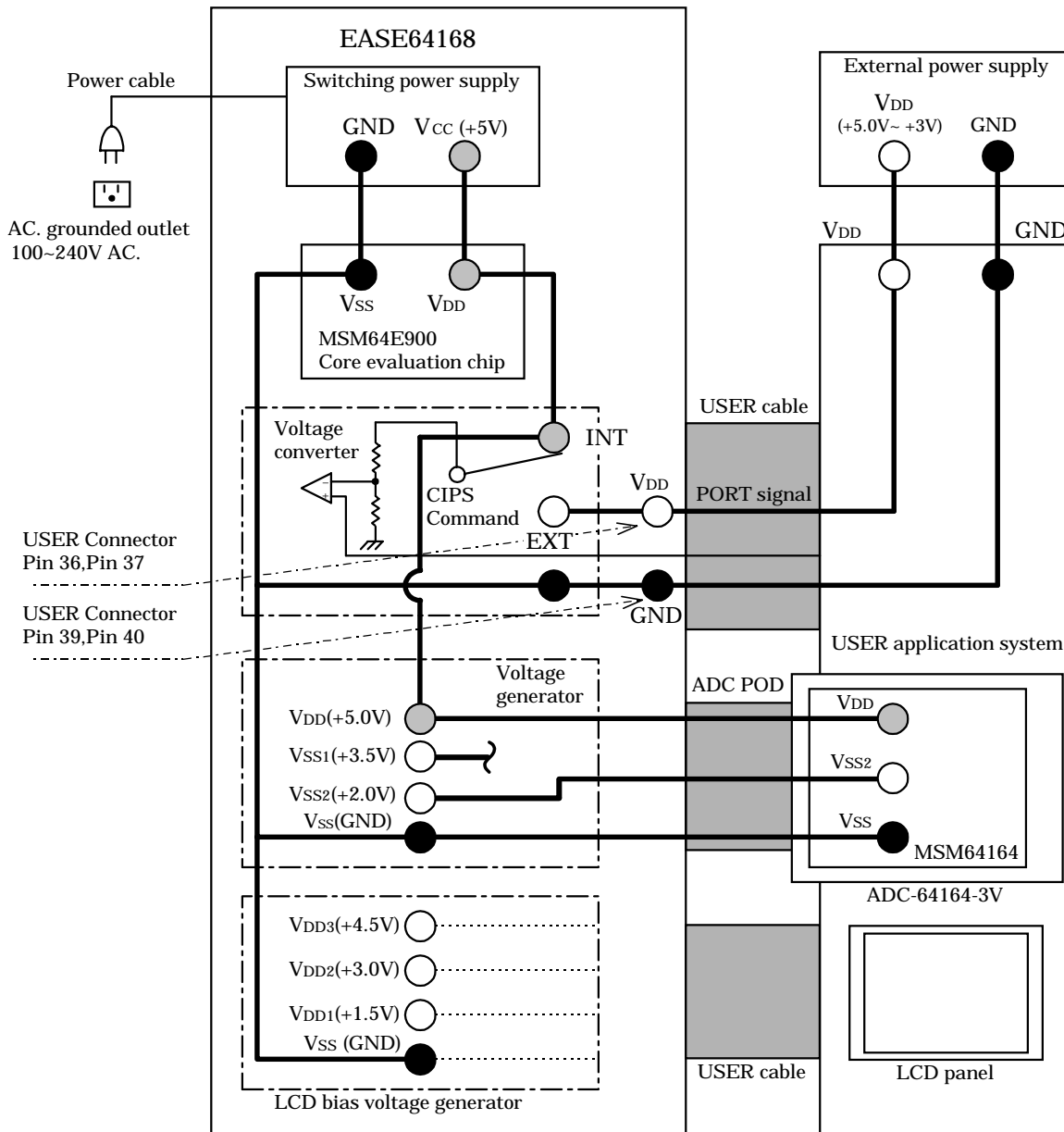


Figure 4-8 Power Supply Connections for -3.0V MSM64162/164

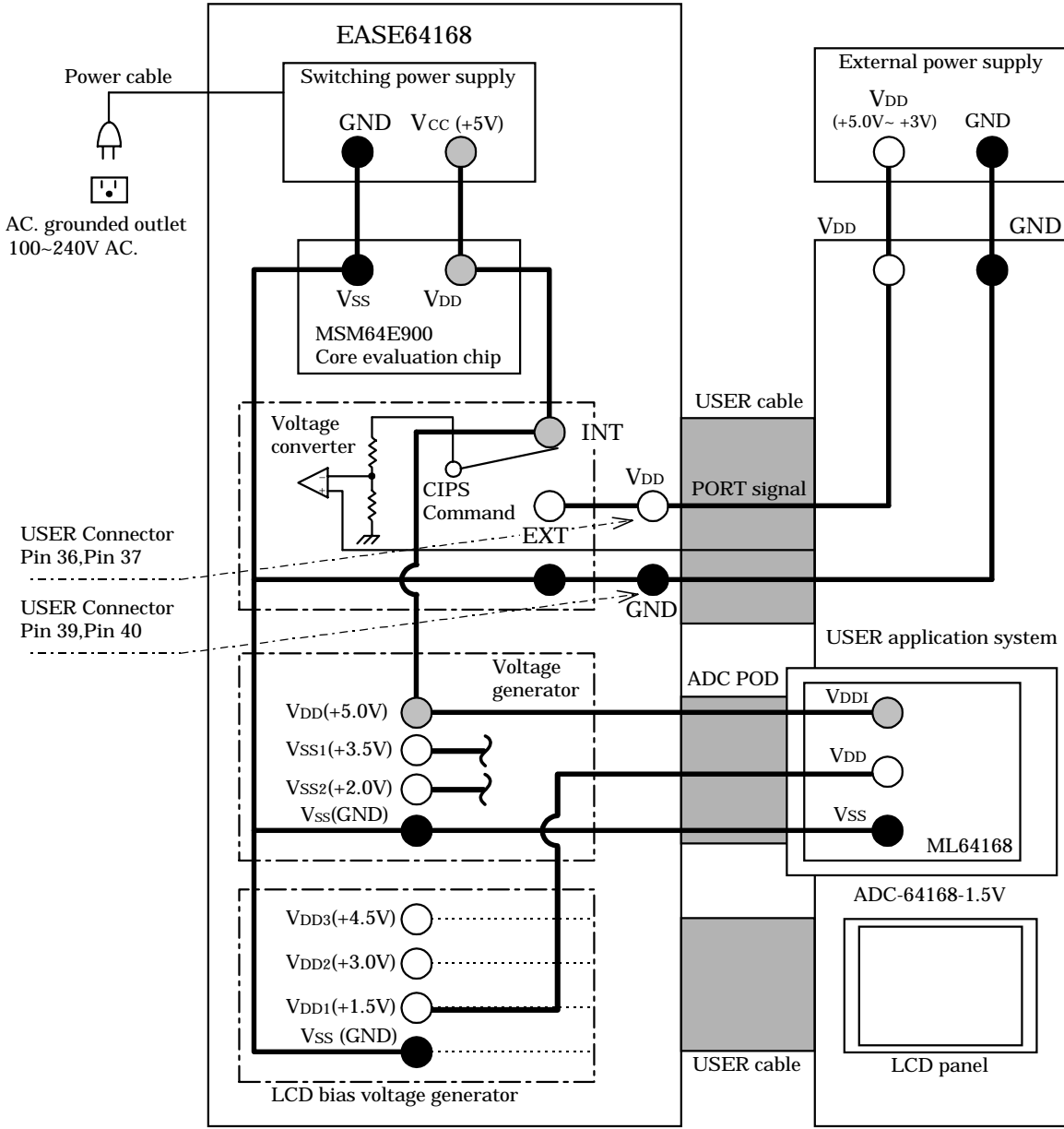


Figure 4-9 Power Supply Connections for 1.5V ML64168

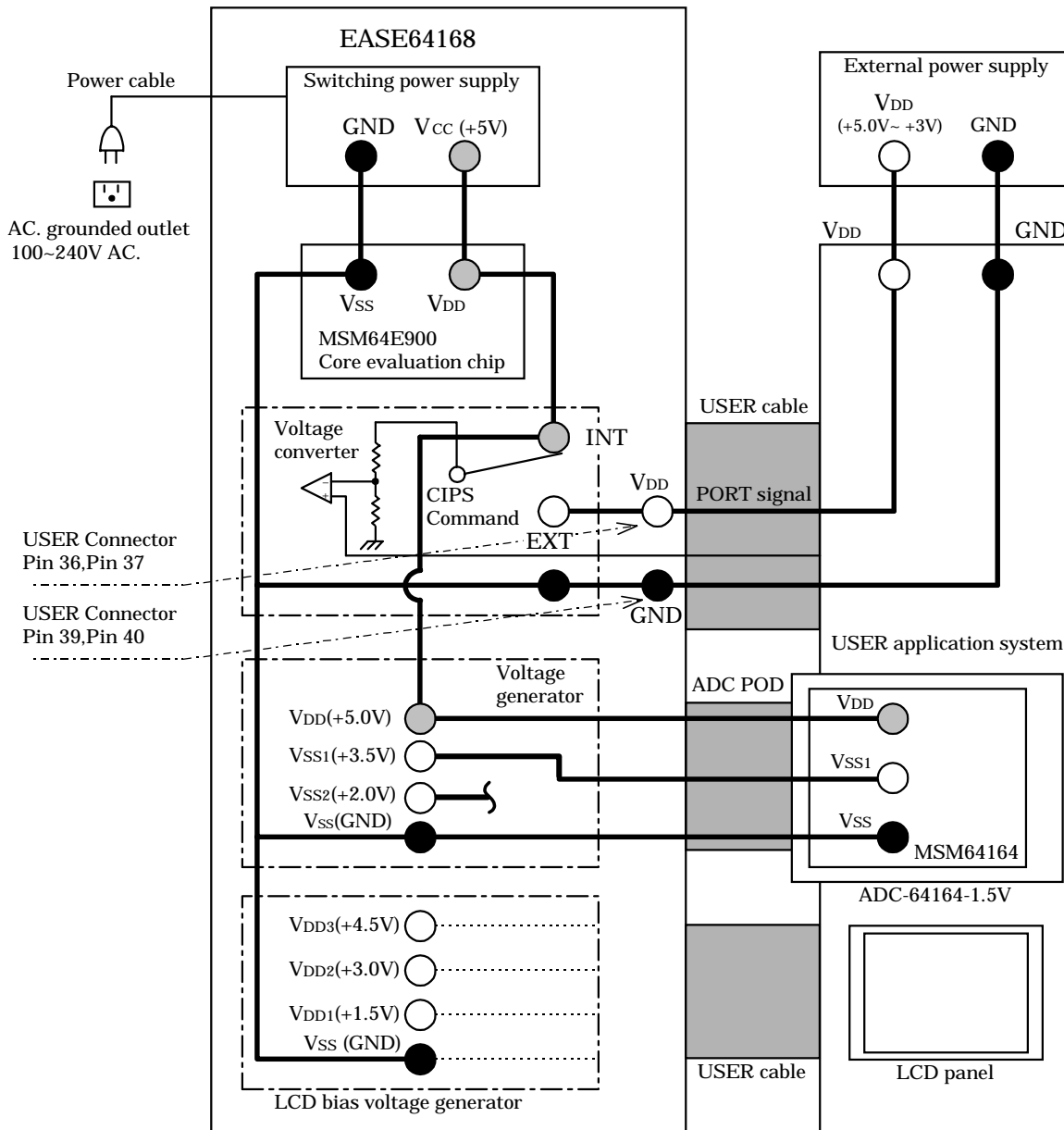


Figure 4-10 Power Supply Connections for -1.5V MSM64162/164

1.12 A/D converter counter A registers

In the EASE64168 emulator, the A/D converter counter A registers (CNTA0~3) are made by the BCD counter LSI (HC192). When the value of "0AH" ~ "0FH" is written to CNTA0~3, CNTA0~3 are set to the written value. This operation is different from those of the M6416x series microcontrollers. The value of "0AH" ~ "0FH" should not be written to the A/D converter counter A registers in the application program.

1.13 Warning messages (Mask Options)

EASE64168 emulator judges the mask option data which was generated in the mask option generator (MASK162/162A/164/168), and there is the function which output warning message in the case of a wrong used way. Please confirm contents of the mask option and the application program if the following warning messages were output on CRT. (Even though warning messages are output, **they do not have influence on emulation and an assignment of a segment by the mask option at all.**)

Table 4-2 Warning messages

Judgment standard	Message	Judgment time
Is not 1/2duty which can not be realized by EASE64168 set up ?	Warning 1 : The 1/2 bias signal cannot be output.	At the time of LCDX command.
Does not duty of the display control register (DSPCON) which is set up on a program differ from the mask option data which was loaded in LCDX command ?	Warning 2 : The LCD driver duty disagreed with mask option.	After the emulation is finished. (At the time of break occurs.)

However, so as to write it below, when the used way is right even, warning messages are sometimes output. (EASE64168 emulator searches from segment-L0 of loaded mask option data in order, and judges duty of the data which segment data have described it at the beginning of the first.) If warning messages are output at the time of the following conditions, **disregard warning messages after it was checked that value of the display control register (DSPCON) is set up correctly.**

(1) In case of 1/4duty is chosen by mask option (part 1)

If 1/4duty is chosen and the segment which does not assign data to COM3 and COM4 both exists, EASE64168 emulator judges that mask option data choosing 1/2duty. (See table 3.) In this case, the warning message 1 is output at the time of LCDX command. Moreover, when the display control register (DSPCON) has be set up except for 1/2duty (even 1/4duty is set up correctly) at the time of break occurs, the warning message 2 is output.

Table 4-3 A mask option layout example 1

SEG	SIGNAL	C/S/P	COM1		COM2		COM3		COM4	
			DATA	DSPR	DATA	DSPR	DATA	DSPR	DATA	DSPR
L0	/COM2/ /	C								
L1	1a/ 1b/ / /	S	B	05	D	06				
L2	AL/MODE/ 4f/ 4a	S	A	03	B	03	C	06	B	10
L3	2b/ 2c/ 2d/ 2f	S	A	07	B	08	C	09	D	07
L4	COM1/ / /	C								

(2) In case of 1/4duty is chosen by mask option (part 2)

If 1/4duty is chosen and the segment which does not assign data to COM4 exists, EASE64168 emulator judges that mask option data choosing 1/3duty. (See table 4.) In this case, when the display control register (DSPCON) has be set up except for 1/3duty (even 1/4duty is set up correctly) at the time of break occurs, the warning message 2 is output.

Table4-4 A mask option layout example 2

SEG	SIGNAL	C/S/P	COM1		COM2		COM3		COM4	
			DATA	DSPR	DATA	DSPR	DATA	DSPR	DATA	DSPR
L0	/COM2/ /	C								
L1	1a/ 1b/ 1c/ /	S	B	05	D	06	A	11		
L2	AL/MODE/ 4f/ 4a	S	A	03	B	03	C	06	B	10
L3	2b/ 2c/ 2d/ 2f	S	A	07	B	08	C	09	D	07
L4	COM1/ / /	C								

(3) In case of 1/3duty is chosen by mask option

If 1/3duty is chosen and the segment which does not assign data to COM3 exists, EASE64168 emulator judges that mask option data choosing 1/2duty. (See table 5.) In this case, the warning message 1 is output at the time of LCDX command. Moreover, when the display control register (DSPCON) has be set up except for 1/2duty (even 1/3duty is set up correctly) at the time of break occurs, the warning message 2 is output.

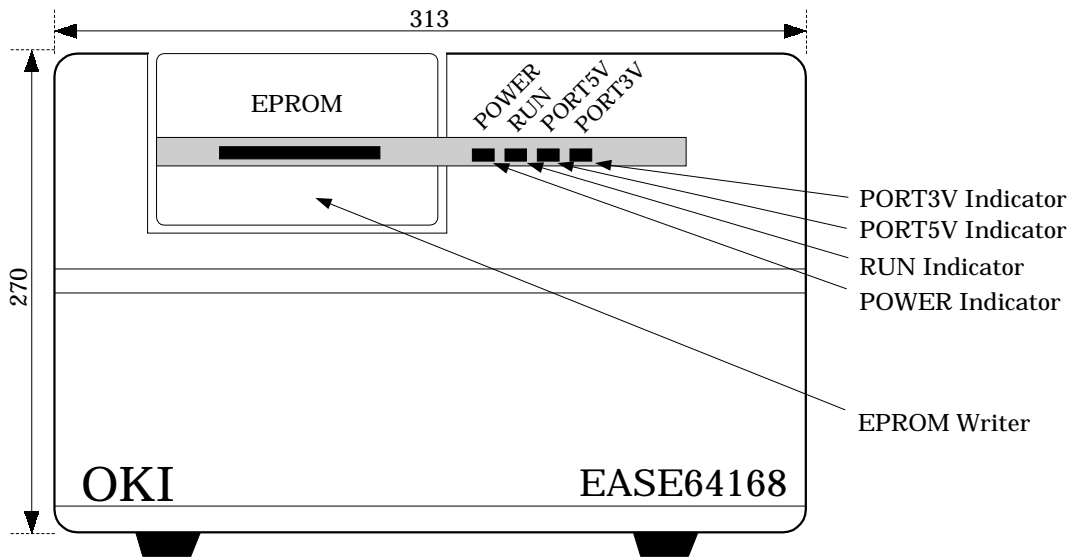
Table4-5 A mask option layout example 3

SEG	SIGNAL	C/S/P	COM1		COM2		COM3		COM4	
			DATA	DSPR	DATA	DSPR	DATA	DSPR	DATA	DSPR
L0	/COM2/	C								
L1	1a/ 1b/ /	S	B	05	D	06				
L2	AL/MODE/ 4f	S	A	03	B	03	C	06		
L3	2b/ 2c/ 2d	S	A	07	B	08	C	09		
L4	COM1/ /	C								

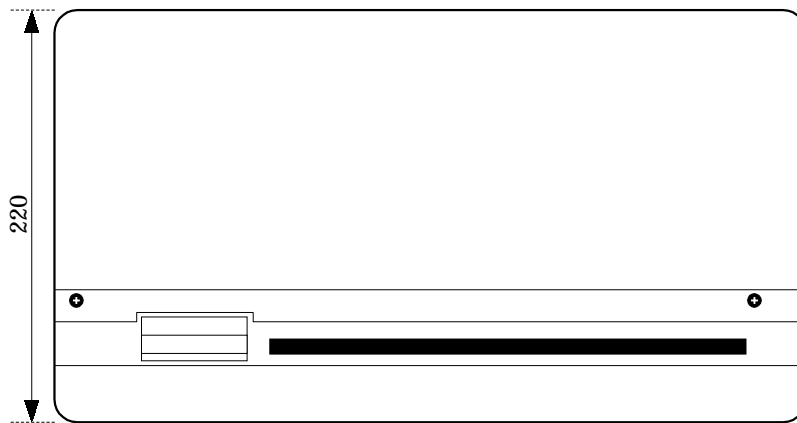
Appendix

- 1 EASE64168 External Views
- 2 User Cable Configuration
- 3 Pin Layout of User Connectors
- 4 RS232C Cable Configuration
- 5 Emulator RS232C Interface Circuit
- 6 If EASE64168 Won't Start
- 7 Mounting EPROMs
- 8 Error Messages
- 9 Hardware Specifications
- 10 Command Summary

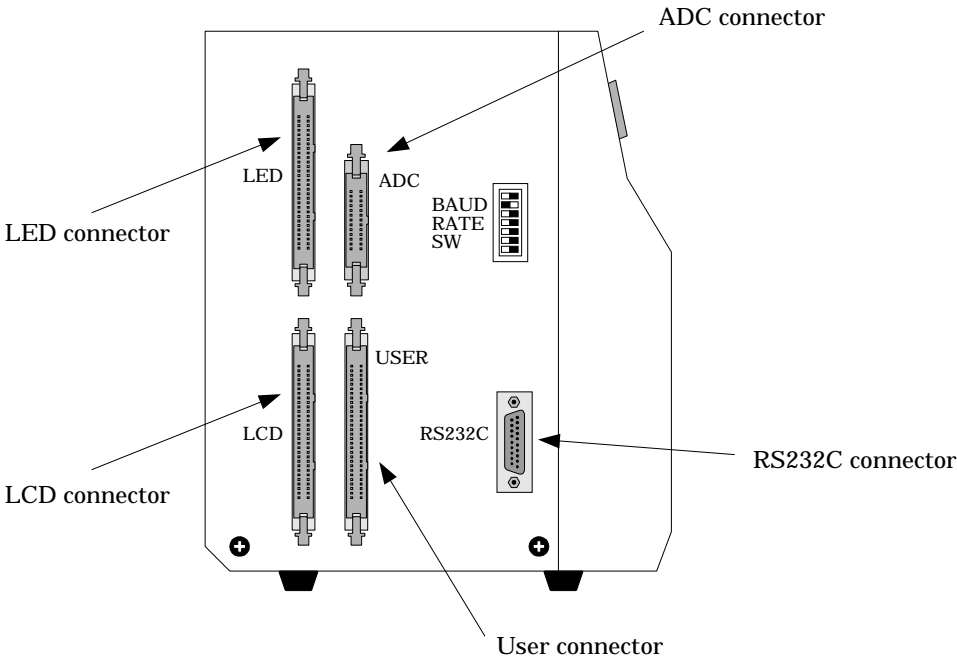
1. EASE64168 External Views



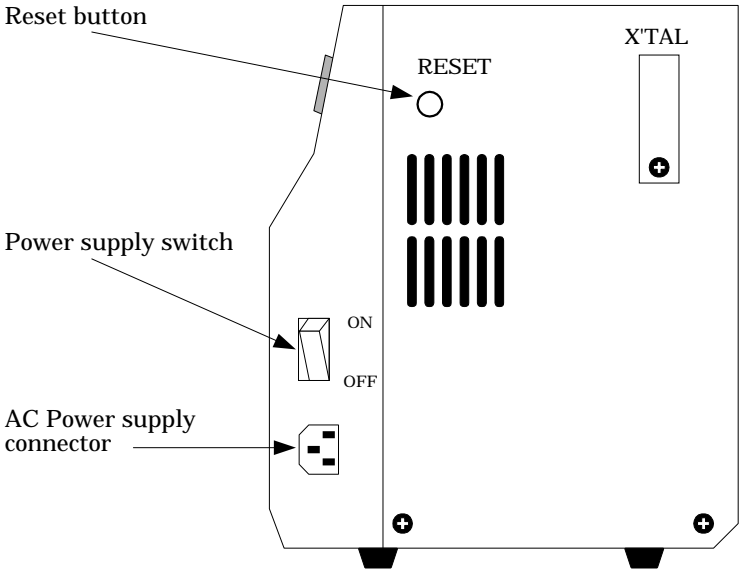
Front View



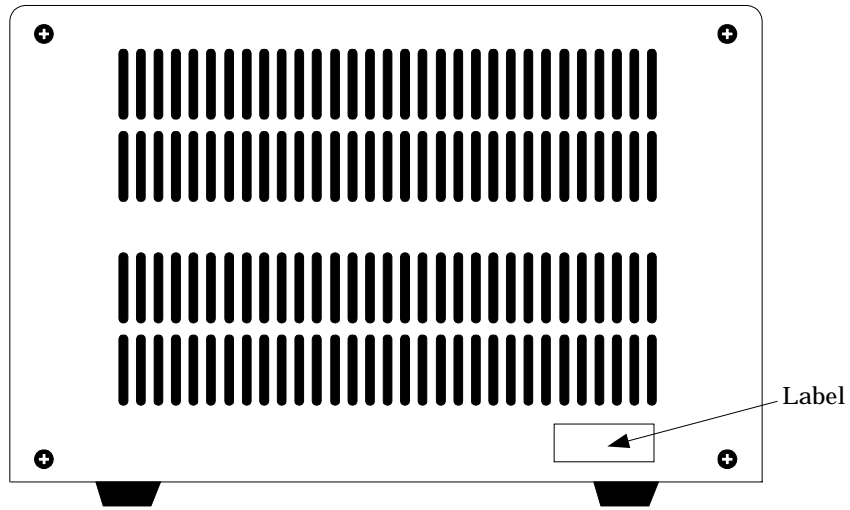
Top View



Left View



Right View



Rear View

2. User Cable Configuration

Figure A-1 shows the configuration of the accessory user cables (two 40-pin cables). Table A-1 gives the connector part number for the user cable.

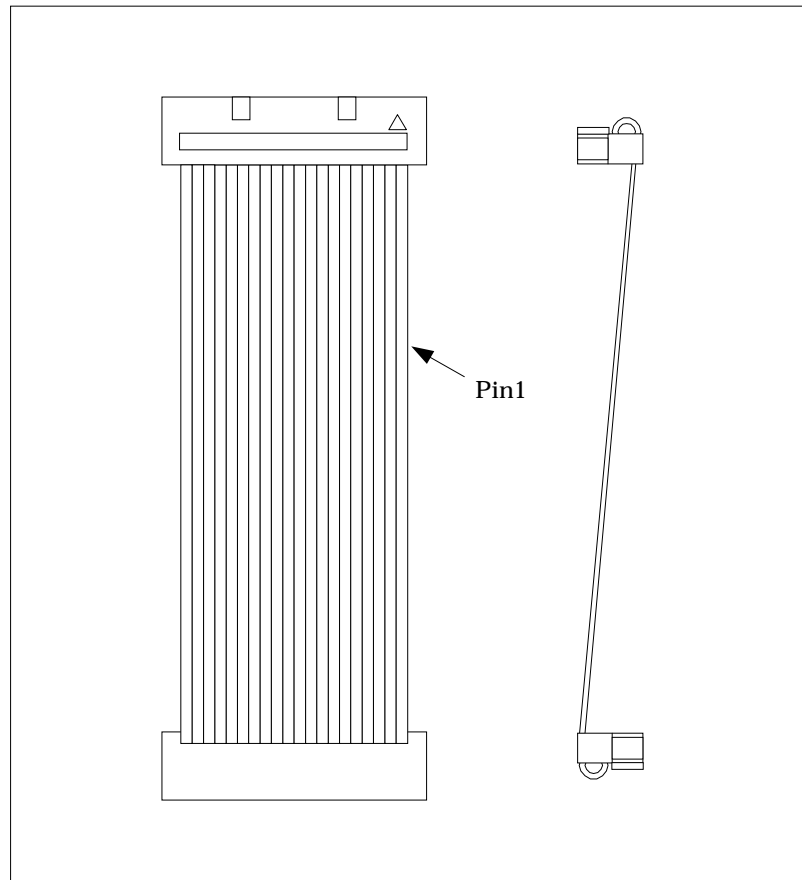


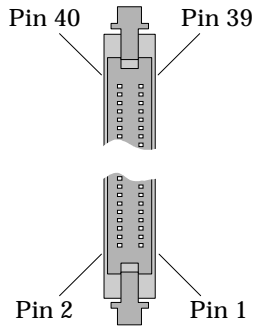
Figure A-1 User Cable Configuration

Table A-1 User Cable Connector Part Number Information

Cable	Maker	Connector Model
TCU-64164	Hirose Electric	HIF3BA-40D-2.54R

3. Pin Layout of User Connector

(1) User Connectors



* As shown at left, the user connector is a 40-pin connector with pin 1 at lower right.

* The voltage level of the user connector interface power supply can be switched by the CIPS command to either an internal power supply voltage (5V) or an external power supply voltage (3V~5V). However, the switching of the interface power supply has no relationship with the selection of the 1.5V or 3.0V versions of the MSM64162/164 ADC POD and CROSC board.

* The HALT pin is a monitoring pin that outputs an "H" level in halt mode.

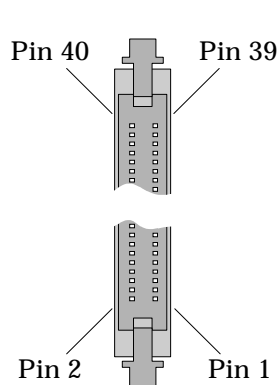
The P5.0~P5.3 pins and P6.0~P6.3 pins will be output pins when the LCD driver pins (L26~L33 or L16~L23) are set by mask option as output ports. They will output the contents of the display registers (DSPR00, DSPR01).

- * When ON is specified by the URST command, the RESET pin becomes valid. When it is valid, an "L" level input during realtime emulation will reset the evaluation board.
- * When LOUT is specified by the CCLK command, the XT pin becomes valid. When it is valid, the XT pin inputs a low-speed clock.
- * When HOUT is specified by the CCLK command, the OSC1 pin becomes valid. When it is valid, the OSC1 pin inputs a high-speed clock.
- * When the user connector interface power supply is set to be an external power supply by the CIPS command, supply a voltage from 3V to 5V on the VDD pin.

Table A-2 User Connector Pin List

Pin Number	Signal Name	Pin Number	Signal Name
1	P2.0	21	BD
2	P2.1	22	P5.0(DSPR00 a)
3	P2.2	23	P5.1(DSPR00 b)
4	P2.3	24	P5.2(DSPR00 c)
5	P3.0	25	P5.3(DSPR00 d)
6	P3.1	26	P6.0(DSPR01 a)
7	P3.2	27	P6.1(DSPR01 b)
8	P3.3	28	P6.2(DSPR01 c)
9	P4.0	29	P6.3(DSPR01 d)
10	P4.1	30	-
11	P4.2	31	RESET
12	P4.3	32	HALT
13	P0.0	33	XT
14	P0.1	34	OSC1
15	P0.2	35	-
16	P0.3	36	VDD
17	P1.0	37	VDD
18	P1.1	38	-
19	P1.2	39	GND
20	P1.3	40	GND

(2) LCD connector



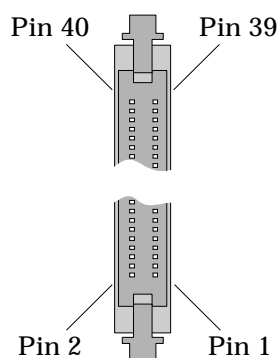
* As shown at left, the LCD connector is a 40-pin connector with pin 1 at lower right.

* The LCD connector corresponds to the L0~L33 pins of the M6416x series microcontrollers. It outputs LCD driver signals 0V to 4.5V.

Table A-3 LCD Connector Pin List

Pin Number	Signal Name	Pin Number	Signal Name
1	L0	21	L20
2	L1	22	L21
3	L2	23	L22
4	L3	24	L23
5	L4	25	L24
6	L5	26	L25
7	L6	27	L26
8	L7	28	L27
9	L8	29	L28
10	L9	30	L29
11	L10	31	L30
12	L11	32	L31
13	L12	33	L32
14	L13	34	L33
15	L14	35	-
16	L15	36	-
17	L16	37	-
18	L17	38	-
19	L18	39	-
20	L19	40	-

(3) LED connector



* As shown at left, the LED connector is a 40-pin connector with pin 1 at lower right.

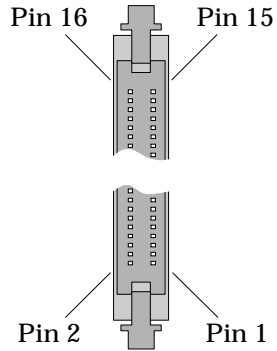
* The LED connector corresponds to the L0~L33 pins of the M6416x series microcontrollers. It outputs LED driver signals 0V to 5V.

Table A-4 LED Connector Pin List

Pin Number	Signal Name	Pin Number	Signal Name
1	L0	21	L20
2	L1	22	L21
3	L2	23	L22
4	L3	24	L23
5	L4	25	L24
6	L5	26	L25
7	L6	27	L26
8	L7	28	L27
9	L8	29	L28
10	L9	30	L29
11	L10	31	L30
12	L11	32	L31
13	L12	33	L32
14	L13	34	L33
15	L14	35	-
16	L15	36	-
17	L16	37	-
18	L17	38	-
19	L18	39	GND
20	L19	40	GND

(4) ADC connector

* The ADC connector is a 16-pin connector with pin 1 at the lower right.



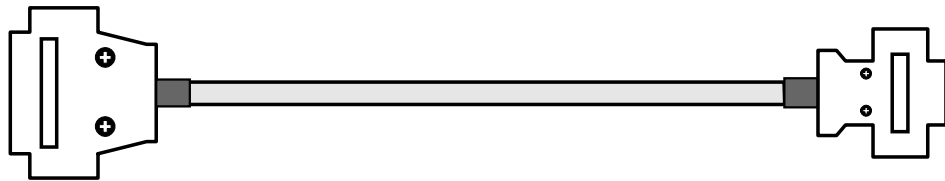
* The ADC connector connects to the accessory ADC POD.

Table A-5 ADC Connector Pin List

Pin Number	Signal Name	Pin Number	Signal Name
1	P0.0	2	P0.1
3	P0.2	4	N.C.
5	P4.3	6	\overline{XT}
7	\overline{RESET}	8	P0.3
9	VSS1 (+3.5V)	10	VSS2 (+2.0)
11	VDD1 (+1.5V)	12	VDD (+5.0V)
13	VDD (+5.0V)	14	VDD2(+3.0V)
15	VSS (GND)	16	VSS (GND)

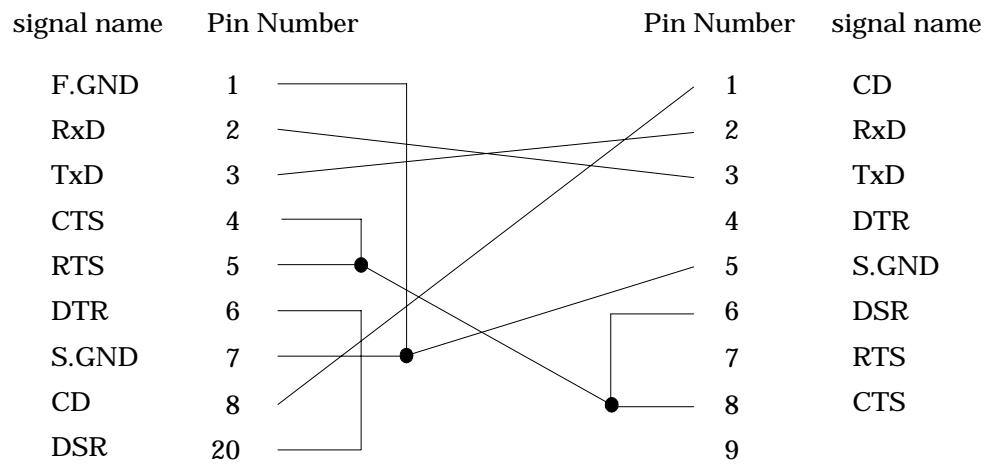
4. RS232C Cable Configuration

For IBM PC/AT computers (TCS-OMIBM)

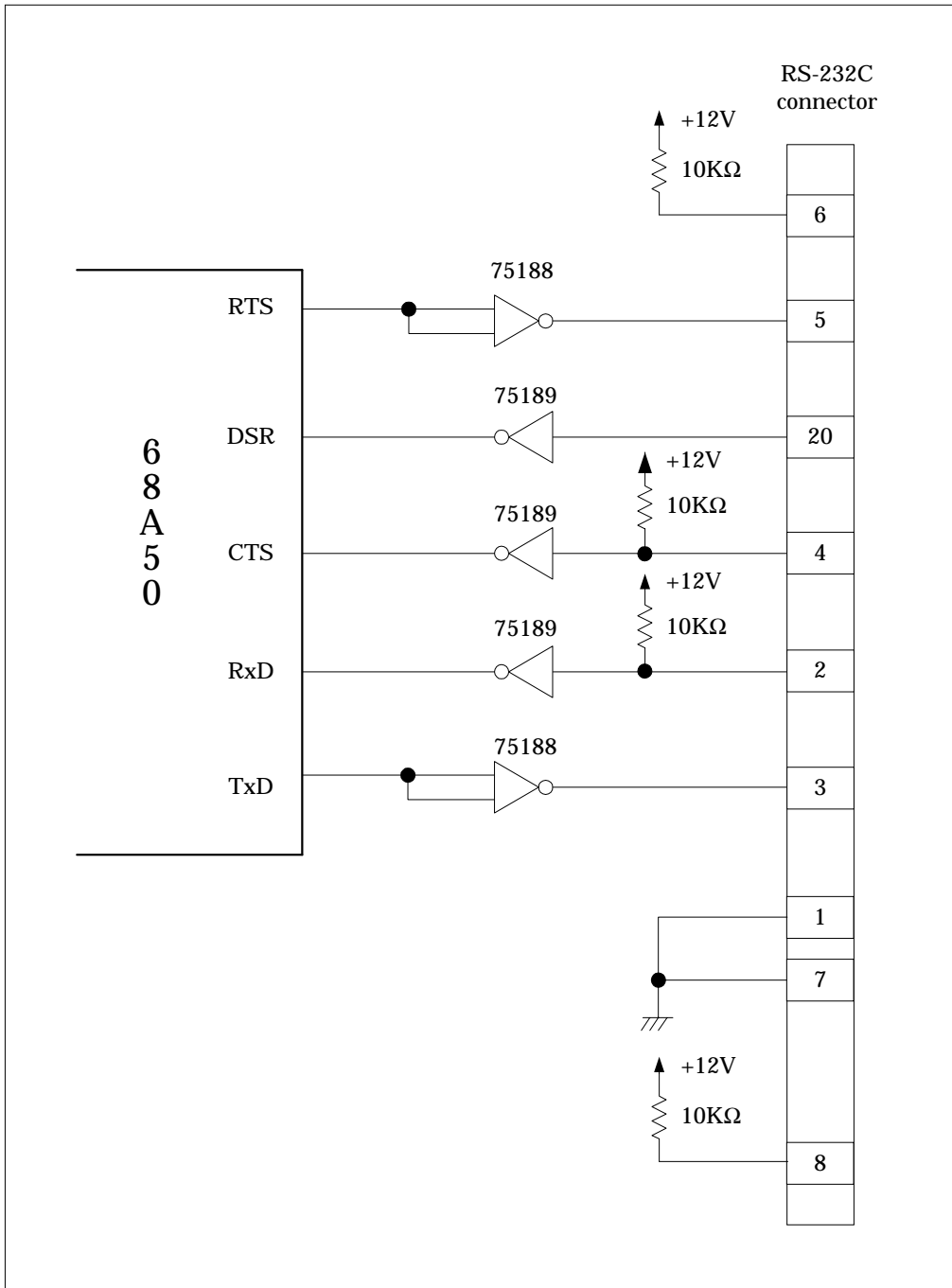


Emulator Serial Port

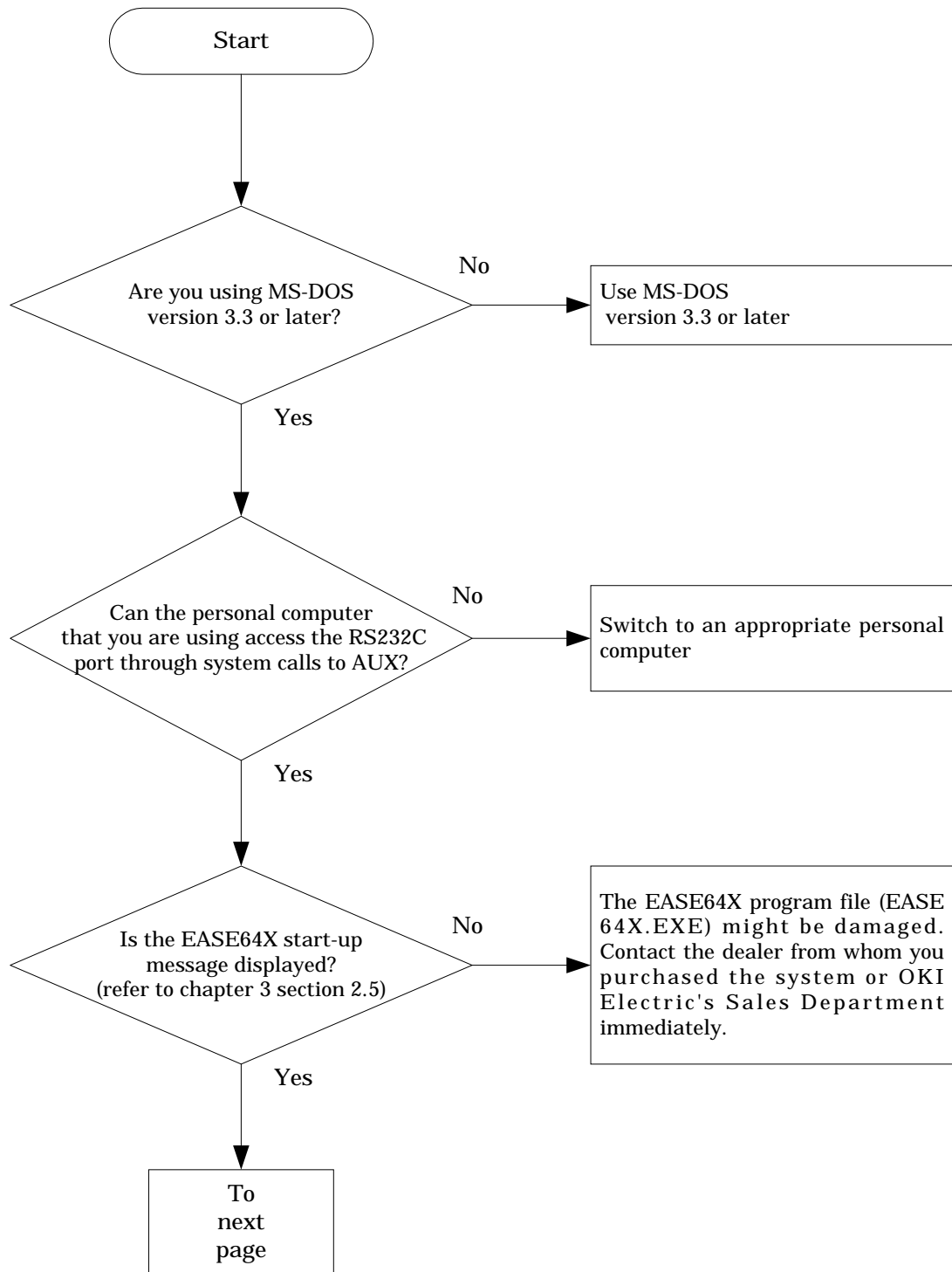
Host Computer Serial Port

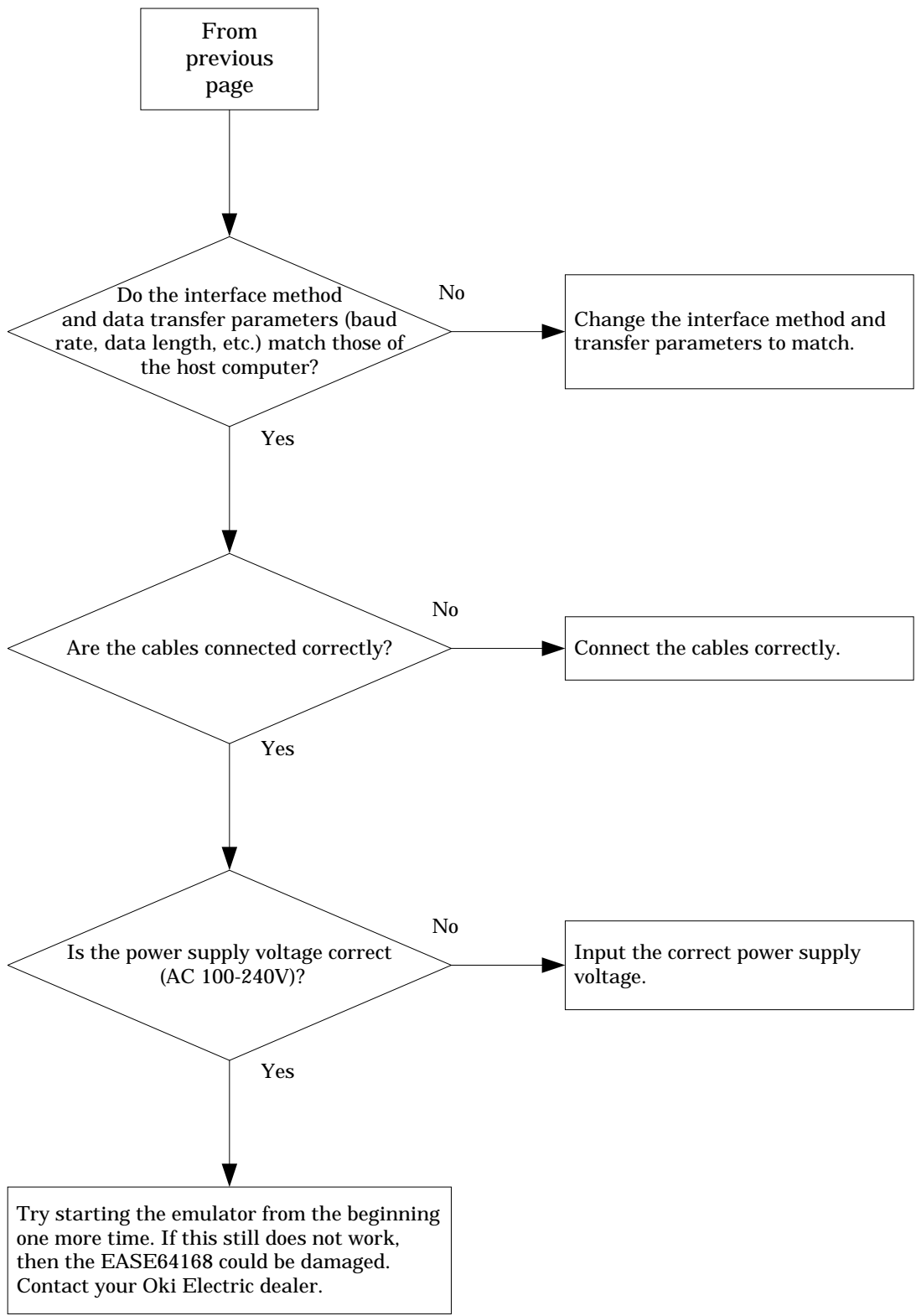


5. Emulator RS232C Interface Circuit



6. If EASE64168 Won't Start

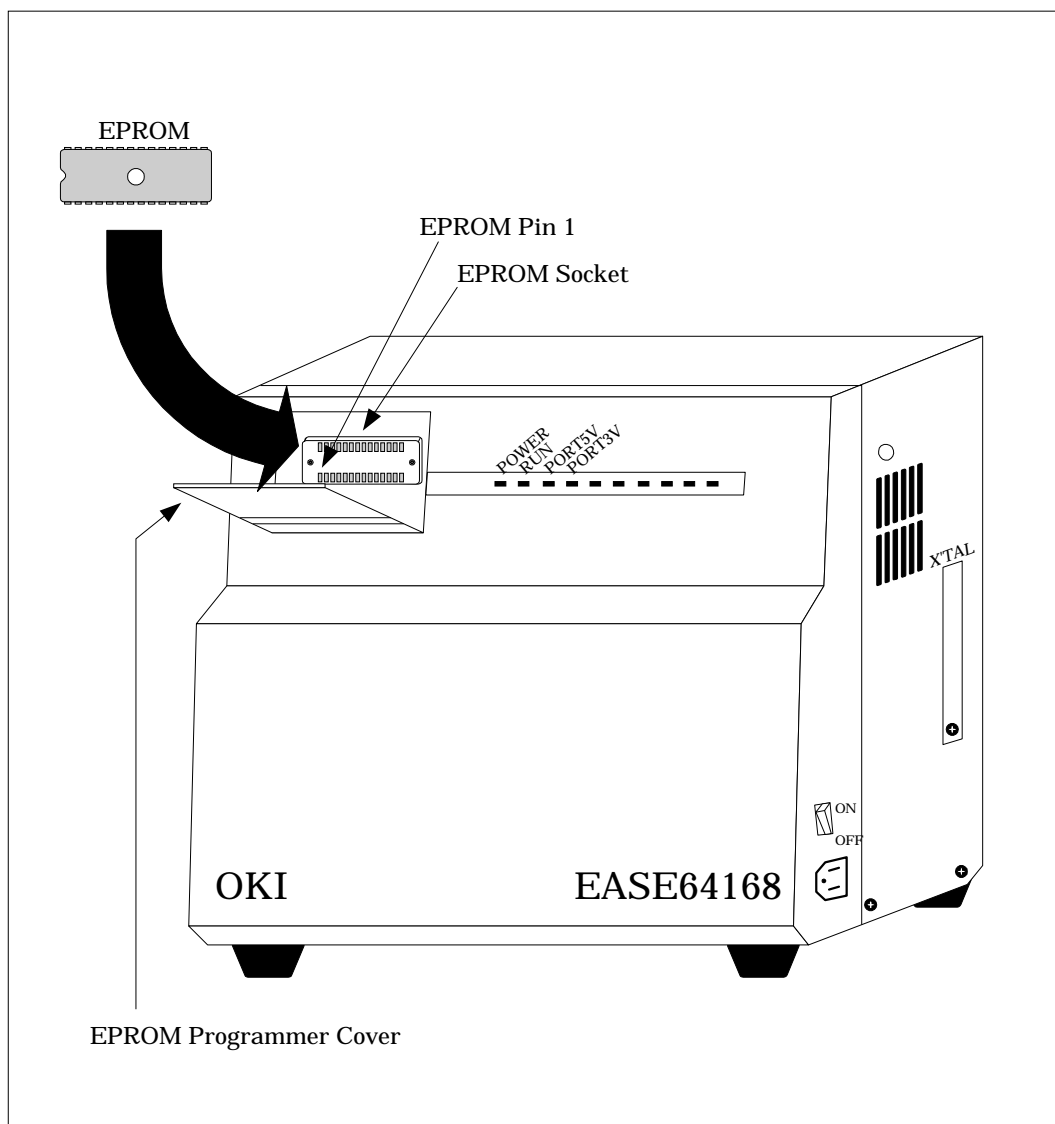




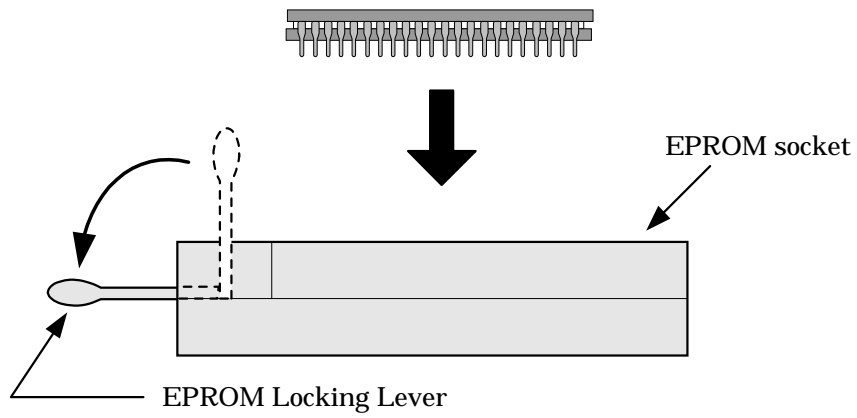
7. Mounting EPROMs

Follow the procedure below to insert an EPROM into the EASE64168's EPROM socket.

- (1) Open the EPROM programmer cover in the upper left of the EASE64168, as shown below.



(2) Next, set the EPROM to be written or read in the EPROM socket, as shown below.



To set the EPROM, insert the EPROM in the EPROM socket while the EPROM locking lever is up, and then flip the EPROM locking lever to the horizontal position.

8. Error Messages

**** Error 1: Data address error ****

The input address was not an allowable value.

**** Error 2: Data error ****

The input data value was not an allowable value.

**** Error 3: Illegal format ****

The command syntax contains an error.

**** Error 4: Command not found ****

The input command does not exist.

**** Error 5: Break status not found ****

The break status does not exist.

**** Error 6: Trace data not ready ****

No data has been traced into trace memory.

**** Error 7: File not found ****

The input file name cannot be found.

**** Error 8: Command input too long ****

The number of characters input exceeds 256.

**** Error 9: EPROM abnormal ****

Programming of the EPROM was not performed correctly.

**** Error 10: File not found ****

The specified file name cannot be found.

**** Error 11: Illegal file ****

The specified file is not in Intel HEX format.

**** Error 12: Illegal file ****

The specified HEX file contains an error.

**** Error 13: Abort ****

Communications were terminated abnormally.

**** Error 14: Cannot create file ****

The specified file cannot be created.

**** Error 15: Disk full ****

The disk is full.

**** Error 16: File write error ****

The specified file cannot be written correctly.

**** Error 17: File read error ****

The specified file cannot be read correctly.

**** Error 18: File open error ****

The specified file cannot be opened.

**** Error 19: File close error ****

The specified file cannot be closed.

**** Error 20: Illegal code accepted ****

The emulator received an illegal code.

**** Error 21: Communication buffer overflow ****

An abnormal condition occurred during communication.

**** Error 22: Already diagnostic sequence ****

A batch file is already open.

**** Error 23: List file already open ****

A list file is already open.

**** Error 24: List file already closed ****

No list file is open.

**** Warning 1: The 1/2 bias signal cannot be output ****

A 1/2 bias waveform cannot be output.

**** Warning 2: The LCD driver duty disagreed with mask option ****

The LCD duty setting differs from the loaded mask option data.

9. Hardware Specifications

Electrical Specifications

Parameter	Symbol	MAX	TYP	MIN
Internal supply voltage	VCC	5.25V	5V	4.75V
User connector interface supply voltage	VDD	5V	-	3V
Power supply input voltage (50/60Hz)	-	AC240V	-	AC 100V

Frequency characteristics (VDD=5V)

Operating Frequency		MAX	TYP	MIN
Input low speed side clock		66kHz	32.768kHz	30kHz
Input high speed side clock	MSM64164C mode	620kHz	400kHz	300kHz
	ML64168 mode	800kHz	700kHz	300kHz

Operating Conditions (without condensation and vibration shock)

Parameter	Maximum rating	
	MAX	MIN
Operating temperature	50 °C	5 °C
Operating humidity	70%	30%

10. Command Summary

Evaluations Board Access Commands		Page
1	CHIP	Set target chip
	CHIP [mnemonic]	
	mnemonic : 64162, 64164, 64168	
2	D	Display contents of target chip registers
	D or D <i>mnemonic</i>	
	<i>mnemonic :</i> PC , P0 , CAPR1 , IRQ0 B , P1D , CAPCON , IRQ1 A , P2D , TBCR , IRQ2 HL , P3D , DSPCON , BUPCON XY , P4D , CNTA , MIEF CY , SBUF , CNTB SP , SCON , ADCON0 BSR0 , FCON , ADCON1 BSR1 , BDCON , IE0 BCF , BFCON , IE1 BEF , CAPR0 , IE2	
3	C	Change contents of target chip registers
	C <i>mnemonic data</i>	
	<i>mnemonic : (note1)</i> PC (0 to 7DF or 0 to FDF) B (0 to F) , CAPR0 (0 to FF) , TBCR (0 to F) , P20CON (0 to F) A (0 to F) , CAPR1 (0 to FF) , DSPCON (0 to 3) , P21CON (0 to F) HL (0 to FF) , CAPCON (0 to 1) , IE0 (0 to F) , P22CON (0 to F) XY (0 to FF) , CNTA (0 to 79999) , IE1 (0 to F) , P23CON (0 to F) SP (0 to FF) , CNTB (0 to 3FFF) , IE2 (0, 1) , P30CON (0 to F) BSR0 (0 to F) , ADCON0 (0 to 3) , IRQ0 (0 to F) , P31CON (0 to F) BSR1 (0 to F) , ADCON1 (0 to F) , IRQ1 (0 to F) , P32CON (0 to F) BCF (0, 1) , SBUF (0 to FF) , IRQ2 (0 to F) , P33CON (0 to F) BEF (0, 1) , SCON (0 to F) , MIEF (0, 1) , P40CON (0 to F) P1D (0 to F) , FCON (0, 1) , , P41CON (0 to F) P2D (0 to F) , BDCON (0 to F) , , P42CON (0 to F) P3D (0 to F) , BFCON (0, 1 or 0 to F) , , P43CON (0 to F) P4D (0 to F) , BUPCON (0 to 3 or 0, 1) , , P01CON (0 to F)	

Evaluations Board Access Commands (cont.)			Page
4	DDSPR	Display Display Register	3-93
	DDSPR		
5	CDSPR	Change Display Register	3-71
	CDSPR <i>mnemonic</i>		
	<i>mnemonic</i> : 0~20 ... MSM64162 mode 0~30 ... MSM64164C and ML64168 mode		

Code Memory Commands		Page
1	DCM	Display Code Memory
	DCM <i>address</i> [, <i>address</i>] or DCM *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : displays entire address range	
2	CCM	Change Code Memory
	CCM <i>address</i>	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
3	FCM	Fill Code Memory
	FCM <i>address</i> , <i>address</i> [, <i>data</i>] or FCM * [, <i>data</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : fills entire address range <i>data</i> : 0 to FF	
	3-112	
4	LOD	Load Disk file program into Code Memory
	LOD <i>fname</i>	
	<i>fname</i> : [Pathname] filename [extension]	
5	SAV	Save Code Memory into Disk file
	SAV <i>fname</i> [<i>address</i> , <i>address</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode <i>fname</i> : [Pathname] filename [extension]	
3-138		

Code Memory Commands (cont.)		Page
6	VER	Verify Disk file with Code Memory
	VER <i>fname</i> [<i>address</i> , <i>address</i>]	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode <i>fname</i> : [<i>Pathname</i>] <i>filename</i> [<i>extension</i>]	
7	ASM	Line Assembler Command
	ASM <i>address</i>	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
8	DASM	Disassemble Command
	DASM <i>address</i> , [<i>address</i>] or DASM *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : displays entire address range	

Data Memory Commands		Page
1	DDM	Display Data Memory
	DDM <i>address</i> [, <i>address</i>] or DDM *	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode * : displays entire address range	
2	CDM	Change Data Memory
	CDM <i>address</i>	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode	
3	FDM	Fill Data Memory
	FDM <i>address, address</i> [, <i>data</i>] or FDM * [, <i>data</i>]	
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode * : files entire address range <i>data</i> : 0 to FF	

Emulation Commands		Page
1	STP	Step Execution
	STP [<i>number</i>] [<i>address</i>] or STP *	
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : executes 65535 steps <i>number</i> : 1 to 65535	
2	G	Realtime Emulation (continuous execution)
	G [<i>address</i>] [, <i>parm</i>]	
	<i>parm</i> : <i>address</i> [, <i>address</i> ..., <i>address</i>] RAM (data-count) BAR (data-count) <i>address</i> (count) <i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	

Break Commands			Page
1	DBC	Display Break Condition Register	3-83
	DBC		
2	SBC	Set Break Condition Register	3-139
	SBC		
3	DBS	Display Break Status	3-86
	DBS		
4	DBP	Display Break Point Bits	3-84
	DBP <i>address</i> [, <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
5	EBP	Enable Break Point Bits	3-106
	EBP <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
6	RBP	Reset Break Point Bits	3-129
	RBP <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode	
7	FBP	Fill Break Point Bits	3-111
	FBP <i>address, address</i> [, <i>data</i>] or FBP * [, <i>data</i>]		
	<i>address</i> :	0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : fills entire address range <i>data</i> : 0,1	

Trace Commands		Page	
1	DTM	Display Trace Memory	3-97
	DTM <i>-number_{step} number_{step}</i> or DTM <i>number_{TP} number_{step}</i> or DTM *		
	<i>number_{step}</i> : number of steps to go back (1~8192) <i>number_{step}</i> : number of steps to display (1~8192) <i>number_{TP}</i> : value of TP at which to start display (0~8191) * : Display the entire area of trace memory		
2	CTO	Change Trace Object	3-76
	CTO		
3	DTO	Display Trace Object	3-101
	DTO		
4	CTDM	Change Trace Data Memory	3-75
	CTDM [<i>address</i>]		
	<i>address</i> : 780 to 7FF ... MSM64162 mode 700 to 7FF ... MSM64164C mode 600 to 7FF ... ML64168 mode		
5	DTDM	Display Trace Data Memory	3-96
	DTDM		
6	STT	Set Trace Trigger	3-146
	STT		

Trace Commands (continued)			Page
7	DTT	Display Trace Trigger	3-105
	DTT		
8	RTT	Reset Trace Trigger	3-136
	RTT		
9	DTR	Display Trace Enable Bits	3-103
	DTR <i>address</i> [, <i>address</i> ..., <i>address</i>] or DTR *		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : displays entire address range		
10	ETR	Enable Trace Enable Bits	3-107
	ETR <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
11	RTR	Reset Trace Enable Bits	3-135
	RTR <i>address</i> [, <i>address</i> ..., <i>address</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
12	FTR	Fill Trace Enable Bits	3-114
	FTR <i>address</i> , <i>address</i> [, <i>data</i>] or FTR * [, <i>data</i>]		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : fills entire address range <i>data</i> : 0,1		

Trace Commands (continued)			Page
13	DTP	Display Trace Pointer	3-102
	DTP		
14	RTP	Reset Trace Pointer	3-134
	RTP		

Reset Commands			Page
1	RST	Reset System and Evaluation Chip	3-132 3-133
	RST	Reset the system	
	RST E	Reset the evaluation chip.	
2	URST	Set User Reset Terminal (on user connector)	3-153
	URST	[<i>mnemonic</i>]	
	<i>mnemonic</i>	: ON, OFF	

Performance/Coverage Commands			Page
1	DCC	Display Cycle Counter	3-87
	DCC		
2	CCC	Change Cycle Counter	3-65
	CCC [-] <i>number</i>		
	<i>number</i> : 0 to 4294967295		
3	SCT	Set Cycle Counter Trigger	3-141
	SCT		
4	DCT	Display Cycle Counter Trigger	3-90
	DCT		
5	RCT	Reset Cycle Counter Trigger	3-130
	RCT		
6	DIE	Display Instruction Executed Bits	3-94
	DIE <i>address</i> [, <i>address</i>] or DIE *		
	<i>address</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode		
	* : displays entire address range		
7	RIE	Reset Instruction Executed Bits	3-131
	RIE		

EPROM Program Commands			Page
1	TYPE	Set EPROM Type	3-152
	TYPE <i>mnemonic</i>		
	<i>mnemonic</i> : 64, 128, 256, 512		
2	PPR	Program EPROM	3-126
	PPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] PPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : programs entire address range <i>address_{EPROM}</i> : EPROM write address		
3	TPR	Transfer EPROM into Code Memory	3-149
	TPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] TPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : programs entire address range <i>address_{EPROM}</i> : EPROM read address		
4	VPR	Verify EPROM with Code Memory	3-157
	VPR <i>address_{Code}</i> , <i>address_{Code}</i> [, <i>address_{EPROM}</i>] VPR *		
	<i>address_{Code}</i> : 0 to 7DF ... MSM64162 mode 0 to FDF ... MSM64164C mode 0 to 1FDF ... ML64168 mode * : verifies entire address range <i>address_{EPROM}</i> : EPROM comparison address		

Mask Option File Commands			Page
1	LODM	Load Disk file Mask Option into System memory	3-123
	LODM <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
2	VERM	Verify Disk file with System Memory	3-156
	VERM <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
3	PPRM	Program Mask Option Data into EPROM	3-128
	PPRM		
4	TPRM	Transfer EPROM into System Memory	3-151
	TPRM		
5	VPRM	Verify EPROM with System Memory	3-159
	VPRM		

Commands for Automatic Command Execution			Page
1	BATCH	Batch Processing	3-59
	BATCH <i>fname</i>		
	<i>fname</i> : [<i>Pathname</i>] <i>filename</i> [<i>Extension</i>]		
2	PAUSE	Pause Command Input	3-125
	PAUSE		

Other Commands			Page
1	LIST	Listing (Redirect the Console output to Disk file)	3-121
	LIST <i>fname</i>		
	<i>fname</i> : [Pathname] filename [Extension]		
2	NLST	No Listing (Cancel the Console output Redirection)	3-124
	NLST		
3	>	Call OS Shell	3-56
	> DOS command		
4	CCLK	Display/Change Clock Mode	3-66
	CCLK [<i>mnemonic</i>]		
	<i>mnemonic</i> : HIN, HOUT, LIN, LOU		
5	CIPS	Display/Change Interface Power Supply	3-74
	CIPS [<i>mnemonic</i>]		
	<i>mnemonic</i> : INT, EXT		
6	EXPAND	Expand Code Memory	3-109
	EXPAND [<i>mnemonic</i>]		
	<i>mnemonic</i> : ON, OFF		
7	EXIT	Terminate the Debugger and Exit to OS	3-108
	EXIT		