

EMULOGIC[™]

ECL-3211 Microprocessor Development System User's Guide

Order Number: MAN-0801-02

PREFACE

This manual describes the features and commands of Emulogic's ECL-3211 microprocessor development system. It explains how you can apply this sophisticated system to develop, test, and modify software and hardware for your target microprocessor system.

The manual has been written for the experienced designer, as well as for the newcomer to the field of microprocessor development.

RELATED MANUALS

Besides the ECL-3211 User's Guide, there are several other manuals you may wish to reference for additional information related to ECL-3211 operation. Initially, you may find the following most useful:

Chip Supplements to the ECL-3211 User's Guide

Each supplement provides microprocessor specific information the user will need for emulating his/her target.

ECL-3211 Microprocessor Development System Installation Guide

This manual explains how to install the complete ECL-3211 system. Configurations for board installation and cabling are included, as well as a section on diagnostics. The software section of the manual contains information about formatting and initializing diskettes, and how to boot the operating system. Also described are frequently used commands. This manual describes installation procedures for modified systems as well (e.g., ECL-3211-AMR and ECL-3211-AMS).

Emulogic Relocatable Macro Cross Assembler

This guide contains reference material and procedures for developing source programs to be assembled by the Emulogic relocatable macro cross assembler. This manual also discusses the Emulogic linker, which is invoked to link object files into an executable program file.

Chip Supplements to the Cross Assembler Manual

Each supplement provides chip-specific information for invoking the Emulogic relocatable macro cross assembler and linker to produce programs that can be run on the user's target microprocessor.

Emulogic PROM Burner Utility Program User Guide

This manual explains the features of the PROM burner utility program and details the procedures for its use.

Introduction to RT-11 (DEC No. AA5281B-TC)

This manual provides you with an overview of Digital Equipment Corporation's RT-11 operating system - its features and commands.

PDP-11 Keypad Editor User's Guide (DEC No. AAH853A-TC)

The commands and functions of the keypad editor are delineated in this guide for creating and modifying disk files.

VT100 User Guide (DEC No. EK-VT100-UG-002)

You can interface with the ECL-3211 via the VT100 video display terminal. This manual explains its features and operation.

* * *

Additional manuals providing extensive, in-depth support for DEC system software are listed in Appendix B. These manuals should be viewed as library reference material for advanced system hardware and software topics.

ORGANIZATION OF THE MANUAL

The ECL-3211 User's Guide is organized as follows:

Chapter 1 Introduction

An overview of the ECL-3211 system that briefly describes the basic features of the system and the advantages of using it to develop your microprocessor system.

Chapter 2 System Description

A detailed breakdown of the ECL-3211 system into its hardware and software components.

Chapter 3 General Operating Procedures

An explanation of how you begin using the RT-11 and ECL-3211 systems, and also how you use keyboard functions and modes, read and modify the ECL-3211 screen display, and enter commands.

Chapter 4 Memory Management

A description of ECL-3211 system memory and its relationship to memory in your target system. Also an explanation of how you can map logical addresses to different physical addresses, access memory, move or relocate a memory segment to a different memory area, save memory in a disk file, load a disk file into memory, display and alter memory contents, or disassemble memory.

Chapter 5 Controlling and Monitoring Emulation

A discussion of ECL-3211 emulation management. It explains how to set up and use the system's 8 breakpoints to develop simple-to-complex controls to test and monitor program emulation on your target, without sacrificing full-speed operation. This chapter tells you how you can use switches, counters, external inputs, trace, and phantom jumps and calls to enhance testing and control. Also discussed are setting logical entities, stepping through program execution, and using symbolic debugging.

Chapter 6 Creation and Use of Command Files

A discussion of creation and use of command files for streamlining repetitive ECL-3211 tasks. Command files can be used to start up emulation, set parameters for serial emulations or run a series of test cycles. Specific ECL-3211 command file characteristics and options are discussed, including the use of Emulogic FasKeys to access command file commands and to execute user-designated command files.

Chapter 7 Command Dictionary

A comprehensive description of each ECL-3211 emulation system command, including format, function, and examples. The commands are arranged in alphabetical order for easy reference.

Chapter 8 Command Summary

A quick run-through of all the ECL-3211 emulation system commands, categorizing them according to related function.

Appendix A ECL-3211 Error Messages

A listing and brief explanation of the error messages you may encounter while running the ECL-3211 emulation system.

Appendix B DEC System Software Manuals

The library of DEC system software manuals relating to operation of the ECL-3211.

Appendix C Emulogic Chip Emulation Programs

A table listing the chips supported by Emulogic and the corresponding ECL-3211 emulation programs.

Appendix D FasKey Menu Summary

A display of all the available FasKey keypad configurations for fast entry of ECL-3211 commands.

READING THE MANUAL

This manual is designed for both new and experienced users. If you are a new user, we suggest you read Chapters 1 through 6 to familiarize yourself with ECL-3211 concepts and facilities. You may wish to refer to the "Command Summary" (Chapter 8) and "Command Dictionary" (Chapter 7) as you read these chapters, as well as the Chip Supplement to the ECL-3211 User's Guide for your target chip.

The experienced user may wish to use the "Command Dictionary" and the "Appendix D - FasKey Menu Summary" as reference tools when using the ECL-3211 system.

Once you are familiar with the ECL-3211 system, the ECL-3211 Reference Card, providing brief descriptions of the commands, can be used as a quick reference to ECL-3211 command syntax.

Preface.....	iii
CHAPTER 1 INTRODUCTION.....	1-1
The User Interface.....	1-1
Software Development.....	1-1
Memory Management.....	1-2
Full-Speed Emulation.....	1-2
CHAPTER 2 SYSTEM DESCRIPTION.....	2-1
General Hardware Configuration.....	2-1
Display and Keyboard.....	2-1
Electronics Module.....	2-2
Disk Storage Devices.....	2-2
Hardware Options.....	2-3
Emulation Support Package.....	2-3
High-speed Memory.....	2-4
EMUNET Multi-User System.....	2-4
Hardware Summary.....	2-4
General Software Configuration.....	2-4
Cross Assemblers and Linkers.....	2-5
Emulation Software.....	2-5
Software Options.....	2-6
Prom Burner Utility.....	2-6
High-Level Compilers.....	2-6
Communication Software.....	2-6
Utility Software.....	2-7
Software Summary.....	2-7
CHAPTER 3 GENERAL OPERATING PROCEDURES.....	3-1
Bringing up the System.....	3-1
Running the Emulation System.....	3-2
Selecting Start-up Command and Data Files.....	3-3
Command File Options.....	3-3
Data File Options.....	3-3
Examples to Illustrate the Options.....	3-4
Emulation System Screen Displays	3-5
Modes of Operation.....	3-13
Alter Mode.....	3-13
Command Mode.....	3-13
Emulate Mode.....	3-14
Mapping Mode.....	3-14
Step Mode.....	3-14
Special Key Functions.....	3-15
Use of the FASKEY Capabilities.....	3-16
FASKEY 1 Menu.....	3-17
FASKEY 2 Menu.....	3-18
Programmable FASKEYS.....	3-19
Command File Pause Keys.....	3-19

CHAPTER 4	MEMORY MANAGEMENT.....	4-1
	Introduction.....	4-2
	ECL-3211 Memory Characteristics.....	4-3
	External Memory.....	4-3
	ECL-3211 Memory.....	4-3
	Memory Speed.....	4-4
	Memory Mapping.....	4-4
	Map Commands.....	4-6
	General Map Commands.....	4-6
	Internal Map Commands.....	4-8
	High-Speed Memory Map Commands.....	4-9
	Saving Emulation Parameters.....	4-12
	ECL-3211 Map Display Facilities.....	4-12
	Viewing the Contents of Memory.....	4-13
	Altering the Contents of Memory.....	4-15
	Viewing the Contents of Memory in ASCII.....	4-16
	Disassembling the Contents of Memory.....	4-17
	Saving Data Memory.....	4-19
	Loading Disk Files into Memory.....	4-20
	Relocating Mapped Memory.....	4-20
	Moving Memory Segments.....	4-21
CHAPTER 5	CONTROLLING AND MONITORING EMULATION.....	5-1
	Introduction.....	5-1
	Breakpoints.....	5-2
	Breakpoint Structure.....	5-2
	Setting Actions and Conditions.....	5-4
	Controlling Program Flow.....	5-8
	Halt Action.....	5-8
	Pause Action.....	5-9
	Phantom Action.....	5-9
	Command File Action.....	5-10
	Setting and Clearing Switches.....	5-10
	Controlling Counters.....	5-11
	Controlling the Trace Buffers.....	5-11
	Breakpoint Conditions.....	5-11
	System Conditions.....	5-12
	Target Conditions.....	5-12
	Bit-Test Conditions.....	5-13
	Word-Test Conditions.....	5-13
	Turning Breakpoints On and Off.....	5-13
	Breakpoint Relationships.....	5-14
	Logical.....	5-14
	Priority.....	5-16
	Direct Control.....	5-17

System Signals.....	5-18
Switches/Triggers.....	5-18
Counters.....	5-19
Range of Counter.....	5-20
Counter Modes.....	5-20
Using Counters with Breakpoints.....	5-21
External Inputs.....	5-22
Phantom Programs.....	5-22
Phantom Jumps.....	5-22
Phantom Calls.....	5-23
Conditional Phantom Jumps and Calls - a Warning.....	5-23
The Trace Buffer.....	5-24
Setting the Clock Source and Frequency.....	5-26
Setting Logical Entities.....	5-27
Starting Emulation.....	5-27
Stepping Through a Program.....	5-28
Using the Symbolic Debugger.....	5-28
 CHAPTER 6 CREATION AND USE OF COMMAND FILES.....	6-1
Introduction.....	6-1
Creating Command Files.....	6-3
Creating a Command File under RT-11.....	6-3
To Run a Stored RT-11 Command File.....	6-4
Creating ECL-3211 Command Files.....	6-4
Creating an ECL-3211 Command File Under RT-11 Edit.....	6-5
Creating an ECL-3211 Command File Using the ECL-3211 LOG Command.....	6-5
Running a Stored ECL-3211 Command File.....	6-6
ECL-3211 Command File Characteristics and Options....	6-8
Pauses	
The Command File Pause (CF/P) Command.....	6-8
To Interrupt Execution of a Command File.....	6-8
To Pause during Emulation.....	6-9
Command File Error Display.....	6-9
Use of Control Keys During a Command File Pause.....	6-9
Command File Comment Lines.....	6-9
Prompting For Command Line Input.....	6-10
Use of the ESC Key.....	6-10
Use of the Keyboard Bell by a Command File...6-10	
Changing the Command File Extension Default..6-11	
Viewing the Contents of a Command File.....6-11	
Terminating Command File Execution.....6-11	
Calling a Command File as a Breakpoint Action.....	6-12
Glossary of ECL-3211 Command File Mnemonics.....	6-14
Programmable FASKEY Access to Command Files.....	6-16
Command Files Used to Run Diagnostics - An Example using Control and Command Files	6-17

CHAPTER 7 ECL-3211 MDS COMMAND DICTIONARY.....7-1

 Command Usage.....7-1

 Command Syntax Conventions.....7-1

 ECL-3211 Commands (in alphabetical order).....7-4

CHAPTER 8 COMMAND SUMMARY.....8-1

 Memory Management Commands.....8-1

 Emulation Management Commands.....8-5

 Screen Management Commands.....8-8

 Command File Management Commands.....8-9

 Miscellaneous Commands.....8-12

APPENDIX A ECL-3211 EMULATION SYSTEM ERROR MESSAGES.....A-1

APPENDIX B DEC SYSTEM SOFTWARE MANUALS.....B-1

APPENDIX C EMULOGIC MICROPROCESSOR EMULATION SOFTWARE.....C-1

APPENDIX D FASKEY MENU SUMMARY.....D-1

APPENDIX E LIST OF FIGURES.....E-1

APPENDIX F LIST OF TABLES.....F-1

CHAPTER 1

INTRODUCTION

The ECL-3211 microprocessor development system provides easy-to-use, and yet, sophisticated facilities for software development, and for full-speed, in-circuit emulation of microprocessor-based applications.

As a universal development system, the ECL-3211 is capable of emulating a variety of microprocessors. Its software-driven design enables you to change development support from one microprocessor to another by loading software for the new chip and plugging in an Emulation Support Package pod for that microprocessor family. No hardware, other than the pod, need be changed to support different microprocessors.

The system is built around Digital Equipment Corporation (DEC) hardware and software. It utilizes PDP-based, 16-bit processors, and standard DEC terminals and peripherals. DEC's RT-11 operating system and utilities provide support for your program development.

THE USER INTERFACE

Your interface with the ECL-3211 is via the system's video display terminal. When the system is processing in the RT-11 keyboard monitor, the terminal keyboard accepts any RT-11 command. If you are running ECL-3211 emulation software, you can enter ECL-3211 commands, data, and cursor directives.

When you load emulation software, the terminal changes its display to a split-screen format to monitor your target system and the program you execute during emulation. You can modify memory contents and register values directly on the screen via ALTER mode. (The system has several operation modes that determine which functions you may perform.)

SOFTWARE DEVELOPMENT

You can develop your software on the system's PDP-11 computer by using the file management and text editing features of the DEC RT-11 operating system. Emulogic provides a relocatable macro cross assembler for each supported microprocessor. PASCAL and C compilers are also available for some microprocessors.

Emulogic's PROM Burner Utility allows you to convert debugged object code to a variety of formats compatible with various PROM programmers.

MEMORY MANAGEMENT

The ECL-3211 map facility enables you to use ECL-3211 memory and Emulogic high-speed memory in conjunction with the memory in your target system. The high-speed memory is available to ensure emulation at the full-rated speed of your target chip.

Memory contents can be altered directly at the keyboard, relocated to a new logical address range, stored on disk, or reloaded from disk.

The ROM simulation feature allows you to restrict access to a segment of memory to read-only operations during emulation. When you are not emulating, you can still modify the memory contents at the keyboard.

The Disassemble Memory facility is also available to enable the user to disassemble memory contents.

FULL-SPEED EMULATION

To emulate your target system, you connect the ECL-3211 system with a chip-specific Emulation Support Package pod directly to the target board in place of the target microprocessor. The ECL-3211 effectively becomes the target microprocessor. By using Emulogic high-speed memory, you can emulate at the full-rated speed of the target chip. Various facilities are available to monitor and control emulation without degrading target performance.

Program flow can be controlled by setting one to eight 78-channel breakpoints. Each breakpoint may be defined as a logical function: it performs a specified set of actions when the logical state of the system equals the logical state described by the breakpoint's logical conditions. Each condition can be set to 1 (asserted = true), 0 (disasserted = false), or X (don't care). All breakpoints are transparent to the target system. They do not steal any time, space, or interrupts from the target.

Breakpoints may be used independently, or they may be concatenated, using logical switches, to increase the number of possible trap conditions. The trap conditions you can define are literally endless.

The "phantom" breakpoint feature adds even more versatility to your development tools. This feature allows you to jump to coded instructions, or patches, that you have inserted in memory without changing any in-line code. If need be, you can access up to eight patches, one per breakpoint.

The 511-record trace buffer stores address, data, port, status and control line information. The trace functions as

a logical state analyzer, monitoring the status of the target chip, as well as the status of eight internal inputs (located on the pod). By using breakpoints to turn the trace on and off, you can perform pre- and post-triggering to capture only relevant data.

Two trigger outputs on the pod can be used to trigger or synchronize external instruments such as oscilloscopes, or stand-alone timing analyzers.

Program measurement is provided by two 31-bit counters that may be used as timers, cycle counters, instruction counters, or breakpoint-tripped counters. Since these function at full speed, no correction factors are needed.

Memory images, trace records, externally generated serial data, and command sequences can be stored in disk files for analysis or for later emulations.

Symbolic debugging allows the user to substitute symbols for hex values in commands.

Use of the Emulogic FasKey facility allows coded keystroke rapid entry of ECL-3211 commands and keystroke access to user-designated command files.

All of these features are explained in detail in the following chapters.

SYSTEM CONFIGURATION AND FUNCTION

The Emulogic ECL-3211 Microprocessor Development System (MDS) consists of several hardware and software components. The modular packaging of these components permits you to customize a system that fits a particular set of requirements. The ECL-3211 MDS is based on the popular Digital Equipment Corporation (DEC) LSI-11 family of microprocessors. Emulogic provides hardware and software packaged for a variety of LSI-11 system configurations. This chapter describes the types and functions of the ECL-3211 MDS modules.

GENERAL HARDWARE CONFIGURATION

Typically, an ECL-3211 MDS station consists of:

- . An ASCII-type video display and keyboard unit
- . An electronics module
- . A magnetic disk data storage device

The features of these items, as supplied by Emulogic, Inc. are described in the following sections.

DISPLAY AND KEYBOARD

The display module consists of a molded cabinet housing an 80-character by 24-line video display screen. At the rear of the cabinet are a power switch, fuse, and board-edge connections for the keyboard and electronics modules. The keyboard consists of a sloped, detached molded case on which are mounted two keypads. The large central keypad contains mostly the alphabetic, numeric and special character keys. There are also terminal and screen cursor control keys in this keypad. The layout of keys in the central keypad is similar to the common typewriter key arrangement. A small keypad is located on the right of the keyboard. The keys in this keypad have been programmed to perform special ECL-3211 "FasKey" functions which enable quick entry of ECL-3211 commands. Operation and maintenance for the display and keyboard modules are described in the terminal operator's guide packed with your system. Appendix D at the back of this manual displays the special keypad configurations for the "FasKey" command entry mode.

ELECTRONICS MODULE

The electronics module is housed in a compact rectangular cabinet. The front panel is plain and contains no controls or indicators. At the back of the cabinet are a power switch, a fuse, and an access door to the circuit boards containing the major system electronics. The boards are held in place and connected electrically by the Q-BUS (LSI-11 bus) "backplane" assembly which faces the rear door.

The boards are grouped on the backplane by function. As viewed from the rear of the cabinet, these groups occupy upper left, central left, upper right, and lower full-width areas. The board groups represent, respectively:

- . DEC LSI-11 family general-purpose computer circuitry (4-5 boards, upper left)
- . EMUNET DMA and DATACOM circuitry (optional, 2 boards, central left)
- . EMULOGIC high-speed memory (HSM) (optional, 1-4 boards, upper right)
- . EMULOGIC Map and Control circuitry (2 boards, lower full-width).

The half-width boards (called dual-size) each occupy two backplane connectors of a backplane slot. Full-width boards (quad-size) occupy four connectors each. In a normal configuration, only the left side connectors supply data and address bus lines; the right side connectors supply direct-current power only.

Multi-pin connector sockets on the outward (nearest the cabinet door) edges of some boards accept connecting cables to other boards and to disk or accessory modules. An adjustable opening at the door bottom permits necessary cables to extend beneath the door when it is closed.

DISK STORAGE DEVICES

One or more magnetic disk storage devices can be connected to the DEC controller board in the electronics module.

The devices available are flexible diskette (floppy disk) drives or hard-platter cartridge disk drives or both.

The device types offer a wide range of cost, data storage quantity, access speed, and media portability options.

The diskette drive offered by Emulogic, Inc. is a 2-unit model. The molded front panel has a wide horizontal recess in which the two access doors are mounted side-by-side. Each latching door opens to accept a standard 8-inch diskette. When a diskette has been inserted and the door closed, the loaded unit is ready for access. On the back of the cabinet are a power switch and a 40-wire ribbon cable to join the drive to the DEC controller board in the electronics module.

The hard-platter cartridge drive offered by Emulogic, Inc. has a molded front panel containing a long horizontal slot-like recess concealing the air intake. A smaller horizontal recess above the intake contains two pushbuttons (yellow) and two status lights (white, red). A large latching access door on the top of the cabinet opens into a spindle well to receive the disk cartridge. A cartridge must be inserted, the access door closed, and the "LOAD" button pressed to ready the drive for access. On the back of the cabinet are a power circuit-breaker and multi-pin sockets to connect the data and control cable from the disk controller in the electronics module.

Instructions and maintenance information for the particular disk drive or drives on your system are contained in the operator's guides provided with those modules.

HARDWARE OPTIONS

Your ECL-3211 MDS station may have one or more Emulogic, Inc. hardware options to expand the capabilities of the system. A modular Emulation Support Package (ESP) pod can be added to permit real-time, full-function microprocessor emulation. High-speed memory boards can be added in 16 k-byte increments to allow full-speed emulation using large development programs. The EMUNET multi-user hardware lets up to 15 ECL-3211 MDS stations use the disk resources of a PDP-11 or VAX 11/7XX or compatible host computer.

EMULATION SUPPORT PACKAGE

The EMULOGIC Emulation Support Package (ESP) is a custom microprocessor emulation kit. It includes a microprocessor-specific pod with connecting cables and a copy of appropriate driving software. The ESP pod is the emulation hardware interface required to link the basic ECL-3211 MDS station to a target system. The pod consists of a molded case containing the microprocessor (chip) emulator, chip support, and signal sensor circuitry. Along one outer edge is a row of BNC-type connectors; two for trigger output signals and eight for input sense signals. From one end of the pod extends a short twisted-pair ribbon cable, terminated with a DIP plug, to be inserted in the microprocessor socket of the target circuitry. From the opposite end of the pod extend a pair of long 40-wire twisted-pair cables to be inserted in sockets of the EMULOGIC Map and Control boards in the electronics module. The ESP pod allows continuous reading of the microprocessor status as well as some target conditions. ESPs are available for many commercially available microprocessors.

HIGH-SPEED MEMORY

EMULOGIC high-speed memory (HSM) modules are dual-size boards loaded with 16 or 32 k-byte blocks of random access memory (RAM). HSM boards are mounted in the electronics module of the ECL-3211 MDS and are "mapped" via the system's mapping facility. The high-speed read and write access cycles of these memory modules permit full-speed emulation of any available microprocessor. In addition, one or more boards of HSM can be designated as read-only memory (ROM) to simulate running the development program from an in-target fixed program store. When creating and testing large microprocessor development programs, particularly in 16- and 32-bit applications, HSM modules allow fast and realistic emulation conditions.

EMUNET MULTI-USER SYSTEM

Emulogic's EMUNET system hardware permits several ECL-3211 MDS stations to share disk resources on a "host" computer system. The system hardware consists of a pair of dual-size boards and cabling to join the boards to each other and to link the stations (satellites) and host. (Emulogic, Inc. supplies the host-resident EMUNET hardware and software packages for RSX and VAX/VMS operating systems as well as some configurations of PDP-11 computer systems.) When an ECL-3211 MDS station is used as a multi-user satellite, its local disk storage requirements can be reduced or eliminated. Thus, for sites where multiple ECL-3211 MDS stations are used, the EMUNET multi-user system provides an added measure of resource efficiency and data manageability.

HARDWARE SUMMARY

The display and keyboard, electronics, and disk drive modules form the core of the EMULOGIC ECL-3211 Microprocessor Development System. With appropriate software, this basic system can be used to develop control software for most of the currently available microprocessors. Addition of an emulation support package (ESP) and emulation software allows you to test the control program in real-time with your target hardware. Emulogic's high-speed memory (HSM) permits full-speed emulation of large and very large programs with the ECL-3211 MDS. For Emulogic, Inc. system users with several ECL-3211 MDS stations, cost savings in disk resources and time savings in data file management may be realized through the EMUNET multi-user hardware and software options.

GENERAL SOFTWARE CONFIGURATION

There are several modules of the ECL-3211 MDS system software, each of which provides specific control and support capability. There are the Emulogic Macro Cross Assemblers and Linkers that provide mnemonic coding capability -- identical to the language provided by the microprocessors'

original manufacturers -- for microprocessor system development. There are ECL-3211 MDS microprocessor emulation programs which -- with an appropriate ESP -- allow full speed emulation and run-time monitoring as well as interactive testing and debugging utilities. Modular format offers ease of installation and maintenance, cost effectiveness, and conservation of system resources. The following sections briefly describe typical EMULOGIC software modules installed in an ECL-3211 MDS.

CROSS ASSEMBLERS AND LINKERS

The purpose of the EMULOGIC Macro Cross Assembler and Linker packages is to allow you to create microprocessor system development programs

- . Using the mnemonic assembler language originally designed for the microprocessor
- . Taking advantage of the power and built-in features of the DEC MACRO-11 language and assembler
- . Assembling and linking numerous sub-programs (sections) into a single loadable module.

Familiar mnemonics, combined with macro-level syntax, speeds development time and reduces the frequency of coding errors. The assembly and linking processes produce fully-resolved LDA-format code which can be loaded and executed in the ECL-3211 MDS. Thus, the EMULOGIC Macro Cross Assembler and Linker package carries the control programs for microprocessor system development from the design phase to "live" emulation readiness. The Cross Assemblers and Linkers run within the DEC RT-11 operating system environment or under RSX and VMS using RT-11 emulators.

EMULATION SOFTWARE

The most visible of Emulogic, Inc.'s software modules, the ECL-3211 MDS emulation software is responsible for control of all emulation related functions. This software processes all set-up commands prior to running an actual emulation. It handles all ECL-3211 MDS commands, including those entered interactively and those contained within "command files." Emulation software manages the mapping facility, interpretation of signals and data passing to and from the ESP pod, and control of emulation breakpoints. You can request memory disassembly and instruction traces through the emulation software. Finally, the emulation software provides the means for pre- and post-emulation file management. Like the Cross Assemblers and Linkers, the emulation software is designed to run within the DEC RT-11 operating system environment. The remaining chapters of this manual are primarily devoted to instructing you in the capabilities and use of the ECL-3211 MDS and the emulation software.

SOFTWARE OPTIONS

There are software options that allow program development in high-level languages (C and Pascal), conversion of memory image files to PROM or EPROM formats, and inter-system data communication. These software kits permit additional tailoring of your system to meet particular system configurations or production needs. The available EMULOGIC software options for the ECL-3211 MDS are described briefly in the following sections.

PROM BURNER UTILITY

The Prom Burner utility program offered by Emulogic, Inc. converts completed microprocessor memory image files to suitable formats for ROM chip encoding. After completing the assembly and linking of a program (usually after successful emulation, as well), the Prom Burner utility prepares the LDA-format code for one of the many available ROM, PROM, or EPROM devices. This EMULOGIC software utility produces a "ROM map," accepts parameters to define the characteristics of the device of choice, and -- finally -- generates a load-ready image for burning into the ROM device. The program thus prepared can be fed to any commercially available PROM encoding devices over the RS-232 port of the ECL-3211 MDS. Details of this utility and its use are provided in the Emulogic PROM Burner Utility User's Guide.

HIGH-LEVEL COMPILERS

Emulogic, Inc. offers the C and Pascal programming language compilers by Whitesmith, Ltd. for system development of selected microprocessors. Using the power of these high-level languages, you can produce LDA-format load modules for emulation or ROM encoding. Particularly when system development time rather than compactness of code is of higher priority, the efficiency and convenience of high-level language could offer productivity gains. Your EMULOGIC product representative can advise you if there are C and Pascal cross compilers available for the microprocessor model you have chosen for your target system.

COMMUNICATION SOFTWARE

Emulogic, Inc.'s EMUNET software package resides in a host computer and includes a control program to manage high-speed serial data transfers between the host's disks and the multi-user hardware network. The primary function of this program is to process disk access requests (for storage or retrieval) transmitted by the ECL-3211 MDS stations (satellites). Processing includes keeping track of each request, translating virtual file requests to actual data file designations,

and passing requests and data between the EMUNET and the disk control mechanism of the host. The EMUNET hardware and software make it possible for all ECL-3211 MDS satellites on the network to use host disk space, transparently accessing their files through the concept of "virtual" data volumes. The EMUNET Multi-User system is described in detail in the Emulogic EMUNET Multi-User System Guide.

UTILITY SOFTWARE

A number of EMULOGIC utility programs are available to simplify microprocessor system development activities. In this category are the FXC file transmit-and-receive, TERM terminal emulator, and BINHEX data converter utilities. Using FXC software, you can send data to or receive data from practically any computer system that handles ASCII data. The TERM program allows an ECL-3211 MDS station connected to a host computer to act as a normal terminal of that host (that is, the MDS can support log-on, log-off, and all general system functions and services available under the host computer). BINHEX is a program which converts ASCII data to binary or LDA-format or LDA to ASCII. These utilities offer the means to pass microprocessor development programs in either LDA- or ASCII-format between computers, including the ECL-3211 MDS. In addition, you can use an ECL-3211 MDS station as the terminal of a larger host computer.

SOFTWARE SUMMARY

Much of the power of the ECL-3211 MDS is derived from EMULOGIC software packages. The principle packages, Macro Cross Assemblers and Linkers and the emulation software, support development of microprocessor memory image files as well as comprehensive real-time emulation and monitoring. The PROM Burner software converts programs from LDA-format to ROM-ready code. For selected microprocessors, Emulogic, Inc. offers C and Pascal cross compilers to make programming even more convenient. EMUNET software drives the multi-user system capability, providing cost savings and enhanced data management for sites with multiple ECL-3211 MDS stations. The FXC, TERM, and BINHEX utilities make possible conversion or transmission of programs and data between systems. You can select the main and optional EMULOGIC software packages that will properly support the application of your MDS stations.

GENERAL OPERATING PROCEDURES

This chapter discusses the procedures for bringing up your ECL-3211 in the DEC RT-11 operating system and running the ECL-3211 emulation software.

The emulation system has its own unique screen display and modes of operation. These are described, along with the special keyboard functions you may use while running the emulation system.

BRINGING UP THE SYSTEM

The ECL-3211 emulation system is started by powering up the hardware and then bootstrapping the RT-11 operating system. Power up the hardware in this order:

- (1) Video terminal
- (2) Data storage device
- (3) Electronics module

NOTE: If you plug all three of these into a power strip, you need only turn on the power strip.

Next, install the RT-11 system diskette in unit 0 (the lefthand door), if using the RX02 double density floppy diskette drive. If using the RL02 hard disk drive, mount the RT-11 system disk. Once the disk or diskette is mounted, the ECL-3211 system will automatically bootstrap the RT-11 operating system.

NOTE: If for some reason the system does not bring up the RT-11 system, press the BREAK key and type 173000G after the @ prompt displayed on the video screen.

When the software is up and running, it displays:

```
RT-11SJ  V05.00
```

System configuration information may appear with this system message, depending upon the contents of the indirect command file STARTS.COM which is automatically invoked when RT-11 is bootstrapped. For example your terminal might display

RT-11SJ V05.00

.SET TT: NOQUIET

.SET TT: SCOPE

.SET EDIT KED

.ASS LS LP

.SET LP: CSR=176500

.SET LP: VECTOR=300

.SET KMON IND

.ASS DY0: HLP:

which shows the various system configuration parameters set up by STARTS.COM.

Note the command ".ASS DY0: HLP:". You must assign the physical device which contains file LOXX00.HLP, the Emulogic Help file.

The . prompt indicates that the interactive keyboard monitor is now functioning. Any keyboard monitor command may be entered in response to this prompt. Refer to RT-11 documentation for further information about the RT-11 system features and facilities.

RUNNING THE EMULATION SYSTEM

To start up the ECL-3211 emulation system from the keyboard monitor, enter:

RUN LOXX00.SAV<cr>

where:

LOXX00 is the chip emulation program name (maximum 6 characters)

<cr> is a carriage return

For example,

RUN L00100.SAV<cr>

executes the emulation program for the 8048 microprocessor.

NOTE: If you are operating with RT-11 Version 5 software, you may omit the .RUN portion of the command and simply enter the Emulogic program name followed by a carriage return.

All L-series emulation program files with the

.SAV extension are designed to run on the RT-11 operating system. Emulogic has named these files with L-numbers for our own filing purposes. You may rename them if you wish.

Refer to Appendix C for a list of available microprocessor-specific emulation programs.

SELECTING START-UP COMMAND AND DATA FILES

The next step in bringing up the system occurs with the screen display of the ECL-3211 prompt:

(* or COMMAND FILE)(,* or DATA FILE)<cr>

Your response to this prompt establishes ECL-3211 start-up conditions. Your options are as follows:

If you respond with a <cr>:

If you respond to the prompt by entering a carriage return, the ECL-3211 software will run using both default start-up command and data files.

If you respond with the <cr> to the above prompt, and you have not established a default start-up command file (LOXX00.COM), the ECL-3211 will bring up an unfilled emulation display screen and will display the message "COMMAND FILE NOT FOUND - LOXX00.COM". The system is ready for you to initiate start-up from the command line.

COMMAND FILE OPTIONS:

You have a choice of two responses for command file, if you have not answered with a <cr>:

If you enter a named command file, the system will use the commands in the named command file for start-up. (Refer to Figure 3.1.)

If you enter an "*" for command file option, the system takes the command file status (including the active command file name, command file stack information, and default command file extension) from the appropriate data file and invokes this status at start-up. (In other words, you start where you left off.)

DATA FILE OPTIONS:

You have a choice of two responses for data file, if you have not answered the prompt with a <cr>:

If you enter a comma followed by a named data file (,FILE.DAT) , the system loads the named data file emulation parameters and data for start-up. (Refer to Figure 3.1.)

If you enter ",*" for data file option, the system creates a blank data file - LOXX00.DAT. You start up with a clean slate.



```
.RUN L00500
(* or COMMAND FILE)(,* or DATA FILE)<cr>
SORT.COM,SORT.DAT_
```

FIGURE 3.1 INITIAL START-UP PROMPT SCREEN

Some examples to illustrate the options:

Example 1 - To start up with a clean slate:

,*<cr>

Example 2 - To start up with emulation parameters where you left off:

*<cr>

Example 3 - To start with named files:

TEST1.COM,VARY2.DAT<cr>

Example 4 - To start with default files LOXX00.COM and LOXX00.DAT:

<cr>

Emulation System Screen Display

Following response to the start-up prompt, the emulation system screen display appears. The program is initially in the COMMAND mode, as indicated in the lower right-hand corner of the screen. The cursor appears on the command input line, labeled "C:" in the lower left-hand corner of the screen. You may now enter any ECL-3211 command. Refer to Chapter 7 - The Command Dictionary for a complete description of each command and its syntax.

The ECL-3211 emulation system screen display lists the current status of the target microprocessor in six specific areas on the screen, as indicated in TABLE 3.1 on the following page. All areas except the register area and the central scroll area have the same format regardless of the target chip. The formatting of the register area (in the upper left-hand section of the screen) and the central scroll area (in the center portion of the screen) varies in design according to the characteristics of the specific target chip.

TABLE 3.1 - EMULATION SCREEN DISPLAY OF CHIP STATUS

SCREEN AREA	INFORMATION DISPLAYED
Registers (top left)	Contents of general and working registers and status of flags
Memory Map (top center)	Memory range and type of memory currently mapped
Trace Status (upper right)	Trace ON or OFF and identification of breakpoints controlling trace
Breakpoints (far upper right)	Status of system's eight breakpoints: ON/OFF/UNDEF (undefined)/address/LOGICAL
Central Area (center)	User selected displays - contents of memory, trace buffer, breakpoint parameters, settings for switches and counters
Command Input and Mode	
Lower Left:	
C:	User command input. When cursor is positioned here, any legal ECL-3211 command may be entered.
S:	System status and error messages
E:	System messages
Lower Right:	
TYPE:	Name of microprocessor
FREQ:	Optional clock frequency
FasK:	Current FasKey operation
MODE:	Current system operating mode

Initially, unless prefilled by a start-up command file and data file, the emulation screen display shows all values as zero, blank or undefined. For example, Figure 3.2 illustrates the screen display for the Z80 microprocessor as it first appears. However, by issuing various ECL-3211 commands, you can modify these values.

GENERAL	REGISTERS	HS MAPPING		TRACE ON OFF	SYSTEM BREAK POINT
		RANGE	ASSG TYPE		
PC 0000	B 00 C 00	0000-07FF	0 RAM		B0: UNDEF OFF
SP 0000	D 00 E 00	0000-0000	1 OFF		B1: UNDEF OFF
IX 0000	H 00 L 00 SZXHPNC	0000-0000	2 OFF		B2: UNDEF OFF
IY 0000	A 00 F 00[00000000]	0000-0000	3 OFF		B3: UNDEF OFF
I 00	B' 00 C' 00	0000-0000	4 OFF		B4: UNDEF OFF
R 00	D' 00 E' 00				B5: UNDEF OFF
	H' 00 L' 00 SZXHPNC				B6: UNDEF OFF
	A' 00 F' 00[00000000]				B7: UNDEF OFF

C:
S:
E:

TYPE: Z80
FREQ: 2500 KH
FasK: FasKey 1
MODE: COMMAND

FIGURE 3.2 INITIAL SCREEN FORMAT FOR A Z80 CHIP
(No parameters have been set)

In Figure 3.3, a Z80 user has requested a display of memory beginning at address 800. The system is in command mode.

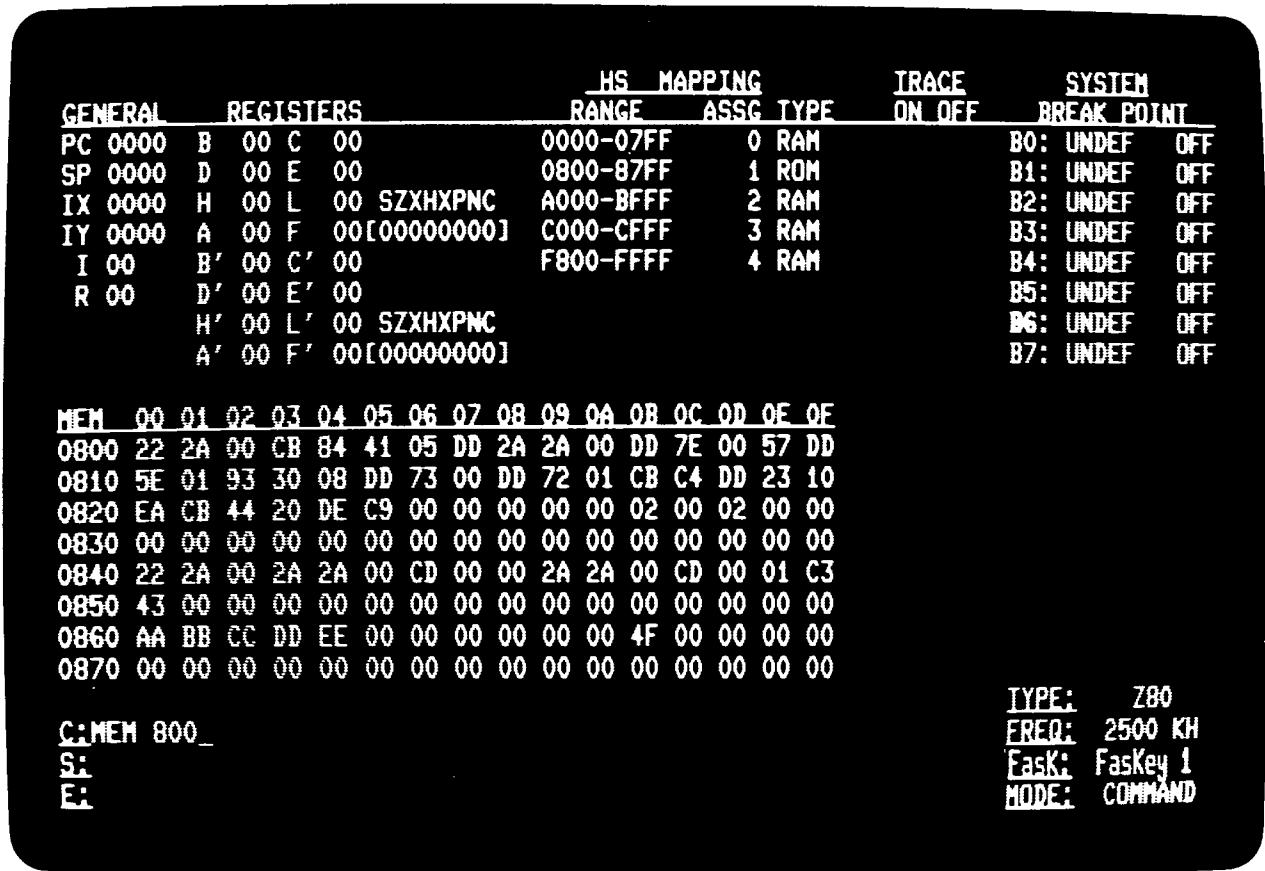


FIGURE 3.3 Screen Display of Z80 Memory Range

In Figure 3.4 below, a Z80 user is setting the parameters for breakpoint 2. Note that the switches and registers have been set, high-speed memory is mapped, traces have been set, and six breakpoints have been defined and are turned on.

GENERAL		REGISTERS		HS MAPPING		TRACE		SYSTEM	
				RANGE	ASSG TYPE	ON	OFF	BREAK POINT	
PC 0000	B 00 C 00			0000-07FF	0 OFF	B5	B7	B0:	0865 ON
SP 006B	D 7F E 00			1000-1FFF	1 RAM	B6		B1:	LOGICAL ON
IX 027F	H 03 L 00	SZXHXPNC		8008-8008	2 ROM			B2:	LOGICAL ON
IY 0000	A 00 F 10	[00010000]		0000-0000	3 OFF			B3:	UNDEF OFF
I 00	B' 00 C' 00			0000-0000	4 OFF			B4:	LOGICAL ON
R 00	D' 00 E' 00							B5:	LOGICAL ON
	H' 00 L' 00	SZXHXPNC						B6:	LOGICAL ON
	A' 00 F' 00	[00000000]						B7:	LOGICAL ON

ACTIONS		ACTIONS		BREAKPOINT 2 TRAP CONDITIONS	
	PH=1000	E0: X	SW1: X	ADDR:	1001011101101110 [976E]
		E1: X	SW2: X	DATA:	XXXXXXXX
		E2: X	SW3: X	HALT: X	BUSRQ: X RD: X
		E3: X	SW4: X	WAIT: X	BUSAK: X WR: X
		E4: X	ROM: X	MREQ: X	NMI: X
		E5: X	INS: 1	RFSH: X	INT: X
		E6: X	CO1: X	IORQ: X	M1: X
		E7: X	CO2: X		

C:BR 2 PH=1000/PC=976E_	TYPE:	Z80
S:	FREQ:	6000 KH
E:	FasK:	FasKey 1
	MODE:	COMMAND

FIGURE 3.4 Screen Display of Breakpoint Setting (Z80 Chip)

In Figure 3.5 below, a Z80 user has requested a display of the instruction trace buffer.

GENERAL		REGISTERS				HS MAPPING		TRACE		SYSTEM									
						RANGE	ASSG TYPE	ON	OFF	BREAK POINT									
PC	004F	B	00	C	80	0000-0000	0 OFF	B3		B0:	LOGICAL ON								
SP	006D	D	7E	E	7F	0000-07FF	1 RAM			B1:	LOGICAL ON								
IX	027F	H	02	L	00 SZXHPNC	0000-0000	2 OFF			B2:	LOGICAL ON								
IY	0000	A	FF	F	55[01010101]	0000-0000	3 OFF			B3:	LOGICAL ON								
I	00	B'	00	C'	00	0000-0000	4 OFF			B4:	UNDEF OFF								
R	71	D'	00	E'	00					B5:	UNDEF OFF								
		H'	00	L'	00 SZXHPNC					B6:	LOGICAL ON								
		A'	00	F'	00[00000000]					B7:	UNDEF OFF								
						EXTERNAL	H B W N I M R I M B R W												
LOC	ADDR	INSTRUCTION				DATA	76543210	T	Q	T	M	N	Q	F	Q	1	K	D	R
504	0123	BIT	0,H			CB	00000000	1	1	1	1	1	0	1	1	0	1	0	1
505	0124					44	00000000	1	1	1	1	1	0	0	1	0	1	0	1
506	0125	JR	NZ,DC			20	00000000	1	1	1	1	1	0	0	1	0	1	0	1
507	0126					DC	00000000	1	1	1	1	1	0	0	1	1	1	0	1
508	0127	RET				C9	00000000	1	1	1	1	1	0	1	1	0	1	0	1
509	006B					4F	00000000	1	1	1	1	1	0	0	1	1	1	0	1
510	006C					00	00000000	1	1	1	1	1	0	1	1	1	1	0	1
511	004F																		
																TYPE:		Z80	
C:TRACE_																FREQ:		6000 KH	
S:																Fask:		FasKey 1	
E:																MODE:		COMMAND	
ECL disconnected																			

FIGURE 3.5 Trace Buffer Display (Z80 chip)

Figure 3.6 demonstrates a screen display of user-set switches and counters for an 8-bit Z80 microprocessor. The user is setting the values for switches 1, 2, and 4.

GENERAL		REGISTERS		HS MAPPING		TRACE	SYSTEM	
				RANGE	ASSG TYPE	ON OFF	BREAK POINT	
PC 0000	B 00	C 00		0000-07FF	0 RAM		B0: UNDEF	OFF
SP 0000	D 00	E 00		0000-0000	1 OFF		B1: UNDEF	OFF
IX 0000	H 00	L 00	SZXHPNC	0000-0000	2 OFF		B2: UNDEF	OFF
IY 0000	A 00	F 00	[00000000]	0000-0000	3 OFF		B3: UNDEF	OFF
I 00	B' 00	C' 00		0000-0000	4 OFF		B4: UNDEF	OFF
R 00	D' 00	E' 00					B5: UNDEF	OFF
	H' 00	L' 00	SZXHPNC				B6: UNDEF	OFF
	A' 00	F' 00	[00000000]				B7: UNDEF	OFF
SWITCHES		COUNT		RESET COUNT		CLOCK SRC		
SW 1=1	CO 1	10000.	CC RI	10000.		INT		
SW 2=0	CO 2	50.	IS					
SW 3=0								
SW 4=1								
C:SWITCH 1=1,2=0,4=1_						TYPE:	Z80	
S:						FREQ:	2500 KH	
E:						FasK:	FasKey 2	
						MODE:	COMMAND	

FIGURE 3.6 Setting Switches
(8-bit Chip)

The screen display of memory range demonstrated in Figure 3.7 displays memory contents as four hex characters, since the target microprocessor (68000) is a 16-bit chip.

REGISTERS				HS MAPPING				TRACE		SYSTEM					
D0	00000000	A0	00000000	T: 0	FROM	TO	ASSG TYP	ON	OFF	BREAK	POINTS				
D1	00000000	A1	00000000	S: 0	000000	0007FF	0 RAM			B0	UNDEF OFF				
D2	00000000	A2	00000000	I: 0	000000	000000	1 OFF			B1	UNDEF OFF				
D3	00000000	A3	00000000	X: 0	000000	000000	2 OFF			B2	UNDEF OFF				
D4	00000000	A4	00000000	N: 0	000000	000000	3 OFF			B3	UNDEF OFF				
D5	00000000	A5	00000000	Z: 0	000000	000000	4 OFF			B4	UNDEF OFF				
D6	00000000	A6	00000000	V: 0						B5	UNDEF OFF				
D7	00000000	US	00000000	C: 0						B6	UNDEF OFF				
PC	000000	SS	00000000	SR 0000						B7	UNDEF OFF				
MEM 000200															
00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
0110	0111	0112	0113	0114	0115	0116	0117	0118	0119	011A	011B	011C	011D	011E	011F
0220	0221	0222	0223	0224	0225	0226	0227	0228	0229	022A	022B	022C	022D	022E	022F
0330	0331	0332	0333	0334	0335	0336	0337	0338	0339	033A	033B	033C	033D	033E	033F
0440	0441	0442	0443	0444	0445	0446	0447	0448	0449	044A	044B	044C	044D	044E	044F
0550	0551	0552	0553	0554	0555	0556	0557	0558	0559	055A	055B	055C	055D	055E	055F
0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	066A	066B	066C	066D	066E	066F
0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	077A	077B	077C	077D	077E	077F
0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	088A	088B	088C	088D	088E	088F
												TYPE: 68000			
C:MEM 200_												FREQ: 8000 KH			
S:												FasK: FasKey 1			
E:												MODE: COMMAND			

FIGURE 3.7 Screen Display of Memory Range for 16-Bit 68000 Microprocessor

MODES OF OPERATION

The ECL-3211 emulation system always runs in one of six possible operating modes. The current mode is always displayed in the lower right-hand corner of the screen (MODE:). TABLE 3.2 summarizes the six emulation system modes. Each mode is discussed in the text which follows the table summary.

TABLE 3.2 EMULATION SYSTEM MODES OF OPERATION

MODE	ENABLES
ALTER	Modification of screen values via the keyboard
COMMAND	User command input
EMULATE	Execution of target chip program
MAPPING	Management of mapped memory (logical relocation of segments of memory)
STEP	Stepping through target chip program, instruction by instruction or breakpoint by breakpoint
HELP	Provides access to the ECL-3211 HELP file for user assistance in command syntax and operation of the ECL-3211 system.

ALTER MODE

The system is placed in ALTER mode by depressing either the PF1 or the PF2 key. While the system is operating in the ALTER mode, the user can modify values directly from the keyboard on either the Memory or Breakpoint screen displays or in the register information area.

The system accepts any changes you have made when you exit ALTER mode and return to COMMAND mode. To exit ALTER mode, press the RETURN key. The system returns to COMMAND mode and places the cursor on the command line.

To move through the Memory and Breakpoint displays, use the arrow keys. The memory display can be scrolled backward or forward using the arrow keys; the breakpoint display does not scroll.

COMMAND MODE

ECL-3211 system commands may be entered on the command input line (C:) whenever the system is processing in COMMAND mode. COMMAND mode is the processing default mode to which the system returns

when operations in an alternate mode have completed.

NOTE: ENTERING A CARRIAGE RETURN WHILE THE SYSTEM IS PROCESSING IN A MODE OTHER THAN STEP MODE RETURNS THE SYSTEM TO THE COMMAND MODE. THE SYSTEM AUTOMATICALLY RETURNS TO COMMAND MODE UPON COMPLETION OF AN EMULATION RUN.

While in COMMAND mode, any command may be entered using the "FasKey" functions of the COMMAND mode. These functions allow for quick input of commands using Emulogic's specifically programmed "FasKeys" on the keypad at the right-hand side of the keyboard. For a more detailed discussion of the available FasKey capabilities, see the topic USE OF THE "FASKEY" CAPABILITIES at the end of this chapter. (APPENDIX D - FASKEY MENU SUMMARY, at the back of this manual, provides illustrated menus and submenus for use of the FasKey Keypad.)

EMULATE MODE

The system enters EMULATE mode when you enter the EMULATE command. The system begins executing the development software code in the ESP microprocessor.

To exit from EMULATE mode and return to command mode, press the RETURN key. The system will also end emulation and return to COMMAND mode when a breakpoint HALT action is tripped.

MAPPING MODE

Whenever you enter a MAP command, the ECL-3211 enters MAPPING mode. MAP mode enables you to map (logically relocate) Emulogic high-speed memory or DEC internal memory into various segments of logical address space in the target microprocessor.

STEP MODE

The STEP command places the ECL-3211 system in STEP mode. During STEP mode, the target chip program "steps" through program execution one instruction at a time. Depending on how you enter the STEP mode, the ECL-3211 will either step through the program one instruction cycle at a time or until a breakpoint with a pause, halt, or command file action is encountered (and will then halt emulation and update the screen display).

Each time you press the RETURN key, the system executes the next step (instruction or breakpoint). To exit from STEP mode, enter any other ECL-3211 command.

HELP MODE

The HELP command places the ECL-3211 system in the HELP mode. HELP mode gives the user access to the system help file for assistance in ECL-3211 command syntax and operation of the system. The help file (LOXX00.HLP) must reside on the pseudo device, "HLP:". To assign this pseudo device, you use the RT-11 ASSIGN command. For example, enter ASSIGN SY: HLP: if the help file resides on your booted device or, enter ASSIGN DY1: HLP: if the help file resides on floppy drive 1, etc.

Entering the HELP command directly from the command line will bring up the ECL-3211 help file. The HELP command is described in detail in Chapter 7 - The Command Dictionary. The specially programmed comma (,) FasKey on the separate keypad on the right-hand side of your keyboard may also be used to access the help file. (See Use of FasKey Techniques later in this chapter). To exit HELP mode, press the RETURN key.

SPECIAL KEY FUNCTIONS

=====

Several keys on the terminal keyboard can perform special functions, such as modifying the current mode or moving the cursor on the screen display. Table 3.3 lists and describes each of the special key functions. As indicated by this table the cursor can jump from character to character, from parameter field to parameter field or from one screen display area to another.

TABLE 3.3 SPECIAL KEY FUNCTIONS

KEY FUNCTION	DESCRIPTION
<cr> (RETURN)	Returns ECL-3211 to COMMAND mode from any mode (except STEP mode) When in STEP mode, <cr> executes next step.
<u>Character Jumps</u>	
↑ (up arrow)	Moves cursor up on the Memory, Map, Trace, and Breakpoint displays. When at the top of the register field, LOOPS TO BOTTOM OF THE REGISTER FIELD. MOVES UP ONE LINE IN help FILE DISPLAY. CAN BE USED TO SCROLL BACK IN THE MEMORY AND TRACE DISPLAYS.
↓ (DOWN ARROW)	MOVES CURSOR DOWN ON THE MEMORY, MAP, TRACE, AND BREAKPOINT DISPLAYS. WHEN AT THE BOTTOM OF THE REGISTER FIELD, LOOPS TO TOP OF REGISTER FIELD. MOVES DOWN ONE LINE IN THE help FILE DISPLAY. CAN BE USED TO SCROLL FORWARD IN THE MEMORY, TRACE, AND DISASSEMBLY DISPLAYS.

TABLE 3.3 SPECIAL KEY FUNCTIONS (cont.)

KEY FUNCTION	DESCRIPTION
→ (right arrow)	Moves cursor right on alterable displays.
← (left arrow)	Moves cursor left on alterable displays.
<u>Parameter Field and Area Jumps</u>	
PF1 (up-area jump)	Moves cursor up from command line to central scroll area to register display, then back to command line. Scrolls forward one page in the HELP file.
PF2 (down-area jump)	Moves cursor from register field to central area, down to command line, then back to the register field again. Scrolls back one page in the HELP file display.

USE OF FASKEY TECHNIQUES

The separate keypad on the right-hand side of your keyboard has been programmed to perform special Emulogic "FasKey" functions when the ECL-3211 is in command mode. The FasKey functions allow you to enter commands through a single or few keystrokes. Keys 0-9, minus, comma, period, and ENTER on the small keypad have programmed special functions depending upon the particular FasKey menu, FasKey 1 or FasKey 2, in which you are operating.

The minus (-) key allows you to move back and forth between the two FasKey menus. The comma key gets you the applicable HELP file for the FasKey operation you have selected.

To display the FasKey menu in the central area, press the comma (,) key on the small keypad. The menu label "FasKey 1" will appear in the lower right hand corner of the display screen on the FasK line. The menu showing the FasKey 1 keystroke functions appears in the central area of the display. To view the FasKey 2 menu and select a FasKey 2 function, press the minus key on the keypad. The system presents the FasKey 2 menu in the central area and "FasKey 2" on the FasK line.

Figure 3.8 demonstrates the menu for use of keys in the FasKey 1 command mode. Refer to Appendix D of this manual for a complete display of all the available FasKey keypad configurations.

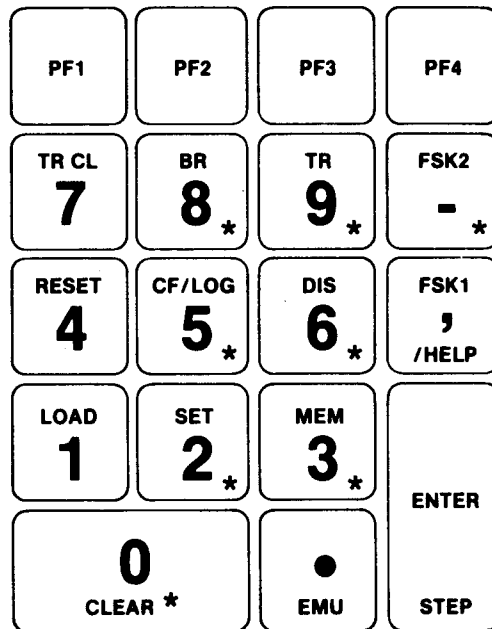


FIGURE 3.8 THE FASKEY 1
COMMAND MENU

Submenus are provided for the following FasKey command keys (shown with asterisks in the above display) on the FasKey 1 configuration of the keypad:

- 0 - CLEAR
- 2 - SET
- 3 - MEM (Memory commands)
- 5 - CF/LOG (Command File commands)
- 6 - DIS (Disassemble Memory commands)
- 8 - BR (Breakpoint commands)
- 9 - TR (Trace commands)

The submenus provide you with subset arguments of the basic commands and with additional FasKey capabilities related to the specific commands.

Figure 3.9 displays the basic FasKey 2 command menu. The FasKey 2 menu is accessed by pressing the minus key on the FasKey 1 menu.



FIGURE 3.9 THE FASKEY 2 MENU

There are no submenus which branch from the FasKey 2 menu.

Programmable FasKeys

A feature of the command file FasKey software allows you to program FasKeys 0-9 to call user-created command files so that you may have your most frequently used command files "at your fingertips". You may assign FasKeys 0-9 for each default command file extension.

Directions for use of this capability have been provided in Chapter 6, CREATION AND USE OF COMMAND FILES.

Command File Pause Keys

Four keys on the keypad have special functions when, and only when the ECL-3211 is in command file pause state (whether the pause has been user-commanded or system-induced due to an error in command file processing). The four CF/P keys are the minus, comma, ENTER and period keys. The minus key causes the command file to "backspace" and reexecute the last command. The comma key calls up HELP information in the central scroll area to direct the user in the actions of the CF/P keys. The period key causes the command file to resume processing. The ENTER key ends the particular command file being paused, closing the file and removing it from the command file stack.

Refer to Chapter 6 CREATION AND USE OF COMMAND FILES for a complete discussion of the command file pause.

Introduction:

Through its memory mapping facility, the Emulogic ECL-3211 microprocessor development system reserves memory for two functions: microprocessor software development and microprocessor emulation. This chapter discusses emulation memory management. Emulation memory consists of low and high speed memory modules which can supplement or supplant any memory in your target system.

Emulation memory can be "mapped" in the ECL-3211 MDS with byte resolution (in 8-bit emulators) or word resolution (in 16-bit emulators). Memory mapping allows your target memory to be brought "inside" the ECL-3211 system for easy access and control. This can be done to provide emulation memory when none may exist in the current target system. MDS emulation memory can also save the cost and time involved in burning ROM or PROM in-target memory. Mapped memory not only provides additional memory for emulation, but it also ensures that the microprocessor can operate at full speed during emulation, allowing emulation to be transparent to the target.

An additional capability allows ECL-3211 high-speed memory to simulate ROM to mimic the target environment during software development. The ability to simulate ROM also enables you to proceed with software development and microprocessor emulation in the absence of physical target memory.

During emulation, the contents of any RAM or simulated ROM location can be dynamically modified by entering new memory values via two available change memory commands, the SET command and ALTER mode .

The ECL-3211 Microprocessor Development System not only provides a wide range of mapping options, it also offers many mapping and memory status displays. Upon command, the ECL-3211 will display screens showing current map assignments and the current contents of memory.

Any of the commands discussed in this chapter may be entered using the FasKey capability of the command mode. We suggest that you first familiarize yourself with the command syntax as demonstrated in this chapter. Refer to Appendix D at the back of this manual for the specific FasKey keypad configurations which apply to memory management.

ECL-3211 memory management also allows you to halt emulation and to write current emulation parameters to a data file, so that emulation can be resumed at a later time using the currently set parameters for the processor registers, map settings, breakpoint definitions, and clock frequencies.

For increased data memory capacity or to save data memory for subsequent emulation sessions, memory images in either your target or the ECL-3211 can be stored on disk and later reloaded, as necessary.

The memory management capabilities of the ECL-3211 also make it possible for you to relocate portions of memory to a different region of logical address space. Relocation can be directed to address either DEC internal memory or Emulogic's high-speed memory by changing the logical address range specified by the current map.

All of these memory management capabilities will be discussed in this chapter in sequence as follows:

- o ECL-3211 MDS Memory Characteristics
 - DEC Internal Memory
 - Emulogic High-Speed Memory
- o Memory Mapping - Internal and High-speed
 - The Commands
 - The Displays
- o Saving Emulation Parameters
- o Viewing the Contents of Memory
- o Altering the contents of Memory
- o Viewing the Contents of Memory in ASCII
- o Disassembling the Contents of Memory
- o Saving Data Memory
- o Loading Disk Files into Memory
- o Relocating Mapped Memory

ECL-3211 MEMORY CHARACTERISTICS

There are two memory sources available while using the ECL-3211 system :

- External memory
- ECL-3211 memory.

External Memory

External memory is located in the target system. The characteristics of this memory, such as type (absolute RAM or ROM) and rate of access are dependent on the configuration of the particular target.

ECL-3211 Memory

ECL-3211 memory is internal to the Emulogic microprocessor development system. There are two types of memory available for mapping in the ECL-3211:

- o DEC internal memory
- o Emulogic high-speed memory.

DEC internal memory has from 8K to 64K bytes of available RAM memory (depending upon your system's configuration). Internal memory cannot be designated as ROM and may not be able to support microprocessor emulation at high clock rates.

Emulogic, Inc. provides 2K bytes of high-speed memory (HSM) as a standard feature of all ECL-3211 Map boards. This memory, the system's MAP 0, is sometimes all the high-speed emulation memory required. MAP 0 cannot be designated as ROM.

Additional Emulogic high-speed memory modules are available in 16k and 32k byte increments. This memory, in addition to supporting high-speed emulation of large development programs, can optionally be designated as ROM memory. As simulated ROM, the memory segment is identified as "read only" by the microprocessor. However, for development purposes, the you can alter the contents of memory through the ALTER mode and SET command, which will be discussed in the "Commands" section of this chapter.

Please Note:

A very important characteristic of ECL-3211 memory mapping is that:

THE SYSTEM ASSUMES ANY MEMORY REFERENCE THAT IS NOT IN A "MAPPED" SEGMENT IS A REQUEST FOR AN ABSOLUTE ADDRESS IN EXTERNAL (TARGET) MEMORY.

Memory Speed:

The three types of emulation memory available to your target microprocessor using ECL-3211 - internal, high-speed and target - may operate at different speeds. Consequently, full-speed emulation of the target system can be affected by the type of memory accessed.

Internal DEC memory is typically slower than external target memory, limiting real-time execution to 1-2 MHz. While this may be quite adequate for some applications and processors, many targets require faster cycle rates. You can supplement target size and also ensure full-speed emulation by using Emulogic high-speed emulation memory. Either internal or high-speed memory can be used with external memory, as long as the clock frequency is set to accommodate the slowest memory device.

NOTE: for the remaining discussion of memory in this manual, we will refer to DEC memory as "internal memory" (or INT memory) and Emulogic high-speed emulation memory as "high-speed memory" (or HSM).

MEMORY MAPPING

Memory mapping allows your target memory (program and data) to be brought "inside" Emulogic ECL-3211 memory. This may be done to provide emulation memory when none may exist on the target board or when target memory is ROM or PROM and it is desirable to test changes and to avoid costly and time-consuming burning of ROMS and PROMS. Mapping also provides a convenient way to evaluate a variety of memory organization stratagems.

Mapping facilities enable you to map either internal memory or high-speed memory.

The segment to which memory is mapped may be as small as one byte (for 8-bit emulators) or one word (for 16-bit emulators), or as large as the entire logical address space of the microprocessor (providing that sufficient memory space has been installed in the ECL-3211). You must keep in mind, however, that when mapping a particular segment (map board) of memory, that if only one byte of the segment is mapped, the rest of memory in that particular segment becomes unavailable for mapping. Mapping a particular range allows access to the mapped area. Segments can quickly be re-mapped to redefine memory.

On start-up, one HS memory map, MAP 0, is defined across the logical range of 0-7FF as RAM type memory. (See Figure 4-1.) All other HS map segments installed must be defined by range and type as either RAM or simulated ROM.

GENERAL	REGISTERS	HS MAPPING		TRACE ON OFF	SYSTEM BREAK POINT
		RANGE	ASSG TYPE		
PC 0000	B 00 C 00	0000-07FF	0 RAM		B0: UNDEF OFF
SP 0000	D 00 E 00	0000-0000	1 OFF		B1: UNDEF OFF
IX 0000	H 00 L 00 SZXHPNC	0000-0000	2 OFF		B2: UNDEF OFF
IY 0000	A 00 F 00[00000000]	0000-0000	3 OFF		B3: UNDEF OFF
I 00	B' 00 C' 00	0000-0000	4 OFF		B4: UNDEF OFF
R 00	D' 00 E' 00				B5: UNDEF OFF
	H' 00 L' 00 SZXHPNC				B6: UNDEF OFF
	A' 00 F' 00[00000000]				B7: UNDEF OFF

C:
S:
E:

TYPE: Z80
FREQ: 2500 KH
FasK: FasKey 1
MODE: COMMAND

FIGURE 4.1 HIGH-SPEED MAPPING AT START-UP

Initially, all HS map segments are defined by default as RAM. HS map segments, other than MAP 0, may be redefined as ROM to simulate target ROM. The capability to simulate ROM enables you to experiment with various memory configurations while designing target software and hardware. It also allows you to proceed with software development and microprocessor emulation in the absence of a physical target chip.

MAP COMMANDS

ECL-3211's map commands are divided into three categories: general, internal, and high-speed map commands.

On initiation, memory is defined as High-speed MAP 0. MAP 0 is the 2K of high-speed RAM memory available on the ECL-3211 map board. To turn on and map internal memory and to access any additional high-speed memory boards, you enter the appropriate map commands as described below.

All map commands are shown with required input capitalized. The entire command may be entered or, to save time, the abbreviated capitalized portion of the command may be entered. The symbol <cr> indicates that you must enter a carriage return immediately following the command.

General Map Commands

Entering MAP mode will display (in the central scroll area) definitions of current available memory(ies). To enter MAP MODE, enter:

MAp<cr>

To clear all current map declarations, enter:

MAp CLear<cr>

To turn on all curent map declarations, enter:

MAp ON<cr>

To turn off all map declarations so that mapping is now external (all addresses refer to absolute addresses in the target memory), enter:

MAp OFF<cr>

Figure 4.2 shows you a screen with all mapping cleared.

GENERAL	REGISTERS	NO MAPPING		TRACE ON OFF	SYSTEM	
		RANGE	ASSG TYPE		BREAK	POINT
PC 0000	B 00 C 00	0000-0000	0 OFF	B0:	UNDEF	OFF
SP 0000	D 00 E 00	0000-0000	1 OFF	B1:	UNDEF	OFF
IX 0000	H 00 L 00 SZXHPNC	0000-0000	2 OFF	B2:	UNDEF	OFF
IY 0000	A 00 F 00[00000000]	0000-0000	3 OFF	B3:	UNDEF	OFF
I 00	B' 00 C' 00	0000-0000	4 OFF	B4:	UNDEF	OFF
R 00	D' 00 E' 00			B5:	UNDEF	OFF
	H' 00 L' 00 SZXHPNC			B6:	UNDEF	OFF
	A' 00 F' 00[00000000]			B7:	UNDEF	OFF

C:MAP CLEAR_
S:
E:

TYPE: Z80
FREQ: 2500 KH
FasK: FasKey 2
MODE: COMMAND

FIGURE 4.2 MAP CLEAR COMMAND

Internal Map Commands

To turn on DEC INT memory mapping, enter:

```
MAP INT<cr>
```

MAP INT allows changes to be made to the range and offset of the internal memory map. The heading INT MAPPING is displayed in the center of the ECL-3211 display screen at the top of the screen. Please note that all emulation memory is initially defined as HS (high-speed) at start-up.

Please note that the range of available internal memory will depend on your DEC system's configuration. A system with the minimal 8K can only be mapped in the address range 0-1FFF; whereas a system with expanded DEC memory of 64K bytes can be mapped in the address range of 0-FFFF. (Be very careful to map internal memory within the boundary of your particular system since the ECL-3211 does not "size" internal memory. Attempting to map beyond the available range for internal memory mapping could interfere with DEC memory allocated for system functions.)

Once the MAP INT command is recognized by the system, subsequent map commands are assumed to refer to internal memory until a map high-speed command is given to alter the mapping status. The system cannot map to internal and high-speed memory simultaneously.

To map a particular address range starting at a physical address offset in internal memory, enter:

```
MAP AAAA-BBBB=OFFSET<cr>
```

Where AAAA-BBBB represents the particular address range and OFFSET represents the physical starting address in internal memory.

Please Note:

The OFFSET address is not an absolute address in DEC memory, but starts at 0 and ranges upward to 8 or 64K depending on the system you have purchased. It is defined relative to the start of that portion of DEC memory that is available for emulation.

Figure 4.3 displays a screen response to the MAP INT command of a system with 8K of available DEC internal memory.

GENERAL	REGISTERS	INT MAPPING		TRACE ON OFF	SYSTEM BREAK POINT
		RANGE	ASSG TYPE		
PC 0000	B 00 C 00	0000-1FFF	0000 RAM		B0: UNDEF OFF
SP 0000	D 00 E 00				B1: UNDEF OFF
IX 0000	H 00 L 00 SZXHPNC				B2: UNDEF OFF
IY 0000	A 00 F 00[00000000]				B3: UNDEF OFF
I 00	B' 00 C' 00				B4: UNDEF OFF
R 00	D' 00 E' 00				B5: UNDEF OFF
	H' 00 L' 00 SZXHPNC				B6: UNDEF OFF
	A' 00 F' 00[00000000]				B7: UNDEF OFF

C:MAP INT_
S:
E:

TYPE: Z80
FREQ: 2500 KH
FasK: FasKey 2
MODE: MAPPING

FIGURE 4.3 THE MAP INTERNAL COMMAND

High-Speed Memory Map Commands

Initially, only high-speed memory MAP 0 (built into the basic ECL-3211 system) is mapped and active on start-up. Any additional high-speed map boards must be turned on (by command) and mapped by you. Map numbers are assigned by the system for as many as 4 additional high-speed memory boards displaying maps 0 through 4 on the map display.

To evoke high-speed mapping, enter:

Map HS<cr>

The heading HS MAPPING will now appear in the ECL-3211 display in the center at the top of the screen. The available HS memory maps are displayed ready to be mapped. If high-speed memory was mapped previously, the previous high-speed map parameters will display.

Once the MAP HS command is recognized by the system, subsequent map commands are relative to high-speed status.

Individual high-speed maps may now be turned on.

To turn individual HS memory maps on, a memory mapping declaration must be entered:

MAp AAA-BBB=N (type)

where AAA-BBB is the range of memory to be mapped on map N and (type) is the optional input of memory type RAM or ROM. If type of memory is not designated, the system will default to RAM.

For example, to turn on address range 0-7FF in HS memory map 1 and to type the mapped range as simulated ROM, you would enter:

MAp 0-7FF=1 ROM

(Remember the exception of HS MAP 0, which cannot be declared ROM).

WARNING: Results are unpredictable if you assign the same address to more than one active map.

To clear a specific HS map, enter:

MAp N CLeAr<cr>

where N represents the number of the high-speed map being cleared. The map numbers appear in the ECL-3211 map display in the central scroll area under the heading "OFFSET/#". Clearing a map declaration turns it off and removes its beginning and ending addresses from the "FROM" and "TO" portions of the map display. Any HS map declaration that is cleared defaults to RAM if it had been previously declared ROM.

To turn OFF a specific HS map (but retain its definition for resetting the map to ON), enter:

MAp N OFF<cr>

where N represents the number of the HS map being deactivated.

To turn ON a specific HS map, enter:

MAp N ON<cr>

where N is the number of the specific HS map being activated.

The definition of the map, previous to being turned OFF, will be reactivated; however the map declaration may be redefined to create new parameters.

To Type High-Speed Memory:

High-speed map segments may be declared RAM or ROM in a command independent of setting the mapped memory range, Either before or after the range is declared, enter:

Map N ROM(or RAM)<cr>

where N equals the number of the HS map segment. (Please remember that MAP 0 cannot be declared ROM.)

This command can be used to change the type of memory of a mapped segment of HS memory.

Figure 4.4 demonstrates a command to map addresses 8000-FFFF of high-speed map 3 as ROM. (68000 microprocessor)

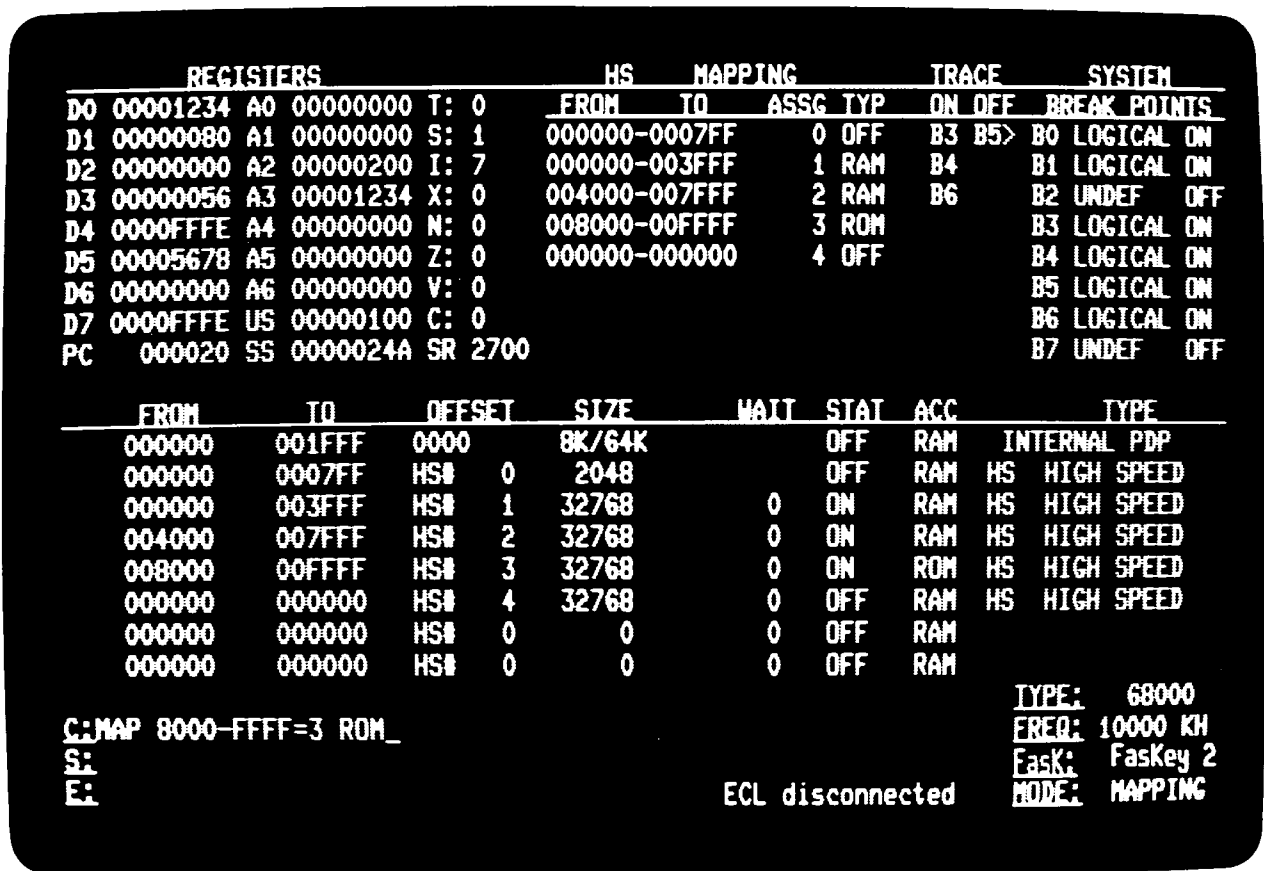


FIGURE 4.4 MAPPING HIGH-SPEED MEMORY AS SIMULATED ROM (68000 chip)

SAVING EMULATION PARAMETERS

You can save current emulation parameters (map settings as well as the contents of registers, counters, and breakpoint definitions) for a future emulation.

Emulation parameters may be saved via two methods. "SAVE FILENAM" will save parameters without exiting. "EXIT/S" or "EXIT/S FILENAM" will save current parameters and also exit from ECL-3211 screen handler, back to RT-11 monitor. These parameters can be re-established at start-up time by specifying the name of the "saved" data file.

To save current map settings, enter:

```
EXIT/S FILE.EXT<cr>
```

The "S" option saves the emulation parameters in a data file (shown above as FILE.EXT). If no data file is specified, the data will be stored in a default data file, LOXX00.DAT. If the filename is specified, but no extension is given, the ECL-3211 system will append a default extension ".DAT" to the file name when writing the file to disk.

ECL-3211 MAP DISPLAY FACILITIES

You can display the characteristics of all available memory maps by entering the command:

```
Map<cr>
```

Refer to figure 4.5 for a display of the map command.

The map display shows:

FROM	the starting logical address of the particular mapped segment
TO	the ending logical address of the mapped segment
OFFSET/#	for internal memory -- the starting physical address of the particular mapped segment
	for high-speed memory -- the HS segment map number (the number of the HS map board)

SIZE the physical memory allocation in bytes
(For DEC internal memory, the display will show 8K/64K. The system will not provide an exact size of available internal memory. You must be aware of the available internal DEC memory in your particular system.)

STAT the activity status (on/off) of the maps

ACC type of access (RAM or ROM)

TYPE INT (internal) or HS (high-speed) memory

GENERAL		REGISTERS		HS MAPPING		TRACE	SYSTEM	
				RANGE	ASSG TYPE	ON OFF	BREAK	POINT
PC	0000	B	00 C 00	0000-07FF	0 RAM		B0:	UNDEF OFF
SP	0000	D	00 E 00	0800-87FF	1 ROM		B1:	UNDEF OFF
IX	0000	H	00 L 00 SZXHPNC	A000-BFFF	2 RAM		B2:	UNDEF OFF
IY	0000	A	00 F 00[00000000]	C000-CFFF	3 OFF		B3:	UNDEF OFF
I	00	B'	00 C' 00	F800-FFFF	4 RAM		B4:	UNDEF OFF
R	00	D'	00 E' 00				B5:	UNDEF OFF
		H'	00 L' 00 SZXHPNC				B6:	UNDEF OFF
		A'	00 F' 00[00000000]				B7:	UNDEF OFF

FROM	TO	OFFSET/#	SIZE	STAT	ACC	TYPE	
0000	1FFF	0000	8K/64K	OFF	RAM	INTERNAL	PDP
0000	07FF	HS# 0	2048	ON	RAM	HS	HIGH SPEED
0800	87FF	HS# 1	32768	ON	ROM	HS	HIGH SPEED
A000	BFFF	HS# 2	32768	ON	RAM	HS	HIGH SPEED
C000	CFFF	HS# 3	32768	OFF	RAM	HS	HIGH SPEED
F800	FFFF	HS# 4	32768	ON	RAM	HS	HIGH SPEED
0000	0000	HS# 0	0	OFF	RAM		
0000	0000	HS# 0	0	OFF	RAM		

C:MAP	TYPE:	Z80
S:	FREQ:	2500 KH
E:	FasK:	FasKey 2
	MODE:	MAPPING

FIGURE 4.5 DISPLAYING MAP PARAMETERS

VIEWING THE CONTENTS OF MEMORY

To view the contents of memory, use the MEM command. The contents of memory are displayed in the central scroll area, beginning at the address location input with the MEM command.

For 8-bit processors, 128 bytes of memory are displayed; for 16-bit processors, 128 words. (If you enter an odd 16-byte/word memory argument, the system will default to the last even 16-byte/word memory location.)

When memory is displayed, the arrow keys can be used to scroll the display. Pressing the UP-ARROW key scrolls backward 1 line; pressing the DOWN-ARROW key scrolls forward 1 line.

8048 Users Please Note:

When emulating the 8048 family of microprocessors (the 8048, 8035, 8039, 8049 and 8749 devices), a different set of memory commands applies for displaying and setting memory. Refer to the Emulogic 8048 User's Guide Supplement for complete instructions.

To view a particular location in memory, enter:

MEM AAAA<cr>

where AAAA is the starting address for the memory area you want to display. In the central scroll area, 128 bytes/words of memory will be displayed beginning at the specified address.

In Figure 4.6 below, memory is displayed beginning at address 800.

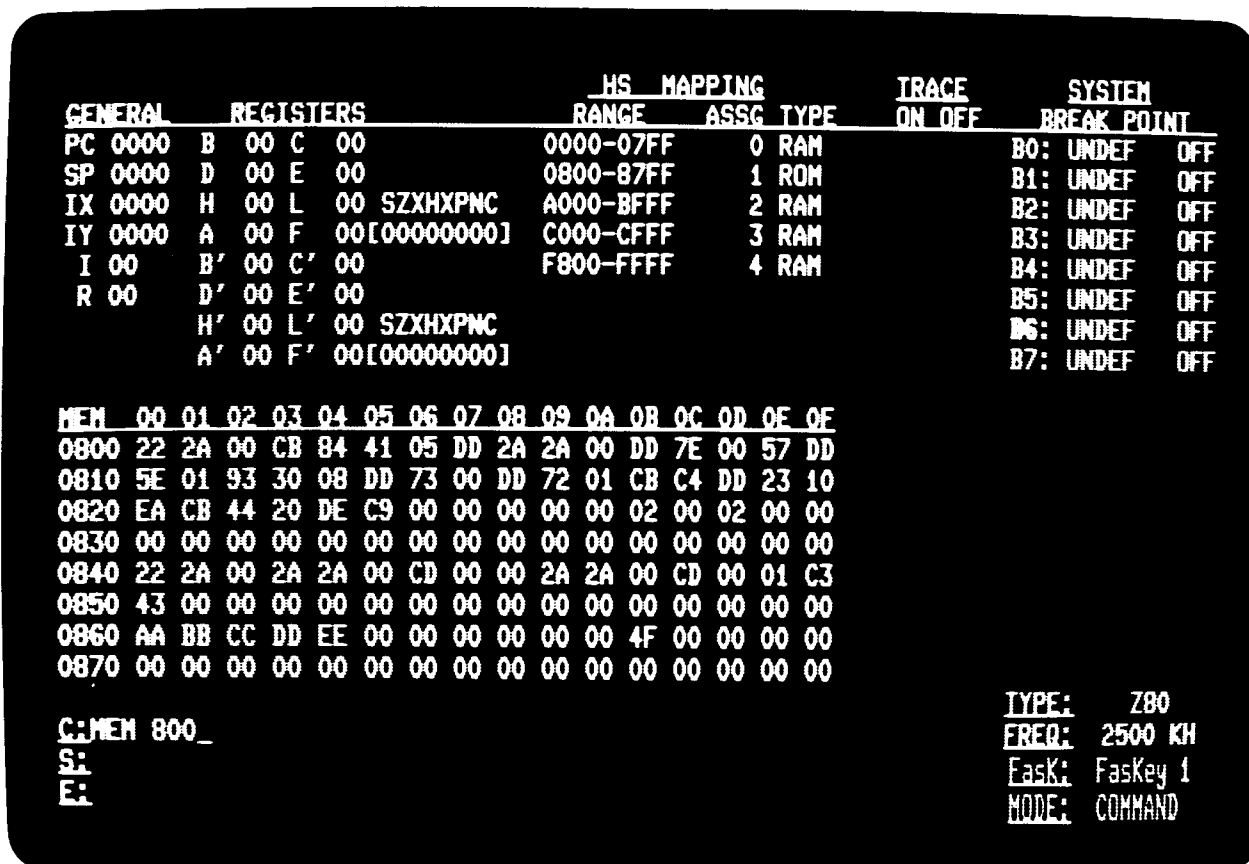


FIGURE 4.6 MEMORY DISPLAY - The MEM Command

ALTERING THE CONTENTS OF MEMORY

ALTER mode allows you to make changes directly to memory contents, either RAM or simulated ROM, without using commands.

With memory displayed in the central scroll area, depressing the PF1 key once will position the cursor at the first byte/word in memory. Use the ARROW KEYS to move the cursor about the display to the location at which you want to make changes. Type the desired changes directly. Pressing the carriage return updates memory with the input changes, brings the system back to command mode and moves the cursor to the bottom of the screen to the command line (C:).

All memory displays are updated on the screen after a halt or pause in emulation.

To Alter the Contents of Memory Using the SET Command:

You can also use the SET command to alter the contents of one or more locations of memory.

Suppose you want to change the contents of address 500 to the value C3 (hex), enter:

```
SEt MEM 500=C3<cr>
```

Location 500 will now contain the value C3. However, suppose that you also want to change the contents of the contiguous locations 501 and 502 to "12" and "34", respectively. Instead of entering three separate commands, you can change the contents of contiguous addresses 500 through 502 by entering:

```
SEt MEM 500=C3,12,34<cr>
```

If you want to change the contents of an entire block of memory to the same value of "FF", for example, you can enter the SEt command showing a range of memory:

```
SEt MEM 300-380=FF<cr>
```

TO VIEW THE CONTENTS OF MEMORY IN ASCII:

The /A switch (ASCII switch), used with the MEM command, displays the contents of 64 hex bytes/words of memory and their ASCII equivalents. The beginning address for display should be given, using the command format:

MEM/A AAAA<cr>

Please note:

If you enter an odd 16-byte/word memory argument, the system defaults to the last even 16-byte/word boundary.

Figure 4.7 below demonstrates a MEM/A display beginning at address 220.

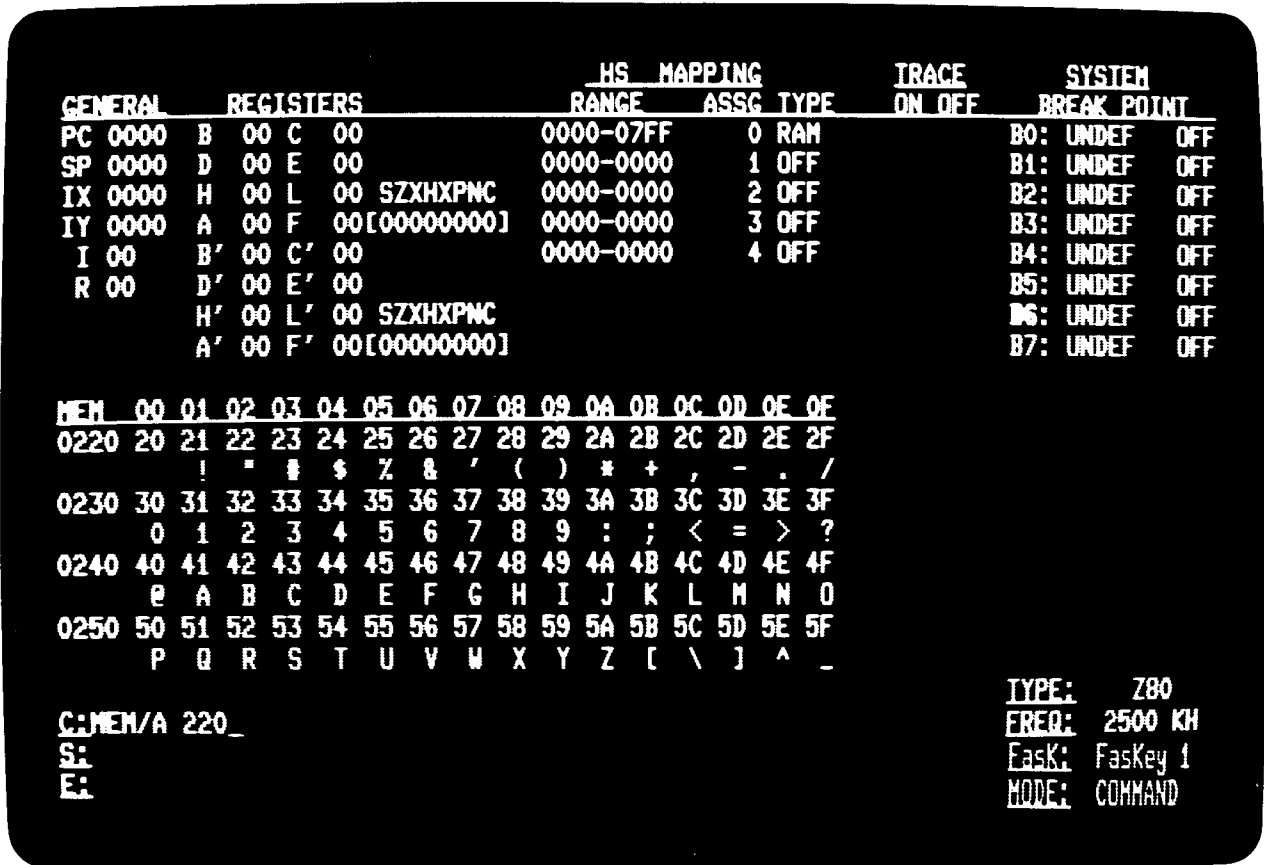


FIGURE 4.7 DISPLAYING MEMORY IN ASCII AND HEX

TO VIEW MEMORY CONTENTS IN QUICK FORMAT

The quick switch (Q) can be used with the MEM command to display the contents of up to three memory ranges. Each specified range can be one or more bytes/words. To request a quick format display of memory, enter a command in the format:

```
ME/Q aaaa-bbbb(,cccc-dddd,eeee-ffff)
```

For example, any of the following are legal quick format requests:

```
ME/Q 1234
ME/Q 10-20,3346,100-500
ME/Q 567,890,ABCD
```

If the range or ranges you specify will display more memory data than can fit within the central area, the system will automatically scroll the display records. Use the NO SCROLL key to stop and start the data display.

The Q switch can be used in conjunction with the ASCII display described previously. Either the A or Q switch can be placed first in the command.

DISASSEMBLING THE CONTENTS OF MEMORY

The DISASSEMBLE MEMORY feature enables you to disassemble the contents of memory. When you enter:

```
DIS AAAA
```

the ECL-3211 disassembles and then displays the next 8 instructions in memory, beginning at the given address AAAA. You can move through the displayed disassembled memory using the keyboard movement arrow keys.

The example in figure 4.8 shows disassembled memory starting at the address 400.

If you define the DIS memory range as AAAA-BBBB, by entering:

```
DIS AAAA-BBBB
```

and range AAAA-BBBB is more than 8 instructions, the DIS display will scroll through the memory range. The NO SCROLL key may be used to stop the scroll action so that a selected portion of the range can be viewed. Pressing the NO SCROLL key again resumes scrolling.

The starting memory address AAAA must be an instruction boundary, or the memory disassembly command is meaningless.

GENERAL	REGISTERS	HS MAPPING		TYPE	TRACE ON OFF	SYSTEM BREAK POINT	
		RANGE	ASSG				
PC 0000	B 00 C 00	0000-07FF	0	RAM		B0: UNDEF	OFF
SP 0000	D 00 E 00	0000-0000	1	OFF		B1: UNDEF	OFF
IX 0000	H 00 L 00 SZXHXPMC	0000-0000	2	OFF		B2: UNDEF	OFF
IY 0000	A 00 F 00[00000000]	0000-0000	3	OFF		B3: UNDEF	OFF
I 00	B' 00 C' 00	0000-0000	4	OFF		B4: UNDEF	OFF
R 00	D' 00 E' 00					B5: UNDEF	OFF
	H' 00 L' 00 SZXHXPMC					B6: UNDEF	OFF
	A' 00 F' 00[00000000]					B7: UNDEF	OFF

ADR	DATA	INSTRUCTION
0400 22 2A 00		LD (002A),HL
0403 CB 84		RES 0,H
0405 41		LD B,C
0406 05		DEC B
0407 DD 2A 2A 00		LD IX,(002A)
0408 DD 7E 00		LD A,(IX+00)
040E 57		LD D,A
040F DD 5E 01		LD E,(IX+01)

TYPE: Z80
 FREQ: 2500 KH
 FAsK: FasKey 1
 MODE: COMMAND

C:DIS 400_
 S:
 E:

FIGURE 4.8 Disassembling Memory

You can also send disassembled memory to a printer or write it to a file.

To Print Disassembled Memory:

To print disassembled instructions, starting at address AAA and ending at address BBB, enter:

DIs PPrint AAA-BBB<cr>

To Write Disassembled Memory to a File:

To write disassembled instructions, for an address range, to a file on a given disk device, enter:

```
DIIs WRite DEV:FILE.EXT AAA-BBB<cr>
```

where AAA-BBB is the address range.

Please note: The ALTER mode cannot be used with a DIS memory display.

SAVING DATA MEMORY

Data from any memory location can be stored in a file on a floppy diskette or hard disk, depending on your system configuration. The WRite command writes out the contents of memory in DEC LDA format at the specified address. The current map settings are in effect. Depending on the region of memory that you specify and the current map, you can store data that is currently in external, internal or high-speed memory.

To Store Contents of Memory

The contents of memory can be written directly to a storage file.

Enter the following:

```
WRite FILE.EXT AAA-BBB<cr>
```

For example, to write the records for address 800-BFF to the file XGA.LDA you would enter:

```
WR XGA.LDA 800-BFF<cr>
```

These records would be retrieved from internal, high-speed or external memory depending upon the current map configuration.

To Write a Program Residing in Non-contiguous Segments to Disk

You can also write a program that resides in non-contiguous segments of memory to a disk file using the WRite command.

For example,

```
WRite YDF.LDA 500-7FF,2000-21FF<cr>
```

writes a non-contiguous program into file YDF.LDA . You can enter up to 60 characters on the command line.

LOADING DISK FILES INTO MEMORY

Disk files that have been written in LDA format using the WRite command and files containing previously assembled and linked programs can be loaded into memory with the LLoad command. These files already contain the absolute address at which the file contents (hexadecimal data) must be placed. Reloading these images into memory is subject to the current map.

For example, to reload disk file XGA.LDA, which contains the memory segment written to disk in the previous example, enter:

```
LLoad XGA.LDA<cr>
```

Regardless of the current map specifications, the file contents will be loaded into the same logical addresses at which they were located prior to being transferred to disk, for example, 800 through BFF in external memory.

If the map has been changed during the interim, the file contents will be loaded into addresses 800 through BFF in external, internal or high-speed memory depending on the mapping configuration at the time of the load.

RELOCATING MAPPED MEMORY

You can relocate internal memory or high-speed memory to a different region of logical address space by changing the logical address range specified in the current map.

For example, suppose that internal memory is mapped

```
0000-FFF=0000 RAM
```

If the target system references a subroutine in logical addresses 100-3FF, it actually accesses internal memory locations 100-3FF.

You can relocate this subroutine to logical addresses 800-AFF without changing its physical location in internal memory by redefining the map as, for example,

```
0700-16FF=0000 RAM
```

The program is stored in internal memory 100-3FF, but now the target must send address requests in the range 800-AFF to read it.

This same procedure can be applied to relocate high-speed memory. Suppose data is stored in high-speed memory segment 0, which is mapped into logical addresses 000-7FF:

000-7FF=0 RAM

You can relocate this segment to another region of logical address space by redefining the map. For example to map this segment of memory into addresses 500 through CFF, enter:

MAp 500-CFF=0<cr>

MOVING MEMORY SEGMENTS

The MOVE command allows you to physically move blocks of data to a new stated address.

To move a block of data, enter:

MOve AAAA-BBBB CCCC<cr>

where AAAA-BBBB is the address range (inclusive) of the data to be moved and CCCC is the starting address for the desired new location.

For example:

High-speed memory is mapped for the range 100-7FF, and you wish to move external memory segment 30-4F to high-speed memory beginning at address 100, you would enter the following:

MOVE 30-4F 100<cr>

CONTROLLING AND MONITORING EMULATION

Introduction:

Once you have developed software to run on your target system, you can load the program into emulation memory and execute it in a real-time environment. Because the ECL-3211 ESP emulates the target microprocessor, you can test and control signals on the ESP and also manipulate software execution.

Eight breakpoints enable you to control program flow, system signals, and the system trace. They are easy to use, and yet more versatile and sophisticated than most emulator breakpoints. Each breakpoint consists of actions and conditions. When setting a breakpoint, you specify various conditions that must be met before the breakpoint will trip, and actions that will result from the breakpoint trip.

The ECL-3211 has two counters, four switches, and eight external inputs. These control breakpoints and external circuits, count events, measure time intervals, and monitor external system levels.

The "phantom" breakpoint feature enables the ECL-3211 to inject a patch of coded instructions into the executing program as a result of a breakpoint trip. The purpose of a phantom is to introduce changes to your program during emulation without actually modifying the original program code.

Other facilities discussed in this chapter include:

- o The ECL-3211 trace, which enables you to store and then retrieve status information about the system and external inputs at important moments during emulation.
- o The step feature for "stepping" through program execution on the ESP system.
- o The clock and frequency control, which enables you to emulate at either the speed of the target clock or ECL-3211 clock.

Refer to the Command Dictionary (Chapter 7) for alphabetized, complete descriptions of the commands discussed in this chapter. Any of the commands may be entered using the FasKey capability of the command mode. We suggest that you first familiarize yourself with the command syntax as demonstrated in this chapter and in the Command Dictionary. Refer to Appendix D at the back of this manual for specific FasKey keypad entry menus.

BREAKPOINTS

The ECL-3211 provides eight sophisticated breakpoints for trapping faults and monitoring program flow. Each breakpoint can:

- o Monitor ESP processor emulation at full speed,
- o Respond to any combination of the ESP processor 's address, data, and control lines,
- o Respond to any combination of ECL-3211 internal timers, switches, and counters,
- o Respond to any combination of eight external inputs (on pod), thereby functioning as an internal logic analyzer
- o Control program flow by halting, pausing, or injecting phantom machine code
- o Control any combination of ECL-3211 internal timers, switches, and counters (and thereby other breakpoints)
- o Control two external trigger outputs (on pod) to scopes, logic analyzers, or other user hardware
- o Control storage of program and processor information in the ECL-3211 trace buffer
- o Execute a user-created command file

Initially when you bring up the emulation system, all eight breakpoints are turned off. You can set and activate one or more breakpoints. They may be used independently, or may be linked by logical relationships, switches, or breakpoint priority to perform even more complex decisions.

For all their power, the breakpoints are easy to understand and use. The key is to first understand the structure of the breakpoint and the relationship of one breakpoint to another.

Breakpoint Structure

A breakpoint has two major components, an action table and a condition table. The action table contains actions to be performed when the conditions, which correspond to various target and microprocessor signals, are met. You may display, add, or remove entries in these tables with the BREAK command.

For example, to display the action and condition tables for breakpoint 3 in the central scroll area, type

BR 3<cr>

Figure 5.1 illustrates sample action and condition tables for breakpoint 3.

GENERAL	REGISTERS	HS MAPPING		TRACE	SYSTEM
		RANGE	ASSG TYPE	ON OFF	BREAK POINT
PC 0000	B 00 C 00	0000-07FF	0 RAM	B2	B0: 0100 ON
SP 0000	D 00 E 00	0000-0000	1 OFF		B1: UNDEF OFF
IX 0000	H 00 L 00 SZXHPNC	0000-0000	2 OFF		B2: LOGICAL ON
IY 0000	A 00 F 00[00000000]	0000-0000	3 OFF		B3: LOGICAL ON
I 00	B' 00 C' 00	0000-0000	4 OFF		B4: UNDEF OFF
R 00	D' 00 E' 00				B5: UNDEF OFF
	H' 00 L' 00 SZXHPNC				B6: UNDEF OFF
	A' 00 F' 00[00000000]				B7: UNDEF OFF
ACTIONS		BREAKPOINT 3 TRAP CONDITIONS			
PH=F000		E0: X	SW1: X	ADDR: 0000000110000000	[0180]
		E1: X	SW2: X	DATA: XXXXXXXX	
		E2: X	SW3: X	HALT: X	BUSRQ: X RD: X
		E3: X	SW4: X	WAIT: X	BUSAK: X WR: X
		E4: X	ROM: X	MREQ: X	NMI: X
		E5: X	INS: 1	RFSH: X	INT: X
		E6: X	CO1: X	IORQ: X	M1: X
		E7: X	CO2: X		
					PH= JMP
					TYPE: Z80
					FREQ: 2500 KH
					FasK: FasKey 1
					MODE: COMMAND

FIGURE 5.1 DISPLAYING A SPECIFIC BREAKPOINT'S PARAMETERS

During emulation, the ECL-3211 monitors all the signals relating to breakpoint decisions, and compares them to the contents of the condition tables for all eight breakpoints. If the signals match all of the conditions in a breakpoint's condition table, the ECL-3211 performs all the actions specified in that breakpoint's action table.

(The number of conditions that may be specified is extensive. For example, for the 8048, the ECL-3211 monitors 51 processor signals and 14 ECL-3211 system signals -- a total of 65 signals. Within a breakpoint, each of these signals can be specified independently. There are 3.6×10^{19} unique combinations that can be formed from 65 binary signals. However, since each condition may be specified in the breakpoint as 1 or 0, there are 10^{31} unique combinations of conditions that may be specified to trip the breakpoint.)

Setting Actions and Conditions

Breakpoint actions and conditions are set with the BReak command. The format for setting a breakpoint is

BReak N ACTIONS/CONDITIONS

The letter N represents the breakpoint number 0-7. The slash (/) separates actions from conditions in the command.

For example:

BR 2 CF/PC=102<cr>

Breakpoint 2 is set to execute a user-created command file when the first byte or word of an instruction is fetched from address 102.

Actions are added to a breakpoint's action table by entering the appropriate action codes separated by commas or spaces. For example, entering

BReak 0 HL,ST<cr>

adds the HALT and SET TRACE actions to the action table for breakpoint 0.

You can delete an action from the table by specifying the action code with the > prefix. For example,

BREAK 0 >ST<cr>

deletes the ST (SET TRACE) action from the action table for breakpoint 0.

To set breakpoint conditions, enter a slash (/) mark after the last action followed by your string of condition codes. Separate the conditions with the appropriate separator character as shown in Table 5.1. Each time you specify a breakpoint, the system modifies that breakpoint's actions and condition tables. Displaying a given breakpoint will present the actions and conditions of the current tables. The "greater than" (>) character is used only to delete a condition previously specified for a breakpoint.

TABLE 5.1 BREAKPOINT CONDITION SEPARATORS

CHARACTER	CONDITION STATE	BREAKPOINT RESPONSE
, or +	Set to logical 1	Respond when condition is true (logical 1). These separators are optional for before the first condition in a condition string.
-	Set to logical 0	Respond if false (logical 0). Prefix is invalid for conditions accepting word or mask arguments.
>	Null	Deletes condition from table.

Initially, there are no actions in the action table, and each condition is designated in the display by "X"

Consider the following command that sets breakpoint 1 for 8048 emulation.

```
BR 1 HL,STSW1,ENC01/PC=102,T0,RD
```

This command sets a breakpoint to execute three actions if the three conditions are met. If the instruction at address 102 is fetched and the T0 flag and the RD line are active (1) simultaneously, then the breakpoint halts emulation, sets logical switch 1 to "1", and enables counter 1. Note that the plus (+) mark would have the same effect as the commas separating the conditions in the example above.

If you want to set conditions without also specifying any actions, you simply precede the string of conditions with the slash (/).

For example:

```
BR 1/C01
```

This command sets the C01 condition to logical 1 (that is, counter 1 expired) for breakpoint 1.

NOTE: Our directions for stating a condition assume the condition to be active, asserted, true and electrically HIGH when the setting for the condition is logically 1. There are processor specific exceptions because of variations in hardware design. Please refer to your microprocessor supplement to this manual for verification.

Breakpoint Actions

=====

When a breakpoint is tripped, all of the actions in its action table are executed. The results are valid as of the beginning of the next processor machine cycle.

There are five types of actions a breakpoint can perform:

- o Control program flow
- o Control a switch
- o Control a counter
- o Control the trace buffer
- o Execute a command file

The breakpoint actions are listed below in Table 5.2

TABLE 5.2 BREAKPOINT ACTIONS

Action Mnemonic	Explanation
=====	
CF	Command File: The CF argument, as a breakpoint action, halts emulation switching system control to the default breakpoint command file (BRn.COM where "n" is the number of the breakpoint). For information concerning creation of breakpoint command files, see Chapter 6 - Creation and Use of Command Files.
HL	Halt: Stop emulation, update the screen display and go to command mode.
PA	Pause: Stop emulation, update the screen display and resume emulation.

TABLE 5.2 BREAKPOINT ACTIONS

Action Mnemonic	Explanation
STSW1	Set switch 1: Set logical switch 1 (and thus set trigger 1).
STSW2	Set switch 2: Set logical switch 2 (and thus set trigger 2).
STSW3	Set switch 3: Set logical switch 3.
STSW4	Set switch 4: Set logical switch 4.
CLSW1	Clear switch 1: Clear logical switch 1 and thus reset trigger 1.
CLSW2	Clear switch 2: Clear logical switch 2 and thus reset trigger 2.
CLSW3	Clear switch 3: Clear logical switch 3.
CLSW4	Clear switch 4: Clear logical switch 4.
CPSW1	Complement switch 1: complement logical switch 1 and thus complement trigger 1.
CPSW2	Complement switch 2: complement logical switch 2 and thus complement trigger 2.
CPSW3	Complement switch 3: complement logical switch 3.
CPSW4	Complement switch 4: complement logical switch 4.
ENCO1	Enable counter 1. Counter one is enabled for decrementing by cycles, instructions or ESP processor ticks.
ENCO2	Enable counter 2. Counter two is enabled for decrementing by cycles, instructions or ESP processor ticks.
DICO1	Diasable counter 1.

TABLE 5.2 BREAKPOINT ACTIONS

Action Mnemonic	Explanation
DIC02	Diasable counter 2.
DEC01	Decrement counter 1.
DEC02	Decrement counter 2.
ST	Set the Trace.
RT	Reset the Trace.
PH=AAA	Jump to phantom program or call phantom program. (For information on the type of phantoms supported by your microprocessor, refer to the appropriate, specific User's Guide supplement and to your processor specific HELP file provided in your emulation ECL-3211 software.)

Controlling Program Flow

You can set a breakpoint to alter program flow by including a HALT, PAUSE, PHANTOM or COMMAND FILE action in its action table. These actions are discussed in the following paragraphs.

HALT Action:

The HALT action (HL) causes emulation to stop. The emulator retrieves all the information needed to update the display, including the internal registers and status of the ESP processor. The ECL-3211 returns to the COMMAND mode. You can then enter any user command.

Meanwhile, the processor is left in an idling state. The ECL-3211 utilizes a branch-on-self instruction to prevent the pin values of the ESP processor from being affected by the HALT. If you enter the EMULATE command, the ECL-3211 restores the contents of the processor's internal registers, and the processor begins executing your program. If you have changed the values of the registers using ALTER mode or the SET command, the new values are loaded into the processor before the new emulation run begins.

PAUSE Action:

The PAUSE action (PA) is the same as HALT, except that the ECL-3211 temporarily stops emulation, updates the display screen, and then automatically restarts the emulation at the next instruction in the executing program. PAUSE allows you to view the updated register and status information (including the central scroll area). However, you will not have the opportunity to change register values before emulation continues.

The halt (HL), pause (PA), and command file (CF) actions create actual breaks in the processor's program flow.

PHANTOM Action:

A PHANTOM action can be one of two types: either a jump or a call. Whether your emulation software will support a phantom jump or a phantom call depends upon the particular processor you are emulating. Emulogic software adjusts for the capabilities of each processor. Refer to your processor specific Users' Guide Supplement to determine which action, jump or call, is supported by your processor.

A phantom jump causes the ESP processor to fetch its next instruction from the address specified in the breakpoint. It is exactly the same as inserting an absolute jump instruction into the program.

A phantom call is the logical equivalent of a subroutine call inserted at the next instruction after the breakpoint is tripped.

NOTE: The phantom call is NOT functional with every processor. Some processors do not have "calls", and others respond according to conditions peculiar to those. Refer to the appropriate User's Guide supplement for information about using calls with your emulation microprocessor.

When the breakpoint is tripped, the system saves the return address, and the ECL-3211 injects a jump or call instruction to the address you have specified. The processor then executes the machine code you have stored at the specified address. Just as with any subroutine call, a phantom subroutine must end with a return instruction to continue executing the original program. A phantom "jump" program must be terminated with an absolute jump rather than a return.

Suppose that you write a program patch into memory starting at logical address 750. To set breakpoint 5 to execute a phantom jump or call to this location when PC=FF, you enter:

```
BR 5 PH=750/PC=FF<cr>
```

Refer to "Phantom Programs" later in this chapter for a detailed explanation of use of phantoms.

COMMAND FILE Action

A breakpoint command may be used to execute a command file when a specific condition or set of conditions is encountered, thus performing a series of ECL-3211 commands. For example, the breakpoint command:

```
BR 2 CF/ADDR=102<cr>
```

instructs the system to execute the default breakpoint command file BR2.COM when the processor fetches the instruction at address 102.

Any legal ECL-3211 command may be used in an ECL-3211 breakpoint command file, thus giving you great flexibility for manipulating memory and system parameters in response to the occurrence of a user defined condition. The breakpoint command file may even call another command file. See Chapter 6 for complete information concerning the creation and use of breakpoint command files.

Setting and Clearing Switches

Each breakpoint can set, clear, or complement any or all of the four software switches provided in the ECL-3211. These switches may be interrogated after the emulation run, or used as input conditions to control other breakpoints.

Two of the switches, 0 and 1, are connected to the pod as trigger outputs. These switches can be used in conjunction with breakpoints to trigger scopes, logic state analyzers, logic timing analyzers, and target circuitry. Refer to "System Signals" later in this chapter for more detailed information about switches.

Controlling Counters

Each breakpoint can control either or both of the counters in the ECL-3211. There are three actions that control the counters: DECO_n, ENCO_n and DICO_n, where *n* refers to counter 1 or 2.

The DECO action decrements the specified counter by one each time the breakpoint is tripped.

The ENCO and DECO actions enable or disable, respectively, the specified counter respectively. These actions are taken only if the specified counter is in Interval Timer mode, Cycle Counter mode, or Instruction mode. Otherwise, these actions are ignored. When the breakpoint is tripped, the counter is enabled or disabled as of the beginning of the next machine cycle.

For more detailed information on counters, refer to the SYSTEM SIGNALS section later in this chapter.

Controlling the Trace Buffer

A breakpoint can also enable the trace buffer to begin storing machine states, starting with the next state. For example, entering

```
BReak 2 ST<cr>
```

places the trace enable action in breakpoint 2's action table. When this breakpoint is activated, the trace buffer begins storing the status of various microprocessor and target signals. The trace will be active until it is disabled or emulation is stopped.

Once emulation halts, you can display the contents of the trace buffer by typing

```
TRace<cr>
```

The trace contains information about the previous 511 machine cycles. It is particularly useful for studying program instructions and signal status preceding a fault in the executing program. Refer to the section "The Trace Buffer" in this chapter for more information about using the trace.

Breakpoint Conditions

As already mentioned, all of the conditions for a breakpoint must be met (that is, logical AND functional) before the breakpoint trips and executes the actions in its action table. The conditions correspond to ESP pod microprocessor and target system signals. Each of these signals are sampled

at the target processor's address valid time or data valid time (depending on the processor's timing) and then compared to the conditions at the end of the machine cycle.

System Conditions

System conditions refer to ECL-3211 system signals and can always be included in the breakpoint condition table, regardless of the particular microprocessor being emulated. They allow breakpoints to be controlled by switches, counters, and external inputs. These signals may, in turn, be controlled by breakpoints, thereby enabling breakpoints to control each other or even themselves. System conditions are bit-test conditions as shown in Table 5.3.

TABLE 5.3 SYSTEM CONDITIONS

CONDITION	DESCRIPTION
EXTERNAL INPUTS (0-7)	Allow you to trip breakpoints based on the state of any signals applied to the external input lines.
SWITCHES (1-4)	Provide control either prior to execution (by setting switches via keyboard commands) or during execution (by setting switches with breakpoints).
COUNTERS (1-2)	Allow breakpoints to respond to time limits, cycle counts, instruction counts, or breakpoint event counts.

Each of these signals is discussed in the text which follows.

Target Conditions

Target conditions are processor specific. (For example, the address word may be 8, 12, 16, or 32 bits wide. A processor may or may not have ports. Status and control lines may also vary among processors.) Refer to the appropriate processor supplement to this guide for a list of the breakpoint conditions you can set.

Some microprocessor conditions are bit-test conditions, and others are word-test conditions.

Bit-Test Conditions

Bit-test conditions are entered in the condition table as 0 or 1. If the table entry is 1, the breakpoint is tripped when the corresponding signal is 1 and all other conditions in the breakpoint table are met. If the table entry is 0, the breakpoint will be tripped when the signal is 0 and all other conditions are met. Any bit-test conditions which are not specified for a given breakpoint will appear as "X" on the breakpoint display. These are inactive and have no bearing on the breakpoint condition test.

On startup, all breakpoint conditions are inactive and appear as Xs in the display. This allows you to enter new conditions without having to keep track of which conditions are already set. If a breakpoint containing only X's in its condition table is turned on, it trips on every cycle (subject to priority rules), since its "conditions" are always met.

Word-Test Conditions

A word-test condition has two forms: word match or bit mask.

The word match tests for identity between the signals and the specified word. For example, for an eight-bit data bus, the eight signal lines are treated as two hexadecimal digits, and are compared to the two bytes of the condition specification. If they are equal, the breakpoint will be tripped (if all other conditions are also met).

The bit mask form allows the inclusion of X's in the test. The signals are then treated as eight separate bits to be tested individually against the corresponding bits in the specified mask. This form can be used to cause a word-test condition to be asserted over a range of data or addresses. For example, if you want to trip a breakpoint on every instruction below FF, you can set the PC (program counter) condition to PC=M00000000XXXXXXXX (for a 16-bit address).

Turning Breakpoints On and Off

Initially when you bring up the emulation system, all of the breakpoints are disabled. When you set actions and or conditions for a breakpoint, it is at the same time automatically activated.

You can clear and disable one or more breakpoints by entering the BReak command with the CLeAr option:

BReak N CLeAr<cr>

For example:

BR 0-7 CL<cr>

clears all eight breakpoints of their current definitions.

You can temporarily turn off one or more active breakpoints by including the OFF option, instead of CLeAr.

BR 2 OFF<cr>

turns off breakpoint 2.

To turn the breakpoint back on, later, simply enter

BR 2 ON<cr>

Breakpoint Relationships

Each breakpoint is defined separately and may be used to perform independent test and control functions. However, breakpoints may also be used in conjunction with one another to execute more complex functions.

Breakpoints may be linked to each other in three ways, by:

- logical relationships
- priority
- direct control

Logical Relationships

Each breakpoint responds to the logical AND of all its conditions. If two breakpoints contain the same actions but different conditions, the action will occur if the conditions for either breakpoint are met. This represents the logical OR function. If an action must take place on the result of a complex logical expression, then one breakpoint is needed for each OR function in the expression. To use the breakpoints efficiently, the expression must be reduced to the form containing the least number of OR functions. The required form is referred to as the MINTERM or "sum-of-products" form of the expression.

In formal notation, the minterm form is described as:

$$F = ABC + DEF + GHJ$$

Where + is the OR function. The AND function is represented by adjacent variables (that is, AB is A AND B).

F represents the fact that the actions will be taken, and the other variables in the expression represent conditions in the breakpoints' condition tables. In the above expression, three breakpoints would be needed.

For example, suppose you want to halt (HL) emulation if either a read operation (RD) from location 400 or a write (WR) to location 500 occurs. This could be represented in formal notation as:

Let $A = (PC = 400)$
 $B = (RD)$
 $C = (PC = 500)$
 $D = (WR)$
and $F = (HL)$

Then $F = AB + CD$

The final expression contains two terms, thereby requiring two breakpoints, for example: BR 0 = HL/PC=400+RD and BR 1 = HL/PC=500+WR.

If the condition expression is not in minterm form, then it must be rewritten so the terms may be allocated to the required breakpoints. For example, the expression $AB(CD + EF)$, would have to be factored using the rules of Boolean algebra to give $ABCD + ABEF$, which can then be entered into two breakpoint tables as $F = ABCD$ and $F = ABEF$.

Priority

Functions that cannot be implemented directly within a breakpoint or by the logical OR function between breakpoints, can be implemented by taking advantage of breakpoint priority.

WHEN THE CONDITIONS FOR TWO SEPARATE BREAKPOINTS
ARE SIMULTANEOUSLY MET, ONLY THE LOWER-NUMBERED
BREAKPOINT IS TRIPPED.

For example, breakpoint 0 has priority over breakpoint 1.

Using Breakpoint Priority for Condition False:

You can use a breakpoint with no actions in its table to prevent another breakpoint from tripping until a desired situation occurs. The most important use of this technique is to create the equivalent of a "-" prefix (that is, respond if false) on a word match condition. This cannot be done directly, because there is no provision for it in the breakpoint table. However, this prefix can be simulated with two breakpoints.

For example, set up breakpoint 0 with the word match or mask condition, but no actions. Then, set up breakpoint 1 with all its conditions set to X, and with the action that is to be taken when the word match or mask condition in breakpoint 0 fails.

When the word match or mask condition is met, the conditions for both breakpoints are met. Because breakpoint 0 has priority, it is tripped, and breakpoint 1 is not. Since breakpoint 0 has no actions in its table, nothing happens.

When the condition for breakpoint 0 fails, then only the conditions for breakpoint 1 will be met. Breakpoint 1's action will then occur. In this case, breakpoint 0 inhibits breakpoint 1 until its condition fails. Breakpoint 1 waits for that failure, and then trips when it occurs.

To illustrate this technique, let's restrict program flow to a region in memory. Suppose we want to halt emulation if the target processor tries to access memory above hex FF. To do this, set breakpoint 0 for no actions, and set the address lines bit mask to ADDR=M00000000XXXXXXX (we are assuming a 16-bit address).

BR 0/ADDR=M00000000XXXXXXX

Also set breakpoint 1 for a halt action (HL) with no conditions.

BR 1=HL

For addresses below hex FF, breakpoint 0 will trip on every instruction and do nothing. Breakpoint 1, which would normally trip on every cycle, is prevented from tripping by priority. For an address above hex FF, breakpoint 0 fails to trip, thereby allowing breakpoint 1 to halt emulation, as desired.

Direct Control

The third way to link breakpoints is through explicit control using software switches and counters, which allow breakpoints to control each other or themselves. Switches can be set, cleared, or complemented by breakpoints, and may also serve as input conditions to breakpoints. For example, if breakpoint 0 sets a switch that is a condition for breakpoint 1, breakpoint 0 may explicitly enable breakpoint 1.

Suppose you want to trip a breakpoint at a point in a subroutine where a fault occurs, but only when the subroutine is called from a particular location in the main program. You can set breakpoint 0 to trip at the calling instruction and to set switch 1 to logical 1 prior to calling the subroutine. Breakpoint 1 can be set to trip when both the fault occurs and switch 1 happens to be set to logical 1. In this situation, breakpoint 0 is activated only when the subroutine is called from a specific location in memory, and is used to explicitly control activation of breakpoint 1. Breakpoint 1 will not trip if the subroutine is called from some other location.

A switch can also control the breakpoint that sets it. To trip a breakpoint on only the first occurrence of an event, you might set a switch from the keyboard (using the Switch command) before starting emulation, and also set a breakpoint to trip when the event occurs and the switch is set. If you define this breakpoint to also reset switch n when it trips (with the CLSWn action), then the breakpoint will trip the first time the event occurs and will inhibit itself for all future occurrences of the event.

By using a counter, you can perform "N and only N" sequences. This can be done by setting a breakpoint with the DECON action (decrement counter n) in its action table. Each

time the breakpoint trips, it will decrement the specified counter. The breakpoint will trip as long as the counter has not yet reached zero. The breakpoint is thus enabling itself for a specific number of times. (Refer to "Counters" later in this chapter.)

Further uses of explicit control are limited only by your imagination. With four switches and two 31-bit counters, and eight breakpoints that may be coupled, there are many complex control structures that may be set up to accomplish sophisticated fault isolation.

SYSTEM SIGNALS

In addition to the signals generated by the ESP processor and the target system, there are several signals generated by the ECL-3211 which may be used to control breakpoints, measure program and hardware parameters, and control external circuitry. The system signals are listed in Table 5.4.

Switches/Triggers

The ECL-3211 has four software switches that can be used to control breakpoints and two external triggers. These switches are flip-flops on the ECL-3211 control board that may be set to a 1 or 0 state, or complemented, from the keyboard or as the result of a breakpoint action.

TABLE 5.4 SYSTEM SIGNALS

SIGNAL	FUNCTION
Switches/Triggers	Control breakpoints and external circuits
Counter/Timers	Count events, control breakpoints, and measure intervals in time, cycles, or instructions
External Inputs	Monitor external system levels that are not otherwise provided for, and use these levels to control breakpoints

The switches are numbered 1-4, and each responds to a command or breakpoint according to its number. For example, to set switch 1 to 1 via the keyboard, enter the Switch command:

```
SW 1=1<cr>
```

The switch 1 flip-flop will set at a 1 state. To clear this switch, type

```
SW 1=0<cr>
```

and the switch will set at the 0 state.

You can complement switch n during emulation by including the CPSWn action (n=1,2,3, or 4) in the breakpoint action table.

Switches 1 and 2 are switch/triggers connected to TRIGGER 1 and TRIGGER 2 outputs, respectively, on the pod assembly. These switches may be used to trigger an external logic state analyzer, oscilloscope, or target circuitry.

For example, to observe the exact timing of an interrupt line during a fault condition, a breakpoint may be set to respond when the fault occurs. If the breakpoint contains the set switch 1 (STSW1) action and TRIGGER 1 is connected to a logic analyzer, you can obtain a center triggered record with 5 nS resolution to determine whether the interrupt signal is being presented to the ESP processor at the right time in its cycle.

Another use for the trigger might be to fire a pulse generator to initiate a signal in the target system.

When they are not being used as external triggers, switches 1 and 2 may be used in exactly the same way as the other two switches.

Switches 3 and 4 are not brought out to the external system; they are used only as software switches to control breakpoints or to record simple events under breakpoint control.

The switches actually change state at the time when the breakpoint actions are taken -- after the data strobe time of the ESP processor, and before the start of the next processor machine cycle.

Counters

There are two 31-bit counters in the ECL-3211, which may be used in any of several ways to measure and control the program under test. These counters operate in one of four operating modes:

- o Real-time interval measurement
- o Target processor cycle count
- o Target processor instruction count
- o Breakpoint-controlled event count

Each of these counters may be set and read from the keyboard, tested for termination in breakpoints, or controlled by breakpoints, depending on the operating mode.

The numerical value and operating mode of a counter may only be set from the keyboard, using the COUNt command.

Range of Counter

The count value may range from 0 to 2,147,483,647 ($(2^{31})-1$).

Counter Modes

An active counter that has been set to real-time interval mode (IN) is decremented once per clock cycle. The clock source can be either the ECL-3211 clock or the external clock. You can select the clock source (and its frequency in the case of the ECL-3211 clock) by entering the FREQ= command. (Refer to "Setting the Clock Source and Frequency".) To set counter 1 to real-time interval mode with an initial value of 999, enter

```
CO 1=999 IN<cr>
```

Setting a counter to cycle count mode (CC), automatically decrements the counter once per machine cycle. If, instead, you set the counter to instruction count mode (IS), the counter automatically decrements once per instruction.

A counter set to IN, CC, or IS mode can be turned on or off via breakpoint control using the ENCO_n (enable counter *n*) and DICO_n (disable counter *n*) actions. Then, the breakpoints effectively define the portion of the program to be measured by the counter.

The DECO_n action (decrement counter *n*) has no effect on a counter in any of these modes.

However, the DECO breakpoint action does have an effect on a counter set to breakpoint-controlled event count mode. The counter is decremented whenever a breakpoint trips with the DECO action in its action table. In this mode, though, the counter will not respond to ENCO and DICO breakpoint actions. To set a counter to "breakpoint" mode, enter the COUNT command with a count value, but without a mode type. For example,

```
CO 2=25<cr>
```

sets counter 2 to a value of 25 in breakpoint mode.

If you include the RI option with the COUNT command when setting the count value and mode, the counter will re-initialize whenever emulation is interrupted.

CO 2=25 RI<cr>

Counter 2 will be re-initialize to its original value, 25, whenever you reenter emulation.

Using Counters With Breakpoints

A breakpoint can test for a counter's terminal count, but not its actual value. The condition signal for a counter becomes 1 when the counter passes through zero. A breakpoint can test the condition bit for 1 or 0 (or, of course, X).

A breakpoint that lists a count condition as 1 will trip when the counter expires. A breakpoint that tests the condition bit for 0 will trip on every cycle (if all its other conditions are met) until the counter expires. Thus a counter can be used to inhibit a breakpoint until it finishes its count, or enable the breakpoint while it is counting.

External Inputs

The ECL-3211 provides eight external inputs, numbered 0-7, that are connected to BNC connectors on the pod assembly. These inputs may be driven by TTL levels from the target system circuitry. You can set a breakpoint to respond to any of these input lines. If the trace is on, the state of each input is recorded in the buffer.

These eight external inputs provide the same full-speed testing and control functions that are available for the address, data, and control lines on the ESP processor, but allow you to decide which additional signals should be included in the testing procedure. Through judicious selection of the signals applied to these inputs, the ECL-3211 can be used for many tests that would otherwise require the use of external test equipment (e.g., a logic state analyzer).

PHANTOM PROGRAMS

The following section explains the concepts behind phantom programs. (For information on phantom command syntax, refer to the section "Breakpoint Actions" earlier in this chapter and to Chapter 7 - The Command Dictionary.)

The provision for phantom programs in the ECL-3211 simplifies the modification and testing of target system software. A phantom program may be effectively inserted into the test program in response to a breakpoint without modifying the original machine code sequence.

A phantom program is a machine code segment located in some specific position in memory (usually in a patch code area reserved for the purpose). The phantom program is executed when a breakpoint is tripped that contains the phantom's address in its action table. The transfer of control to the phantom code segment is accomplished by the ECL-3211 at full speed.

The ECL-3211 performs a transparent jump or call by synthesizing the machine code required and force-feeding it to the processor. Thus, no instructions are actually inserted into the target program code. Because the jump or call is transparent, it takes up no space in the memory, and does not need to be located in RAM. This feature makes it possible to insert modified code segments of arbitrary length into an existing machine code program in ROM or in space-limited RAM. The modifications may then be fully tested at full speed before burning a new ROM or permanently committing RAM space.

The phantom program itself may be generated in assembly language using the assembler and loaded into memory as a unit, or it may be generated at the keyboard via ALTER mode.

There are two types of phantom program segment: the phantom jump and the phantom call. Each has its special requirements and uses.

Phantom Jump

The phantom jump code segment is a simple program patch. It is entered by a synthesized jump instruction in response to a breakpoint. Instructions in the segment are simply performed in sequence. No registers or addresses are saved. The phantom jump code segment must provide the means of returning the processor to the main program.

Phantom Call

The phantom call code segment is a true subroutine, called by the ESP processor exactly the same way as any other subroutine call. This type of phantom code may only be used if the processor has a CALL instruction in its instruction set (refer to the appropriate User's Guide supplement). If not, the ECL-3211 will support only a phantom jump.

Like any other subroutine, the phantom call will cause the return address to be saved. This is done by the processor, according to its own architecture, and is not affected by the ECL-3211 in any way. The ECL-3211 does not know what the return address is; only the ESP processor knows. Therefore, the processor must recall the return address exactly the same way as it does for any other subroutine in the program.

Using the phantom call does not give the ESP processor an extra level of subroutine call. If the program uses subroutines elsewhere, then you must take all the steps necessary to ensure that the subroutine stack does not overflow when a phantom subroutine is added. As with any other subroutine, the phantom call code segment must end with a proper return instruction, and any stack manipulations and register saving or other housekeeping must be taken care of.

Conditional Phantom Jumps and Calls - A Warning

It is important to remember what is being done by the program software and what is being done by the ECL-3211. Since a phantom code segment is entered in response to a breakpoint, and a breakpoint is tripped on a set of conditions (which could be quite complex), it is possible to create a phantom code segment that appears to give the ESP processor some magical qualities.

If a breakpoint is set up to respond to a complex situation, and a phantom segment is then used to take care of that situation, the resulting combination of phantom software and breakpoint control may perform the task much faster than the processor could do by itself. A working program might result that will delude the programmer into thinking that the job is done. However, the processor must still be programmed to duplicate the work done by the breakpoint. In some cases, that work might take a significant amount of time and memory, as well as programming effort, since the ECL-3211 is using a high-speed, dedicated processor to evaluate the breakpoint conditions in real time.

To avoid this problem, you must strongly resist the temptation to use the power of the breakpoints to substitute for programming. The best and simplest solution is to restrict the condition code in a phantom breakpoint to an address, and force the ESP processor to make all the other decisions.

THE TRACE BUFFER

The ECL-3211 provides users with a full-speed trace capability. The trace feature may be turned on and off under breakpoint control to monitor relevant information during specific moments of emulation.

The trace buffer consists of 512 records, each of which can store up to 72 bits of information. (The number of bits actually used depends on the particular processor being emulated). One record represents one machine cycle. Each address, data, and control line on the ESP and each of the eight ECL-3211 external input lines (providing a logic state analyzer function) is sampled at full speed when it becomes valid for inclusion in the trace buffer. The address bus contents, for example are latched using the "address ready" signal of the ESP processor itself. The data bus contents are disassembled for display, so that software information may be gathered in terms of assembly language mnemonics.

Use of the trace buffer does not carry any penalty in execution speed. The processor clock may be set at its maximum specified rate, and the timing of the test program may be determined. No correction factors must be added.

Because the trace buffer may be turned on or off as a breakpoint action, the trace information may be qualified by any combination of conditions that may be specified in a breakpoint. (No time penalty is incurred when controlling the buffer with breakpoints.)

When the trace is turned on, the trace buffer begins storing information in buffer location 511. Each new record will occupy logical location 511; the previous records will move down one location to make room. When the buffer is full, each record that is added to the buffer thereafter displaces the record at buffer location 0. The displaced record is lost. When the trace is turned off, the buffer retains the most-recently stored 511 records. You can examine the contents of the trace buffer in COMMAND mode by typing the command

```
TRace<cr>
```

The eight most recently stored records will appear in the central scroll area. The remainder of the trace buffer may be viewed by using the up- and down-arrow keys to scroll through the buffer. Figure 5.2 displays the contents of a sample trace.

You can specify a record number at which the system should start displaying the trace. For example,

```
TR 495<cr>
```

displays the trace, starting at record 495.

You may also specify a range of records for the system to display. For example,

```
TR 480-511<cr>
```

The contents of the entire trace buffer may be printed by using the TRace PRint command. A portion of the trace buffer may be selectively printed by specifying a range of locations. You can also store the trace buffer contents on floppy disk by using TRace WRite FILE.EXT command.

GENERAL		REGISTERS		HS MAPPING		TRACE		SYSTEM	
				RANGE	ASSG TYPE	ON	OFF	BREAK POINT	
PC	004F	B	00 C 80	0000-0000	0 OFF	B3		B0:	LOGICAL ON
SP	006D	D	7E E 7F	0000-07FF	1 RAM			B1:	LOGICAL ON
IX	027F	H	02 L 00 SZXHPNC	0000-0000	2 OFF			B2:	LOGICAL ON
IY	0000	A	FF F 55[01010101]	0000-0000	3 OFF			B3:	LOGICAL ON
I	00	B'	00 C' 00	0000-0000	4 OFF			B4:	UNDEF OFF
R	71	D'	00 E' 00					B5:	UNDEF OFF
		H'	00 L' 00 SZXHPNC					B6:	LOGICAL ON
		A'	00 F' 00[00000000]					B7:	UNDEF OFF
LOC		ADDR		INSTRUCTION		DATA		EXTERNAL	
								76543210	H B W N I M R I M B R W
504	0123	BIT	0,H		CB	00000000	1 1 1 1 1 0 1 1 0 1 0 1		
505	0124				44	00000000	1 1 1 1 1 0 0 1 0 1 0 1		
506	0125	JR	NZ,DC		20	00000000	1 1 1 1 1 0 0 1 0 1 0 1		
507	0126				DC	00000000	1 1 1 1 1 0 0 1 1 1 0 1		
508	0127	RET			C9	00000000	1 1 1 1 1 0 1 1 0 1 0 1		
509	006B				4F	00000000	1 1 1 1 1 0 0 1 1 1 0 1		
510	006C				00	00000000	1 1 1 1 1 0 1 1 1 1 0 1		
511	004F								
C:TRACE_								TYPE:	Z80
S:								FREQ:	6000 KH
E:								FastK:	FastKey 1
ECL disconnected								MODE:	COMMAND

Figure 5.2 Sample Trace

SETTING THE CLOCK SOURCE AND FREQUENCY

Emulation speed is controlled by either the target (external) clock or the ECL-3211 internal frequency synthesizer. The frequency of the external clock varies according to your processor. The ECL-3211 clock has a default value, which can be modified.

Initially, the clock source is the ECL-3211 clock. The ECL-3211 clock is set to a default value that is appropriate for your processor. This value may vary among processors.

You can use the FREQ command to select the target clock as the clock source, or to change the frequency of the ECL-3211 clock. For example,

```
FR EXT<cr>
```

sets the clock source to the target's clock.

```
FR 2000<cr>
```

sets the clock source to the ECL-3211 clock and changes the frequency to 2000 KHz (2 MHz).

If you include the IND option when setting the ECL-3211 clock frequency, the ECL-3211 will not only drive the ESP processor, but also peripheral devices in the target system. For example:

```
FReq=2000 IND<cr>
```

The ECL-3211 clock will drive both the pod and external peripherals at 2000 KHz. (This feature is not available for all processors; please refer to the specific processor supplement to this manual to determine if the IND option is available for your specific processor.)

Whenever you enter the FREQ command, the emulation system suspends processor access for 30 seconds while the emulation frequency is adjusted, and then performs a hardware reset. During the 30 second wait period, any command which requires running the emulator (such as EMULATE, SET, MEM, SET MEM, LOAD, WRITE, DIS) will be deferred until the time-out period has expired. A message will appear on the screen indicating that the system is waiting. When the command is executed, the message "RESET PERFORMED" will appear on the S: line to remind you that a hardware reset (the equivalent of the ECL-3211 RESET command) has been performed.

SETTING LOGICAL ENTITIES

The screen display shows the current value for each logical entity (registers and flags) associated with the target processor. To set or change these hex values, you can enter ALTER mode. In the ALTER mode, move the cursor to the appropriate bit and type in the new value. You may also use the SET command to establish or change the value for each logical entity. For example, to set the program counter (PC) to 8FF, you can enter:

```
SET PC=8FF<cr>
```

Available logical entities are processor-dependent; their code designations vary by vendor and microprocessor. Refer to the appropriate User's Guide supplement for the logical entities that can be specified for your development microprocessor.

STARTING EMULATION

Before you can execute a program on your target system, you must first load your program into emulation memory. You may also want to activate some of the emulation facilities (e.g., breakpoints, trace) and set target or ECL-3211 values.

You can simplify this setup procedure by creating an indirect command file with the KED keypad text editor, and then writing the appropriate ECL-3211 commands into the file. Each time you wish to start emulation under the same conditions, you need only call the indirect command file.

Consider the following sample indirect command file called PLOG1.COM, which contains these commands:

```
LOAD PLOG.DAT
MAP INT
MAP 1F00-1FFF=0000
BR 0 ST,PA/PC=2000
BR 1 PA/+RD-WR,PC=2001
FREQ=1000
SET PC=1F00,A=5,B=9,SP=2300
MEM 2000
EMU
END
```

Once you are operating in the ECL-3211 emulation system, you can set up and start emulation by entering

```
CF PLOG1.COM<cr>
```

This file will load your program, relocate internal memory, activate breakpoints 0 and 1, set the ECL-3211 clock frequency, set the program counter and ESP processor registers, display a memory segment, and start emulation at the current program counter.

For more detailed information on the creation and use of command files, refer to Chapter 6 of this manual. Refer to Chapter 3 for more information on start-up options.

STEPPING THROUGH A PROGRAM

The ECL-3211 also enables you to "step" through your program. When the system is in step mode, the program executes a step, stops emulation, and updates the screen display. When you enter a carriage return, the ECL-3211 executes the next step.

The STEP command places the system in STEP mode. If you enter the command:

```
STEP<cr>
```

the system steps one instruction cycle, halts emulation, updates the display screen, and then awaits your carriage return to execute the next step (the next instruction cycle).

When you enter the STEP command:

```
STEP B<cr>
```

the system steps to the next breakpoint that executes a HALT or PAUSE action, halts emulation, and updates the screen. Entering a carriage return causes emulation to continue until the next breakpoint that executes a halt or pause.

USING THE SYMBOLIC DEBUGGER

The Emulogic symbolic debugger enables you to enter symbolic arguments in place of hex values in ECL-3211 commands.

A symbolic argument may be entered in any of these three formats:

```
AAAAAA
```

```
AAAAAA+BBBBB
```

```
AAAAAA-BBBBB
```

where AAAAAA is a legal RADIX50 name, and BBBBB is a hexadecimal number.

For example, you might use symbolic arguments as in the following commands:

```
SET PC="ACE",A="QUEEN+4"<cr>
```

```
BR 7="JACK"<cr>
```

The ECL-3211 searches the internal symbol table and any external symbol table files that have been opened for the number associated with the specified symbol. The number in the table is then combined with BBBB to obtain the result, which is returned as a hex number.

Figure 5.3 illustrates use of the symbol "KING" to set a simple breakpoint. Because the value for KING is 450 (hex), breakpoint 3 will trip at address 450 and will halt emulation.

The internal symbol table contains the sixteen most recently created symbols. The SYmbol command enables you to add, delete, or modify internal symbol table information, or to load the internal symbol table from an external file.

GENERAL		REGISTERS		HS MAPPING		TRACE		SYSTEM	
				RANGE	ASSG TYPE	ON	OFF	BREAK POINT	
PC	0000	B	00 C 00	0000-07FF	0 RAM			B0:	UNDEF OFF
SP	0000	D	00 E 00	0800-87FF	1 ROM			B1:	UNDEF OFF
IX	0000	H	00 L 00 SZXHPNC	A000-BFFF	2 RAM			B2:	UNDEF OFF
IY	0000	A	00 F 00[00000000]	C000-CFFF	3 RAM			B3:	0450 ON
I	00	B'	00 C' 00	F800-FFFF	4 RAM			B4:	UNDEF OFF
R	00	D'	00 E' 00					B5:	UNDEF OFF
		H'	00 L' 00 SZXHPNC					B6:	UNDEF OFF
		A'	00 F' 00[00000000]					B7:	UNDEF OFF
ACTIONS		ACTIONS		BREAKPOINT 3 TRAP CONDITIONS					
		HALT		E0: X	SW1: X	ADDR:	0000010001010000 [0450]		
				E1: X	SW2: X	DATA:	XXXXXXXX		
				E2: X	SW3: X	HALT: X	BUSRQ: X	RD: X	
				E3: X	SW4: X	WAIT: X	BUSAK: X	WR: X	
				E4: X	ROM: X	MREQ: X	NMI: X		
				E5: X	INS: X	RFSH: X	INT: X		
				E6: X	CO1: X	IORQ: X	M1: X		
				E7: X	CO2: X				PH= JMP
								TYPE:	Z80
								FREQ:	2500 KH
								FasK:	FasKey 1
								MODE:	COMMAND
C: BREAK 3="KING" _									
S:									
E:									

You can reference a maximum of four external symbol table files. These files are written in symbol table format (STB), and can be obtained as optional output from the Emulogic linker (ELINKx).

To add SOFT to the internal symbol table with a value of 250, and change the value of LIFT to 99, enter

```
SY SOFT=250,LIFT=99<cr>
```

To reference symbols contained in an external file named COORD.STB, type

```
SY COORD.STB<cr>
```

If you want to delete SOFT and LIFT from the internal symbol table, type this command:

```
SY CL SOFT,LIFT<cr>
```

To clear the entire internal symbol table, enter:

```
SY CL<cr>
```

You may find it convenient to use an indirect command file to maintain a list of frequently used symbols.

CHAPTER 6

CREATION AND USE OF COMMAND FILES

The ECL-3211 provides the capability to create, store and execute command files. A command file is a user-created file containing a series of commands. The system performs the commands in sequence without keyboard intervention. By using command files, you can streamline recurrent tasks, saving input time and eliminating errors. Command files can be used for such tasks as to:

- . set startup emulation parameters
- . run emulation
- . write traced instructions following a breakpoint to a file or printer
- . write memory images to disk or a printer
- . perform one or more ECL-3211 commands as a breakpoint action when a given breakpoint condition has been encountered
- . Compare two files and display their differences, write the differences to file, or print.

Creative use of command file capabilities depends on the ingenuity of the user. There are two types of command files which may be used with the ECL-3211 development system: the DEC RT-11 operating system command file and the Emulogic ECL-3211 command file.

In this chapter, we have provided instructions for:

- Creating command files
 - . in RT-11 edit
 - . from the ECL-3211 command line using the LOG command
- Use of ECL-3211 command files

- Use of RT-11 command files
 - . in conjunction with ECL-3211 command files
 - . in conjunction with IND control files and ECL-3211 command files
- Programmable "FasKey" access to Command Files

At the end of the chapter, we have included a glossary of command file mnemonics and have provided some sample command files.

CREATING COMMAND FILES

Both RT-11 and ECL-3211 command files can be created using RT-11 edit facilities. ECL-3211 command files can also be created while operating in ECL-3211 command mode by using the ECL-3211 LOG command.

Both ECL-3211 and RT-11 command files may be modified only by using RT-11 edit facilities.

CREATING A COMMAND FILE UNDER RT-11

The command file name is limited to six characters. Assign the three character command file extension you will be using in ECL-3211. The ECL-3211 default command file extension for calling a previously created command file is ".COM". (This default, assumed in ECL-3211 only, may be changed via user invoked options.)

Use RT-11 syntax and commands in RT-11 edit when constructing an RT-11 command file (a command file designed to run in RT-11). For more complete information concerning creation and utilization of RT-11 command files, refer to the appropriate RT-11 USER'S GUIDE.

Use ECL-3211 syntax and commands for the command file text when designing an ECL-3211 command file in RT-11 edit. Any ECL-3211 command may be used in an ECL-3211 command file; however, the last command must be "END".

A Sample RT-11 Command File

To demonstrate RT-11 command file syntax, we have created a sample RT-11 command file below. We have called the command file CYCLE1.COM. The command file directives entered for the sample command file CYCLE1.COM are as follows:

(Input of a carriage return is assumed after each command.)

```
RUN L00500.SAV
TEST1.COM
RUN L00500.SAV
TEST2.COM
DIFF/BINARY/OUTPUT:COMP1 TEST1.DAT,TEST2.DAT
```

Two ECL-3211 start-up command files, TEST1.COM and TEST2.COM, are passed by CYCLE1.COM to the Emulogic start-up program (L00500.SAV in this example) to be executed in ECL-3211. The two files are previously established ECL-3211 command files which can be accessed by the RT-11 system. After execution of the second command file, the data files (created by commands within TEST1.COM and TEST2.COM) are compared by the SRCCOM RT-11 utility.

TO RUN A STORED RT-11 COMMAND FILE

To run a previously created RT-11 command file, enter

@FILE.EXT

following the RT-11 command mode prompt "."

For example, to run the previously created command file CYCLE1.COM, you would enter:

@CYCLE1.COM

CREATING ECL-3211 COMMAND FILES

ECL-3211 command files can be created in two ways:

- By using RT-11 EDIT to create the command file
- By using the ECL-3211 LOG command while operating within the ECL-3211 development system software.

A Sample ECL-3211 Command File

To demonstrate ECL-3211 command file syntax, we have created a sample ECL-3211 command file below. We have called the command file TEST1.COM. The ECL-3211 command file commands entered for "TEST1.COM" are as follows:

(Assume that a carriage return has been entered after each command entry.)

Command	Procedure
=====	=====
!FILENAME: TEST1.COM	Display "FILENAME: TEST1.COM"
SET PC=123,SP=7,IX=45,IY=34	Set chip specific registers
MAP HS	Map high-speed memory
MAP 500-AFF=1	Define High-speed map 1 as addresses 500-AFF
BR 0 ST/PC=123	Define breakpoint 0 to set trace when PC=123
BR 1 CF/SW4	Define breakpoint 1 to execute command file when switch 4 is on.
CF VARY1.COM	Execute command file VARY1.COM
BR 2 HL/PC=2ED	Define breakpoint 2 to halt when PC=2ED
EM	Run emulation
TRACE WRITE TRACE1.TRA 400-511	Write trace records 400-511 to named file, TRACE1.TRA
SAVE TEST1.DAT	Save current emulation status and write data to named file, TEST1.DAT
EXIT	Exit from ECL-3211 and return to RT-11
END	Close command file

(Please note that in our sample an EXIT command precedes the ECL-3211 command file END command. We have ended the command file in this manner because we want the command file to exit from ECL-3211 after execution to return to a "parent" RT-11 command file - CYCLE1.COM, to continue our example.)

Creating an ECL-3211 Command File Under RT-11 Edit

An ECL-3211 command file can be created using RT-11 edit facilities in exactly the same manner as you would to create an RT-11 command file. The only difference in procedures is that you will enter ECL-3211 commands for your command file text and, most important, YOU MUST REMEMBER TO ENTER THE COMMAND "END" AS YOUR LAST COMMAND ENTRY.

Creating an ECL-3211 Command File Using the LOG Command

To log plus execute:

The ECL-3211 LOG command has been designed to log or keep track of all subsequent commands entered from the keyboard at the command line of the ECL-3211 until "LOG END" is entered. All commands entered while the system is in LOG mode are executed by the system and written to a user-designated file (FILE.EXT). When you end the logging session, you enter "LOG END" from the ECL-3211 command line which enters an END command to the log file you have created and closes the file. You have now conveniently created a file which can be accessed as a command file.

To log without execution:

An additional LOG command, LOG/C FILE.EXT, allows you to create a command file using the log capability WITHOUT EXECUTING THE ENTERED COMMANDS. This can be a definite advantage as an onboard mini editing feature for creating command files, for you will most likely want to make some additions or changes in a command stream without executing the entire stream.

The command files you have created using the LOG command may be accessed and executed in the same ways as any other ECL-3211 command file.

Any legal ECL-3211 commands may be used while in the LOG mode, except cursor movement commands.

As with ECL-3211 command files created under RT-11, you may edit the log-created command files using the LOG command only under RT-11 edit.

This "onboard" ability to build a command file can be used creatively in many ways, such as inserting command text as needed, for expanding emulation capabilities. It can also be used to test alternatives for command file command flow, or to experiment with new emulation parameters.

Sample Use of the LOG Command to Establish ECL-3211 Command Files

Example 1: To create a command file called VARY1.COM without executing the command session, enter:

```
LOG/C VARY1.COM
```

enter the desired ECL-3211 commands, each followed by a carriage return, then terminate the LOG session and store the newly created command file by entering:

```
LOG END
```

Example 2: To run a command session and LOG the session to a file called VARY1.COM, enter the command:

```
LOG VARY1.COM.
```

enter the desired ECL-3211 commands, then terminate the LOG session and store the newly created command file by entering:

```
LOG END
```

TO RUN A STORED ECL-3211 COMMAND FILE

An ECL-3211 command file may be executed in a number of ways:

- It may be called from the command line by a "CF" command while operating in ECL-3211 command mode.
- It may be called by another ECL-3211 command file
- It may be called as an action of a breakpoint
- It may be passed as a startup command file (see Chapter 3, Bringing Up the System).

Let's take a look at the sample command file TEST1.COM again:

```
!FILENAME: TEST1.COM
SET PC=123,SP=7,IX=45,IY=34
MAP HS
MAP 500-AFF=1
BR 0 ST/PC=123
BR 1 CF/SW4
CF VARY1.COM
BR 2 HL/PC=2ED
EM
TRACE WRITE TRACE1.TRA 475-511
SAVE TEST1.DAT
EXIT
END
```

To execute our sample command file while in the ECL-3211 command mode, you would simply enter the following from the ECL-3211 command line:

CF TEST1.COM

In our sample command file, the nested command file VARY1.COM is opened and executed. The parent command file TEST1.COM is placed on the command file stack when the nested command file is opened so that processing may return to TEST1.COM when the END command is executed by VARY1.COM.

ECL-3211 command files can be nested to a depth of five. If an attempt is made to nest beyond this level, an error message is given and the command stack is cleared.

If a CF command is the last command before the parent command file's END command, the parent command file will not be loaded onto the stack and the CF command will in effect act as a link or bridge to the requested command file.

A command file can call itself as the last command in the file, and thus create a command loop.

Needless to say, command files may be called by command files. A breakpoint command file may call another command file. The only warning to bear in mind is that the command files may not be nested to a depth greater than five. (Please note also that the CF/T command which allows you to display a command file in the central scroll area during a pause will add the command file from which you are currently operating to the command file stack.)

PAUSES

The CF/P Command - Command File Pause

A pause in command file execution can be user-initiated by including the command "CF/P", command file pause, in the command file text. While a pause is operative, commands may be inserted from the command line by the user. This can be a useful tool when you wish to test variables at a given moment in a command stream. When the command to pause is being executed, "CF/P" and the name of the command file will display on the E: Line of the screen display.

To insert a pause in the command file execution when building the command file text, type:

CF/P

To resume execution of the command file, type the following from the ECL-3211 command line:

CF/R

"CF/R" will resume execution of the command file.

To Interrupt Execution of a Command File:

Execution of a command file may also be interrupted directly from the keyboard at any time during the processing of the command file. To stop execution of a command file, enter CTRL C twice (simultaneously depress the "Control" and "C" keys two times).

The message "CF/P" and the name of the interrupted command file will appear on the E: Line of the screen display.

To resume execution of the command file, enter:

CF/R

To clear the command file and the command file stack completely, enter:

CF/C

To Pause During an EMULATE Command from a Command File:

If the ECL-3211 is processing an EMULATE command from a command file, any keyboard interrupt, such as entering a carriage return, will halt emulation. At this time the command file will enter a command file pause (CF/P) state. The message "CF/P" and the name of the command file will appear on the E: line of the screen display.

Command File Error Display

A pause in the execution of a command file will occur and be signaled to the user when an error in command file execution has occurred.

Any legal ECL-3211 commands (except cursor movement commands) are allowed in emulation command files. If an illegal command is encountered, the system pauses during the run of the command file and "CF/P" and the name of the command file are displayed on the E: line of the screen display. The illegal command is not executed.

To resume processing, enter CF/R (command file resume). ECL-3211 system control will move to the next command following the illegal command.

Use Control Keys During a Command File Pause

Four keys in the control keys keypad (the keypad on the right side of the keyboard) have been programmed to have special capabilities when the ECL-3211 is in command file pause (CF/P is displayed on the E: Line). The cause of the pause is irrelevant, whether it be operator induced or system-induced due to an error.

During any command file pause, the - , ' , . , and ENTER keys on the keypad on the right side of the keyboard have the following special capabilities:

- Use of the minus key causes the command file to "backspace" to the last performed command in the command file and to re-execute that command.
- ' Use of the comma key gives you a help file defining the special functions of the CF/P control keys.
- . Depression of the period key serves as a CF/R command and resumes command file processing.

ENTER Use of the ENTER key ends execution of the current command file.

Command File Comment Lines

A comment line may be inserted in an ECL-3211 command file by

using an exclamation mark (!) as the first character of the intended comment line. Each line of the comment must be preceded by an exclamation mark, as in the example:

```
!THIS IS A COMMAND FILE COMMENT LINE
!A COMMAND FILE COMMENT LINE CANNOT EXCEED 60 CHARACTERS
```

The comment line(s) will be displayed on the S: Line of the ECL-3211 screen display when the command file executes.

Prompting for Command Line Input

A command line can be set to wait for keyboard input by including an @ as the last character of the command line. The user can also create an accompanying comment line to appear below the fill-in line to prompt for specific input. Two steps are needed to create the prompt and fill-in message:

First, using a comment line command (!), enter the desired text for the prompt followed by a carriage return.

Next, enter the command line text:

```
COMMAND @<cr>
```

where COMMAND is the desired ECL-3211 command and @ precedes the position for fill-in.

For example, to prompt a user to execute a SET MEM command, you could enter the following two commands in the command file stream:

```
!ENTER "ADDRESS=VALUE"
SET MEM @
```

The comment line will appear on the S: Line of the screen display.

It is not necessary to input a comment line, you may enter the prompted command line only, if you wish. However, if you wish to accompany a prompted command line with a comment, you must remember to enter the comment line command first in your command file string.

Use of the ESC key:

One of the benefits of using the prompted command is that the user can modify and also circumvent command file input as necessary. By pressing the ESC key (upper left-hand corner of the keyboard) as a response to the @ of the prompted command line, the command is bypassed and the command file proceeds to its next command.

Use of the Keyboard Bell by a Command File

To sound the keyboard bell during a particular moment in command file execution, insert a "control G" on a comment line, i.e., type an exclamation point then press the CTRL and G keys simultaneously.

For example, if you want the keyboard bell to ring following the comment lines given in the previous example, you would enter:

```
!THIS IS A COMMAND FILE COMMENT LINE
!A COMMAND FILE COMMENT LINE CANNOT EXCEED 60 CHARACTERS
!<ctrl>G
```

Changing the Command File Extension Default

The user has the available option to assign any desired three character command file extension default in lieu of the ECL-3211 system assumed command file extension of "COM".

The user-assigned command file extension will now apply to any command file command (CF command) or LOG command, if no extension has been designated.

To assign a new default command file extension, enter:

```
CF/D XXX
```

where XXX is the new default command file extension.

To View the Contents of a Command File while in ECL-3211

You may display the text of a designated command file in the central scroll area while operating in ECL-3211 by using the CF/T command. (Any ASCII file can be displayed using this command.) If the file has more than eight command lines (or records, if not a command file) the file will scroll through the display area. The NO SCROLL key may be used to control scrolling.

The CF/T command uses one command file nesting level since the command file from which you are operating is placed on the stack while the ECL-3211 is executing the CF/T command.

To display a command file, enter the following:

```
CF/T FILE.EXT
```

To Terminate Command File Execution

The CF/C command is used to terminate execution of all command files; i.e., the command file being executed is terminated and the command stack is cleared. The message CLEAR COMMAND FILE STATUS will appear on the S: Line of the ECL-3211 screen display.

To terminate execution of all command files, enter:

CF/C

from the ECL-3211 command line.

To terminate execution of an individual paused command file:

When the ECL-3211 is in the CF/P (command file pause) state, you can terminate the currently executing command file without affecting any command files present on the command file stack. To terminate a paused command file and remove it from the command file stack, press the ENTER key on the keypad on the right-hand side of your keyboard.

Calling a Command File as a Breakpoint Action

A user-created command file may be called to be executed as a breakpoint action when a given breakpoint condition or set of conditions is encountered.

In the previous command file example in which the command file TEST1.COM was created, Breakpoint 1 called a command file as a breakpoint action when the condition SW4 (switch 4 on) was encountered. The breakpoint command was as follows:

Break 0 CF/SW4

Notice that the specific command file to be called was not named in the breakpoint command. The command file name is assumed by the system when calling the command file from the breakpoint.

Breakpoint command files follow a strict naming rule which must be adhered to when building the breakpoint command file. When creating the name for a breakpoint command file, you must follow the format:

BRn.COM

where n is the number of the breakpoint (0- 7) and .COM is the command file extension. (If you have changed the command file extension default, use the new default extension when naming the breakpoint command file. Thus, you may maintain several command files for each breakpoint simply by changing the 3-character extension.)

When executing the CF breakpoint action command, the ECL-3211 will automatically look for the command file with the corresponding breakpoint number. If you have not properly identified the breakpoint command file, and you hit the breakpoint coming out of emulation, an error message will appear on the S: Line to inform you that the breakpoint command file was not located.

Create the breakpoint command file as you would any other ECL-3211 command file, using either an RT-11 edit function or the ECL-3211 LOG command.

To continue our previous example, the breakpoint command file BR1.COM might contain the following types of commands:

```
!SWITCH 4 IS ON, BREAKPOINT TRIGGERED  
CF/P  
CO 1=10 RI  
END
```

The message concerning switch 4 would appear on the S: line of the display. Then the command file would obey the next command and pause. At this point, a user could enter keyboard commands or could simply continue with the execution of the two remaining breakpoint commands by entering CF/R from the command line or by depressing the period (.) key on the FasKey keypad.

=====

GLOSSARY OF ECL-3211 COMMAND FILE MNEMONICS

=====

For diagrams of the command file FasKey menus for quick entry of ECL-3211 command file commands, refer to APPENDIX D at the end of this manual.

The following mnemonics are specifically designed for use as commands within and with ECL-3211 command files:

Mnemonic (manual entry)	Translation	FasKey Input (menu: key)
CF FILE.EXT	Execute a Command File FILE.EXT	FasKey 1: 5 CF/LOG: .
CF/P	Command File Pause	
CF/R	Command File Resume Execution	FasKey 1: 5 CF/LOG: 3
CF/T	Command File Type	FasKey 1: 5 CF/LOG: 1
CF/C	Command File Clear	FasKey 1: 5 CF/LOG: 0
CF/D	Command File Default	FasKey 1: 5 CF/LOG: 2
CF/D COM	Reset Command File Default to COM	FasKey 1: 5 CF/LOG: 9
!	Command File Comment Line	
!<cntrl>G	Ring Keyboard Bell	

The commands listed above may be used within a command file or directly from the keyboard.

Each of the CF/ commands are described below.

CF FILE.EXT	FasKey 1: 5 CF/LOG: .
-------------	--------------------------

The CF command when accompanied by the name and extension of a command file, executes the named command file. This command may be used directly from the ECL-3211 command line to execute a command file. It may be used as a command within a "parent" command file.

The CF/P command is inserted to cause a pause in the execution of the command file. During a command file pause, ECL-3211 keyboard commands may be entered allowing you to try new parameters, change memory values, create a separate command file through use of the log command, i.e., you may perform any legal ECL-3211 command. The interrupted command file will not resume until CF/R is entered from the keyboard.

While the command file is in a pause state, the statement CF/P FILE.EXT will display on the E:line of the ECL-3211 screen display (where FILE.EXT is the name of the command file in pause state). FasK: CF/P ,HL will display in the lower right-hand corner of the screen display.

During a command file pause, The minus, comma, ENTER and period keys on the keypad on the right-hand side of the keyboard have special command file pause functions. The hyphen key backspaces the command file to the last executed command and re-executes that command. The comma calls up a CF/P help file to instruct you in the use of the CF/P keypad keys. The ENTER key ends the paused command file (but does not clear the command file stack). The period keys issues a CF/R (resume command file) command.

CF/R

FasKeY 1: 5

CF/LOG: 3

The CF/R command is used to resume execution of a command file that has been in a pause state when any of four actions have caused the execution pause:

- . A CF/P has been executed from a running command file.
- . A keyboard interrupt (such as a carriage return) has been entered while emulation was in progress.
- . An error has been encountered by the system while executing a command file and the command file has paused and displayed CF/P.
- . Execution of the command file has been interrupted by entry of a double <CTRL>C .

CF/T FILE.EXT

FasKeY 1: 5

CF/LOG: 1

To display the text of a designated command file in the ECL-3211 central scroll area, enter:

CF/T FILE.EXT

Use the NO SCROLL key to control scrolling in displayed files with more than eight command lines.

CF/C

FasKey 1: 5
CF/LOG: 0

Terminates execution of all command files and clears the command file stack. The message CLEAR COMMAND FILE STATUS will appear on the S: line of the screen display.

CF/D

FasKey 1: 5
CF/LOG: 2

Is used to change the system default for command file extension. The ECL-3211 system-provided default for command file extensions is "COM". If you wish to alter this extension, enter:

CF/D YYY

where YYY is the new extension to which the system will default.

The user-assigned command file extension will now apply to any command file command (CF command) or LOG command, if no extension has been designated.

To return to a default extension of COM, you may use the FasKey 1 menu; enter FasKey 5 (CF/LOG) followed by Faskey 9 (CF/D COM).

Programmable "FasKey" Access to Command Files

Emulogic has provided special "FasKey" logic so that you may program special fasKeys in the keypad to access frequently used command files. Using the keys 0-9 of the keypad on the right-hand side of the keyboard, ten command files may be accessed by FasKey action for each command file extension you have assigned to command file names. The FasKey command file table will access the command files with the extension currently declared as the default. (See the CF/D command.)

A FasKey text file FKTXT.EXT must be created for each default command file extension. List the keys (0-9) and their corresponding command files, either by name or by providing a brief statement of purpose. One such file, FKTXT.COM has been provided as a sample.

To access a command file using the FasKey capability, the command file must be named using the following format:

FKEYn.EXT

where n is the number, 0-9, of the FasKey being coded to access the command file and EXT is the default command file extension currently recognized by the system.

To set up a command file for FasKey access, you may create the command file in RT-11 edit, by using the ECL-3211 LOG or LOG/C command or by re-naming an existing command file using the FasKey command file naming convention.

To create the FasKey command file using the ECL-3211 LOG/C command use the following format:

```
LOG/C FKEY(n).EXT<cr>
```

Next, enter the desired command file commands (each followed by a carriage return).

When all the desired commands have been entered, end the newly created FasKey command file with:

```
LOG END<cr>
```

For example, to create a FasKey command file to be accessed by the "3" FasKey, you would enter commands such as the following:

```
LOG/C FKEY3.COM<cr>
SET MEM 5FF=47<cr>
SW 1=1<cr>
SET PC=104<cr>
CF TEST1.COM<cr>
LOG END<cr>
```

By accessing the FasKey 1 menu, selecting item 5 (CF/LOG), next selecting the ENTER key and then keypad key 3, the command file FKEY3.COM will be executed.

To display the user-created FKTEXT.EXT menu of current FasKey accessible command files for the current default command file extension when in FasKey 1, depress the following keypad keys:

```
5 (CF/LOG)
```

```
ENTER (Keyboard=CF)
```

Command Files Used to Run Diagnostics:

The following file "IND.68" comes from our own Emulogic test library and is an example of a control file designed to be executed by the Indirect Control Processor (IND) under RT-11, Version 5. This particular control file was designed to run testing of an Emulogic ESP-68000 pod when the ECL-3211 system is working with from one to four additional Emulogic high speed memory boards. (We have provided program directives for the portion of the diagnostic testing for up to and including two high-speed memory boards.)

This type of operation can be adapted to run in non-IND environments on RT-11 Version 5 or RT-11 Version 4.

```

.START: .ENABLE QUIET
        COP SORT68.COM TS1.DIF
        DEL/NOQ *.DIF
.25:    .ASK    DTACK    IS TARGET DTACK (P1-10) GROUNDED
        .IFF    DTACK    .GOTO    25
        .ASK    LOOP     HALT AT END OF TEST
.50:    .ASKN   MEM      HOW MANY 32K H.S. MEMORY BOARDS PRESENT
        .IF     MEM = 0 .GOTO 100
        .IF     MEM = 1 .GOTO 200
        .IF     MEM = 2 .GOTO 300
        .IF     MEM = 3 .GOTO 400
        .IF     MEM = 4 .GOTO 500
        .IF     MEM > 4 .GOTO 50
.100:   L01200  SORT0.EX0,EX0.DAT
        DIFF/BINARY/OUTPUT:SORT0 SORT68.MEM,SORT0.DAT
        DIFF/BINARY/OUTPUT:SORTD SORT68.MEM,SORTD.DAT
        RUN TRPREP TIGO68.DAT
        DIFF/OUTPUT:TIGO68 TIGO68.MEM,TRTEMP.TMP
        RUN TRPREP PHANTOM.DAT
        DIFF/OUTPUT:PHANTM PHANTM.MEM,TRTEMP.TMP
        .GOSUB  TEST
        .IF <FILERR> EQ 1      .GOTO ERROR
        .IFF    LOOP         .GOTO 100
.DONE:  TYPE DONE.MSG
        .STOP
.ERROR: TYPE ERROR.MSG
        .STOP
.200:   L01200  SORT0.EX1,EX1.DAT
        DIFF/BINARY/OUTPUT:SORT0 SORT68.MEM,SORT0.DAT
        DIFF/BINARY/OUTPUT:SORT1 SORT68.MEM,SORT1.DAT
        DIFF/BINARY/OUTPUT:SORTD SORT68.MEM,SORTD.DAT
        DIFF/BINARY/OUTPUT:HSSHFL HSSHFL.MEM,HSSHFL.DAT
        DIFF/BINARY/OUTPUT:HSADRT HSADRT.MEM,HSDART.DAT
        DIFF/BINARY/OUTPUT:ROM1 68DAT1.DAT,ROM1.DAT
        DIFF/BINARY/OUTPUT:SAVE EX3.DAT,SAVE.DAT
        RUN TRPREP TIGO68.DAT
        DIFF/OUTPUT:TIGO68 TIGO68.MEM,TRTEMP.TMP
        RUN TRPREP PHANTM.DAT
        DIFF/OUTPUT:PHANTM PHANTM.MEM,TRTEMP.TMP
        .GOSUB  TEST
        .IFF    LOOP         .GOTO 200
        .GOTO   DONE
.300:   .....ETC.

.TEST:  .TESTFILE SORT0.DIFF
        .IF <FILERR> EQ 1      .GOTO ERROR
        .TESTFILE SORT1.DIFF
        .IF <FILERR> EQ 1      .GOTO ERROR
        .TESTFILE SORT2.DIFF
        .IF <FILERR> EQ 1      .GOTO ERROR
        .TESTFILE SORT3.DIFF

```

```

.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE SORT4.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE SORTD.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE BLKMAP.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE HSSMEM.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE HSSHFL.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE HSDART.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE TIGO68.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE PHANTM.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.TESTFILE WAIT.DIFF
.IF <FILERR> EQ 1      .GOTO ERROR
.RETURN

```

COMMAND FILE "ERROR.MSG"

ERROR HALT

COMMAND FILE "DONE.MSG"

END OF TEST
NO ERRORS

COMMAND FILE "SORT0.EX0"

```
!FILENAME: SORT0.EX0
!  
CLEAR  
FREQ=7500  
CF/D EX0  
DTACK TARGET  
WAIT OFF  
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0  
SET A0=0,A1=0,A2=0,A3=0,A4=0,A5=0,A6=0,PC=0  
BR 0-7 CLEAR  
SET A2=200,D1=80,US=100,SS=100,SR=0  
BR 1 PA,DECO1/PC=4  
BR 3 HL/CO1  
CF CO1.EX0  
CL 0-300  
LOAD BUBB68.LDA  
LOAD 68DAT1.DAT  
SW  
!RUNNING "SORT" IN MAP 0  
EM 0  
!DISPLAY AND SAVE SORTED DATA  
MEM 200  
SAVE EX0.DAT  
WRITE SORT0.DAT 200-2FF  
CF SORTD.EX0  
END
```

NOTE: 1)The changed default command file extension, "EX0".

2)Two ECL-3211 command files are executed from this command file: "CO1.EX0" and "SORTD.EX0". Command file "CO1.EX0" is a nested command file and causes the parent command file "SORT0.COM" to be placed on the command stack. When running command file SORTD.EX0, "SORT0.EX0" is not nested, since command file SORTD.EX0 is the last command before the END command. Using an execute command file as the last command in a command file command stream chains the command file from which you are exiting to the next command file.

COMMAND FILE "SORTD.EXO"

```

!FILENAME: SORTD.EXO
CLEAR
FREQ=1500
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0
SET A0=0,A1=0,A2=0,A3=0,A4=0,A5=0,A6=0
SET A2=200,D1=80,US=100,SS=100,SR=0
BR 4 CL
BR 5 HL/CO1
CF CO1.EXO
MAP INT
CLEAR 0-300
LOAD 68DAT1.DAT
LOAD BUBB68.LDA
!RUNNING "SORT" IN DEC INT.
SW
EM 0
!DISPLAY AND SAVE SORTED DATA
MEM 200
WRITE SORTD.DAT 200-2FF
CF TIGO.EXO
END

```

COMMAND FILE "TIGO.EXO"

```

!FILENAME: TIGO.EXO
!
CL
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0
SET A0=0,A1=0,A2=0,A3=0,A4=0,A5=0,A6=0,US=0
BR 0-7 CLEAR
MAP HS
FREQ=7000
SET SR=2700,SS=24A
CLEAR 0-200
LOAD TXTST.LDA
SET MEM 0=4EF8,20
BR 0 HL/ADDR=84
BR 3 ST
TR CLEAR
!EXECUTING "TIGO"
EM 0
EM 0
TRACE WRITE TIGO68.DAT 426-511
CF PHANTM.EXO
END

```

COMMAND FILE "PHANTOM.EXO"

```
!FILENAME: PHANTM.EXO
!  
CL  
MAP HS  
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0  
SET A0=0,A1=0,A2=0,A3=0,A4=0,A5=0,A6=0,US=0  
BR 0-7 CLEAR  
MAP 1000-17FF=0 RAM  
CLEAR 1000-10FE  
LOAD ADDQ.LDA  
SET MEM 1040=4E71,4E71,4E71,4EF8,1008  
BR 6 HL/ADDR=101E  
BR 4 JMP  
BR 4 PH=1040/PC=1006  
TR CLEAR  
SET SR=2700,SS=1070  
!EXECUTING PHANTOM JUMP  
EM 1000  
SET MEM 1046=5D97,4E75  
BR 0 CLSW1/PC=1000  
BR 1 STSW1/DATA=0C00,ADDR=106C  
BR 4 CALL  
!EXECUTING PHANTOM CALL  
EM  
TR WRITE PHANTM.DAT 444-511  
CF HSSHFL  
END
```

NOTE: the command file HSSHFL is called with the assumed default extension, ".EXO" established in command file "SORT0.EXO"

COMMAND FILE "HSSHFL.EXO"

```
!FILENAME: HSSHFL.EXO
!  
CL  
FR=7000  
MAP 0-7FF=1  
CL 00-FE  
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0  
SET A0=F0,A2=0,A3=0,A4=0,A5=0,A6=0,US=0  
SET A1=7FE  
SET PC=0,SS=0,SR=0  
BR 0-7 CLEAR  
BR 0 PA,ST,DECO1/ADDR=38  
BR 1 PA,ST,DECO1/ADDR=6A  
BR 6 CF/ADDR=A0  
BR 7 HL/CO1  
LOAD HSSHFL.LDA  
CF CO1.EXO  
!EXECUTING "HSSHFL"  
CL 700-7FE  
MEM 700  
EM 0  
CF HSADRT  
END
```

NOTE: Breakpoint 6 contains a command file as a breakpoint action. The name of the command file is BR6.EX0, assumed by the ECL-3211 breakpoint command file naming convention, since the command file extension default has been previously changed by command file "SORT0.EX0" to EX0.

COMMAND FILE "BR6.EX0"

```
!FILENAME: BR6.EX0
!  
!HIGH SPEED MEMORY ERROR HALT!  
WRITE HSSMEM.DIF 0-2E  
END
```

COMMAND FILE "HSADRT.EXO"

```
!FILENAME: HSADRT.EXO
!  
BR 0-7 CLEAR  
CL  
SET AD=F0,PC=0,SS=0,SR=0  
SET D0=0,D1=0,D2=0,D3=0,D4=0,D5=0,D6=0,D7=0  
SET A2=0,A3=0,A4=0,A5=0,A6=0,US=0  
SET A1=7FE  
BR 0 ST,PA/ADDR=34  
BR 1 DECO1,PA/ADDR=62  
BR 2 PA/ADDR=7C  
BR 3 PA/ADDR=C4  
BR 6 CF/ADDR=D0  
BR 7 HL/CO1  
LOAD HSADRT.LDA  
CF CO1.EXO  
!EXECUTING "HSADRT"  
CL 700-7FE  
MEM 700  
EM 0  
EXIT  
END
```

COMMAND FILE "CO1.EXO"

```
!FILENAME: CO1.EXO  
!  
CO 1=4  
CO 2=2  
END
```

ECL-3211 MDS COMMAND DICTIONARY

The Emulogic ECL-3211 Microprocessor Development System (MDS) is directed through a unique command language. You will use this language to map memory, load emulation programs and data, set breakpoints, and perform all other necessary functions related to the emulation process. This chapter describes the format and syntax of the commands and how the commands are used to carry out emulation procedures. Techniques of using the commands are the subject of other chapters of this guide.

COMMAND USAGE

ECL-3211 MDS commands can be used in two ways. You can enter commands directly as keyboard entries or indirectly as statements in command files. Either way, the command formats and syntax are identical.

Direct commands are commonly used for setting up the emulation environment, controlling the system display or adjusting emulation parameters. You make direct command entries by typing a command on the MDS display command line in the lower left corner of the screen.

Indirect commands perform similar functions, but they are executed automatically from within command files. These indirect command files are useful for performing repetitive or routine tasks and for set-up sequences. You can create command files using a text editor, or you can use the ECL-3211 LOG command to perform the file creation. Details of command file creation and use are provided in Chapter 6 - Creation and Use of Command Files.

COMMAND SYNTAX CONVENTIONS

In this chapter, commands are shown in the general format:

COMMAND(/option) argument 1(,argument 2,. . . ,argument n)

This structure is to be interpreted as follows:

COMMAND	the legal command abbreviation is used in all syntax examples. The full form of the command is shown in the section heading and is used in text to avoid any possible confusion. The portion of the command which is optional input is shown in lower case. Either the full form or the abbreviated form is acceptable in a command line.
---------	---

() (parentheses) parentheses are used to show optional elements of the command format. You can include or ignore these elements, depending on the function you want the command to perform. The parentheses merely signal an optional input choice; do not include them in your entries.

{ } (curly braces) curly braces are used to show a choice of options, keywords, or arguments available. Enter one and only one of these elements. If braces are enclosed in parentheses ({ }), any one of the choices will satisfy the option. The curly braces merely indicate that a choice can be made between mutually exclusive options; do not include them in your entries.

NOTE: The notation {+|-} means "plus or minus" and indicates that you should use one of these logical operators as a separator character in the argument list.

. . . elipsis -- three spaced dots -- indicates that multiple arguments are allowed for the command. You may enter arguments up to the allowed number. Use a comma (,) to separate each argument from the one preceding. Do not include the dots; they merely show you that more arguments can follow the first.

/ (slash) the slash mark indicates that the next character or character string represents a command switch. In some commands, several switches may be used together. When you use more than one switch in a command, place a slash (/) immediately before each switch code.

UPPER CASE UPPER CASE or capital letters are used to show the mandatory input portions of command words, switches and argument words as they appear in a correct command entry. Enter these elements exactly as shown when you use them in that command.

lower case lower case, or small letters, is used to show optional input, generic switches, and arguments for which you must substitute appropriate words, codes, or numeric values. You should refer to the description of these elements for the command you want to enter to determine the correct format of the substitution. For example, you might replace the term "addr" with the hexadecimal address value, 1A6F, to complete a MEM command.

Example: MEm addr
ME 1A6F

In this command dictionary, each command is accompanied by a description of the switches, options, and arguments allowed. In addition, the function of each allowable combination of the command syntax elements is provided.

Be sure to enter the command elements as shown in this dictionary, including spaces and commas where indicated.

Conclude every command by pressing the RETURN key to enter the command. For simplicity, these RETURNS are not shown in the format examples.

Any of the commands discussed in this chapter may be entered using the FasKey capability of the command mode. We suggest that you first familiarize yourself with the command syntax as demonstrated in this chapter. Refer to Appendix D at the back of this manual for the specific FasKey keypad configurations.

ECL-3211 COMMANDS

=====

HElp Abbreviation: HE

HELP
HE

Functions:

The HELP command presents general and specific displays on the ECL-3211 MDS screen designed to assist the user in the use of ECL-3211 commands.

Format:

HE({command word,special keyword})

Command Words:

BReak	EXIT	MOve	REset
CLear	FReq	SAve	TYpe
CF	HElp	SEt	WRite
COunt	LOad	STep	RUn (an RT-11 command)
DIIs	LOG	SWitch	
DUp	MAp	SYmbol	
EMulate	MEM	TRace	

Special Keywords:

ABort	BA (Breakpoint Action)	BT (Breakpoint Triggers)
ALter	BC (Breakpoint Conditions)	FAskey (keypad FasKey operation)
ERrors	BE (Breakpoint Externals)	
CHip	BM (Breakpoint Masks)	FKeys (keypad special command file option)

Display the Primary HELP File

To enter the Help mode and get general information on using this utility, enter:

HE

This command presents the general help mode display which shows all the system commands and special keywords. From this display, you can gain access to detailed information on any of the commands or keywords listed. To do this, simply type the command or keyword code for which you want information.

To exit from the help mode back to the ECL-3211 MDS command mode, press the RETURN key.

Display Information on a Specific Item

To look up information regarding one of the commands or special keywords without first entering the general help mode display, enter:

HE {command word or special keyword}

For example: HE BR will present a display of breakpoint information.
HE AB will present a display of information on how to abort a command.

To return to command mode, press the RETURN key.

PLEASE NOTE:

The help file (LOXX00.HLP) must reside on the pseudo device, "HLP:". To assign the help file to the pseudo device, use the RT-11 ASSIGN command. For example, enter:

ASSIGN SY: HLP: if the help file resides on your
booted device.

or enter:

ASSIGN DY1: HLP: if the help file resides on floppy
drive 1

BREAK

Abbreviation: BR

Functions:

The BREAK command displays, sets, clears, enables, disables, or modifies any one or more of the 8 ECL-3211 MDS breakpoints.

Format:

BR N(N-N) ({=addr} {action 1...,action n/condition 1...,condition n})

Where N equals breakpoint number 0-7. A range of breakpoint numbers may be provided instead of a single breakpoint number.

For example:

BR 2 HL/PC=143+C01

BR 3=1A6F

Keywords, options, and expressions:

- OFF the OFF option indicates that a specified breakpoint or range of breakpoints is to be disabled.
- ON the ON option indicates that a specified breakpoint or range of breakpoints is to be enabled.
- Clear the CLEAR option indicates that a breakpoint or range of breakpoints is to be cleared of all actions and conditions.
- addr an address (addr) is a unique location in system memory designated by a hexadecimal value representing an ECL-3211 mapped logical address.

Breakpoint Actions:

CF	CLSW4	CPSW4	DIC02	PA	STSW1
CLSW1	CPSW1	DEC01	ENC01	PH=addr	STSW2
CLSW2	CPSW2	DEC02	ENC02	RT	STSW3
CLSW3	CPSW3	DIC01	HL	ST	STSW4

One or more breakpoint actions may be specified, each separated by a comma from the preceding one.

Breakpoint Action Mnemonics are translated as follows:

Action Mnemonic	Explanation
=====	
CF	Command File: The CF argument, as a break-point action, halts emulation switching system control to the default breakpoint command file (BRN.COM where N is the number of the breakpoint). For information concerning creation of breakpoint command files, see Chapter 6 - Creation and Use of Command Files.
HL	Halt: Stop emulation, update the screen display and go to command mode.
PA	Pause: Stop emulation, update the screen display and resume emulation.
STSW1	Set switch 1: Set logical switch 1 (and thus set trigger 1).
STSW2	Set switch 2: Set logical switch 2 (and thus set trigger 2).
STSW3	Set switch 3: Set logical switch 3.
STSW4	Set switch 4: Set logical switch 4.
CLSW1	Clear switch 1: Reset logical switch 1 and thus reset trigger 1.
CLSW2	Clear switch 2: Reset logical switch 2 and thus reset trigger 2.
CLSW3	Clear switch 3: Reset logical switch 3.
CLSW4	Clear switch 4: Reset logical switch 4.
CPSW1	Complement switch 1: complement logical switch 1 and thus complement trigger 1.
CPSW2	Complement switch 2: complement logical switch 2 and thus complement trigger 2.
CPSW3	Complement switch 3: complement logical switch 3.
CPSW4	Complement switch 4: complement logical switch 4.

Action Mnemonic	Explanation
=====	
ENC01	Enable counter 1. Counter one is enabled for decrementing by cycles, instructions or target chip ticks.
ENC02	Enable counter 2. Counter two is enabled for decrementing by cycles, instructions or target chip ticks.
DIC01	Diasable counter 1.
DIC02	Diasable counter 2.
DEC01	Decrement counter 1.
DEC02	Decrement counter 2.
ST	Set the Trace.
RT	Reset the Trace.
PH=AAA	Jump to phantom program or call phantom program. (For information on types of phantoms supported by your microprocessor, refer to the appropriate, specific Chip Supplement to the USERS' GUIDE and to your chip specific HELP file provided in your ECL-3211 software.)

Breakpoint Conditions:

A variety of system, microprocessor, and target conditions can be specified depending on the model of microprocessor.

Refer to the HELP Utility or to the appropriate supplement to the ECL-3211 User's Guide.

Display Breakpoint Settings

To display the setings of a breakpoint, enter:

BR N

where N equals the number of the breakpoint (0-7)

The system presents a display showing all actions and conditions in effect for the specified breakpoint.

To Turn Breakpoints On and Off

To activate or deactivate one or more breakpoints, enter:

BR N ON
OFF

For example: BR 5-7 ON
BR 0 OFF

The system turns on or off the breakpoint or range of breakpoints specified. The new breakpoint status is shown in the breakpoint area in the upper right of the display.

Clear Breakpoint Settings

To clear the settings (actions and conditions) of a breakpoint or range of breakpoints, enter:

BReak N CLear

For example: BR 1-5 CL

The system removes all actions and conditions from the breakpoint or range of breakpoints specified. The new breakpoint status will now be shown in the breakpoint area of the display as undefined ("UNDEF").

Set Breakpoint on Address

To set a breakpoint to halt emulation at a predetermined address, enter:

BR N=addr

For example: BR 3=23FF

The system sets a breakpoint which is activated when the specified address is placed on the address bus for any reason. The implicit action of this breakpoint is a halt. The appropriate address word length is determined by the addressing scheme of the emulated micro-processor. The break-on-address breakpoint is the simplest form of address trap.

Set Breakpoint on Instruction Fetch

To set a breakpoint to halt emulation when an instruction is fetched from a particular address, use the syntax:

BR N/PC=addr

For example, the command BR 3/PC=23FF will set the breakpoint condition ADDR=23FF and INS (first opcode fetch) = 1.

Set Breakpoint Actions and Conditions

To set a breakpoint that will execute any of the actions listed previously for one or more conditions allowed by the microprocessor, enter:

```
BR N action 1(. . .,action n)/(-)condition 1(. . .{+|-}condition n)
```

The system sets a breakpoint which executes the specified actions if all of the conditions occur during emulation. Emulation will halt or pause only if the HL or PA actions are specified. Note that actions are always entered first, that each additional action is preceded by a comma, and that a slash mark (/) separates the conditions from the actions.

The plus (+) and minus (-) signs are logical operator/separators indicating AND and NOT, respectively for the condition statements. The minus operator cannot precede a multi-bit condition operator such as an address, a port, or data. Plus is the default for the first condition argument.

Remove Actions or Conditions

To remove actions or conditions from an existing breakpoint, enter:

```
BR N >action 1(. . .>action n)/>condition 1(. . .>condition n)
```

The "greater than" (>) symbol before the action or condition indicates that the parameter is to be deleted from the breakpoint specifications. One or more actions, one or more conditions, or any combination may be specified.

Examples: BR 2 >PA>CF/PC=456
BR 1 >/TC=32

Add Actions or Conditions

To add actions or conditions to an existing breakpoint, enter:

```
BR N action 1(. . .,action n)/condition 1(. . .{+|-}condition n)
```

The system adds the specified actions or conditions to the current breakpoint tables. This use of the BREAK command allows you to modify the entries in the action and condition tables. For example, if the program counter (PC) were previously set to activate breakpoint 5 at 2372, the command "BR 5/PC=2567" resets the condition of breakpoint 5 to trip when the PC has a value of 2567. One or more actions, conditions, or any combination may be specified.

CLEAR
CL

CLEAR

Abbreviation: CL

Function:

The CLEAR command, used without arguments, clears the central area of the display on the ECL-3211 MDS screen. Used with a range argument, the command causes the system to clear (set to zero values) the specified range of addresses in memory. If the system is in a memory display when you enter a clear memory command, the system updates the current display.

Format:

CL address1-address2

An address is designated as a hexadecimal value, whose maximum depends on the microprocessor being emulated.

CF (Command File)

Abbreviation: CF

Functions:

The CF command displays, clears, initiates, and directs execution of ECL-3211 MDS command files. A command file may be invoked from the command line of the system display, as an action in a breakpoint, or through a command in a command file (command files may be nested to a depth of 5 and may call themselves). When a CF command is the last command in a command file, no nesting occurs; the first command file simply exits into the next.

When used to invoke a command file from a breakpoint, the CF command does not use a filespec. Instead, the system invokes the command file whose name is that of the current breakpoint (that is, BR0 through BR7) and whose extension matches the current default. Thus, if a command file is invoked from breakpoint 4 and you have not altered the default extension, the system will attempt to locate a command file named "BR4.COM".

Format:

CF(/switch) (file.ext)

Switches and Arguments:

/D ext the D switch sets the default command file specification extension to the 3-character alphanumeric code you specify as "ext".

If you do not specify an extension, command files created with the LOG command will have the ECL-3211 system default ".COM" filespec extension. When a command file is called, the system searches for a file with the current default extension unless you explicitly provide some other extension with the CF command filespec.

/P the P switch can be used only in indirect commands; it causes a pause in the execution of command file statements. The system shows the condition "CF/P" on the execution line of the system display. You can now enter keyboard commands on the command line. You can create a pause in command file execution at any time from the keyboard by entering <CTRL/C> twice.

/R The R switch causes the system to resume execution of a paused command file.

CF
command file

(switches, continued)

/C	The C switch causes a termination of command file execution and <u>clearing</u> of the ECL-3211 MDS command file stack. The system presents the message "CLEAR COMMAND FILE STATUS" on the status line of the display.
/T FILE.EXT	The T switch causes the system to <u>type</u> (list) the contents of a command file in the central scroll area of the system display. If more than 8 command statement lines are contained in the file, the system will scroll them through this area. Use the NO SCROLL key to stop and examine the display.

COUNT

Abbreviation: CO

Function:

The COUNT command can be used to set the 2 ECL-3211 MDS counters. The count value is specified as a decimal number and is decremented by 1 on each signal from the count source. Three selectable sources cause regular automatic decrementing of the counter. The default counter source allows breakpoint actions to decrement a given counter, specifically.

Format:

CO N=value ({source}) (RI)

Options and Arguments:

N	one of the 2 ECL-3211 MDS decrementing counters is designated by a numeral "1" or "2".
value	a decimal number ranging from 0 to $2^{31}-1$ (2,147,483,647) representing the initial count value.
source	a 2-character code (shown elsewhere as "SS") represents the event or source of the count increment: CC selects machine (microprocessor) cycles. IN selects target (external) clock pulses. IS selects program instruction executions.

If you do not enter a source code, the value of the counter can be controlled exclusively through the "counter" breakpoint actions (in particular, DECO (decrement counter)). Under the conditions of this default, a specified counter is decremented with each execution of a DECO breakpoint action.

RI The Re-Initialize option restores the starting value of the counter.

DIS

Abbreviation: DI

Function:

The DIS (disassemble) command causes the system to convert hexadecimal data in memory to source-code format for examination. The resulting information can be displayed on the ECL-3211 screen, written to a file for later study, or printed on the system printer. When a starting address (addr) is provided, the disassembly begins at the specified address and continues for the next 8 instructions. If a range of addresses is provided, the disassembly includes all memory from the starting to ending address, inclusive.

Formats:

```
DIS {addr}{range}
DIS PRint range
DIS WRite filespec range
```

Arguments and Options:

addr	an address is a hexadecimal value indicating the starting point in memory of the disassembly. The system displays 8 sequential disassembled instructions beginning at the specified address. You can use the down arrow key to scroll forward to higher addresses.
range	a range of memory data can be specified by a beginning and ending address in the format: addr-addr If the range includes more than 8 instructions, the system automatically scrolls the instructions through the central area of the display. Use the NO SCROLL key to stop the scrolling and examine the display.
PRint	The PRINT option, abbreviated to PR, indicates that the specified range is to be printed on the system printer. This option requires an address range argument.
WRite	The WRITE option indicates that the specified range is to be written to the named RT-11 file. This option requires "filespec" and "range" arguments.
filespec	an RT-11 file specification is required to indicate the destination file of a DIS command with the WRITE option.

Display Disassembled Memory

To produce a display of disassembled memory, enter the command:

DIIs {addr}{range}

For example, DI 3000-3100

This command causes the system to show the specified memory in source-code format in the central area of the display. When an address argument is used, use the down arrow key to scroll past the first 8 instructions. If a range argument is used, use the NO SCROLL key to stop and examine the display.

Write Disassembled Records to a File

To write a block of disassembled memory to a file, enter:

DIIs WRite filespec range

For example, DI WR MFILE.MEM 2000-2100

This command causes the system to disassemble the specified range of memory and send the resulting information to a named file. The file specification may be any legal RT-11 filename and extension.

Print Disassembled Records

To print a block of disassembled memory on the system printer, enter:

DIIs PRint range

Example: DI PR 800-AFF

This command causes the system to disassemble the specified range of memory and to print the resulting information on a default printing device.

DUMP
DU

DUMP

Abbreviation: DU

Function:

This command causes the system to dump the specified range of memory to the system printer. The DUMP command operates through the ECL-3211 mapping facility, so the address specifications refer to logical addresses. To abort a listing from the DUMP command, press <CTRL/C> twice.

Format:

Dump range

Example: DU 800-AFF

Arguments:

range a range is specified a starting and ending address of the memory to be dumped, represented in the format:

addr-addr

Addresses are hexadecimal values.

EMULATEAbbreviation: EM

Format:

EM (addr)

Option:

addr the address option is a hexadecimal value representing an ECL-3211 MDS mapped logical address. This address is the location of the first emulation instruction to be executed.

Function:

This command starts the ECL-3211 MDS emulation routines. Instructions are fetched from the emulation test program mapped in memory and executed by the ESP processor. Signal tests and breakpoints are handled according to your pre-set directions entered during emulation set-up. If the optional address is not supplied, the EM command causes emulation to use the current value of the program counter (PC) as the starting instruction address.

EXITAbbreviation: EXIT

Function:

This command terminates the ECL-3211 monitor process and display and returns you to the primary operating system control, RT-11 monitor. The SAVE option (S) allow you to save the current emulation environment in a binary data file. The "save" file can then be used to set up the ECL-3211 MDS when it is started through the RUN command. Once you have issued the EXIT command, no other ECL-3211 commands are valid.

Format:

EXIT({/S}{S/filespec})

Options and Arguments:

/S	When the S option is specified, the system saves the emulation environment in a disk file. The default "save" file is named L0xx00.DAT; xx represents a 2-digit code for the microprocessor type.
filespec	You may use a standard RT-11 file specification to indicate a "save" file other than the default.

FREQ

Abbreviation: FR

Function:

This command sets the emulation frequency and specifies the clock signal source. The "type" options allow you to use the ECL-3211 MDS for ESP only or for ESP and target clock signals.

Format:

FR {frequency}{EXT} (type)

Options and Arguments:

frequency Frequency is designated by a decimal value representing in thousands of hertz (KHz) the frequency of the emulation clock.

type 2 types of clock signals can be generated by the ECL-3211 MDS:

- INT (Internal) The clock synthesizer of the ECL-3211 MDS provides a signal of the specified frequency to the Emulogic ESP pod only. All other target devices must use the target clock. The internal (INT) setting is the default clock source type.
- IND (Internal Driver) The clock synthesizer provides a signal of the specified frequency to the Emulogic ESP pod and to your target devices. This capability is not supported by all microprocessors; consult the microprocessor vendor's hardware specifications and the Emulogic chip supplement.
- EXT The EXT option shuts off the ECL-3211 MDS clock synthesizer signal to the ESP pod: the emulation microprocessor runs on a signal from the target clock.

LOAD
LO

LOAD

Abbreviation: LO

Function:

This command loads an assembled and linked program file in blocked absolute load module format from disk to memory, according to pre-determined mapping directions. This step makes the program code available for emulation.

Format:

LOad filespec

filespec a file specification designates a binary RT-11
LDA-format absolute load module as the file to be
loaded.

LOG

Abbreviation: LOG

Function:

This command opens the specified log file and writes subsequent keyboard commands into the designated log file. You can use the C switch to cancel immediate execution, thus creating a command file which can be executed on command. The END argument is used to mark the end of the logging function.

Formats:

LOG(/C) filespec
LOG END

Switches, Options, and Arguments:

/C	The C switch cancels immediate execution of commands as they are entered in "log" mode. By using this switch, you can create a command file which will not be executed until called by a direct or indirect CF command.
filespec	The file specification designates an RT-11 format file as the output log file. You can use any 3-character extension; if no extension is specified, the system assigns the current default.
END	the END argument designates the end of the log mode process. The "LOG END" command must conclude every log mode activity.

MAP

Abbreviation: MA

Functions:

The MAP command can initiate several map-related functions, depending on your selection of options and arguments. The Emulogic ECL-3211 MDS mapping facility allows you to locate emulation program modules, data files, and other related memory resources in suitably efficient areas of memory. The utility then automatically manages retrieval and storage of all emulation-associated memory requests.

Formats:

```
MAp n (option)
MAp range=n (option)
MAp range=offset (option)
```

Keywords, Options, and Arguments:

- | | |
|--------|--|
| n | The n option is a decimal value from 0 to 4 representing a high-speed map module of the ECL-3211 MDS. Map 0 is always the 2-kilobyte module on the MAP board. |
| range | The range argument consists of a starting and ending memory address in the format:

addr-addr |
| offset | The offset parameter is a hexadecimal value representing a physical range starting address in internal (DEC) memory. Each address is a hexadecimal value. In the case of internal memory, the range is added to an offset address to provide an effective address range. |
| option | You can select a variety of setting and control options for the MAP command: |
| ON | this option turns on the specified ECL-3211 MDS mapping function. If no map is specified, all previously declared maps are enabled. |
| OFF | this option turns off the specified ECL-3211 MDS mapping function. If no map is specified, all previously declared maps are disabled. |
| CL | the CLEAR option deletes the current declarations of the specified map. If no map is specified, all map declarations are deleted. |

(Map Options, continued)

- RAM this option sets the specified map to act as random access memory (RAM); this is the default condition.
- ROM this option sets the specified map to act as read-only memory (ROM) during emulation. Map 0 and internal (DEC) memory cannot be declared as ROM.
- INT this option accesses DEC system memory (internal) and allows you to modify the configuration of this memory area. Either 8 or 64 kilobytes of memory is available, depending on the system configuration.
- HS this option selects presentation of maps in the Emulogic high-speed memory modules. These modules include the 2 kilobytes on the MAP board plus any additional high-speed memory boards installed in the system. Each board is represented by a map number.

Display All Map Information

To review all current settings of ECL-3211 MDS maps, enter:

MAP

The system enters map mode and presents all map conditions at the top of the display. If you have more than 8 maps, use the up and down arrow keys to view all the map entries. You can make changes to map types (RAM, ROM), range, and status, or you can clear all settings by using additional MAP commands.

Display and Access Internal Mapping

To review or modify current settings of DEC (internal) mapped memory, enter:

MAP INT

The system enters internal map mode and presents the internal map

conditions at the top of the display. You can make changes to the range, offset, and status, or you can clear all settings using additional MAP commands.

Display and Access High-Speed Mapping

To review or modify current settings of the Emulogic high-speed mapped memory, enter:

MAp HS

The system enters high-speed map mode and presents the high-speed map conditions at the top of the display.

You can make changes to map type (RAM, ROM), range, and status, or you can clear all settings using additional map commands.

Change Map Status

To change the status (ON, OFF) of one or more maps, you can enter the MAP command with appropriate options. This command affects all internal, or high-speed maps, depending upon which map mode you have previously selected.

Change Status of All Maps

To switch all maps on or off, enter:

MAp ON
OFF

This command turns all current map declarations to ON or OFF, as specified. The status will be updated in the "Status" column of the map display.

Change Status of a Specified Map

To switch the status of an individual high-speed memory map on or off, in high-speed map mode enter:

MAp n ON
OFF

where n is the number (0-4) of the particular high-speed map.

This command turns the current map declarations to ON or OFF, as specified. Since only the Emulogic high-speed (HS) memory can have more than one map (one map per high-speed board), this command applies only to HS maps.

Clear Map Settings

To delete the current settings of one or more maps, you can enter the MAP command with the CLEAR option. This command affects all maps, internal or high-speed, depending upon which map mode was previously set.

Clear All Maps

To clear all current map conditions, from map mode enter:

MAp CLear

This command deletes all map declarations, switches status of all maps to OFF, and resets map types to RAM. Map ranges are set to undefined (000-000).

Clear a Specific Map

To delete the current map declarations of an individual high-speed (HS) map, in high-speed map mode enter:

MAp n CLear

where n is the particular number or numbers (n-n) of high speed map(s) you are clearing.

This command deletes the current declarations of the specified map, switches it off, and resets the map type to the default, RAM. Because only Emulogic high-speed memory can have more than one map, this command applies only to HS maps.

Set Map Range

To set the range (block of addresses) represented by a map, you can use the MAP command with a range argument. The range specification must be a form compatible with the internal or high-speed memory area being established.

Set Range for Internal Memory

To set a range of memory for the internal (DEC) map, in internal map mode enter:

MAp range=offset

Example: MA 000-5FF=000

(Set Range for Internal Memory, continued)

The system establishes the specified block in the internal map to begin at the given offset address. You must ensure that the range and offset you provide in this command do not, together, exceed the maximum address of available internal memory.

Set Range for High-Speed Memory

To set a range of memory for one of the high-speed memory maps, in high-speed memory mode enter:

MAp range=n (RAM)
(ROM)

Example: MA 100-5FF=1 ROM

The system establishes the range in the specified map. You can use the ROM or RAM options to change the memory type of the range being mapped. Remember that ranges in high-speed map 0 are RAM only.

MEM

Abbreviation: ME

Functions:

The MEM command causes the system to present the contents of memory on the central area of the display. The memory display is generated through the mapping facility, so you specify logical addresses or ranges. Memory data presented through the MEM command can be modified through the Alter mode.

Format:

```
MEm(/switch) (addr) (addr 1 . . . ,addr n)
                (range) (range 1 . . . ,range n)
```

Switches and Arguments:

/A The A switch causes the system to present the memory display in hexadecimal and ASCII equivalents.

/Q The Q switch provides a quick presentation of the memory range specified in the command.

NOTE: The Q and A switches can be used together, in either order, to provide hexadecimal and ASCII presentation in quick format.

addr An address is a hexadecimal value representing a starting location in memory to be displayed.

range The range is designated as a starting and ending address in the format:

 addr-addr

Recall Most Recent Memory Display

To recall the most recent memory display, enter:

MEm

The system presents the contents of 128 bytes (8-bit microprocessors) or 128 words (16-bit microprocessors) of memory most recently called with a MEM command in hexadecimal notation. If no previous MEM command has been issued, the system shows 128 bytes or words, beginning at address 000. You can use the up and down arrow keys to scroll through the displayed records.

Display Specified Memory

To view the contents of memory starting at a specified address, enter:

ME_m addr

The system presents a display of 128 bytes (8-bit processors) or 128 words (16-bit processors) of memory data starting at the specified address or the last even 16-byte or -word memory address (an address whose low order digit is 0). Use the up and down arrow keys to scroll through the displayed records.

Display Specified Memory with ASCII

To view the contents of memory starting at a specified memory address in hexadecimal and ASCII representation, enter:

ME_m/A(/Q) addr

The system presents 4 pairs of data-record lines. In each pair, the upper line is hexadecimal notation and the lower line, ASCII. 64 bytes of data is represented for 8-bit machines and 64 words for 16 bit microprocessors, starting at the specified or last lower even 16-byte or word boundary. The display is updated by each pause or halt in emulation. You can use the Alter mode and SET command to modify values which will, in turn, change the ASCII equivalent. The optional Q switch produces a memory display in quick format (see "Display Memory in Quick Format").

Display Memory in Quick Format

To view the contents of memory at a specific address or range of addresses, enter:

ME_m/Q(/A) addr 1 . . . ,addr n
 range 1 ,range n

The system presents the data records from the specified address or range in hexadecimal representation. Any combination of 3 ranges, addresses, or ranges and addresses can be specified in one command. You should note, however, that if the ranges or addresses exceed the 8 records available in the display's central area, the system scrolls all records through to the last. Since the Q switch defeats the use of up and down arrow keys, you must ensure that ranges specified can be viewed without scrolling. The optional A switch produces a quick memory display with ASCII equivalents of each byte or word displayed. You can change data displayed with the MEM/Q command with the SET command.

MOVE

Abbreviation: MO

Function:

This command causes the system to move the block of memory designated by the range to a new location starting at the address specified in the command. Because the MOVE command makes use of the ECL-3211 mapping facility, all addresses refer to logical memory locations. However, the data in the specified range is physically re-written. If the area specified by "addr" (that is, the new location) is not large enough to hold the entire block, the system moves as much of the block as available space will allow.

Format:

MOve {range}{addr}

Arguments:

range The range represents a block of memory addresses between a starting and ending address, inclusive, in the format:

addr-addr

addr Addresses are hexadecimal values representing locations (logical) in mapped memory.

RESET
RE

RESET

Abbreviation: RE

Function:

The system resets the ECL-3211 electronics modules. This resetting function affects logic on the Emulogic Map and Control boards and in the ESP pod. Be aware, however, that this command does not provide a reset signal for your target hardware.

Format:

REset

SAVE

Abbreviation: SA

Function:

The system saves the emulation status data in the designated save file.

Format:

SAve (filespec)

Options:

filespec A legal RT-11 file specification represents the file in which the current ECL-3211 MDS emulation environment is to be stored. The default file extension is ".DAT".

SET

Abreviation: SE

Function:

The SET command is used to set values in memory or to set values for the target chip's logical entities (registers, program counters, etc.)

Refer to your chip-specific Emulogic supplement to the User's Guide for the proper use of chip mnemonics for this command.

Format:

SEt ({chip mnemonic option}{MEM address=value{value1,value2,...}})

Options:

MEM

The MEM option allows the user to set a specific value for a given address or optionally successive addresses.

For example, SET MEM 400=123,124,125 sets the contents of memory at three contiguous addresses beginning at address 400 at the values 123, 124, and 125.

Chip Mnemonic

Sets the hexadecimal value for one of the target's logical entities (registers, program counters, etc.) Mnemonics vary according to the chip being emulated.

For example, SET PC=44,SP=3 .

STEP

Abbreviation: ST

Function:

The system initializes the step counter at 1, commences step mode emulation, executes one instruction, and pauses. Press the RETURN key (a "null" command) to advance to the next step. You cannot use the STEP command to execute instructions within phantom programs or interrupts.

Format:

STep (B)

Options:

B The B option specifies that steps are from breakpoint to breakpoint. The system proceeds with emulation until the conditions are met of a breakpoint containing a HALT, PAUSE, or CF (command file) action.

Any command other than RETURN terminates the step mode emulation and places the system in command mode.

If you include the B option in this command, the system executes program instructions without pausing until the conditions of a breakpoint containing a HALT, PAUSE, or CF are met.

SWITCH

Abbreviation: SW

Functions:

The SWITCH command allows display or manual setting of the 4 ECL-3211 MDS sense switches.

Format:

Switch n=condition(,n=condition...)

For example, SW 1=0,2=1,3=0,4=1

Arguments:

n	The 4 ECL-3211 switches are designated as one-digit numerals: 1, 2, 3, and 4.
condition	The logical conditions, "on" and "off", are represented by the binary values 1 and 0, respectively.

Display Current Switch Settings

To view the settings of the switches, enter the command:

SWitch

The system presents a display of the current switch and counter conditions.

Set Switch Conditions

To set the value of a switch, enter the command:

SW n=condition(,n=condition....)

Example: SW 1=0,4=1

The system sets the logical condition specified on one of the four system switches. Since switches 1 and 2 provide output signals to the ESP pod triggers (TRIG-1, TRIG-2) respectively, you can use the SWITCH command to pulse the trigger lines.

SYMBOL

Abbreviation: SY

Functions:

The SYMBOL command can be used to set and clear symbols from the ECL-3211 MDS internal symbol table or to open program symbol table files that have been linked with the Emulogic Linker. Symbols can be used with offset hexadecimal values and logical operators; the format "symbol+7" or "symbol-2B" can be used as addresses or other numeric values.

Format:

```
SYmbol (CLeAr) (symbol 1(,symbol 2,. . .,symbol n))  
          (symbol=value) (filespec)
```

Options and Arguments:

CLear	The CLEAR option indicates that one or more symbols are to be deleted from the internal (ECL-3211 MDS) symbol table. When not accompanied by symbol arguments, the CLEAR option clears all internal symbols and closes any open symbol files.
symbol	A symbol represents a named variable. Once such a symbol is established, the symbol name can be referenced in any command where a hexadecimal value is normally acceptable. A symbol name must be enclosed in double quotation marks (as: "symbol") when making a substitution.
value	a hexadecimal value is assigned to a symbol created through the SYMBOL command.

Add a Symbol to the Internal Table

To add a symbol or change the value of an existing symbol in the internal table, enter:

```
SY symbol=value
```

Example: SY FISH=0F1F

The system adds the named symbol with the value specified or changes the value of the existing symbol. You may now substitute the symbol name, in double quotation marks (" "), in your program as a reference to the given value.

Open a Symbol Table File

If you have symbol table files created through the Emulogic Linker and want to use those symbols in ELC-3211 MDS commands, enter:

SYmbol filespec

The system opens the specified file and makes the table available for command symbol references. You may keep up to 4 symbol table files open simultaneously.

Clear Internal Tables

To delete symbols from the internal ECL-3211 MDS symbol table, enter:

SYmbol CLear (symbol 1(. . .,symbol n))

Example: SY CL FISH,KING,QUEEN,JACK,ACE

The system clears one or more symbols, as specified, from the internal table. If no symbol names are given, the system clears all symbols in the internal table and closes any open symbol table files.

TRACE

Abbreviation: TR

Functions:

The TRACE command controls your use of and access to the ECL-3211 MDS trace buffer. This buffer holds the last 512 cycles executed by the emulation microprocessor. Records in the trace buffer can be viewed on the system display, written to a file, or printed on a hardcopy device.

Formats:

```
TRace {record}{range}
TRace CLear
TRace WRite filespec (range)
TRace PRint (range)
```

Options and Arguments:

record The ECL-3211 MDS maintains records of cycles executed by the microprocessor in the trace buffer. The buffer holds 512 records, designated by decimal values from 0 to 511. (Record = 0-511.)

range a range of records is designated by a starting and ending record number in the format:

record-record

CL The CLEAR option causes the system to delete all records from the trace buffer.

WR The WRITE option causes the system to write the entire trace buffer or a range of records to a named RT-11 file.

filespec An RT-11 file specification indicates the output destination of a TRACE command with the WRITE option.

PR The PRINT option causes the system to print the entire trace buffer or a range of records on the system printing device.

Display the contents of the Trace Buffer

To view the current contents of the trace buffer, enter:

TRace

The system presents a display of records 504 to 511 (the last 8) in the trace buffer in the central area of the system display. You can use the up and down arrow keys to scroll backward or forward through the trace record display.

Display Selected Trace Records

To view selected records from the trace buffer, enter:

TR {record}{range}

The system presents the selected records in the central area of the system display. If you specify a single record, the system shows 8 records beginning with the record specified. If you specify a range of records, the system scrolls the records beginning at the start-of-range. Use the NO SCROLL key to halt the scrolling and examine records.

Clear the Trace Buffer

To clear all records from the trace buffer, enter:

TR CL

This command deletes all current instructions from the trace buffer.

Write Trace Records to a System File

To copy the entire trace buffer or a range of trace records into an RT-11 file, enter:

TR WR filespec (range)

The system copies the specified records into the file designated by "filespec". If no range is given, the system writes out all of the records in the trace buffer.

Print Trace Records

To print the entire trace buffer or a range of trace records, enter:

TR PR (range)

The system prints the specified records on the system printing device. If no range is given, the system prints all 512 records from the buffer.

TYPE

Abbreviation: TY

Function:

This command sets the ECL-3211 MDS for operation with the specified microprocessor type.

Format:

TY device type

Argument:

device type a device type specifies the model of microprocessor used for system development. The Emulogic ECL-3211 MDS recognizes most popular microprocessor types by their generic numeric codes. For example, the Intel 8048 is device type "8048"

WRITE

Abbreviation: WR

Function:

This command writes the specified block or blocks of data to the named file in LDA format. This file can later be reloaded using the ECL-3211 MDS LOAD command. Since the WRITE command uses the mapping facility, you need specify only logical addresses in range arguments.

Format:

WR filespec range 1(. . .,range n)

Arguments:

filespec The RT-11 file specification designates an output file to receive the data indicated in the command.

range A range of memory addresses to be written out by the command is represented by a starting and ending address in the format:

addr-addr

The address is a hexadecimal value whose maximum depends on the microprocessor being emulated. One or more ranges (separated by commas) may be provided in one command.

This chapter may be used as a quick reference to the basic commands that are available in the ECL-3211 microprocessor development system. We have summarized the commands, categorizing them into four functional areas:

- o Memory Management
- o Emulation Management
- o Screen Management
- o Command File Management

Those few commands that do not fall under these four functional areas are listed under "Miscellaneous Commands".

NOTE: Some ECL-3211 commands are capable of performing more than one function. This chapter gives only a cursory definition of the commands. For a full scope of each command's capabilities, refer to Chapter 7 "Command Dictionary".

In the command tables that follow, the capitalized portion of the command mnemonic is required input. The lower case portion of the mnemonic is optional input. After typing a command on the command line of the display, press the RETURN key (<cr>).

Any of the commands listed in the tables may be entered using the FasKey capability of the command mode. Refer to Appendix D at the back of this manual for specific FasKey keypad configurations.

TABLE 8.1 below lists the ECL-3211 commands used to manipulate ECL-3211 and target memory.

TABLE 8.1
MEMORY MANAGEMENT COMMANDS

COMMAND	FUNCTION
CLear	Clears a specified range of memory, either a specific address range or a named map. (See also Screen Management.)

TABLE 8.1 (cont.)
MEMORY MANAGEMENT COMMANDS

COMMAND	FUNCTION
DIs	Memory contents are disassembled as instructions, starting at given address DIs AAAA or DIs AAAA-BBBB DIs PRint YYYY-ZZZZ prints disassembled contents of memory (as instructions) in given address range. DIs WRite FILE.EXT YYYY-ZZZZ Disassembled memory from address YYYY-ZZZZ (HEX) inclusive is written to file FILE.EXT
DUmp	Dumps the specified portions of memory (in HEX) to the system printer. Use the following command format: DUmp AAAA-BBBB
LOad	Loads LDA-formatted files into memory from disk. LOad FILE.EXT
MAp	Assigns a range of logical addresses to a corresponding range of physical addresses in ECL-3211 internal memory or high-speed memory. There are twelve available map functions: MAp displays map assignments MAp INT maps internal DEC memory MAp HS maps high speed memory MAp AAAA-BBBB=OFFSET maps internal memory for the given address range beginning at the given OFFSET location into internal memory MAp AAAA-BBBB=N (RAM or ROM) maps high speed map N for the given address range (as either RAM or simulated ROM - HS Map 0 cannot be declared as ROM)

TABLE 8.1 (cont.)
MEMORY MANAGEMENT COMMANDS

COMMAND	FUNCTION
	Map ON or Map OFF Turns on or off all maps Map Clear Clears all map definitions Map N ON Turns on high speed map N Map N OFF Turns off high speed map N Map N Clear Clears high speed map N Map N RAM or ROM Declares map N to be RAM or simulated ROM (HS Map 0 and DEC Internal memory cannot be declared as ROM)
ME m	Displays memory contents in the central scroll area of the screen display. The following displays of memory are available: ME m AAAA (AAAA=beginning location) displays memory beginning at location AAAA in HEX. The optional switch "/A" causes memory to be displayed as ASCII format as well as in hexadecimal format. ME m Last Mem display is redisplayed or if no previous display, defaults to 128 Byte/words beginning at address 0. ME m/Q(/A) AAAA-BBBB(,CCCC-DDDD,EEEE-FFFF) Quick display of memory in given memory ranges (maximum of three ranges.) The optional switch "/A" causes memory to be displayed in ASCII as well as in hexadecimal format.

TABLE 8.1 (cont.)
MEMORY MANAGEMENT COMMANDS

COMMAND	FUNCTION
MOve	The block of memory encompassed by the given address range is moved to a new stated address. MOve 30-4F 100 where 30-4F is the address range being moved and 100 is the starting address for the desired new location.
SEt	Sets contents of specified memory locations. SEt MEM 0-32F=12 Sets the constant value 12 into memory locations 0 to 32F. SEt MEM 0=C3,58,00 Stores the hex values, C3, 58, and 00 in sequential addresses beginning at address 0.
WRite	Stores contents of a range of memory into a diskfile in LDA format. WRite FILE.EXT AAAA-BBBB(,CCCC-DDDD,EEEE-FFFF,....) (See emulator L0ad command to load file)

EMULATION MANAGEMENT

There are several basic commands that you use to control emulation of your target system. These commands are listed below in TABLE 8.2 .

TABLE 8.2
EMULATION MANAGEMENT COMMANDS

COMMAND	FUNCTION
BReak	Defines a breakpoint and displays its conditions and actions. There are nine breakpoint functions: BReak N displays setting for breakpoint N, where N = 0-7 BReak N ON (OFF) Turns a given breakpoint on or off BReak N-N ON (OFF) Turns on or off breakpoints N-N ex: BR 1-4 ON BReak N(-N) CLeaR Clears a particular breakpoint or, optionally, a range of breakpoints BReak N=AAAA (sets halt at address AAAA) BReak N A1(,A2,A3,A4)/C1(+C2,-C3+C4...) sets actions and conditions for breakpoint n (See Chapter 4 for a complete discussion of breakpoint syntax including action and condition mnemonics.) BReak N >A1/>C2 remove action A1 and remove condition C2 from breakpoint N parameters BReak N A4,A5 add breakpoint actions A4 and A5 to breakpoint N BReak N/C5,C6 add conditions C5 and C6 to breakpoint N

TABLE 8.2(cont.)
EMULATION MANAGEMENT COMMANDS

COMMAND	FUNCTION
COunt	Sets numerical value and type of operation of one of the system's counters. COunt N=DDDDDDDD (XX) (RI) where N = counter number, 1 or 2 DDDDDDDD = decimal number from 0 to 2,147,483,647 ($2^{31}-1$) XX may equal one of the following three count source mnemonics: CC (counts machine cycles) IN (counts target clock ticks) IS (counts instructions) RI resets the counter to initial value at beginning of each emulation
EMulate	EMulate (XXXX) Starts emulation at current program counter address or, optionally at user-specified address XXXX.
FReq	Selects emulator clock source and frequency There are two FReq functions: FReq DDDDD (TTT) where DDDDD is a decimal number optional TTT= INT for internal clock synthesizer = IND for ECL-3211 clock source Note: IND is not available on all systems. Check your specific chip supplement to the ECL-3211 User's Guide. FReq EXT (external target clock source)
REset	Performs hardware reset for ECL-3211 pod to initialize logic in pod chip. (Does not perform a reset to the target system.)
SEt	Sets hex value for one of the ESP pod's logical entities (registers, program counters, etc.) Entity codes are specific to the emulated microprocessor.

TABLE 8.2 (cont.)
EMULATION MANAGEMENT COMMANDS

COMMAND	FUNCTION
STep	Steps through emulation one instruction at a time. Each press of the RETURN key advances emulation through the next step.
STep B	Emulation begins and continues until a breakpoint with a HALT, PAUSE, or COMMAND FILE action is encountered. A carriage return causes the system to progress to the next halting breakpoint. Any command other than a carriage return exits step mode.
SWitch	Sets a system switch There are two SWitch command functions: SWitch switch and counters display is shown in central scroll area SWitch N=1 or 0 where N can be switch 1-4
TRace	Displays, prints or clears trace buffer, or writes buffer contents to a disk file. The following TRace functions are available: TRace trace records 504-511 display in scroll area TRace AAA or (AAA-BBB) first 8 trace records beginning at record AAA (decimal) are displayed in the central area or, optionally, a range of records beginning at record AAA and ending with record BBB (decimal) are displayed in the central area. TRace CLEAR clears trace buffer TRace WRITE FILE.EXT (AAA-BBB) Writes contents of trace buffer record (512 records) to designated file. If optional range (AAA-BBB, decimal) is given, only the specified records are written.

TABLE 8.2 (cont.)
EMULATION MANAGEMENT COMMANDS

COMMAND	FUNCTION
TRace (cont.)	TRace PRint (AAA-BBB) Contents of trace buffer (512 records) are printed. If range of record numbers is given (AAA-BBB, decimal values), only the contents of the trace buffer within the record range will be printed.
TYPe	TY=nnnn<cr> Specifies the type of processor being emulated. This command is used when several devices in a micro-processor family use the same emulation software. Refer to the supplement to this Users's Guide for the processor you are emulating. ex: TY=8049<cr> This sets the device to an 8049 chip, which is a member of the 8048 family.

SCREEN MANAGEMENT COMMANDS

You can display a variety of information relevant to target emulation in the central scroll area of the screen display, using the commands listed below in TABLE 8.3.

TABLE 8.3
SCREEN MANAGEMENT COMMANDS

COMMAND	FUNCTION
BReak N	Displays the breakpoint definition for a given breakpoint N, where N may equal 0-7.
CLear	Clears the central scroll area.
MEm	Displays memory contents in the central scroll area. The last memory display is redisplayed or, if there was no previously requested memory display, defaults to displaying 128 byte/words beginning at address 0. (See Memory Management Commands in TABLE 8.1 or refer to CHAPTER 4 "Memory Management" for more detailed information on memory display commands.)

TABLE 8.3 (cont.)
SCREEN MANAGEMENT COMMANDS

COMMAND	FUNCTION
SWitch	Displays the state of logic switches, counters and clock source in the central scroll area.
TRace	Displays trace records 504-511 in the central scroll area. (See TABLE 8.2 for a complete listing of trace commands.)
DIIs	Displays disassembled memory contents in the central scroll area. (See TABLE 8.1 for a complete list of DIS commands.)
CF/T	This command types the contents of a specified command file in the central scroll area. If the file has more than eight records, the file scrolls.
	Use the following command format:
	CF/T FILE.EXT

=====

COMMAND FILE MANAGEMENT

The following commands listed in TABLE 8.4 below are used in the creation and control of ECL-3211 command files. For more information on use of command files in the ECL-3211 system, refer to the chapter entitled CREATION AND USE OF COMMAND FILES in this manual.

TABLE 8.4
COMMAND FILE MANAGEMENT COMMANDS

COMMAND	FUNCTION
CF	Executes a command sequence contained in the specified command file.
	Use the following command format:
	CF FILE.EXT
	where FILE is the command file name and .EXT is the command file extension.

TABLE 8.4 (cont.)
COMMAND FILE MANAGEMENT COMMANDS

COMMAND	FUNCTION
CF/D	Used to create a user-defined default for the command file extension. (System defined default is ".COM".) CF/D XXX where XXX represents any three character alphanumeric which is be the new command file extension default. This user-defined default will also automatically apply to LOG created command files .
CF/P	Causes a command file pause; can be used only within a command file. Use the CF/R command to resume command file execution.
CF/R	Must be entered from the display command line following a command file pause in order to resume execution of the command file.
CF/C	Terminates execution of all command files and clears the command stack.
CF/T	Types the contents of a named command file in the central scroll area. Use the following command format: CF/T FILE.EXT
END	Functions as a command sequence terminator in a command file.

TABLE 8.4 (cont.)
COMMAND FILE MANAGEMENT COMMANDS

COMMAND	FUNCTION
LOG	The LOG command allows you to keep track of all commands entered from the keyboard at the command line. All commands are executed and written to the user designated FILE(.COM). (See CF/D to change command file extension.) To end the LOG session, enter LOG END, EXIT or EXIT/S at the keyboard. An END command is written to the LOG file, the LOG file is closed and now available for execution as a command file.
LOG/C	The /C switch used with the LOG command suppresses execution of all commands entered from the command line. This command is especially useful for creating an ECL-3211 command file. LOG/C FILE(.EXT) System default for .EXT is .COM
!	To include comments in a LOG file or a command file (the comment will display during execution of the command file), precede the comment with an ! (exclamation mark). Each comment line may be 60 characters in length. !xx Where xxx... is the comment line.
!<CTRL/G>	This command will cause the keyboard bell to ring as a signal. Enter a "!" followed by a <CTRL/G> (depress the CONTROL and G keys simultaneously.)

MISCELLANEOUS COMMANDS

The commands listed in TABLE 8.5 perform miscellaneous ECL-3211 functions.

TABLE 8.5
MISCELLANEOUS COMMANDS

COMMAND	FUNCTION		
EXIT	Returns the ECL-3211 system to the RT-11 keyboard monitor.		
EXIT/S	The S switch used with the EXIT command performs an exit and saves the environment (see SAVE command below). If no data file is specified, the data is stored in a default data file - L0xx00.DAT (xx is an Emulogic code specifying the processor being used). Use the following command format: EXIT/S (FILE.EXT)		
Save	The current status of the emulator (that is, emulation parameters) is saved in the designated file. The default extension is "DAT". The status of the emulator can be saved at any point by the SAVE command. This saved status may then be restored at a subsequent emulation run time. Use the following command format to store the data into a non-default file : Save filespec		
HElp	Provides detailed information about the ECL-3211 system commands and features.		
SYmbol	This command is used for the creation and handling of designated symbols or symbol-table files. When using a symbol, it must be enclosed in double quotes " ". ex: MEM "TAG" The following SYMBOL functions are available: <table><tr><td>SY NAME=HHHH</td><td>The user-defined symbol NAME is loaded into the internal symbol table and assigned the hexadecimal number HHHH. The symbol may now be referred to as "NAME".</td></tr></table>	SY NAME=HHHH	The user-defined symbol NAME is loaded into the internal symbol table and assigned the hexadecimal number HHHH. The symbol may now be referred to as "NAME".
SY NAME=HHHH	The user-defined symbol NAME is loaded into the internal symbol table and assigned the hexadecimal number HHHH. The symbol may now be referred to as "NAME".		

TABLE 8.5 (cont.)
MISCELLANEOUS COMMANDS

COMMAND	FUNCTION
SYmbol (cont.)	SYmbol FILE.EXT The symbol table file, FILE.EXT is opened. With any symbolic reference, the system searches through FILE.EXT. A limit of four symbol table files may be opened for access simultaneously. SYmbol CLEAR Clears the internal symbol table and closes all open symbol table files. SYmbol CLEAR AAAA,BBBB Delete symbols AAAA and BBBB from the internal symbol table.

APPENDICES

APPENDIX A

ECL-3211 ERROR AND SYSTEM MESSAGES

This appendix lists messages you may encounter using the ECL-3211 Microprocessor Development System (MDS) emulation software. Along with the description of the message is a suggested action to correct the condition. In the case of system messages, the system function corresponding to the message statement is described.

In most cases, errors in command strings entered from the command line of the display are flagged by ECL-3211 in the following manner:

The entire command entry appears on the status line of the screen. The displayed string is underlined up to the point where the error in syntax occurred.

To Correct the Error:

Re-enter the command string using the correct syntax and legal references. Refer to the system HELP file, ECL-3211 Users' Guide, or an appropriate supplement to ensure proper command entries.

```
=====
ECL-3211  ERROR MESSAGES
=====
```

CHECKSUM ERROR-BLOCK NUMBER n

(Not currently used by ECL-3211 run-time.)

COMMAND FILE NOT FOUND filespec

The file referenced either at start-up or by a CF command, is not present on the disk.

COMMAND FILE NESTING TOO DEEP

The nesting limit has been overextended. (The nesting limit is 5.) The command stack has been cleared.

DEVICE FULL - WRITE ABORTED

The disk did not have enough contiguous space to write a file from a trace write or disassembly write command. The file has not been deleted. Squeeze the disk or insert a blank disk into the device to which you are writing.

DEVICE FULL - WRITE ABORTED AND FILE DELETED filespec

The disk did not have enough contiguous space to receive a file from a WRITE command. Squeeze the disk or insert a blank disk into the drive to which you are writing.

ERROR CLOSING FILE filespec

An I/O error occurred while attempting to close a file on the disk. This may be caused by a media fault or hard I/O error.

ERROR OPENING FILE filespec

The system was unable to open the requested file and returned to the command level. Check that you have typed the correct file name and mounted the volume on which the file is stored.

ERROR OPENING HELP FILE

The system was unable to open the help text file LOXX00.HLP and returned to command level. Use the RT-11 SHOW command to check that the logical device, "HLP:", has been assigned a physical device name.

ERROR READING FILE filespec

An I/O error occurred while attempting to read a specified file from the disk. This may be caused by a media fault or hard I/O device error.

HELP-NO SUCH KEYWORD

This HELP command error message indicates that the requested keyword is not available for this emulator.

INVALID COMMAND

The command word was not recognized as a valid command for the emulator.

NON-ASCII FILE filespec

An attempt was made to type a non-ASCII file via the CF/T command. This is not a fatal error. The CF/T command types only ASCII formatted files.

SYMBOL NOT FOUND

The symbol referred to in a symbolic reference was not found in the internal symbol table or in the symbol file.

SYNC ERROR

The blocked absolute loader did not find a block header where one was expected. This error may be caused by attempting to load a file not in DEC LDA format or by a data transmission error.

SYNTAX ERROR

The word following HELP in the command string was meaningless. This error is frequently typographical.

TOO MANY FILES

This message indicates that the symbol command has been used in an attempt to open more than the four permitted symbol table files.

=====

ECL-3211 SYSTEM MESSAGES:

=====

BACKSPACE COMMAND FILE

The paused command file has been stepped backwards one or more commands by pressing the minus (-) key on the FasKey keypad.

- BACKSPACE . RESUME E END

This message is displayed if you press the comma (,) key on the FasKey keypad while a command file is paused. It denotes that the minus key has the backspace function, the period (.) key has the resume execution function, and the ENTER key will terminate execution of the command file.

CLEAR REGISTER SCREEN DISPLAY

All register settings have been cleared via the CLEAR OPTIONS submenu of the FasKey 1 menu.

DISASSEMBLY ABORTED

The system has received an instruction (two presses of <CTRL/C>) to abort the disassembly function in progress.

PAUSED COMMAND FILE TERMINATED

Execution of paused command file has ceased. This message usually indicates a deliberate or accidental pressing of the ENTER key on the FasKey keypad.

TRACE ABORTED

The system has received an instruction (two pressed of <CTRL/C>) to abort the trace function in progress.

APPENDIX B

DIGITAL EQUIPMENT CORPORATION

SYSTEM SOFTWARE MANUALS

The following manuals provide in-depth support for Digital Equipment Corporation software and hardware that is used with the ECL-3211 microprocessor development system.

RT-11 Documentation Directory

Introduction to RT-11

RT-11 Installation and System Generation Guide

RT-11 Software Support Manual

System Message Manual

RT-11 Master Index

RT-11 Programmer's Reference Manual

RT-11 System User's Guide

RT-11 System Release Notes

RT-11 Software Dispatch Review

PDP-11 MACRO-11 Language Reference Manual

APPENDIX C

EMULOGIC MICROPROCESSOR EMULATION PROGRAMS

The following table lists the microprocessors supported by Emulogic and the corresponding ECL-3211 emulation programs.

```
=====
MICROPROCESSOR FAMILY      EMULATION PROGRAM
=====
```

8-BIT DEVICES:

NSC800	L01000.SAV
Z80	L00500.SAV
6502	L01500.SAV
6502S	L01400.SAV
6809	L00700.SAV
8031	L00600.SAV
8048	L00100.SAV
8080	L00900.SAV
8085	L00800.SAV
8088	L01600.SAV

16-BIT DEVICES:

8086	L00200.SAV
68000	L01200.SAV
68010	L01700.SAV
80186	L01800.SAV
80188	L01900.SAV
Z8000	L01300.SAV

APPENDIX D

FASKEY MENU SUMMARY

The following diagrams display the "FasKey" keypad designations for two FasKey menus, FasKey 1 and FasKey 2. Additionally, a series of submenus may be accessed from FasKey 1 and are explained in the displays which follow.

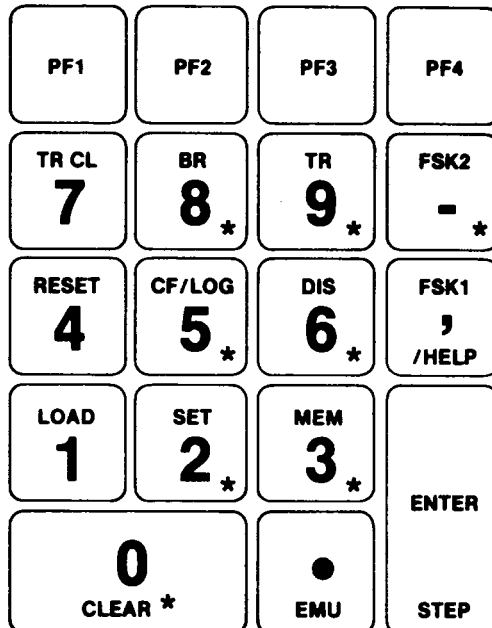
The FasKey 1 menu is accessed at ECL-3211 start-up by pressing the comma key on the Emulogic FasKey keypad on the right-hand side of the keyboard. "FasKey 1" will display on the Fask: line in the lower right-hand corner of the screen display. To access the FasKey 2 menu, press the minus key on the keypad while in FasKey 1.

The asterisked keys in the following diagrams represent keys which will access a FasKey submenu.

The menus and submenus display in the central scroll area. If you do not wish to see the menus, once you are familiar with the FasKeys and their functions, press the "0" key twice on the FasKey 1 keypad.

When you are in FasKey 2 or one of the FasKey 1 submenus, pressing the minus key will always return you to the basic FasKey 1 menu shown directly below:

FASKEY 1 MENU



CLEAR OPTIONS

FASKEY 1: THE "0" KEY

The CLEAR FasKey menu allows you quick input of the MAP CLEAR, BReak-point 0-7 CLEAR, TRACE CLEAR, HELP CLEAR (to access CLEAR help file), CLEAR REGISTERS AND CLEAR MEMORY commands. The "0" key also allows clearing of the central scroll area so that you may input using FasKey without displaying FasKey menus.

FASKEY 1: 0 KEY CLEAR OPTIONS

PF1	PF2	PF3	PF4
MAP CL 7	BR 0-7 CL 8	TR CL 9	FSK1 -
4	5	6	CL , HELP CL
1	CLEAR REG 2	CLEAR MEM 3	ENTER
0 CLEAR SCROLL	•		

SET OPTIONS

FASKEY 1: THE "2" KEY

The SET FasKey menu allows you quick input of the SET REGISTERS, SET MEMORY, SET SWITCH, SET FREQUENCY, SET COUNTERS, and HELP for SET commands.

FASKEY 1: 2 KEY SET OPTIONS

PF1	PF2	PF3	PF4
7	8	9	FSK1 -
SW 4	FREQ 5	COUNT 6	SET , HELP SE
1	SET REG 2	SET MEM 3	ENTER
0		•	

MEMORY OPTIONS

FASKEY 1: THE "3" KEY

The FasKey MEM menu allows you quick input of the following MEM commands: HELP ME (MEM command help file), MEM WRITE, MEM DUMP, MEM/QUICK/ASCII, MEM/ASCII, DISASSEMBLE, MEM/QUICK, SET MEM, MEM display, MEM CLEAR and MOVE (memory segment).

FASKEY 1: 3 KEY MEMORY OPTIONS

PF1	PF2	PF3	PF4
7	WRITE 8	DUMP 9	FSK1 -
MEM/Q/A 4	MEM/A 5	DIS 6	MEM DISPLAY , HELP ME
MEM/Q 1	SET MEM 2	MEM 3	ENTER
0 CLEAR MEM	MOVE ●		

CF/LOG OPTIONS

FASKEY 1: THE "5" KEY

The CF/LOG menu allows quick input of a variety of LOG and Command File commands: HELP LOG (the help file for the LOG command), CF/D COM (to return to the COM default command file extension, LOG, LOG/Create, LOG END, CF/LOG mini help file, CF/Type, CF/Default, CF/Resume, CF Clear, CF File.ext and "enter Keypad CF Mode".

FASKEY 1: 5 KEY CF/LOG OPTIONS

PF1	PF2	PF3	PF4
7	HE LOG 8	CF/D COM 9	FSK1 -
LOG 4	LOG/C 5	LOG END 6	CF/LOG , HELP CF
CF/T 1	CF/D 2	CF/R 3	KEYPAD CF MODE * ENTER
0 CF CLEAR	CF ●		

COMMAND FILE OPTIONS - KEYPAD CF MODE

FASKEY 1: LOG/C (5): KEYPAD CF MODE (ENTER)

**COMMAND FILE OPTIONS
KEYPAD CF MODE
(ENTER FROM CF/LOG)**

PF1	PF2	PF3	PF4
CF 7 FKEY 7	CF 8 FKEY 8	CF 9 FKEY 9	FSK1 -
CF 4 FKEY 4	CF 5 FKEY 5	CF 6 FKEY 6	CF/T FKTXT , HELP CF
CF 1 FKEY 1	CF 2 FKEY 2	CF 3 FKEY 3	CF/LOG ENTER
0 FKEY 0		•	

DIS OPTIONS

FASKEY 1: THE "6" KEY

Input of the keys designated on this menu enables quick entry of the following DISASSEMBLE memory commands: DIS, HELP DI (the disassemble memory help file), DIS PRINT, and DIS WRITE.

FASKEY 1: 6 KEY DIS OPTIONS

PF1	PF2	PF3	PF4
7	8	9	FSK1 -
4	5	DIS 6	DIS OPTIONS , HELP DI
DIS PR 1	DIS WR 2	3	ENTER
0		.	

BREAKPOINT OPTIONS

FASKEY 1: THE "8" KEY

The Breakpoint Menu, accessed by key 8 on the Faskey 1 menu, enables quick input of the following breakpoint commands: BR 0-7 OFF, BR 0-7 CLEAR, Breakpoint Help File, and the basic define breakpoint command BR which is accessed by keys 0-7. When selected, keys 0-7 bring up the BR # Options displayed on the next page. The prompt ENTER BREAKPOINT NUMBER --USE KEYPAD-- appears on the S: Line.

FASKEY 1: 8 KEY BREAKPOINT OPTIONS

PF1	PF2	PF3	PF4
BR 7 7 *	BR 0-7 OFF 8	BR 0-7 CL 9	FSK1 -
BR 4 4 *	BR 5 5 *	BR 6 6 *	BR OPTIONS , HELP BR
BR 1 1 *	BR 2 2 *	BR 3 3 *	ENTER
BR 0 0 *		•	

BR # OPTIONS

FASKEY 1: BR # 0-7 KEYS

Use of this FasKey menu level allows quick entry of the following ECL-3211 breakpoint commands for the specific numbered breakpoint: Display actions, Display conditions (when / entered); set breakpoint parameters for the following commonly used settings -- PA/PC=, PA/ADDR=, Breakpoint (specific number being defined)=, HL/PC=, HL/ADDR=, display breakpoint help file, turn ON specific breakpoint, turn OFF specific breakpoint, display specific breakpoint parameters (key 3) and clear specified breakpoint.

FASKEY 1: BR# 0-7 KEYS BR # OPTIONS

PF1	PF2	PF3	PF4
ACTIONS 7 /COND	PA/PC= 8	PA/ADDR= 9	FSK 1 -
BR #= 4	HL/PC= 5	HL/ADDR= 6	OPTIONS , HELP BR
BR # ON 1	BR # OFF 2	BR # (CR) 3	ENTER
0 BR # CL	.		

TRACE OPTIONS

FASKEY 1: THE "9" KEY

The trace options menu enables quick input of the following TRACE commands: TRACE, the Trace Help File, TRACE PRINT, TRACE WRITE, and TRACE CLEAR. The "3" key has been assigned to issue the DIS (disassemble memory) command.

FASKEY 1: 9 KEY TRACE OPTIONS

PF1	PF2	PF3	PF4
7	8	TRACE 9	FSK 1 -
4	5	6	TR OPTIONS , HELP TR
TR PR 1	TR WR 2	DIS 3	ENTER
0 TR CLEAR		•	

THE FASKEY 2 MENU

This menu, accessed by pressing the minus key while in the FasKey 1 command mode, enables quick entry of the following commands: MAP memory, WRITE, DUMP, SWitch, FREQ, COUNT, EXIT (takes you back to RT-11), MOVE (moves memory segments), SYMbol, SAVE, EMUlate and STEP. There are no submenus accessed from the FasKey 2 command mode. To return to FasKey 1 command mode, press the minus key.

FASKEY 2 MENU

PF1	PF2	PF3	PF4
MAP 7	WRITE 8	DUMP 9	FSK 1 -
SW 4	FREQ 5	COUNT 6	FSK 2 , /HELP
EXIT 1	MOVE 2	SYM 3	ENTER
0 SAVE	● EMU		

CF/P OPTIONS

WHENEVER A COMMAND FILE IS PAUSED

Whenever the ECL-3211 is in a command file pause state (CF/P FILE.EXT is displayed in the lower left-hand corner of the screen), four keys on the keypad assume special capabilities. The hyphen key allows you to backspace to the last previous command in the command file stream and to execute that command. The comma key displays a help file for use of the special CF/P keys. The ENTER key ends the paused command file. The period key issues a CF/R command to resume command file execution.

If the system is operating from a command file at the moment of CF/P, the remaining keys (0-9), will issue the CF FKEY level keypad commands, executing the corresponding numerically coded user-created FasKey command file. Otherwise, the keypad keys will issue the FasKey 1 commands operative at the time of the CF/P. Look at the FasK: Line to determine the level at which the FasKey keypad is operating.

CF/P OPTIONS (Whenever a command file is paused)

PF1	PF2	PF3	PF4
7	8	9	BACK SPACE - 1 LINE
4	5	6	CF/P , HELP
1	2	3	END PAUSED CF ENTER
0	CF/R •		

APPENDIX E

LIST OF FIGURES

Figure Number	Page Number
3.1 Initial Start-up Prompt Screen.....	3-4
3.2 Initial Screen Format for Z80 Chip.....	3-7
3.3 Screen Display for Z80 Memory Range.....	3-8
3.4 Screen Display for Breakpoint Setting.....	3-9
3.5 Trace Buffer Display (Z80 Chip).....	3-10
3.6 Setting Switches (8-bit Chip).....	3-11
3.7 Screen Display of Memory Range for 16-bit 68000 Chip.....	3-12
3.8 FasKey 1 Command Menu.....	3-17
3.9 FasKey 2 Command Menu.....	3-18
4.1 High-Speed Mapping at Start-up.....	4-5
4.2 Map Clear Command.....	4-7
4.3 The Map Internal Command.....	4-9
4.4 Mapping High-Speed Memory as Simulated ROM (68000 Chip)....	4-11
4.5 Displaying Map Parameters.....	4-13
4.6 Memory Display - The MEM Command.....	4-14
4.7 Displaying Memory in ASCII and HEX.....	4-16
4.8 Disassembling Memory.....	4-18
5.1 Displaying a Specific Breakpoint's Parameters.....	5-3
5.2 Sample Trace.....	5-25
5.3 Symbolic Debugging.....	5-29

APPENDIX F

LIST OF TABLES

Table Number	Page Number
3.1 Emulation Screen Display of Chip Status.....	3-6
3.2 Emulation System Modes of Operation.....	3-13
3.4 Special Key Functions.....	3-15
5.1 Breakpoint Condition Separators.....	5-5
5.2 Breakpoint Actions.....	5-6
5.3 System Conditions.....	5-12
5.4 System Signals.....	5-18
8.1 Memory Management Commands.....	8-1
8.2 Emulation Management Commands.....	8-5
8.3 Screen Management Commands.....	8-8
8.4 Command File Management Commands.....	8-9
8.5 Miscellaneous Commands.....	8-12