

8510/a GRAPHICS COMPUTER SYSTEM INSTALLATION & USERS GUIDE

This document describes the installation of a TERAK 8510/a Graphics Computer System. Single and multiple disk drive configurations are presented, along with system acceptance test procedures. Operating environment requirements, a basic troubleshooting guide, maintenance procedures and descriptions of the system components are also included. Contained in appendices are descriptions of the system modules and a part number listing of the major assemblies.



"TERAK CORPORATION believes that the information contained herein is accurate. In no event will TERAK be liable for any losses or damages whether direct or indirect resulting from the use of such information, including, without limitation, losses arising from claims of patent, copyright, and trademark infringement. No license is granted hereby for the use of any patent or patent rights of TERAK. TERAK reserves the right to update the information contained herein at any time without further notice.

The information contained herein is proprietary to TERAK CORPORATION and must be treated as confidential. It may not be disclosed to others or used for any purpose except as expressly consented to by TERAK."

8510/a GRAPHICS COMPUTER SYSTEM INSTALLATION & USERS GUIDE

Part No. 50-0010-001 FIRST EDITION

Copyright 1980 by TERAK CORPORATION "All Rights Reserved"

TERAK is a trademark of TERAK CORPORATION. DEC, PDP-11, RT-11 and LSI-11 are trademarks of DIGITAL EQUIPMENT CORPORATION.

CONTENTS

PAC	
INTRODUCTION	1
SECTION	
1. PRE-INSTALLATION CONSIDERATIONS	
1.1 PHYSICAL INSPECTION OF SYSTEM COMPONENTS	2
1.2 SYSTEM POWER & OPERATING ENVIRONMENT REQUIREMENTS	Z
1.2.1 Input Voltage & Frequency Options	2
1 2 3 Temperature Limits	2
1.2.4 Relative Humidity Range	ა
1.3 UNIT DIMENSIONS	۰۰۰۰
2. SYSTEM INSTALLATION PROCEDURES	
2.1 SINGLE DISK DRIVE CONFIGURATION	
2.2 MULTIPLE DISK DRIVE CONFIGURATION	4
3. SYSTEM ACCEPTANCE TESTS	4
3.1 POWER ON; DISKETTE LOADING	
3.2 SYSTEM ACCEPTANCE TEST PROCEDURES	5
3.3 DISKETTE REMOVAL	
4. SYSTEM EXPANSION	5
4.1 BUS EXTENSION	
4.2 MODULE INSTALLATION	
5. UNIT COVER REMOVAL-REPLACEMENT	8
5.1 8510, 8512 & 8515 UNITS	8
5.2 8532-1 MONITOR	9
6. UNIT AC INPUT VOLTAGE SELECTION	
6.1 8510/12/15 UNITS	
7. UNIT CHANGES FOR 50 Hz & 60 Hz AC POWER FREQUENCIES	12
8. BASIC TROUBLESHOOTING	13
9. SYSTEM DESCRIPTION	17
9.1 8510 DATA PROCESSOR SYSTEM	
9.2 8532-1 MONITOR	
* 9.3 8532-2 KEYBOARD	17
9.4 8512 DISK SUBSYSTEM	
9.5 8515 EXPANSION SYSTEM	19

(,,,,,,

APPENDICES

A RAW OF SHARE VALUE	PAG	GE
A. Diskette Handling/Storage Considerations	A1 thru	A2
B. Suggested Maintenance		B1
C. Processor Reference Guide	C1 thru (C44
D. Video Controller-Memory Reference Guide	D1 thru	D8
E. Diskette Controller Reference Guide	E1 thru I	E19
F. Asynchronous Serial Interface Reference Guide	F1 thru	F5
G. Parallel Interface Reference Guide	G1 thru	G5
H. Use of Control Console Mode (Micro-ODT)	H1 thru	H5
I. TERAK Part Numbers, Major Assemblies		. I1
LIST OF FIGURES	FIGURE N	
System Components		2.1
Single Disk Drive Interconnections		2-1
Multiple Disk Drive Interconnections		۷-2 1.1
8510/a Module Locations & I/O Priority		1.2
Bus Extension Interconnections		1-3
EIB Locations		5-1
8510/12/15 Cover Removal-Replacement & 8510/15 Module Retaining Hardware		5-2
8532-1 Monitor Chassis Removal-Replacement		6-1
8510/12/15 AC Input Voltage Selection		6-2
8532-1 AC Input Voltage Selection		8-1
8532-2 Keyboard Unit Output Codes		9-1
8532-2 Reyboard Unit Output Codes		9-2
Diskette Loading		Á-1
Diskette Handling/Storage Considerations		A-2
General Register Identification		C-1
High and Low Byte		C-2
Word & Byte Addresses for First 4K Bank		C-3
Processor Status Word (PS)		C-4
System Memory Organization		D-1
Display Geometry		D-2
Composite Video Waveform at 8510 Video EIB BNC		D-3
Graphic Address Register (GAR)		D-4
Video Control Register (VCR)		D-0
Generator Ruffer Address Format		D-0
Page Buffer Address Format		D-7
Video Index Register (VIR)		. レー0
Keuhoard Data Buffer (KDB)		. D-9
Keyboard Status Register (KSR)		D-10
Emulator Data Buffer (FDB)		D-11
Emulator Status Register (FSR)		D-12
Diskette Controller Control & Status Register (OXCS)		C-1
Diskette Controller Data Buffer (OXCR)		C-2
Diskette Controller ROM Listing		E-3
Diskette Controller Sample System Bootstran Listing		C-4
Diskette Controller Listing of Dummy Boot Example		E-0
Serial Interface FIB Connector & Switch Functions		r-
Single Serial Current Loop & J3 Wiring	. .	r-2
Parallel Interface FIB Connector & Module Jumper Functions		. U
Data & Control Word Formats	• • • • • • • • • • • • • • • • • • • •	. 6-2
I/O Data Transfers	• • • • • • • • • • • • • • • • • • • •	. U∹
8510/a HALT Switch Module		. IT

 $(\geq$

INTRODUCTION

The TERAK 8510/a is a general-purpose, disk-based computer intended for data processing and medium-resolution graphics applications. A 320 x 240 dot pattern and/or 1920 characters (24 lines of 80) can be displayed (under program control) on a 12 in. monochrome CRT. Four drives, accommodating IBM 3740 formatted (8 in.) floppy diskettes, can be included in a system configuration. The hardware is modularized into a keyboard unit, a CRT display, a data processor unit, bus expansion unit and optional drive units. The system components are shown in Figure 1-1. All components are light-weight and are suitable for office and laboratory environments.

This document describes the installation of a TERAK \$510/a\$ Graphic Computer System. The installation consists of the interconnection and acceptance testing of the system hardware. Also included is a brief trouble-shooting guide and system description. Appendices contain diskette handling/storage considerations, suggested maintenance procedures, a parts list and reference guides for the system hardware.

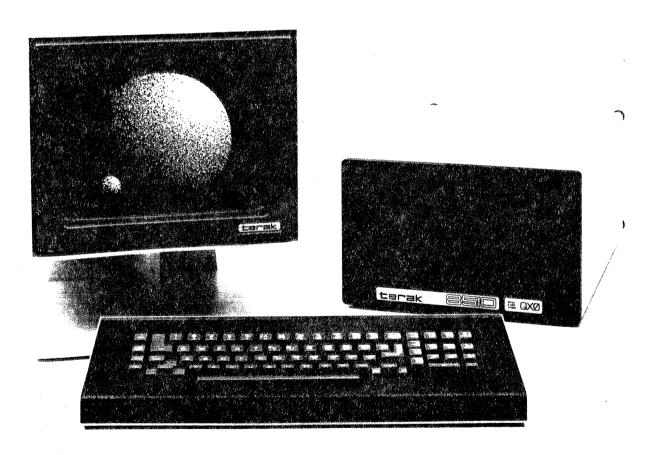


Figure 1-1 System Components

1 PRE-INSTALLATION CONSIDERATIONS

1.1 PHYSICAL INSPECTION OF SYSTEM COMPONENTS

Carefully unpack all the shipping containers; save all the container materials for equipment storage or future reshipment.

Each container is marked with the unit part number(s) and individual serial number(s). (Appendix I lists the part numbers of the major assemblies.) Use the following lists to verify the contents of each container.

8510 Container:
8510 Data Processor 1 ea.
8510/a System Acceptance Disk 1 ea.
8510/a Documentation Kit 1 ea.
AC Power Cord 1 ea.
8515 Container:
8515 Expansion System 1 ea.
Cable, Aux. Drive, Short
8510 Bus Extension EIB (packed inside
the 8515 unit) 1 ea.
AC Power Cord 1 ea.
8512 Container:
8512 Disk Subsystem 1 ea.
Cable, Aux. Drive, Short 1 ea.
AC Power Cord 1 ea.
8532-1, 8532-2 Container:
8532-1 Monitor 1 ea.
8532 Monitor Cable (6 ft.)
8532-2 Keyboard (with integral 8532
Keyboard Cable Assembly) 1 ea.
AC Power Cord

Inspect all items for obvious signs of damage. DO NOT interconnect a damaged item to a system; contact TERAK, or its representative, for further information.

NOTE: The 8510, 8512 and 8515 disk drive heads are protected during shipment by foam blocks that are inserted in the disk drive openings. These blocks ensure that the doors will stay open, so that the drive heads remain unloaded (not down) and therefore protected. Remove and save these blocks for future re-shipment. Shipping the units without the foam blocks in place may result in damage to the drive heads.

1.2 SYSTEM POWER & OPERATING ENVIRONMENT REQUIREMENTS

The TERAK 8510/a system requires single-phase ac power sources and earth-grounded chassis protection. Each system unit that requires ac power input is delivered with an ac power cord that contains three insulated 18-gauge stranded wires. The chassis ground wire is connected to the round pin of the ac power cord.

A grounding conductor that is equal, or greater in size, to the power cord, must be a part of the branch circuit that supplies each unit of the 8510/a system.

The grounding conductor mentioned above must be grounded to earth at the service equipment or other acceptable building earth ground such as the building frame in the case of a high-rise steel frame structure.

The attachment-plug receptacles in the vicinity of each unit of the 8510/a system are all to be of a grounding type, and the grounding conductors serving these receptacles are to be connected to earth ground at the service equipment or other acceptacle building earth ground such as the building frame in the case of a high-rise steel frame structure.

1.2.1 AC Input Voltage & Frequency Options

100 V RMS	50 Hz/60 Hz
120 V RMS	50 Hz/60 Hz
220 V RMS	50 Hz/60 Hz
240 V RMS	50 Hz/60 Hz

(CAUTION)

Each unit has been factory wired for a particular ac input voltage and frequency. Labels, indicating the values, were affixed to the unit when it was initially shipped from the factory. To use an ac voltage other than that indicated by the label, internal jumpers must be changed. See Section 6 for a description of voltage selection.

Each unit has been configured for either 50 Hz or 60 Hz operation. For 50 Hz operation, a label signifying that frequency was affixed to the unit when it was initially shipped from the factory. See Section 7 for a description of the changes necessary to operate at 50 Hz vs 60 Hz.

WARNING

THE 8512, 8532-1 AND 8532-2 UNITS ARE DESIGNED AND INTENDED FOR USE ONLY WITH THE 8510 UNIT. ANY ATTEMPT TO CONNECT THE 8512, 8532-1 OR 8532-2 UNITS TO A DEVICE OTHER THAN THE 8510 UNIT MAY RESULT IN DAMAGE TO THE EQUIPMENT.

1.2.2. Power Requirements

8510 Unit: 250 Watts (Maximum) 8512 Unit: 100 Watts (Nominal) 8515 Unit: 250 Watts (Maximum) 8532-1 Unit: 24 Watts (Nominal) 8532-2 Unit: 2.5Watts (Nominal)

1.2.3 Temperature Limits

System Operating Range: 50 F (10 C) to 100 F

(37.7 C)

Hardware Storage Range: -40 F (-40 C) to 167 F

(75 C)

Media Storage Range: 50 F (10 C) to 120 F

(49 C)

1.2.4 Relative Humidity Range

Operating: 20% to 80% without condensation Storage: 5% to 98% without condensation

1.3 UNIT DIMENSIONS

1

8510: 7.5 in (19 cm) H x 12.2 in (31 cm) W x 18 in. (46 cm) D

8515: Same as 8510

8512: 5.5 in. (14 cm) H x 12.2 in. (31 cm) W x 18 in. (46 cm) D

8532-1: 15.5 in. (39 cm) H x 12.2 in. (31 cm) W x 12 in. (30.5 cm) D

8532-2: 2.5 in. (6.4 cm) H x 17.8 in. (45 cm) W x 7.1 in. (18 cm) D

1.4 UNIT WEIGHT

8510: 45.5 lbs. (Shipping container wt.: 53.5 lbs.) 8515: 41.5 lbs (Shipping container wt.: 50 lbs.) 8512: 31 lbs. (Shipping container wt.: 40 lbs.) 8532-1: 30.8 lbs. 8532-2: 6.3 lbs. (Shipping container wt.: 48.2 lbs.)

2 INSTALLATION PROCEDURES

Installation of an 8510/a system is straight-forward and, typically, can be accomplished within 15 minutes. The system can be configured via plug-in modules, connectors and switches mounted on External Interface Boards (EIBs) that are installed on the rear surfaces

of the units. In the following descriptions, the interconnection diagrams indicate which EIBs are involved in a particular configuration. Labels on each cable identify their use.

2.1 SINGLE DISK DRIVE CONFIGURATION

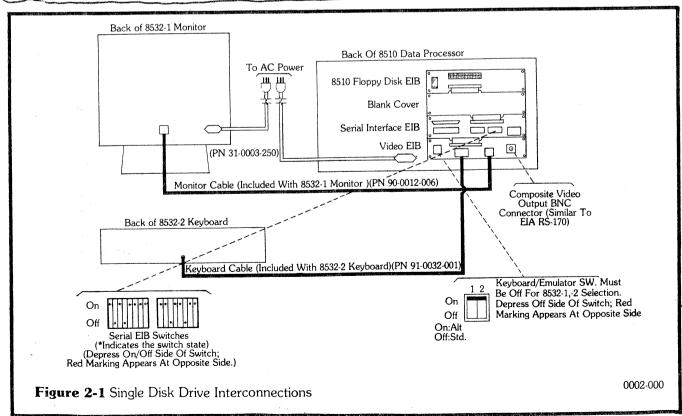
Figure 2-1 depicts the standard configuration of 8510 EIBs as installed at the factory. A six ft. cable, packed in a plastic bag in the 8532 shipping container, is used to connect the 8532-1 Monitor to the Video EIB of the 8510 Data Processor unit. Another six ft. cable, integral to the 8532-2 Keyboard, also plugs into the Video EIB. The connectors on the Video EIB are labeled "Monitor" and "Keyboard", and are of different size, for easy recognition.

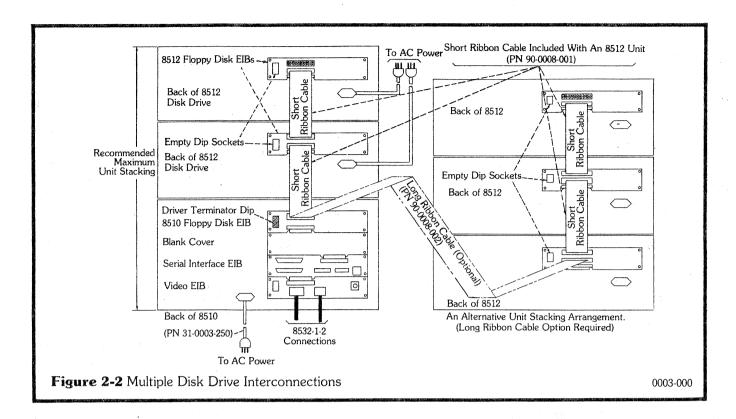
The "Keyboard/Emulator" switch, on the Video EIB, must be in the STD (off) position to use the 8532-1 Monitor and 8532-2 Keyboard as the system console. The switch is OFF when a red bar appears at the ON side of the housing; conversely, the switch is ON when a red bar appears at the OFF side.

The switches on the 8510 Asynchronous Serial Interface EIB must be set as shown in Figure 2-1.

The 8532-1 Monitor and 8510 Data Processor have individual ac power cords. A single switch on the 8510 controls the ac power to both units.

This completes the single disk drive interconnections. Proceed further in Sections 2 & 4 for other system configurations.





2.2 MULTIPLE DISK DRIVE CONFIGURATION

Refer to Figure 2-2. A maximum of four disk drives can be utilized in an 8510/a system. As shown, the 8512 Disk Subsystems can be stacked on top of the 8510, or alternatively, can be stacked separately. Each 8512 has a disk drive "daisy chain" connector EIB; these EIBs are interconnected to the 8510 Floppy Disk EIB by means of 8 inch flat ribbon cables (Part Number 90-0008-001). One ribbon cable is packed inside each 8512 (or 8515) shipping container. An optional 27 inch flat ribbon cable (Part Number 90-0008-002) can be used to position the 8512s at a different location on the user's workspace. The cable and unit connectors are keyed to prevent incorrect coupling; check the cable connector orientation before attempting to insert it in the EIB connector.

Terminating resistors for the disk drive daisy chain are contained in a 14-pin Dual-In-Line Package (DIP), on the 8510 Disk Drive EIB. For convenience, the terminator resistor DIP should remain in the 8510 socket. If the optional 27 in. cable is used, the terminator resistor DIP should be moved to the daisy chain EIB socket on the drive unit that is last in the daisy chain connection.

NOTE: The DIP must be oriented so that pin 1 is at the lower right-hand corner of the socket, when viewed from the rear of the unit. Pin 1 is identified by a dot or depression, or the number 1, on the package.

Each 8512 has an ac input power requirement; however, the control of ac power is by means of the main ac power ON/OFF switch on the 8510 unit.

3 SYSTEM ACCEPTANCE TESTS

After an 8510/a system has been interconnected in accordance with Section 2 of this document, the system should be initially tested by means of the TERAK System Acceptance Disk (Part Number 61-0006-001). The diskette contains GO/NO-GO tests that exercise the hardware subsystems. Each test is preceded by an explanatory message on the display; user instructions will also appear on the display.

NOTE: If the following procedures cannot be accomplished, refer to Section 8 for troubleshooting steps.

3.1 POWER ON; DISKETTE LOADING

Turn the system on by means of the ac power switches located in the upper right corners of the 8510 and 8515 (if used) front panels. If closed, open the diskette loading door on the 8510 (Drive QX0) by depressing the button just below the door.

NOTE: If you are unfamiliar with diskette handling procedures, refer to Appendix A first.

Remove the System Acceptance Diskette from its protective envelope; keep your fingers off of the exposed magnetic surfaces of the diskette. Holding the diskette with the label side up, as shown in Figure A-1, insert it into the QX0 drive until it can go no further. The diskette will "latch" into position. Close the drive door; the drive head-down light (Red Light Emitting Diode embedded in the door button) will turn on.

3.2 SYSTEM ACCEPTANCE TEST PROCEDURES

After the System Acceptance Test diskette has been inserted in drive QX0, and the drive door has been closed, the operating system should load itself (boot) into main memory. Within 20 seconds, an acceptance test-related message should appear on the display.

The test-related message will prompt the user to execute a carriage return to display the "menu" of tests. Select a test by typing the displayed test name, followed by a carriage return. The test procedures will then appear on the display.

NOTE: Once a test has begun, the user should not type anything, unless prompted by a message. Typing during the test execution can produce unpredictable results and an apparent test failure.

NOTE: To exit from any test, at any time, re-boot the operating system by depressing the upper half of the 8510 ac power switch; the acceptance test-related message will re-appear on the display.

3.3 DISKETTE REMOVAL

After the System Acceptance Tests are completed, remove the diskette by depressing the drive door button. The door will open and the diskette will be partially

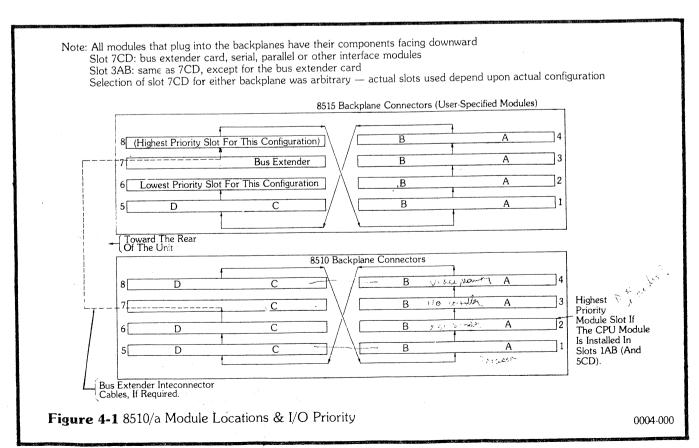
ejected. Remove the diskette and return it to its protective envelope.

4 SYSTEM EXPANSION

A 8510 system can be expanded, by means of modular options, to allow the user to interconnect a variety of peripheral equipment. TERAK, or other LSI-11 buscompatible modules can be installed in the 8510 and 8515 units.

The 8510 and 8515 units each contain an LSI-11 bus-compatible backplane. Each backplane contains sixteen 36 pin connectors, as shown in Figure 4-1. The LSI-11 bus is based on the use of dual-height (8.9 in W x 5.2 in H) modules; that is, modules that plug into pairs of connectors (A&B or C&D).

A module has an I/O priority that is based on its electrical distance from the processor module, as shown in Figure 4-1. For instance, if a quad-height (8.9 in W x 10.5 in H) processor module were installed in slots 1 and 5 of the backplane, the highest priority slot would be #2. The arrows on Figure 4-1 signify the descending order of priority. A quad-height module uses slot connectors A&B for I/O priority signaling; a dual-height module uses either A&B or C&D. An LSI-11 bus-compatible quadheight module maintains continuity of the priority signals through slots C&D by means of etched circuit jumpers on the module



4.1 PROCESSOR BUS EXTENSION

Refer to Figures 4-2 & 4-3. The Bus Extension EIBs and the 8515 Backplane provide the user with a means of extending the 8510's data bus.

An 8515 unit is shipped with a factory-installed Floppy Disk EIB, two Cable Clamp Covers and a Blank Cover, as shown in Figure 4-3. The 8510 Bus Extension EIB has been packed inside the 8515 for shipment; this EIB must be removed from the 8515 and installed in the 8510. In the process, the 8510 Floppy Disk EIB must be relocated.

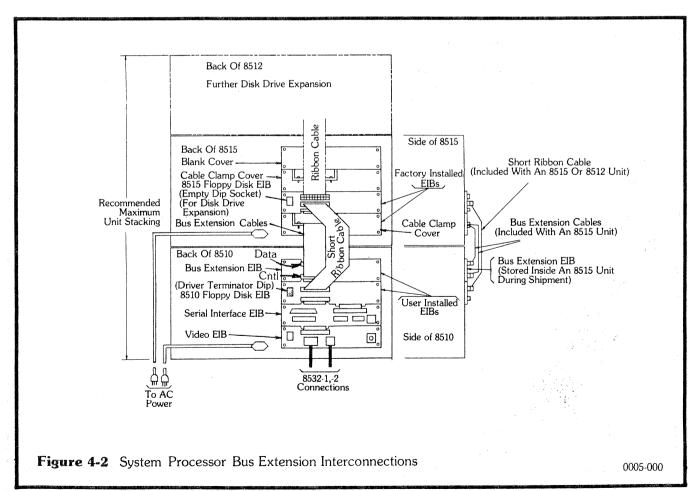
Remove the 8510 and 8515 covers in accordance with Part 5.1, Section 5. Remove the 8510 Bus Extension EIB from the 8515. Check to see that the 8515 Bus Extension EIB cable card is plugged into the 8515 Backplane. (See Figure 4-1 for Backplane locations.) Additional modules may be installed in the 8515 at this time. Note the device I/O priority indicated by Figure 4-1. The 8515 cover can be reinstalled at this time.

All EIBs are attached to the units by means of 4 Phillips head screws, one in each corner of the EIB. Remove the Blank Cover from the 8510; save it for future use. Unfasten the Floppy Disk EIB from the top-most location and let it hang loose.

Remove the module retainer bracket in the 8510 (see Figure 5-1). The Bus Extension EIB card must now be installed in the 8510 backplane slot that is shown in Figure 4-1. The component (connector) side of the card must be facing downward when it is inserted. Lift the ribbon cable through the notch in the card; slide the card through the module guide and insert it into the backplane. Pass the card's ribbon cable and attached EIB through the rear of the 8510. Secure the Bus Extension EIB to the top-most EIB location with four screws, then install the Disk Drive EIB just below it. The 8510 cover can now be re-installed.

As shown in Figure 4-2, the bus extension cables that emerge from the 8515 Cable Clamp Cover are connected to the Bus Extension EIB on the 8510. The shortest (innermost) cable of this pair plugs into the DATA connector on the 8510 Bus Extension EIB; the other cable plugs into the CONTROL connector. An 8 inch ribbon cable, (Part Number 90-0008-001, packed in the 8515 shipping container), connects the 8510 Floppy Disk EIB to the 8515 Floppy Disk EIB.

The 8515 requires ac input power and is controlled from a switch on its front panel. Both the 8510 and 8515 ac power switches must be on for the 8510/a system to operate. The system can be reset from either the 8510 or 8515 power on/reset switches.

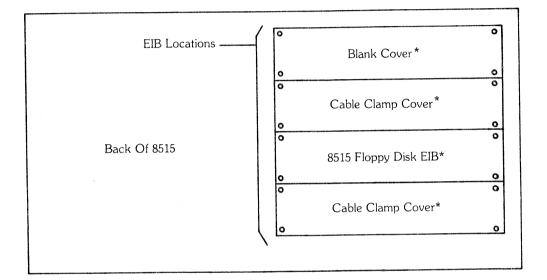


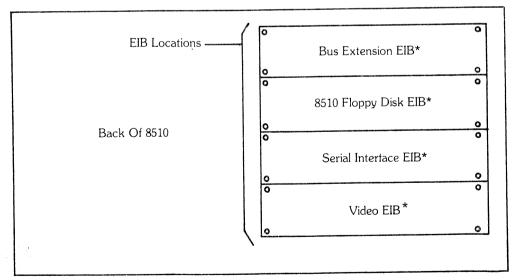
6

Single EIB Location 8512 Floppy Disk EIB*

o

Back Of 8512





^{*}Standard Factory-Installed EIB Configuration; Can Be Altered With Any Combination Of EIBs.

NOTE: All Locations Must Be Filled With An EIB Or A Cover Plate To Maintain Proper Air Flow Through A unit.

Figure 4-3 EIB Locations

0006-000

4.2 MODULE INSTALLATION

The installation of a module is quite straight-forward. As described in Part 4, the I/O priority of module is a function of its electrical distance from the system processor module. Only LSI-11 bus-compatible modules are to be installed. After the unit cover and module retainer bracket have been removed (see Part 5.1 and Figure 5-1), a dual or quad-height module can be plugged into a backplane location. (The power supplies in the 8510 and 8515 units can provide a maximum of 12 A @ 5 VDC and 6 A @ 12 VDC to the backplanes; the available power is a function of the number and type of modules installed.)

Peripheral connections to a module are by means of External Interface Boards (EIBs). These EIBs, located at the rear of the units, as shown in Figure 4-3, are held in place by four Phillips-head screws. With the exception of the bus extension EIBs (described in Part 4.1). the module EIBs can be installed in any location at the rear of the 8510 and 8515 units. Neatly fold the EIB-module ribbon cable(s) along the chassis when re-installing a unit cover, so that the cables are not pinched when the cover is fastened down. EIB connector pin listings and switch settings (if applicable) are described in the Appendices.

5 UNIT COVER REMOVAL— REPLACEMENT

WARNING

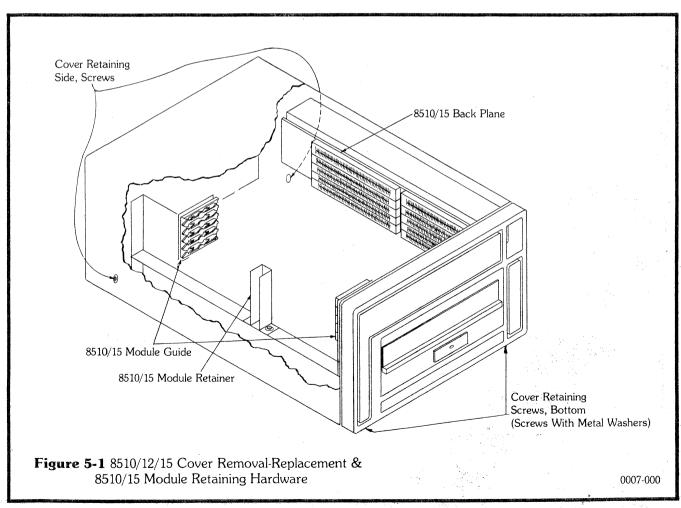
Due to the complex nature of the system components, unit cover removal-replacement, and subsequent maintenance and repairs, should only be performed by technically-qualified personnel.



DISCONNECT THE UNIT AC POWER CORDS BEFORE REMOVING THE COVERS

5.1 8510, 8515 & 8512 UNITS

The unit covers are held in place by four Phillips-head screws, as indicated by Figure 5-1. Remove the two retaining screws and nylon finishing washers at the rearsides of the unit. Loosen the two retaining screws with the metal washers at the bottom-front of the unit. It is not necessary to remove these screws completely. Don't stand a unit on its rear feet to loosen the bottom-front



screws if cables are connected to EIBs. Instead, slide the unit forward until it protrudes far enough beyond the desk or bench edge to allow access to the bottom front screws. Slide the integral front panel-cover forward until it separates from the chassis.

Re-install a cover by sliding it back onto its chassis, taking care not to pinch any ribbon cables that may be protruding slightly. When the cover is in place, tighten the bottom front screws and then replace the rear-side screws.

5.2 8532-1 MONITOR

CAUTION

AVOID STRIKING THE PICTURE TUBE

The Monitor chassis is held in place by 8 Phillips panhead screws; six (#8's) on the bottom and two (#6's) on the rear of the cabinet, as shown in Figure 5-2. Remove these screws and slide the chassis straight back out of its cabinet.

NOTE: Do not remove the flat-head screws securing the stand to the Monitor cabinet.

6 UNIT AC INPUT VOLTAGE SELECTION

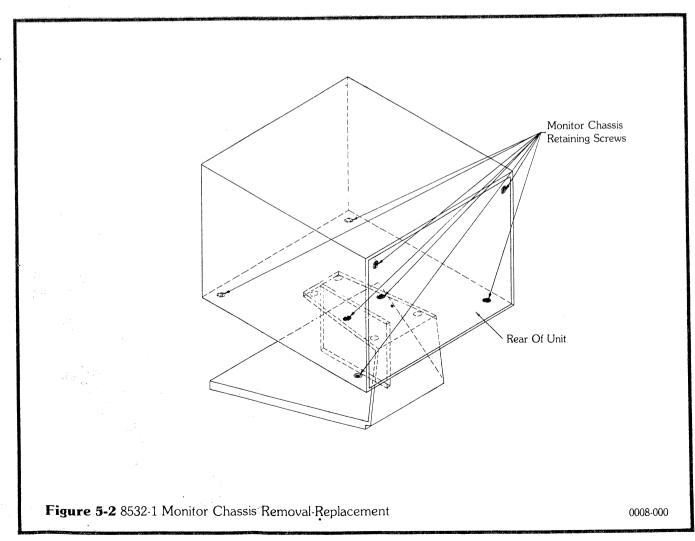
WARNING

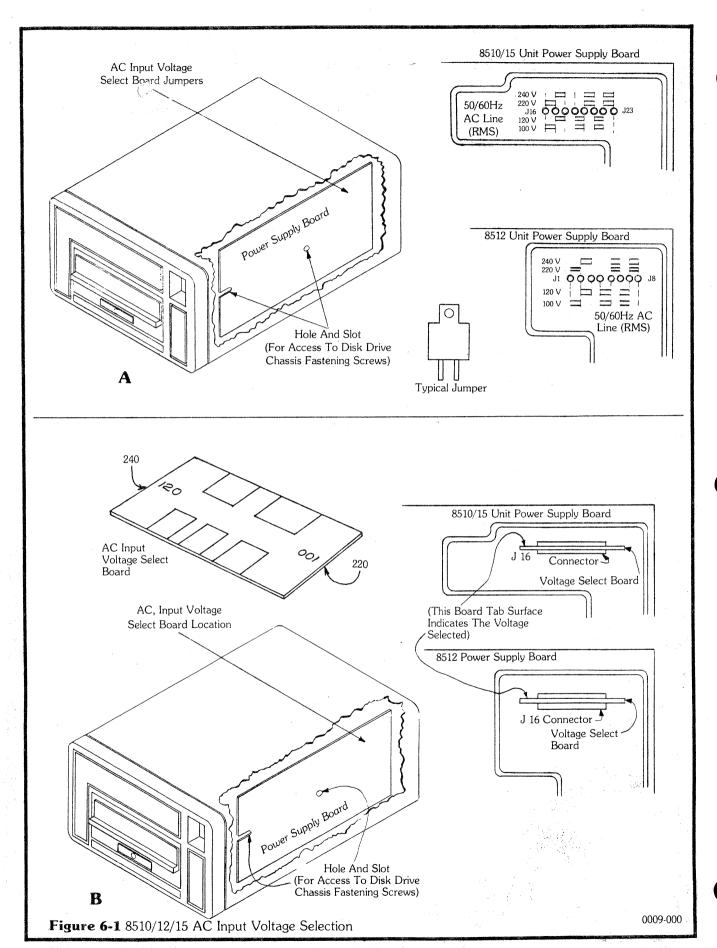
THE VOLTAGES REQUIRED BY THE SYSTEM ARE POTENTIALLY HAZARDOUS: ONLY TECH—NICALLY QUALIFIED PERSONNEL SHOULD PERFORM THE AC INPUT VOLTAGE SELECTION CHANGES.

6.1 8510, 8612 & 8515 UNITS

Remove the covers in accordance with Part 5.1, Section 5. As shown in Figure 6-1, there are two types of power supply boards in current use. Identify the board type in your particular unit, then proceed with the following appropriate paragraph.

As shown in Figure 6-1A, there are three jumpers on the power supply boards of each unit; they are located in the upper right-hand corner of the boards, on the





component side. The 8510 & 8515 jumper jacks are labeled J16 through J23; the 8512 jacks are labeled J1 through J8. Insert the jumpers into the jack pairs that are color-banded together for a particular voltage. For instance, to operate an 8512 unit at 120 V, the jumpered pairs would be: J2 to J3; J4 to J5; J6 to J7.

As shown in Figure 6-1B, a printed wiring board is used for voltage selection. This board must be inserted into a connector located in the upper right-hand corner of the power supply board. The connector is labeled J16. To select a different ac input voltage, remove the board and re-insert it so that the desired voltage is indicated on the board tab surface that is nearest to the front and top of the unit, as shown in Figure 6-1B.

6.2 8532-1 UNIT

Remove the Monitor chassis in accordance with Part 5.2, Section 5. In Figure 6-2, the plugs labeled P1 and P2 contain the jumpers. P2 is actually a pair of plugs: one is for 100V & 120 V operation; the other is for 220V &240V operation. When the pin in P1 must be moved, use a MOLEX HT 2285 Extractor Tool, (or equivalent), to

remove the pin. Insert the pin in its alternate location until it locks in place.

100V Operation

- 1. Insert the "120" labeled P2 plug into J2.
- 2. Move the brown-wired pin in P1 to location 3 of P1.
- 3. Insert P1 into J1.

120V Operation

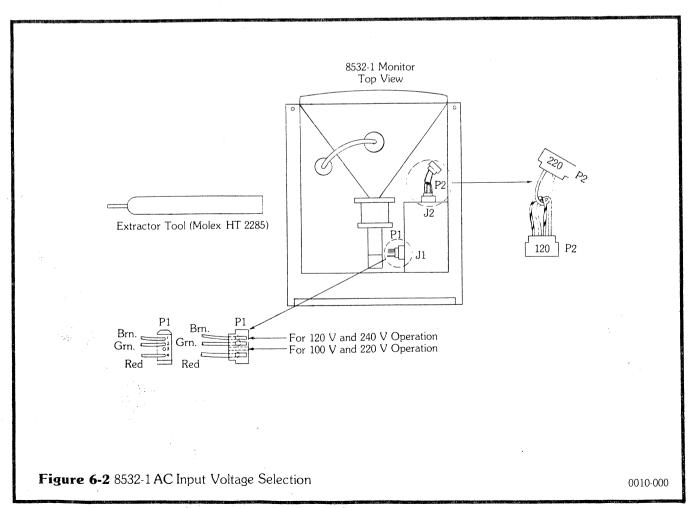
- 1. Insert the "120" labeled P2 plug into J2.
- 2. Move the brown-wired pin in P1 to location 1 of P1.
- 3. Insert P1 into J1.

220V Operation

- 1. Insert the "220" labeled P2 plug into J2.
- 2. Move the brown-wired pin in P1 to location 3 of P1
- 3. Insert P1 into J1.

240V Operation

- 1. Insert the "220" labeled P2 plug into J2.
- 2. Move the brown-wired pin in P1 to location 1 of P1.
- 3. Insert P1 into J1.



7 UNIT CHANGES FOR 50 Hz & 60 Hz POWER FREQUENCIES

The pulleys on the disk drive motors and the drive belts of the 8510, 8512 and 8515 units must be replaced when changing from 50 Hz to 60 Hz operation, or vice versa. The Video Board in the 8510 unit must also be exchanged. The following TERAK part numbers are applicable:

Item	50Hz	60 Hz
Video Board	92-0014-011	92-0014-010
Pulley, motor	39-0012-001	39-0012-002
Belt, disk drive	39-0013-001	39-0013-002

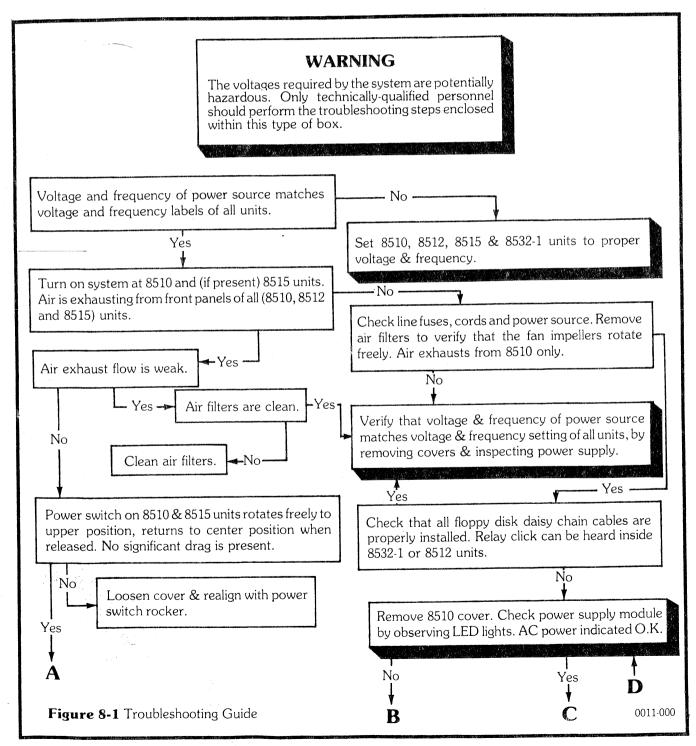
- 1. Remove the unit covers (see Part 5.1, Section 5).
- 2. Remove the 8510, 8515 module retainer brackets (see Figure 5-1).
- 3. Remove the modules from the 8510 & 8515 backplanes.
- 4. Remove the four Phillips-head screws that fasten a disk drive chassis to a unit chassis. Two of these screws are accessed through a hole and a slot in a power supply board (see Figure 6-1). The other two screws are on the unit chassis side that is opposite the power supply board.
- 5. Slide the disk drive chassis forward until the disk drive PWB connectors can be easily disconnected. Remove the disk drive chassis.
- 6. Stand the disk drive chassis on its side so that its PWB is visible. Remove the four slotted hexagonal screws that fasten the PWB to the chassis housing. Swing the PWB clear of the motor pulley and belt drive. There are three connectors still fastened to the PWB; they do not have to be disconnected.
- 7. Remove the disk drive belt.
- 8. Using a 1/16 in. Allen wrench, remove the small pulley on the motor shaft.
- 9. Install the new pulley and new drive belt.
- 10. Re-install the PWB and disk drive unit; re-install all modules except for the Memory-Video module.
- 11. Remove the ribbon cable connectors from P1, then P2, of the Memory board.
- 12. Separate the Memory and Video boards by removing the four corner, and two middle, Phillips-head screws that hold the boards together.
- 13. Fasten the Memory board to the NEW Video board.
- 14. Re-connect the ribbon cables to P2, then P1, of the Memory board.

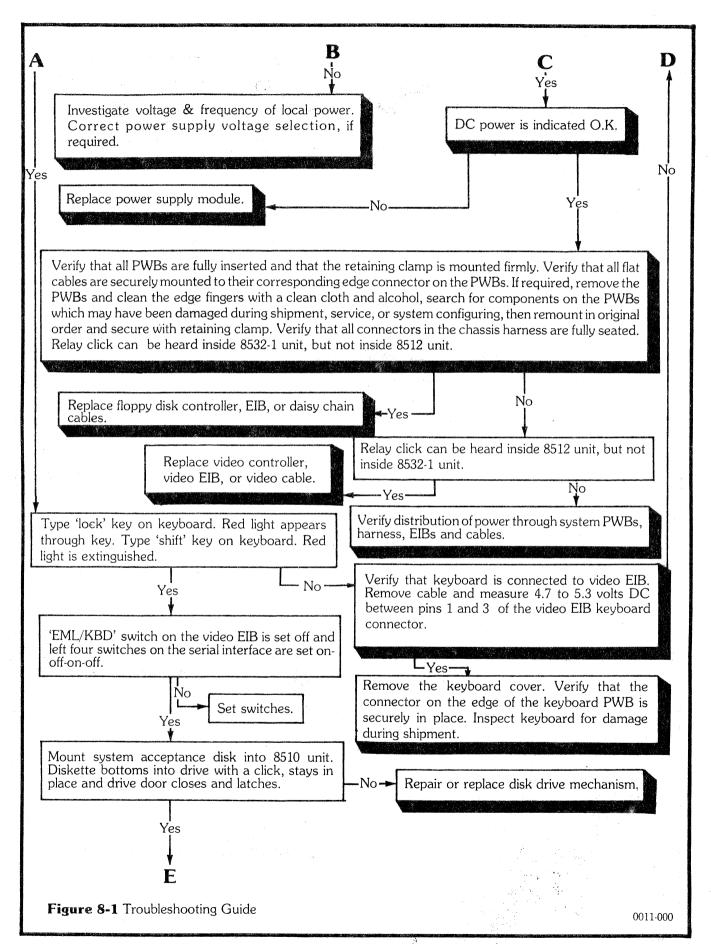
- 15. Re-install the Memory-Video module in the 8510 backplane, then replace the module retainer bracket.
- 16. Re-install the unit covers.

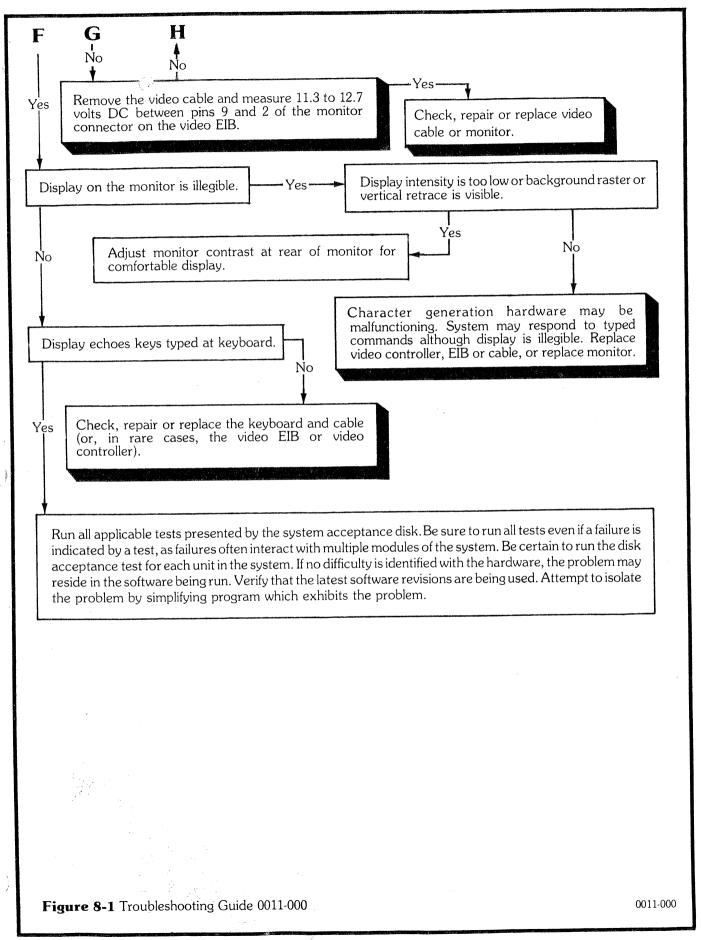
8 BASIC TROUBLESHOOTING

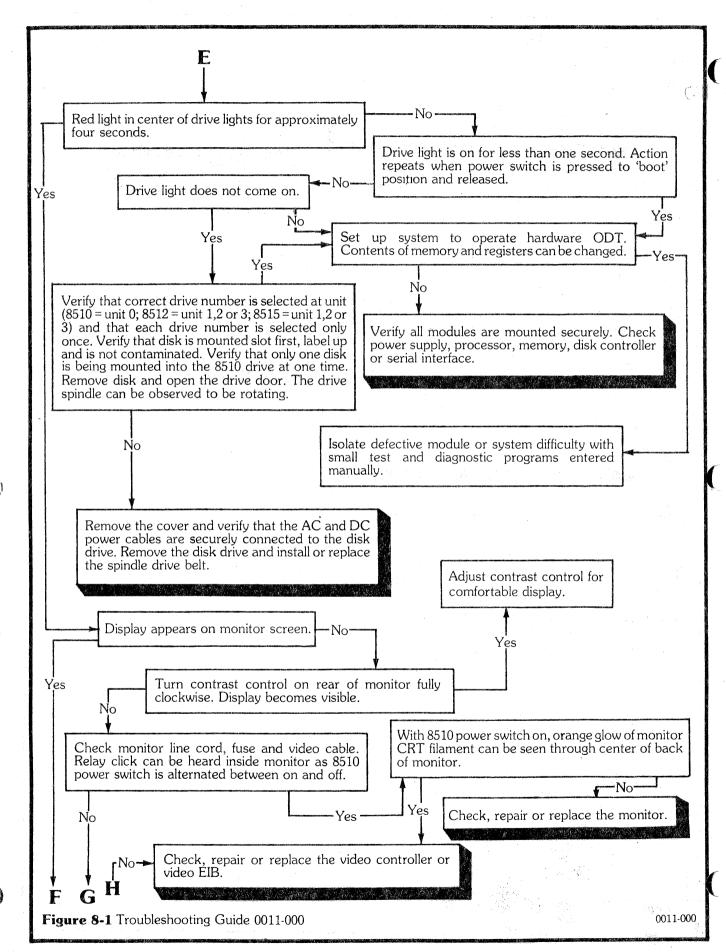
DUE TO THE COMPLEXITY OF THE SYSTEM AND THE REQUIREMENT FOR TEST EQUIPMENT, REPAIRS AND/OR ADJUSTMENTS TO THE VAR—IOUS SUBSYSTEMS OF THE 8510/a SHOULD ONLY BE PERFORMED BY QUALIFIED TECHNICAL PERSONNEL.

The following procedures in Figure 8-1 are intended as a fundamental guide for problems that could prevent the execution of the System Acceptance Tests. These procedures can help to isolate a problem, and in so doing, effect a faster solution when contacting TERAK's Product Service Department. Be sure to include the unit serial numbers in any telephone or letters correspondence with TERAK Product Service.









9 SYSTEM DESCRIPTION

The TERAK 8510/a Graphics Computer is intended for data processing and medium-resolution raster scan graphics applications. The hardware has been partitioned to provide a high degree of flexibility for the user's end-system configuration. This approach provides portability as a result of reasonably light-weight units and an overall economy. User-defined system configurations are made possible by easy access to the processor bus and to peripheral equipment. Each unit (except the 8532-2 keyboard) has its own power supply, so that only low-voltage data and control signals are routed over the 8510/a interconnect cables.

9.1 8510 DATA PROCESSOR

The 8510 is a 16-bit computer that incorporates an LSI-11 processor and an LSI-11 bus-compatible backplane; installed in this backplane are the following four modules:

LSI-11 Processor Module Video Controller-Memory Modules Floppy Diskette Controller Module Asynchronous Serial Interface Module

A disk drive (for 8 in. floppy diskettes) is built into each 8510, 8512 and 8515 unit. Single density, soft sectored, 8 in. IBM 3740 formatted diskettes are supported.

Switch-mode type power supplies convert 100 V, 120 V, 220 V or 140 V RMS, 50 Hz/60Hz ac power to the DC voltages required by the logic and drive circuits.

External Interface Boards (EIB's) plus cables, are used to connect the 8510 to other units such as the 8512/15, 8532-1,-2 and peripheral equipment. The Asynchronous Serial Interface module supports a subset of the ANSI RS-232C standard for Data Terminal Equipment (DTE), Data Communications Equipment (DCE) and 20 mA current loop operation. An optional 16-bit Parallel Interface module, for connecting TTL or DTL devices to the 8510 Bus, is available from TERAK. A composite video signal (horizontal & vertical sync and video) is available at a BNC connector on a Video EIB. This signal, which contains the information presented to the 8532-1 Monitor, can be used to drive other monitors (that contain the necessary separation circuitry).

9.2 8532-1 MONITOR

The 8532-1 Monitor is intended for use as a display device in an 8510/a system. This unit contains a 12 in diagonal monochrome CRT, a video amplifier, horizontal & vertical sync/drive circuits, a power supply and a 2 in speaker. The video, horizontal and vertical synchronization signals produce a 240 visible-

line raster scan of graphics and character dots. The speaker emits software-controlled audible signals.

The Monitor is connected to the Video Controller module by means of a single connector on the 8510 Video External Interface Board (EIB). A Keyboard/Emulator (KBD/EML) switch on the Video EIB (see Figure 2-1) must be in the OFF position to use the Monitor (and the 8532-2 Keyboard) as the system console.

A contrast adjustment is accessible on the rear of the Monitor cabinet. (CW rotation increases the contrast.) The 8532-1 has an ac power cord, but no ac power switch. The 8532-1 power is controlled by the ac power switch on the 8510 unit.

9.3 8532-2 KEYBOARD

The 8532-2 Keyboard is intended for use as an input device in an 8510/a system. This unit contains a matrix of 71 keys and a keystroke to byte-output circuit. There are 138 different byte-output codes, as shown in Figure 9-1. The ASCII set of 96 alphanumeric symbols and 32 control functions are encoded, along with 10 additional codes from the decimal keypad. The keyboard layout is shown in Figure 9-2.

The keyboard circuits provide N-key rollover and automatic repeat on all keys. Capacitive key switches are used to ensure high reliability and long lifetime (approximately 100 million keystrokes). The keyboard layout is similar to modern typewriters.

The 8532-2 is connected to the 8510 Video EIB by means of a single cable. The 8532-2 key codes are input to the Video Controller module. The Video EIB KBD/EML switch determines whether the 8532-2 is used as the system console keyboard (KBD/EML switch set to OFF) or as a distinct peripheral (KBD/EML switch set to ON).

The 8532-2 has no ac power requirement. DC power is supplied to the 8532-2 by the 8510 unit.

9.4 8512 DISK SUBSYSTEM

The 8512 unit is intended for diskette drive expansion in an 8510/a system. This unit contains an 8 in. floppy diskette drive and a power supply for the drive circuits. Connections between the 8510 unit and the 8512 units are made by means of External Interface Boards (EIBs) and cables at the rear of the units. An 8510/a system can have a total of four diskette drives. See Section 4 for a description of drive expansion.

The 8512 units have ac power cords, but no ac power switches. The 8512 power is controlled by the ac power switch on the 8510 unit; that switch must be on to apply power to the 8512 diskette drive circuits.

		- 10.00	***************************************		CONTROL &
KEY	CHAR	UNSHIFT	SHIFT	CONTROL	SHIFT
1	ESC	033	033	033	033
2	TAB	011	011	011	011
3	1!	061	041	061	041
4	2@	062	100	062	034
5	3#	063	043	063	043
6	4\$	064	044	064	044
7	5%	065	045	065	045
8	6^	066	136	066	136
9	7&	067	046	067	046
10	8 *	070	052	070	052
11	9(071	050	071	050
12	0)	060	051	060	051
13	=	055	137	055	137
14	= +	075	053	075	053
15	BACK 5PACE	010	010	010	010
16	Δ	032	032	032	032
17	{7	173	067	041	267
18	}8	175	070	041	270
19	NULL 9	041	071	000	271
20	LINEFEED	012	012	012	012
21	Q	161	121	021	021
22	W	167	127	027	027
23	Е	145	105	005	005
24	R	162	122	022	022
25	T	164	124	024	024
26	Y	171	131	023	023
27	U	165	125	025	025
28	1	151	111	011	011
29	0	157	117	017	017
30	Р	160	120	020	020
31	\ \ \	134	174	134	174
32	11	133	135	036	035
.33	RETURN	015	015	015	015
34	∇	014	014	014	014
35	~ 4	176	064	041	264
36	15	140	065	041	265

KEY CHAR UNSHIFT SHIFT CONTROL CONTROL SHIFT 37 *EOM 6 041 066 031 266 38 DC3 023 023 023 023 39 LOCK
37
38 DC3 023 023 023 023 39 LOCK
39 LOCK Section 1 40 A 141 101 001 001 41 S 163 123 023 023 42 D 144 104 004 004 43 F 146 106 006 006 44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 " 047 042 047 042 51 d 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022
40 A 141 101 001 001 41 S 163 123 023 023 42 D 144 104 004 004 43 F 146 106 006 006 44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 ''' 047 042 047 042 51 4 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3
41 S 163 123 023 023 42 D 144 104 004 004 43 F 146 106 006 006 44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 ''' 047 042 047 042 51 d 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
42 D 144 104 004 004 43 F 146 106 006 006 44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 ;; 073 072 073 072 50 , 047 042 047 042 51 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
43 F 146 106 006 006 44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 ;; 073 072 073 072 50 ' ' 047 042 047 042 51 ⊲ 027 027 027 027 52 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
44 G 147 107 007 007 45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 ''' 047 042 047 042 51 ⊲ 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
45 H 150 110 010 010 46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 ∷ 073 072 073 072 50 ''' 047 042 047 042 51 △ 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
46 J 152 112 012 012 47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 '" 047 042 047 042 51 ⊲ 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
47 K 153 113 013 013 48 L 154 114 014 014 49 :: 073 072 073 072 50 '' 047 042 047 042 51 ⊲ 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
48 L 154 114 014 014 49 ;: 073 072 073 072 50 '' 047 042 047 042 51 □ 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
49 :: 073 072 073 072 50 047 042 047 042 51 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
50 047 042 047 042 51 027 027 027 027 52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
50
52 DC1 1 021 061 041 261 53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
53 DC2 2 022 062 041 262 54 US 3 037 063 037 263
54 US 3 037 063 037 263
-1 03 3 037 003 037 203
55 CONTROL
56 SHIFT
57 Z 172 132 032 032
58 X 170 130 030 030
59 C 143 103 003 003
60 V 166 126 026 026
61 B 142 102 002 002
62 N 156 116 016 016
63 M 155 115 015 015
64 < 054 074 054 074
65 . > 056 076 056 076
66 / ? 057 077 057 077
67 SHIFT
68 🕨 013 013 013 013
69
70 ETX 0 003 060 041 260

Figure 9-1 8532-2 Keyboard Unit Output Codes

0012-000

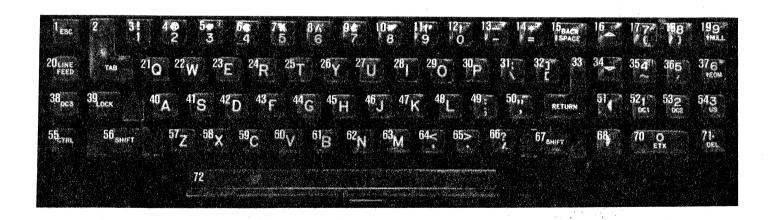


Figure 9-2 8532-2 Keyboard Layout

9.5 8515 EXPANSION SYSTEM

The 8515 unit is intended for LSI-11 bus expansion in an 8510/a system. This unit contains an LSI-11 bus compatible backplane; an 8 in. floppy diskette drive; and power supplies for the backplane and diskette drive. Connections between the 8515 unit and the remainder of an 8510/a system are made by means of External Interface Boards (EIBs) and cables at the rear of the units.

The 8515 backplane accepts TERAK, or other LSI-11 bus compatible modules. As shown in Figure 4-1, one dual-height location of the 8515 backplane is used for bus extension; the other seven are available to the user. See Section 4 for a description of 8510/a system expansion.

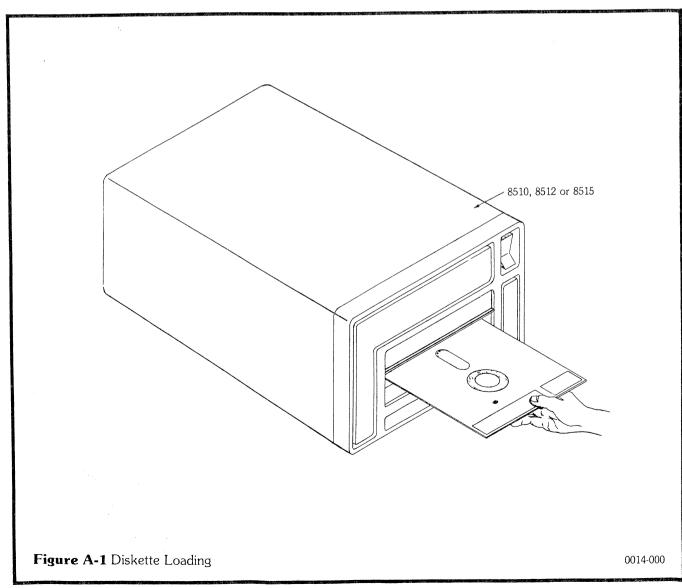
Power to the 8515 unit is controlled by an ac power on/off switch on the unit front panel; this switch must be on to apply power to the backplane and diskette drive circuits. The ac power switch is dual functioning; it can be used to re-boot the operation system by momentarily depressing its upper half.

The diskette drives in the 8510, 8515 an 8512 units are all functionally equivalent.

(5)

APPENDIX A. DISKETTE HANDLING/STORAGE CONSIDERATIONS

The diskettes used with the 8510/a system should be kept in their protective envelopes whenever they are not inserted in a drive unit. Figure A-1 depicts the correct orientation for inserting a diskette into a drive. Figure A-2 describes diskette handling and storage information.



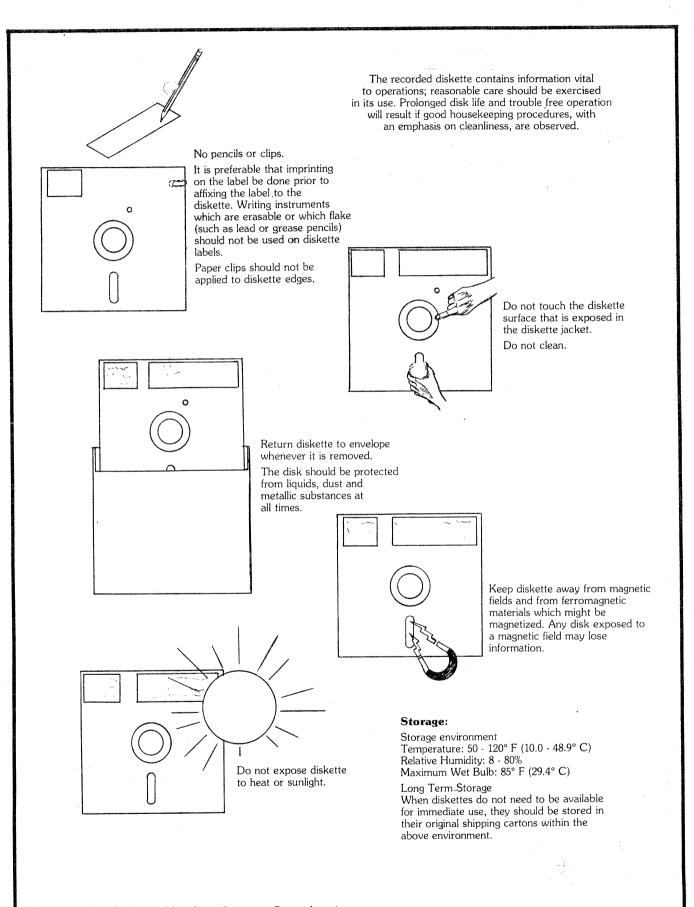


Figure A-2 Diskette Handling/Storage Considerations

0015-000

APPENDIX B. SUGGESTED MAINTENANCE

The filters on the rear of the 8510, 8512 and 8515 units should be cleaned periodically. Generally, it is good practice to clean the filters at least every 30 days.

Turn the ac power off, then remove the filter media by rotating the nylon retaining ring clockwise; this ring is mounted on the fan housing at the rear of the units. Do not remove the 4 screws holding the fan housing in place on the unit.

Vacuum the filter media, then re-install it in the fan housing

With the ac power off, clean the faceplate of the display monitor with a soft cloth that has been wetted with a glass cleaning solution such as Windex, etc.

The diskette drive head should be cleaned at least every six months. Isopropyl alcohol (91% minimum strength) should be used to remove the oxide build-up on the drive head.

Disconnect the ac power and remove the 8510 cover (and 8512/15 covers, if applicable). Remove the 8510 (8515) module retainer bracket(s). (see Figure 5-1), and all the modules in the backplane(s). (There are no modules to remove from the 8512 unit.)

(CAUTION)

If a double-sided (two-headed) disk drive is installed, the read/write heads should not be cleaned or touched. No maintenance is required.

The disk drive head is accessible from the top of the disk drive unit. Lift the head pressure pad (head load button) arm out of the way to expose the drive head. Clean the head surface with the alcohol solution; let the surface air-dry before releasing the pressure pad arm. Use a clean cotton ball or swab; don't use "Q-tip" type swabs because they contain an adhesive that might dissolve in the alcohol and contaminate the head.

Inspect the pressure pad. If it is compacted or worn, with a hard, shiny surface, it should be replaced. A Shugart pressure pad is removed by pinching its bifurcated retaining snaps with a pair of needle-nose pliers and pushing the pad out of the hole in the end of the pressure arm

The unit interior should be inspected at this time for dust and/or debris. Use a small vacuum to clean the unit.

Inspect the disk drive worm gear shaft; a Shugart drive gear must be free of grease and oil.

Before the modules are re-seated in the backplane(s), inspect their edge connectors. Clean off any grease with a degreasing solution (such as alcohol). Remove tarnish with a burnishing tool; do not attempt to clean the contacts with a pencil eraser or any abrasive device, the gold plating will be damaged. Re-seat the modules, replace the cover(s) and re-connect the ac power.

(C) -

APPENDIX C. PROCESSOR REFERENCE GUIDE

C1. GENERAL DESCRIPTION

The 8510 contains an LSI-11 Processor module with a DEC KEV 11 Extended Arithmetic Chip installed. This module contains a processor chip set, clock and bus control circuits.

The module is built around a set of four n-channel MOS LSI chips (data chip, control chip plus two ROMs) that comprise a microprogrammed processor. The microcoded ROMs (microms) contain the code to emulate the basic PDP-11/40 instruction set plus debug routines that function as a console (control panel) emulator. A third microm socket contains the KEV 11 chip that extends the instruction set with fixed and floating point (FIS/EIS) arithmetic instructions.

The LSI-11 Processor Features:

Word (16-bit) or Byte (8-bit) Processing
Stack Processing
Direct Addressing of 32 K-words (64 K-bytes)
Direct Memory Access (DMA)
Single and Double Operand Instructions
Priority-Structured I/O System
Vectored Interrupts
Eight General-Purpose (16-bit) Registers
Power-Fail and Auto Restart Function
Asynchronous Operation
Fixed Point (16-bit) and Floating Point (32-bit)
Arithmetic

Address and data transfers between the processor chip set and the remainder of the 8510/a system are via 16 data/address lines. These lines, plus control and timing signal lines, comprise the 8510/a Bus (which is identical to the DEC Q-Bus).

All modules connected to this bus structure receive the same interface signals. Bus control and data lines are open-collector lines which are asserted when low. All data and most control lines are bidirectional. All transactions on the bus are asynchronous. The processor uses the following bus signals: 16 multiplexed data/address lines, 6 data transfer control lines, 6 system control lines, and 5 interrupt and direct memory access (DMA)control lines.

Interrupt and DMA are implemented with two daisy-chained grant signals which provide a priority-structured I/O system. The highest priority device is the module electrically located closest to the microcomputer module. A device passes grant signals to lower priority devices only when it is not requesting service.

The 8510/a bus provides a vectored interrupt interface for any device. Device polling is not required in processing interrupt requests. When an interrupting device receives a grant, the device passes to the processor an interrupt vector which points to a new processor status word and the starting address of an interrupt service routine for the device.

The 8510/a backplane contains all Q-Bus wiring plus power distribution wiring to all device locations.

Both 16-bit address and 8-bit bytes or 16-bit data words are multiplexed over 16 data/address lines. During a programmed data transfer, the processor will assert an address on the bus for a fixed time. After the address time has been completed, the processor initiates the programmed input or output data transfer. The actual data transfer is asynchronous and requires a reply from the addressed device; bus synchronization and control signals provide this function.

With bidirectional and asynchronous communications on the LSI-11 bus, devices can send, receive, and exchange data at their own rates. The bidirectional nature of the bus allows utilization of common bus interfaces for different devices and simplifies the interface design.

Communication between two devices on the bus is in the form of a master-slave relationship. At any point in time, there is, at most one device that has control of the bus. This controlling device is termed the "bus master." The master device controls the bus when communicating with another device on the bus, termed the "slave." A typical example of this relationship is the processor, as master, fetching an instruction from memory (which is always a slave). Another example is a DMA device interface, as master, transferring data to memory, as a slave. Bus master control is dynamic. The bus arbitrator is in the processor module; it may pass bus control to a DMA device. The DMA device, as bus master, could then communicate with memory (always a slave) without processor intervention.

Since the bus is used by the processor and all I/O devices, a hardware priority structure must determine which device becomes bus master when more than one device requests control of the bus. Every device on the bus which is capable of becoming bus master is assigned a priority according to its electrical position on the bus. The device closest to the processor has the highest priority.

Data transfers on the bus are asynchronous so that communication is independent of the physical bus length and the response time of the slave device. The asynchronous operation between bus master and slave devices precludes the need for synchronizing bus transactions with clock signals. Thus, each device is allowed to operate at the maximum speed.

C2.SYSTEM MICROCOMPUTER

The microcomputer '(processor) of the 8510/a controls the time allocation for bus communiquion and performs arithmetic and logic operations and instruction decoding. It contains multiple highspeed, general-purpose registers which can be used as accumulators, address pointers, index registers, and for other specialized functions. The processor does both single and double operand addressing and handles both 16-bit word and 8-bit byte data. The bus permits DMA data transfers directly between I/O devices and memory without disturbing the processor registers.

The microcomputer architecture provides eight 16-bit general-purpose registers that can perform a variety of functions. These registers can serve as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can be from one general register to another, from one memory location or between memory locations or a device register to another, or between memory locations or a device register and a general register. The eight 16-bit general registers (R0 through R7) are identified in Figure C-1.

Control of the Contro		
General r		
Registers	R0	
	R1	
	R2	
	R3	1
	R4	
	R5	
	R6 (SP)	
	Stack Pointer	
	R7 (PC)	
	Program Counter	
Figure C-1	General Register Identification	0016-000

Registers R6 and R7 in the microcomputer are dedicated. R6 serves as the Stack pointer (SP) and contains the location (address) of the last entry in the stack. Register R7 serves as the processor Program Counter (PC) and contains the address of the next instruction to be executed. Both are normally used for addressing purposes only and not as an accumulator. Register operations are internal to the processor and do not require bus cycles (except for instruction fetch); all memory and peripheral device data transfers do require bus cycles and longer execution time. Thus, general registers used for processor operations result in faster execution times.

The bus cycle required for memory and device references are described below. The bus cycles (with respect to the processor) are:

Mnemo	onic Function	Note
DATI	Data word transfer input	Equivalent to Read Operation
DATIO	Data word transfer input followed by word transfer output	Equivalent to Read- Modify-Write
DATIOB	Data word transfer input followed by byte transfer output	Equivalent to Read- Modify-Write
DATO	Data word transfer output	Equivalent to Write Operation
DATOB	Data byte transfer output	Equivalent to Write Operation

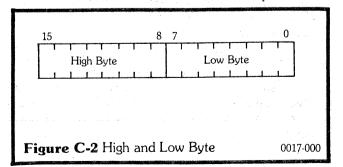
Every processor instruction requires one or more bus cycles. The DATI bus cycle fetches an instruction from the location addressed by the Program Counter (R7). If no further operands are referenced in memory or in an I/O device, no additional bus cycles are required for instruction execution. If memory or a device is referenced, one or more additional bus cycles are required.

Note this distinction between interrupts and DMA operations: interrupts, which may change the state of the processor, can occur only between processor instructions; DMA operations can occur between individual bus cycles since these operations do not change the state of the processor.

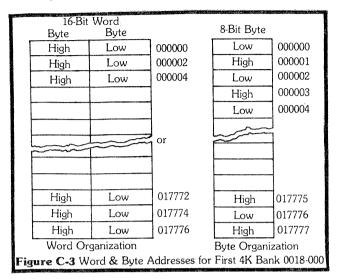
The maximum direct address space of the processor is 32K 16-bit words. The 8510/a memory locations and peripheral device registers are addressed in the same manner. The upper 4096 addresses (28K-32K) are reserved, by convention, for peripheral device addressing. For instance, TERAK uses the space from 28 K through 30 K for character display & generation. The use of this address space is described in the other Appendices for the 8510 modules.

Certain memory locations have been reserved, by convention, for interrupt and trap handling. Addresses from 0 to 376 octal are reserved for trap and device interrupt vector locations. Several of these are reserved for system (processor initiated) traps.

A data word is divided into a high byte and a low byte as shown in Figure C-2. Word addresses are always evennumbered. Byte addresses can be either even or odd

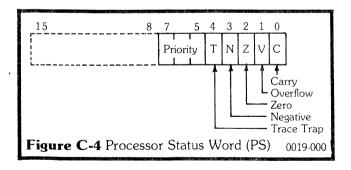


numbered. Low bytes are stored at even-numbered memory locations and high bytes at odd-numbered memory locations. Thus, it is convenient to view the memory as shown in Figure C-3.



C2.1 PROCESSOR STATUS WORD

The Processor Status Word (PS) contains information on the current processor status. This information includes the current processor priority, the condition codes describing the arithmetic or logic results of the last instruction, and an indicator for detecting the execution of an instruction to be trapped during program debugging. The PS word format is shown in Figure C-4. Certain instructions allow programmed manipulation of condition code bits and loading or storing the PS.



C2.1.1 Interrupt Priority Bit

The processor operates with interrupt priority PS bit 7 asserted (1) or cleared (0). When PS bit 7 = 1, an external device cannot interrupt the processor. The processor must be operating with PS bit 7 = 0 for any device request to be serviced by the processor. The processor services interrupts at one priority level. PS bits 6 & 5 may be loaded or stored, but have no effect upon recognition of device interrupt requests.

C2.1.2 Condition Codes

The condition codes contain information on the result of the last CPU operation. The bits are interpretted as follows, (the bits are set after execution of arithmetic or logical, single operand or double operand instructions):

Z = 1, if the result was zero

N = 1, if the result was negative

C = 1, if the operation resulted in a carry from the MSB (most significant bit) or a 1 was shifted from the MSB or LSB (least significant bit)

V = 1, If the operation resulted in an arithmetic overflow

C2.1.3 Trap (T) Bit

The processor can only set the trap bit (T) by an RTT or RTI instruction, with a corresponding bit set or cleared in the stack. When the T bit is set, a processor trap will occur through vector location 14 at completion of the current (in the case of an RTI) or next (in the case of an RTT) instruction execution, and the T bit will be cleared. The T bit causes the same sequence as execution of a BPT instruction. This T bit is especially useful in debugging programs, since it allows programs to be single-instruction stepped under control of another program.

C2.2 Addressing Modes

Data stored in memory must be accessed and manipulated. Data handling is specified by a processor instruction (MOV, ADD, etc.), which usually indicates:

The function (operation code).

A general-purpose register is to be used when locating the source operand and/or a general-purpose register to be used when locating the destination operand.

An addressing mode (to specify how the selected register(s) is/are to be used).

A large portion of the data handled by a computer is usually structured (in character strings, arrays, lists, etc.). The processor addressing modes provide for efficient and flexible handling of structured data.

The general registers may be used with an instruction in any of the following ways:

As accumulators. The data to be manipulated resides within the register.

As pointers. The contents of the register is the address of the operand, rather than the operand itself.

As pointers which automatically step through memory locations. Automatically stepping forward through consecutive locations is known as autoincrement addressing; automatically stepping backwards is known as autodecrement addressing. These modes are particularly useful for processing tabular or array data.

As index registers. In this instance, the contents of the register and the word following the instruction are summed to produce the address of the operand. This allows easy access to variable entries in a list.

An important processor feature, which should be considered in conjunction with the addressing modes, is the register arrangement:

Six general-purpose registers (R0-R5)

A hardware Stack Pointer (SP) register (R6)

A Program Counter (PC) register (R7)

Registers R0 through R5 are not dedicated to any specific function; their use is determined by the instruction that is decoded:

They can be used for operand storage. For example, contents of two registers can be added and stored in another register.

They can contain the address of an operand or serve as pointers to the address of an operand.

They can be used for the autoincrement or autodecrement features.

They can be used as index registers for convenient data and program access.

The processor also has instruction addressing mode combinations that facilitate temporary data storage structures. This can be used for convenient handling of data that must be accessed frequently. This is known as stack manipulation. The register used to keep track of stack manipulation is known as the stack pointer (SP). Any register can be used as a "stack pointer" under program control; however, certain instructions associated with subroutine linkage and interrupt service automatically use Register R6 as a "hardware stack pointer". For this reason R6 is frequently referred to as the "SP".

The stack pointer (SP) keeps track of the latest entry on the stack.

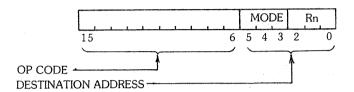
The stack pointer moves down as items are added to the stack and moves up as items are removed. Therefore, it always points to the top to the stack.

The hardware stack is used during trap or interrupt handling to store information allowing the processor to return to the main program.

Register R7 is used by the processor as its program counter (PC). It is recommended that R7 not be used as a stack pointer or program accumulator. Whenever an instruction is fetched from memory, the program counter is automatically incremented by two to point to the next instruction word.

C2.2.1 Single Operand Instructions

The instruction format for all single operand instructions (such as clear, increment, test) is:



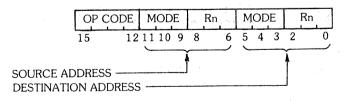
Bits 15 through 6 specify the operation code that defines the type of instruction to be executed.

Bits 5 through 0 form a six-bit field called the destination address field. This consists of two subfields:

- a) Bits 0 through 2 specify which of the eight general-purpose registers is to be referenced by this instruction word.
- b) Bits 3 through 5 specify how the selected register will be used (address mode). Bit 3 is set to indicate deferred (indirect) addressing.

C2.2.2 Double Operand Instructions

Operations which imply two operands (such as add, subtract, move, and compare) are handled by instructions that specify two addresses. The first operand is called the source operand, the second the destination operand. Bit assignments in the source and destination address fields may specify different modes and different registers. The instruction format for most double operand instructions is:



The source address field is used to select the source operand, the first operand. The destination is used similarly, and locates the second operand and the result. For example, the instruction ADD A, B adds the

contents (source operand) of location A to the contents (destination operand) of location B. After execution B will contain the result of the addition and the contents of A will be unchanged.

C2.2.3 Notes On Addressing

)

Examples in this section use the following sample instructions. A complete listing of the processor instructions is located in section C2.3.

Mnem	onic Description	Octal Code
CLR	Clear (zero the specified destination)	0050DD
CLRB	Clear byte (zero the byte in the specified destination)	1050DD
INC	Increment (add 1 to contents of destination)	0552DD
INCB	Increment byte (add 1 to the contents of destination byte)	1052DD
COM	Complement (replace the contents of the destination by their logical complement; each 0 bit is set and each 1 bit is cleared).	0051DD
COMB	Complement byte (replace the contents of the destination byte by their logical complement; each 0 bit is set and each 1 bit is cleared).	1051DD
ADD	Add (add source operand to destination operand and store the result at destination address)	06SSDD
SS =	destination field (6 bits) source field (6 bits) contents of	

Addressing mode refers to the mechanism selected by the mode field of an address to obtain the address of a data item. Note that for most instructions, either source or destination addresses may use any of the eight addressing modes. The eight addressing modes are grouped as:

Direct Addressing- The operand is the content of the selected register.

Deferred (Indirect) Addressing- The contents of the selected register is the address of the operand.

In the following sections, instructions and addressing modes are represented symbolically using the syntax of standard assembler programs. Using symbolic representation and assemblers frees the programmer from concern over the location of specific bits in instructions.

C2.2.4 Direct Addressing

The following table summarizes the four basic modes used with direct addressing.

DIRECT MODES

Мо	de Name	Assembler Syntax	Function
0		Rn Regis	ster contains operand
	INSTRUCTION	OPERA	ND
2	Autoincremer		r is used as a pointer uential data then ented
	Register	ADDRESS	OPERAND
			+ 2 FOR WORD, + 1 FOR BYTE
4	Autodecreme		ter is decremented hen used as a pointer.
INS	STRUCTION	ADDRESS -2 FOR WOI -1 FOR BYT	RD, OPERAND
6	Index >	produc Neither	K is added to (Rn) to e address of operand. X nor (Rn) are
		modifie	d.

C2.2.4.1 Register Mode (Mode 0)

OPR Rn

With register mode, any of the general registers may be used as simple accumulators and the operand is contained in the selected register. Since they are hardware registers, within the processor, the general registers operate at high-speeds and provide speed advantages when used for operating on frequently-accessed variables.

Registers are typically referred to by name as R0, R1, R2, R3, R4, R5, R6 and R7. However R6 and R7 are also referred to as SP and PC, respectively.

Register Mode Examples

(all numbers in octal)

Symbolic

1. INC R3 005203 Increment Add one to the contents of general Operation: register 3. 1 0 0 0 0 REGISTER 15 4 3 OP CODE (INC (0052)) R0 DESTINATION FIELD R1 R2 R3 R4 R5 R6 (SP) R7 (PC)

Octal Code

2. ADD R2, R4 060204

Add

Instruction Name

Operation: Add the contents of R2 to the contents of R4

	BEFORE	_	AFTER
R2	000002	R2	000002
R4	000004	R4	000006

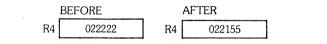
3. COMB R4

105104

Complement Byte

Operation:

One's complement bits 0-7 (byte) in R4. (When general registers are used, byte instructions only operate on bits 0-7; i.e., byte 0 of the register.)



C2.2.4.2 Autoincrement Mode (Mode 2)

OPR (Rn)+

This mode provides for automatic stepping of a pointer through sequential elements of a table of operands. It assumes the contents of the selected general register to be the address of the operand. Contents of registers are stepped (by one for bytes, by two for words, always by two for R6 and R7) to address the next sequential location. The autoincrement mode is especially useful for array processing and stack processing. It will access an element of a table and then step the pointer to address the next operand in the table. Although most useful for table handling, this mode is completely general and may be used for a variety of purposes.

Autoincrement Mode Examples

Symbolic

Octal Code

Instruction Name

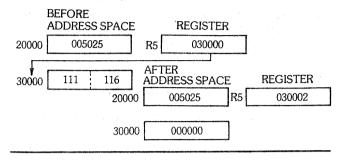
1. CLR (R5)+

005025

Clear *

Oper jon:

Use contents of R5 as the address of the operand. Clear selected operand and then increment the contents of R5 by two.



2. CLRB (R5)+

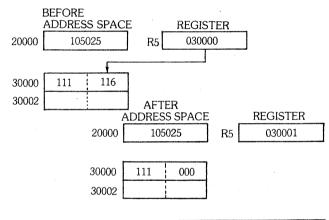
Operation:

105025

Clear Byte

Use contents of R5 as the address of the operand. Clear selected byte operand and

operand. Clear selected byte operand and then increment the contents of R5 by one.

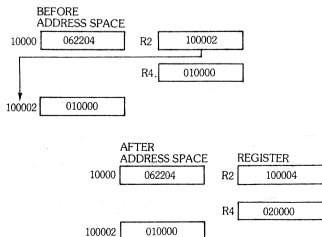


3. ADD (R2)+.R4 062204

Add

Operation:

The contents of R2 are used as the address of the operand which is added to the contents of R4. R2 is then incremented by two.



C.2.2.4.3 Autodecrement Mode (Mode 4)

OPR -(Rn)

This mode is useful for processing data in a list in reverse direction. The contents of the selected general register are decremented (by two for word instructions, by one for bute instructions) and then used as the address of the operand. The choice of postincrement, predecrement features for the LSI-11 were not arbitrary decisions, but were intended to facilitate hardware/software stack operations.

Autodecrement Mode Examples

Symbolic	(Octal Code	Instruction Name
1. INC -(R0)	005240		Increment
Operation:	two ar	The contents of R0 are decremented by two and used as the address of the operand. The operand is incremented by one.	
BEFORE ADDRES	S SPACE	REGIST	ERS
1000 005	240	R0 017	7776
17774 0000	00		
		AFTER ADDRESS SP	ACE REGISTER
	1000	005240	R0 017774
	17774	000001	

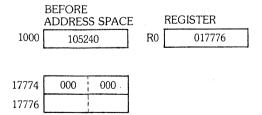
2. INCB -(R0)

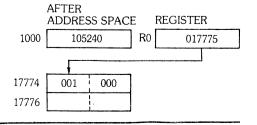
105240

Increment Byte

Operation:

The contents of R0 are decremented by one then used as the address of the operand. The operand byte is increased by one.





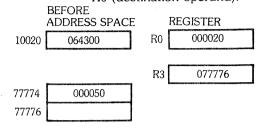
Instruction Name Symbolic **Octal Code** 3. ADD -(R3).R0

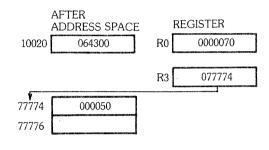
064300

Add

Operation:

The contents of R3 are decremented by 2 then used as a pointer to an operand (source) which is added to the contents of R0 (destination operand).





C2.2.4.4 Index Mode (Mode 6)

OPR X (Rn)

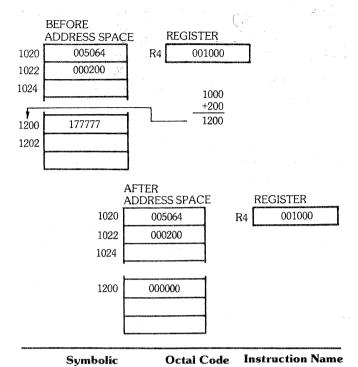
The contents of the seleced general register, and an index word following the instruction word, are summed to form the address of the operand. The contents of the selected register may be used as a base for calculating a series of addresses, thus allowing random access to elements of data structures. The selected register can then be modified by program to access data in the table. Index addressing instructions are of the form OPR X(Rn) where X is the indexed word and is located in the memory location following the instruction word and Rn is the selected general register.

Index Mode Examples

M 11 04	CA PIGUE EM		
	Symbolic	Octal Code	Instruction Name
1.	CLR 200(R4)	005064 000200	Clear
	(C	ontinued on next	page)

Operation:

The address of the operand is determined by adding 200 to the contents of R4. The operand location is then cleared.

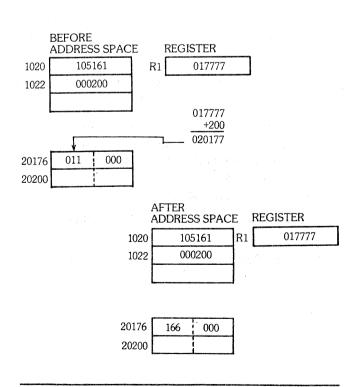


2. COMB 200(R1)

105161 Complement Byte 000200

Operation:

The contents of a location which is determined by adding 200 to the contents of R1 are one's complemented (i.e., logically complemented).



 Symbolic
 Octal Code
 Instruction Name

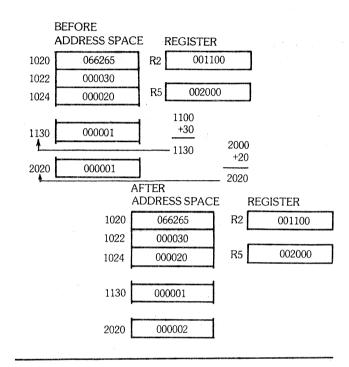
 3. ADD 30(R2),20(R5) 066265
 Add

 000030
 000030

000020

Operation:

The contents of a location which is determined by adding 30 to the contents of R2 are added to the contents of a location which is determined by adding 20 to the contents of R5. The result is stored at the destination address, i.e., 20(R5).

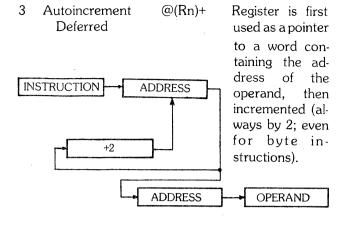


C2.2.5 Deferred (Indirect) Addressing

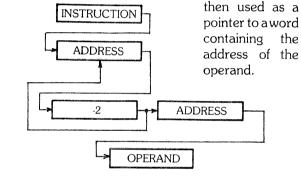
The four basic modes may also be used with deferred addressing. Whereas in the register mode the operand is the contents of the selected register, in the register deferred mode the contents of the selected register is the address of the operand.

In the three other deferred modes, the contents of the register select the address of the operand rather than the operand itself. These modes are therefore used when a table consists of addresses rather than operands. Assembler syntax for indicating deferred addressing is "@" (or "()" when this is not ambiguous). The following table summarizes the deferred versions of the basic modes:

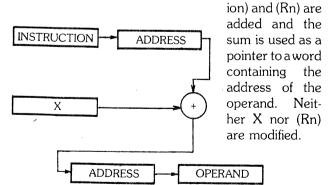
Mode	Name	Assembler Syntax	Function
1	Register Deferred	@Rn or (Rn)	Register contains the address of the operand.
INSTRU	CTION	- ADDRESS	→ OPERAND



Autodecrement @-(Rn) Register is de-Deferred cremented ways by two; even for byte instructions) and

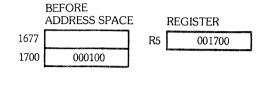


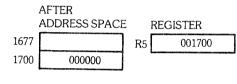
Index Deferred @X(Rn)



The following examples illustrate the deferred modes. Register Deferred Mode Example (Mode 1)

Sym	bolic	Octal (Cod	de Instruction Name	
CLR	@ R5	0050	15	Clear	
Operation:	The clear		of	location specified in R5	





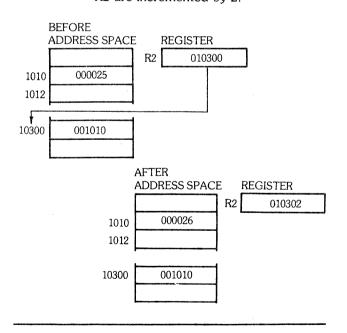
I... - 4---- - 4:- -- - NI-----

Autoincrement Deferred Mode Example (Mode 3)

Sumbalia

Sym	OOIIC	Octai Code	instruction Name
INC@	(R2)+	005232	Increment
Operation:	The co	ontents of R2 are	e used as the address
	of the address of the		operand.
	Operand is increase		by one. Contents of
	R2 are	e incremented b	ou 2.

Ontal Cada



(al-

Value X (stored

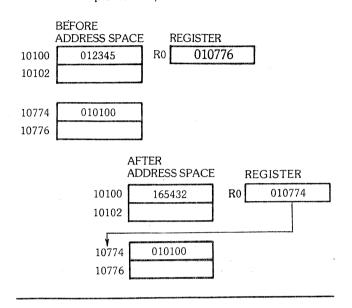
in a word following the instrucAutodecrement Deferred Mode Example (Mode 5)

Symbolic Octal Code Complement COM@-(R0) 005150

Operation:

The contents of R0 are decremented by two and then used as the address of the address of the operand. Operand is one's, complemented. (i.e., logically com-

plemented).

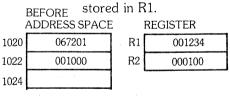


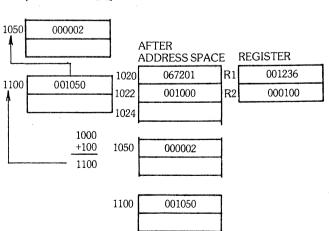
Index Deferred Mode Example (Mode 7)

Symbolic	Octal Code	Instruction Name
ADD@1000(R2),R1	067201	Add
	001000	*

Operation:

1000 and contents of R2 are summed to produce the address of the address of the source operand the contents of which are added to contents of R1; the result is





C2.2.6 Use Of The PC As A General Register

Although Register 7 is a general purpose register, it doubles in function as the Program Counter for the processor. Whenever the processor uses the program counter to acquire a word from memory, the program counter is automatically incremented by two to contain the address of the next word of the instruction being executed or the address of the next instruction to be executed. (When the program uses the PC to locate byte data, the PC is still incremented by two.)

The PC responds to all the standard LSI-11 addressing modes. However, there are four of these modes with which the PC can provide advantages for handling position independent code and unstructured data. When utilizing the PC these modes are termed immediate, absolute (or immediate deferred), relative and relative deferred, and are summarized below:

Mode	Name	Assembler Syntax	Function
2	Immediate	#n	Operand follows instruction.
3	Absolute	@#A	Absolute Address of operand follows instruction.
6	Relative	Α	Relative Address (index value) follows the instruction.
7	Relative Deferred	@A	Index value (stored in the word following the instruction) is the relative address for the address of the operand.

When a standard program is available for different users, it often is helpful to be able to load it into different areas of memory and run it there. The relocation of a program can be accomplished very efficiently through the use of Position Independent Code (PIC) which is written by using the PC addressing modes. If an instruction and its operands are moved in such a way that the relative distance between them is not altered, the same offset relative to the PC can be used in all positions in memory. Immediate, Relative and Relative-deferred modes support this type of PIC. If an operand resides at a constant address in memory (e.g. I/O device registers or vectors) then a PIC reference to the operand would use absolute addressing.

The PC also greatly facilitates the handling of unstructured data. This is particularly true of the immediate and relative modes.

The PC may also be used as a general register as the destination of an instruction. This is useful to effect an execution of an instruction at a calculated address. For example:

ADD R5, PC

would cause the next instruction to be fetched according to the contents of R5.

C2.2.6.1 Immediate Mode

OPR #n.DD

immediate mode is equivalent to using the autoincrement mode with the PC. It provides time improvements for accessing constant operands by including the constant in the memory location immediately following the instruction word.

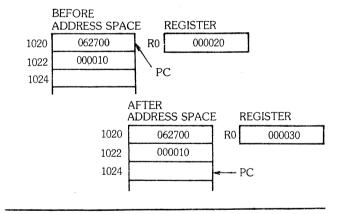
Immediate Mode Example:

Symbolic	Octal Code	Instruction Name
ADD#10,R0	062700	Add
	000010	

Operation:

)

The value 10 is located in the second word of the instruction and is added to the contents of R0. Just before this instruction is fetched and executed, the PC points to the first word of the instruction. The processor fetches the first word and increments the PC by two. The source operand mode is 27 (autoincrement the PC). Thus, the PC is used as a pointer to fetch the operand (the second word of the instruction) before being incremented by two to point to the next instruction.



C2.2.6.2 Absolute Addressing

OPR @#A

This mode is the equivalent of immediate deferred or autoincrement deferred using the PC. The contents of the location following the instruction are taken as the address of the operand. Immediate data is interpreted as an absolute address (i.e., an address that remains constant no matter where in memory the assembled instruction is executed).

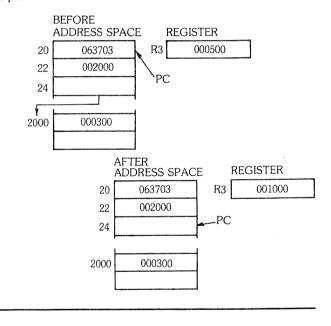
Absolute Mode Examples:

Instruction Name Octal Code Symbolic 005037 Clear 1. CLR@#1100 001100 Clear the contents of location 1100. Operation: AFTER ADDRESS SPACE **BEFORE** ADDRESS SPACE 20 005037 22 001100 20 005037 PC 001100 24 22 000000 1100 1100 177777 1102 1102

 Symbolic
 Octal Code

 2. ADD@#2000,R3
 063703 002000

Operation: Add contents of location 2000 to R3.



C2.2.6.3 Relative Addressing

OPR A or OPR X (PC)

where X is the location of A relative to the instruction.

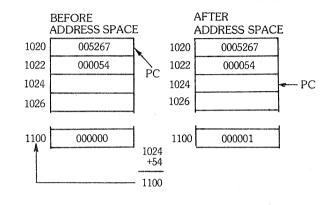
This mode is assembled as index mode using R7. The base of the address calculation, which is stored in the second or third word of the instruction, is not the address of the operand, but the number which, when added to the (PC), becomes the address of the operand. This mode is useful for writing position independent code since the location referenced is always fixed relative to the PC. When instructions are to be relocated, the operand is moved by the same amount.

Relative Addressing Example:

Symbolic	Octal Code	Instruction Name
INC A	005267	Increment
	000054	

Operation:

To increment location A, contents of memory location immediately following instruction word are added to (PC) to produce address A. Contents of A are increased by one.



C2.2.6.4 Relative Deferred Addressing

OPR@A or OPR@X(PC),

where X is location containing address of A, relative to the instruction.

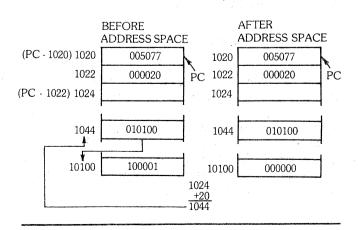
This mode is similar to the relative mode, except that the second word of the instruction, when added to the PC, contains the address of the address of the operand, rather than the address of the operand.

Relative Deferred Mode Example:

Symbolic	Octal Code	Instruction Name
CLR@A	005077 000020	Clear

Operation:

Add second word of instruction to update PC to produce address of address of operand. Clear operand.



C2.2.7 Use Of Stack Pointer As General Register

To processor stack pointer (SP, Register 6) is in most cases the general register used for the stack operations related to program nesting. Autodecrement with Register 6 "pushes" data on to the stack and autoincrement with Register 6 "pops" data off the stack. Index mode with SP permits random access of items on the stack. Since the SP is used by the processor for interrupt handling, it has a special attribute: autoincrements and autodecrements are always done in steps of two. Byte operations using the SP in this way leave odd addresses unmodified.

When running in a typical environment, a program is usually not concerned with the actual address contained in SP; it is assumed to point to valid memory. If interrupts are active, SP is used asynchronously to store and retrieve the state of the processor. Thus, a program should not use memory below SP unless it has been pushed. Here, SP can be used as a general register to allocate memory on the stack:

ADD #6, SP

would be equivalent to:

CLR -(SP)

CLR -(SP)

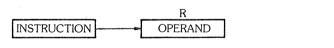
CLR -(SP)

except that the contents of the memory is undefined in the first case.

C2.2.8 Summary Of Addressing Modes C2.2.8.1 General Register Addressing

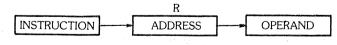
R is a general register, 0 to 7 (R) is the contents of that register

Mode 0 Register OPR R R contains operand



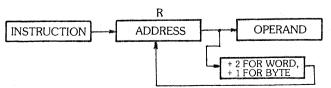
Mode 1 Register deferred OPR (R)

R contains address



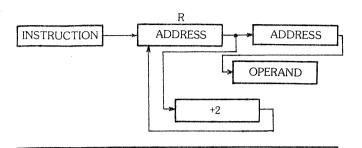
Mode 2 Autoincrement OPR (R)+

R contains address, then increment (R)



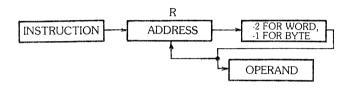
Mode 3 Autoincrement deferred OPR @(R)+

R contains address of address, then increment (R) by 2



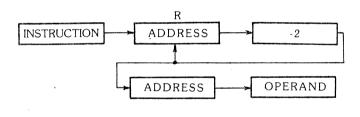
Mode 4 Autodecrement OPR -(R)

Decrement (R), then R contains address

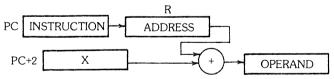


Mode 5 Autodecrement deferred OPR @-(R)

Decrement (R) by 2, then R contains address of address

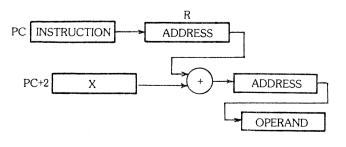


Mode 6 Index OPR X(R) (R) + X is address



Mode 7 Index deferred OPR @X(R)

(R) + X is address of address

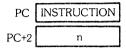


C2.2.8.2 Program Counter Addressing

Register = 7

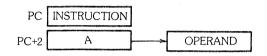
Mode 2 Immediate OPR #n

Operand n follows instruction

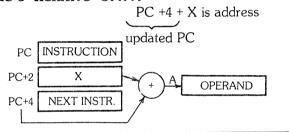


Mode 3 Absolute OPR @#A

Address A follows instruction

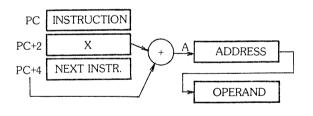


Mode 6 Relative OPR A



Mode 7 Relative deferred OPR @A

PC + 4 + X is address of address updated PC



C2.3 DATA STRUCTURES

The processor instruction set places conventions upon the representation of data. These hardware supported data structures are grouped as logical, integer and floating point. From these structures, a program may build more complex structures such as sets, multiple precision integers, arrays and records.

C2.3.1 Logical Data

Certain instructions manipulate the data in their operands as bits. For these, instructions, bytes, or words are used as sets of 8 or sets of 16 bits, with each bit independent of its neighbors. Bits are usually numbered 0 through 7 for the low to high order bits of a byte, and 0 through 15 for the low to high order bits of a word. See Figure C-2. For example:

BIC #100, R2

Will clear (set to zero) bit number 6 in R2.

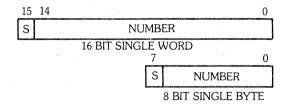
Logical arithmetic is supported by these instructions: BIC(B), BIS(B), BIT(B), ROR(B), ROL(B), XOR, COM(B), BEQ, BNE.

C2.3.2 Integer Data

Integers are stored in bytes or words and are interpreted as being signed or unsigned.

C2.3.2.1 Signed Integer Data

When signed numbers are represented, two's complement notation is used. This allows the most significant bit (bit 7 for bytes, bit 15 for words) to be interpreted as the sign of the number. Operand formats are:



Where: S is the sign bit. S = 0 for positive quantities S = 1 for negative quantities; number is in 2's complement notation

The range of values which may be represented by signed integer words are:

integer wor	ds are:		•
		077777 077776	(32767 decimal)
(Greater)	positive	•	
	zero	000001 000000 177777 177776	(-1 decimal)
		•	
(Lesser)	negative	100001 100000	(-32768 decimal)

The range of values which may be represented by signed integer bytes are:

(Greater)	positive	177 176	(127 decimal)
	zero	001 000 377 376	(-1 decimal)
(Lesser)	negative	201 200	(-128 decimal)

Signed integer arithmetic is supported by these instructions:

CLR(B)	INC(B)	DEC(B)	NEG(B)	TST(B)
ROR(B)	ROL(B)	ASR(B)	ASL(B)	SXT
CMP(B)	ADD	SUB	MUL.	DIV
ASH	BNE	BEQ	BPL	BMI
BVC	BVS	BCC	BCS	BGE
BLT	BGT	BLE		

and by the automatic sign extension of bytes to words performed by MOVB when the destination operand is a register.

C2.3.2.2 Unsigned Integer Data

When unsigned integers are represented, the data is considered to be positive. This representation is useful for counting, and for arithmetic involving memory addresses. The MSB of the data is interpretted as the MSB of a positive number. The range of values which may be represented by unsigned interger words are:

(Higher)	positive	177777 177776	(65,535 decimal)
		100001 100000 077777	(32,768 decimal) (32,767 decimal)
(Lower)	zero	000001 000000	

The range of values which may be represented by unsigned integer bytes are:

_			
(,)		377 376	(255 decimal)
(Higher)	positive		4
		201 200 177	(128 decimal) (127 decimal)
		•	
(Lower)	zero	001 000	

)

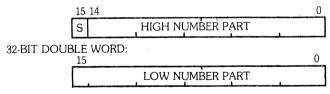
Unsigned integer arithmetic is supported by these instructions:

CLR(B)	INC(B)	DEC(B)	
ROR(B)	ROL(B)	ASR(B)	ASL(B)
CMP(B)	ADD	SUB	ASH
BNE	BEQ	BCC	BCS
BHI	BLOS	BHIS	BLO

Unsigned integers may be operated upon as signed integers, provided that data is precluded from exceeding the signed positive range.

C2.3.2.3 Multiple Precision Integer Data

Integer data may be represented by multiple words or bytes to support extended number ranges. Arithmetic on these operands typically requires multiple processor instructions per operation. The double precision operand format is:



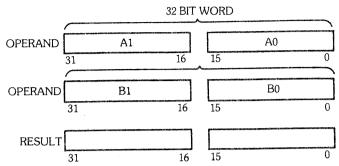
S is the sign bit.

S = 0 for positive quantities

S = 1 for negative quantities; number is in 2's complement notation

The range of values which may be thus represented is: -2,147,419,112 to 2,147,419,111.

For example two 16-bit words may be combined into a 32-bit double precision word and added or subtracted as shown below:



The addition of -1 and -1 could be performed as follows:

-1 = 3777777777

(R1) = 177777 (R2) = 177777 (R3) = 177777 (R4) = 177777

ADD R1,R2 ADC R3 ADD R4,R3

- 1. After (R1) and (R2) are added, 1 is loaded into the C bit
- 2. ADC instruction adds C bit to (R3); (R3) = 0
- 3. (R3) and (R4) are added
- 4. Result is 3777777776 or -2

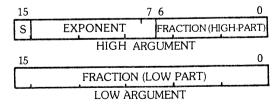
Double precision arithmetic is supported by these instructions:

ADC SBC SXT CMP ADD SUB MUL DIV ASHC Further precision can be supported by appropriate algorithms using these instructions.

C2.3.3 Floating Point Data

Floating point allows the representation of a very large number range at the expense of precision. Standard precision floating point data uses two words, providing a 24 bit mantissa (plus sign), and a 7 bit exponent (plus sign). The number format is essentially a sign and magnitude representation.

The operand format is:



 $S=\mbox{sign of fraction; 0 for positive, 1 for negative}$ Exponent = 8 bits for the exponent, in excess (200)s notation Fraction = 23 bits plus 1 hidden bit (all numbers are assumed to be normalized)

Fraction

The binary radix point is to the left (in front of bit 6 of the High Argument), so that the value of the fraction is always less than 1 in magnitude. Normalization would always cause the first bit after the radix point to be a 1, such that the fractional value would be between ½ and 1. Therefore, this bit can be understood and not be represented directly, to achieve an extra 1 bit of resolution.

The first bit to the right of the radix point (hidden bit) is always a 1. The next bit for the fraction is taken from bit 6 of the High Argument. The result of a Floating Point operation is always rounded away from zero, increasing the absolute value of the number.

Exponent

The 8-bit exponent field (bits 14 to 7) allow exponent values between -128 and +127. Since an excess $(200)_8$ or $(128)_{10}$ number system is used, the correspondence between actual values and coded representation is as follows:

Actual Value	Representation				
Decimal	Octal		Bina	ry	
+127	377	11	111	111	
+1	201	10	000	001	
0	200	10	000	000	
-1	177	01	111	111	
-128	000	00	000	000	

Example of a Number:

$$+(12)_{10} = +(1100)_{2}$$
 $= +(2^{4})_{10} \text{ X } (.11)_{2} \quad [16 \text{ X } (\frac{1}{2} + \frac{1}{4}) = 12]$

S Exponent
Fraction

Representation.

Hidden bit is a 1
Radix point is understood

Floating point arithmetic is supported by these instructions:

FADD FSUB FMUL FDIV BCC BCS BVC BVS BPL BMI

C2.4 INSTRUCTIONS

This section specifies the set of instructions executed by the 8510/a processor. Instructions are encoded in one, two or three words, depending on the addressing modes used. Most are straight-forward, but a few are clarified in the following notes section. To aid reference use, the instructions are listed in alphabetic order.

C2.4.1 Subroutine Instructions (JSR, RTS)

The subroutine call and return instructions provide for automatic (lesting of subroutines, reentrancy, and multiple entry points. Subroutines may call other subroutines (or indeed themselves) to any level of nesting without making special provision for storage of return addresses at each level of subroutine call. The subroutine calling mechanism does not modify any fixed location in memory, thus providing for reentrancy. This allows one copy of a subroutine to be shared among several interrupting processes.

The mechanism which makes this possible is the stack. A subroutine call pushes one word on the stack which contains either the return address or the saved value of a register which carries the return address. Subroutine calls and returns have no effect upon condition codes.

C2. 4. 2 TRAP INSTRUCTIONS, TRAP BIT, AND INTERRUPTS

C2. 4. 2. 1 Trap Instructions

Trap instructions provide for calls to emulators, I/O monitors, debugging packages and user-defined interpreters. A trap is effectively an interrupt generated by software. When a trap occurs, the contents of the current processor Status Word (PS), and then the current Program Counter (PC), are pushed onto the processor stack and replaced by the contents of a two-word vector containing a new PC (at vector address) and new PS (at vector address +2). The return sequence from a trap involves executing an RTI or RTT instruction which restores the old PC and old PS by popping them from the stack. Trap instruction vectors are located at permanently assigned, fixed addresses.

The TRAP instructions are BPT, IOT, EMT and TRAP. The first two have a fixed encoding. EMT and TRAP allow the program to encode the low order byte of the instruction with an order code.

C2. 4. 2. 2 Trap Bit

The TRAP bit (see Section C2. 1. 3) causes the same effect as execution of a BPT instruction.

The Trace Trap is enabled by the Trap (T) bit of the PS and causes processor traps at the end of instruction execution. The instruction that is executed after the instruction that set the T-bit will proceed to completion and then trap through the trap vector at address 14. Usually, the Trace Trap is a system debugging aid and is transparent to the program.

NOTE

The Trap bit of the PS can only be set indirectly by executing a RTI or RTT instruction with the desired PS on the stack. (RTT is used for tracing)

The following are special cases of the T-bit and are detailed in subsequent paragraphs.

- 1. The traced instruction cleared the T-bit.
- 2. The traced instruction set the T-bit.
- 3. The traced instruction caused an instruction trap.
- 4. The traced instruction caused a bus error trap.
- 5. The processor was interrupted between the time the T-bit was set and the fetching of the instruction that was to be traced.
- 6. The traced instruction was a WAIT.
- 7. The traced instruction was a HALT.
- 8. The traced instruction was a Return from Interrupt...

NOTE

The traced instruction is the instruction after the one that set the T-bit.

An instruction that cleared the T-bit — Upon fetching the traced Instruction, an internal flag, the trace flag, was set. The trap will still occur at the end of execution of this instruction. The status word on the stack, however, will have a clear T-bit.

An instruction that set the T-bit — Since the T-bit was already set, setting it again has no effect. The trap will occur.

An instruction that caused an Instruction Trap — The instruction trap is performed and the entire routine for the service trap is executed. If the service routine exists with an RTI or in any other way restores the stacked status word, the T-bit is set again, the instruction following the traced instruction is executed and, unless it is one of the special cases noted previously, a trace trap occurs.

An instruction that caused a Bus Error Trap— This is treated as an Instruction Trap. The only difference is that the error service is not as likely to exit with an RTI, so that the trace trap may not occur.

Note that interrupts may be acknowledged immediately after the loading of the new PC and PS at the trap vector location. To lock out all interrupts, the PS at the trap vector should set Bit 7.

A WAIT — T-bit trap is not honored during a wait.

A HALT — The processor halts. The PC points to the next instruction to be executed. The trap will occur immediately following execution resumption.

A Return from Interrupt — The return from interrupt instruction either clears or sets the T-bit. If the T-bit was set and RTT is the traced instruction, the trap is delayed until completion of the next instruction.

C2.4.2.3 Other Trap Events

Other Traps can occur due to external or internal events — from power failure to floating point error.

Reserved Instruction Traps — These are caused by attempts to execute instruction codes reserved for future processor expansion (reserved instructions) or instructions with illegal addressing modes (illegal instructions). Order codes not corresponding to any of the instructions described are considered to be reserved instructions. JMP and JSR with register mode destinations are illegal instructions, and trap to vector address 4. Reserved instructions trap to vector address 10.

Bus Error Traps — Bus Error Traps are time-out errors; attempts to reference addresses on the bus that have made no response within a certain length of time. In general, these are caused by attempts to reference non-existent memory, and attempts to reference non-existent peripheral devices. Bus error traps cause processor traps through the trap vector address 4.

Power Failure Trap — Occurs when AC power fail signal is received while processor is in run mode. Trap vector for power failure is location 24 and 26. Trap will occur if an RTI instruction is executed in power fail service routine.

Floating Point Trap — If an error is detected during execution of an instruction of one of the floating point instructions (FADD, FSUB, FMUL, FDIV), a trap will occur through vector address 244. Traps will occur due to overflow, underflow, or divide-by-zero conditions. Floating point traps have an important exception to the trap sequence:

The condition codes in the PS that caused a trap to 244 will be set in the PS that was used while the FIS instruction was being executed. Following the trap, this PS will be pushed onto the stack. The stack must be examined following a trap to retrieve the PS and determine the reason for the trap.

Some operating systems will simply perform an RTI instruction returning the error codes to the user program following the erroring instruction. In calculations where error checking between floating point instructions involves an intolerable penalty to speed, the floating point trap allows error service that is not in line with the calculation.

Interrupts are traps caused by a device (BUS) request. The processor can ignore or enable such interrupts by setting or clearing the priority bit in the PS (see Section C2.1.3).

The vector for any interrupt is controlled by the requesting device. Vectors for 8510/a devices are specified in other sections of this document.

Trap Priorities — In case of internal and external multiple processor trap conditions, occurring simultaneously, the following order of priorities is observed (from high policy):

Bus Error Trap Memory Refresh Instruction Traps Trace Trap Power Fail Trap Halt Line

Cannot be disabled

Vertical Retrace Start Interrupt
Device (Bus) Interrupt Request

Can be disabled by priority bit in PS

If a bus error is caused by the trap process handling instruction traps, trace traps, or a previous bus error, the processor is halted. This is called a double bus error.

Note that memory refresh is supported by the microcode of the processor, so that no trap service code is required. Real time applications should be aware of the periodic interruption of the processor (130 microseconds, every 1600 microseconds) if refresh is active.

C2.4.3 Floating Point Arithmetic: FADD, FSUB, FMUL, FDIV

The floating point instructions are unique in that no general addressing modes are supported. A stack (not necessarily SP) pointer is used to specify the operand fields. The assembler format is:

OPR R

where R may be 0 through 6.

Errors will cause a floating point trap. Therefore, trap service code should be present before executing floating instructions.

C2.4.3.1 Registers

There are no pre-assigned registers for the Floating Point option. A general purpose register is used as a pointer to specify a stack address. The contents of the register are used to locate the operands and answer for the Floating Point operations as follows:

(R) = High B argument address

(R)+2 = Low B argument address

(R)+4 = High A argument address

(R)+6 = Low A argument address

After the Floating Point operation, the answer is stored on the stack as follows:

(R)+4 = address for High part of answer

(R)+6 = address for Low part of answer

Where (R) is the original contents of the general register used.

After execution of the instruction, the general register will point to the High answer, at (R)+4.

Condition Codes

Condition codes are set or cleared as shown in the Instruction Descriptions, in the next part of this section. If a trap occurs as a function of a Floating Instruction, the condition codes in the stack are as follows:

	V	N	С	Z
Overflow	1	0	0	0
Underflow	1	1	0	0
Divide by 0	1	1	1	0

Notes

- 1. If the actual value of the exponent is equal to -128, meaning a total value (including the fraction) of less than 2⁻¹²⁸, a trap will occur.
- 2. Following a trap, the general register will be unaltered, as will (R), (R)+2, (R)+4, and (R)+6.
- Following a trap, the error condition codes are placed in the stack (see Sect. C2.4.2). These could be returned in the PS if the trap service code simply executed an RTI instruction.

C2.4.3.2 Interrupts

A Floating Point instruction will be aborted if an interrupt request is issued before the instruction is near completion. The Program Counter will point to the aborted Floating Point instruction so that the Interrupt will look transparent. This avoids excessive interrupt latency due to long floating point execution times.

C2.4.4 Fixed Point Arithmetic: MUL, DIV, ASHC

These instructions are unique in that one operand is a register or a register pair. When a register pair is specified, the register must be even numbered, (e.g. R0, R2, or R4). The low order portion of double precision fixed point number is held in the register +1, (e.g. R1, R3, or R5), and the high order portion (including sign bit) is held in the register.

MUL, when operating on an odd register, produces a 16 bit product. This is useful for single precision fixed point arithmetic. In this case, only the specific register is altered. When the destination register is even, the contents of the associated odd register is contained in the register pair (low order in odd, high order in even). This is the correct format for a subsequent DIV using the same register pair.

DIV must operate upon an even register as a destination operand. The dividend must be a 32 bit quantity. Note that if a 16 bit dividend is loaded into the low order (odd) register of the pair, it must be sign extended into the high order (even) register. The SXT (for unknow sign) or CLR (for positive numbers) instructions are useful for this. After division, the 16 bit quotient is contained in the even register, and the 16 bit remainder is contained in the odd register. This is the correct format for a subsequent MUL using the same register pair.

MUL and DIV operate only on signed integers. Unsigned integers must be restricted in range to avoid errors.

C2.4.5 Branch Instructions

These instructions cause a branch to a location defined by the sum of the offset (multiplied by 2) and the current contents of the Program Counter if:

- a) the branch instruction is unconditional
- b) it is conditional and the conditions are met after testing the condition codes (NZVC)

The offset is the number of words from the current contents of the PC forward or backward. Note that the current contents of the PC point to the word following the branch instruction.

Although the offset expresses a byte address the PC is expressed in words. The offset is automatically multiplied by two and sign extended to express words before it is added to the PC. Bit 7 is the sign of the offset.

If Bit 7 is set, the offset is negative and the branch is done in the backward direction (toward lower memory). Similarly, if it is not set, the offset is positive and the branch is done in the forward direction (toward higher memory).

The 8-bit offset allows branching in the backward direction by 2008 words (400 bytes) from the current PC, and in the forward direction by 1778 words (376 bytes) from the current PC. Branching longer distances can be accomplished either by cascading branch instructions or by using JMP instructions.

The assembler handles address arithmetic for the user and computes and assembles the proper offset field for branch instructions in the form:

Bxx loc

Where "Bxx" is the branch instruction and "loc" is the address to which the branch is to be made. The assembler gives an error indication in the instruction if the permissible branch range is exceeded. Branch instructions have no effect on condition codes. Conditional branch instructions where the branch condition is not met, are treated as NO OP's.

The same arithmetic instruction (e.g. CMP) may be used to operate upon signed or unsigned integer data. The distinction between the two types is made in the

conditional branch instruction used to interpret the resulting condition codes. BGE, BLT, BGT and BLE test the results of operations in which the operands were considered as signed (2's complement) integers. BHI, BLOS, BHIS and BLO test the results of operations in which the operands were considered as unsigned integers (e.g. addresses).

C2.4.6 Legend

The specification for each instruction includes the mnemonic, octal code, a symbolic notation describing its execution and the effect on the condition codes, a description, special comments, and examples.

Mnemonic: This is indicated at the beginning of each specification. When the word instruction has a byte equivalent, the byte mnemonic is also shown.

Instruction Format: The diagrams in Section C2.4.7 show the octal op code, the binary op code, and bit assignment. (Note that in byte instructions the most significant bit (bit 15) is always a 1.)

Symbols:

() = contents of

SS or src = source address

DD or dst = destination address

loc = location

← = becomes

= "is popped from stack"

= "is pushed onto stack"

 Λ = boolean AND

v = boolean OR

₩ = exclusive OR

~ = boolean not

Reg or R = register

B = byte

= 0 for word

1 for byte

, = concatenated

l,h = double word register

1 = low order

h = high order

C2.4.7 Instruction Formats

The following formats include all instructions used in the processor. Refer to individual instructions for more detailed information.

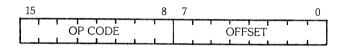
1. Single Operand Group (CLR, CLRB, COM, COMB, INC, INCB DEC, DECB, NEG, NEGB, ADC, ADCB, SBC, SBCB, TST, ASTB, ROR, RORB, ROL, ROLB, ASR, ASRB, ASL, ASLB, JMP, SWAB MFPS, MTPS, SXT, XOR)

15	6	5	0
OP CODE		DD (SS)	

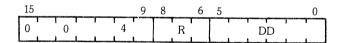
2. Double Operand Group (BIT, BITB, BIC, BICB, BIS, BISB, ADD, SUB, MOV, MOVB, CMP, CMPB)

15 12	11 6	5 0
OP CODE	SS	DD

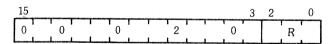
- 3. Program Control Group
 - a. Branch (all branch instructions)



b. Jump To Subroutine (JSR)



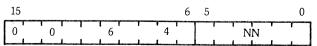
c. Subroutine Return (RTS)



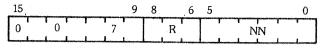
d. Traps (break point, IOT, EMT, TRAP BPT)



e. Mark (MARK)

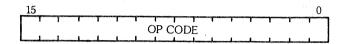


f. Subtract I and branch (if = 0)(SOB)



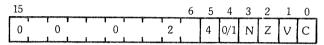
4. Operate Group

(HALT, WAIT, RTI, RESET, RTT, NOP)

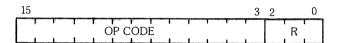


5. Condition Codes Operators

(all condition code instructions)



6. Fixed and Floating Point (optional EIS/FIS)(FADD,
Arithmetic FSUB, FMUL, FDIV,
MUL, DIV, ASH, ASHC)



C2.4.8 List of Instructions by Function

PROGR	AM CONTROL		Mnemonic	Instruction	Op Code
Mnemoni	c Instruction	Op Code	Condition	on Code Operators	
Branch	R	or ase Code	CLC	clear C	000241
BR	branch (unconditional)		CLV	clear V	
BNE	branch if not equal (to zero)	001000	CLZ	clear Z	000242
BEQ	branch if equal (to zero)		CLN	clear N	000244
BPL	branch if plus		CCC	clear all CC bits	000250
BMI	branch if minus		SEC	set C	000257
BVC	branch if overflow is clear		SEV	set V	000261
BVS	branch if overflow is set	102400	SEZ	set Z	000262
BCC	branch if carry is clear	103000	SEN		
BCS	branch if carry is set		SCC	set Nset all CC bits	000270 000277
Signed (Conditional Branch			OPERAND	
BGE	branch is greater than or equa	al	General	OPERAND	
	(to zero)		CLR(B)	clear dst	■050DD
BLT	branch if less than (zero)	002400	CLN(B) COM(B)	complement dst	■050DD
BGT	branch if greater than (zero)	003000	INC(B)	increment dst	■051DD
BLE	branch if less than or equal (to zero	o) 003400	DEC(B)	decrement dst	■052DD
Unsiane	d Conditional Branch		NEG(B)	negate dst	■053DD
BHI	branch if higher	. 101000	TST(B)	test dst	■054DD
BLOS	branch if lower or same		101(D)	test ust	= 007DD
BHIS	branch if higher or same		Shift &	Rotate	
BLO	branch if lower		ASR(B)	arithmetic shift right	■062DD
Jumn &	Subroutine	. 100.00	ASL(B)	arithmetic shift left	■063DD
JMP	jump	. 0001DD	ROR(B)	rotate right	■060DD
JSR	jump to subroutine		ROL(B)	rotate left	■061DD
RTS	return from subroutine		SWAB	swap bytes	0003DD
MARK	mark		ASH	arithmetic shift	072RSS
SOB	subtract one and branch (if \neq 0)		Multiple	Precision	
	• • • • • • • • • • • • • • • • • • •	07,71,00	ADC(B)	add carry	■055DD
Tran &	Interrupt		SBC(B)	subtract carry	■056DD
EMT	emulator trap 104000	104277	SXT	sign extend	0067DD
TRAP	trap		ASHC	arithmetic shift combined	073RSS
BPT	breakpoint trap		PS Wor	d Operators	
IOT	input/output trap		MFPS	move byte from PS	1067DD
RTI	return from interrupt		MTPS	move byte to PS	1064SS
RTT	return from interrupt			E OPERAND	100100
			General		
Miscella	neous		MOV(B)	move source to destination	■1SSDD
HALT	halt	000000	CMP(B)	compare src to dst	■1SSDD ■2SSDD
WAIT	wait for interrupt	000001	ADD	add src to dst	06SSDD
RESET	reset external bus	000005	SUB	subtract src from dst	16SSDD
NOP	no operation	000240	MUL	multiply	070RSS
Reserve	d Instructions		DIV	divide	071RSS

00021R 00022

Mnemonic	Instruction	Op Code
Logical		
BIT(B)	bit test	3SSDD
BIC(B)	bit clear	4SSDD
BIS(B)	bit set	■5SSDD
XOR	exclusive or	074RDD
Floating	Point	
FADD	floating point add	07500R
FSUB	floating point subtract	07501R
FMUL	floating point multiply	07502R
FDIV	floating point divide	07503R

C2.4.9 List of Instructions by Operation Code

OP Code Mnem	onic OP Code	Mnemonic	OP Code	Mnemonic	OP Code	Mnemonic
00 00 00 00 HA 00 00 01 W. 00 00 02 RT 00 00 03 BF 00 00 04 IO 00 00 05 RE 00 00 06 RT 00 00 07 OU 00 01 DD JN 00 02 0R RT 00 02 10 RT 00 02 10 RT 00 02 40 NC 00 02 41 CCC 00 03 DD SV 00 04 XXX BI 00 04 XXX BI 00 10 XXX BI 00 10 XXX BI 00 10 XXX BI 00 24 XXX BI 00 24 XXX BI 00 30 XXX BI 00 34 XXX BI	ALT 00 50 DD AIT 00 51 DD T 00 52 DD T 00 53 DD T 00 54 DD T 00 56 DD T 00 56 DD T 00 57 DD T 00 60 DD T 00 61 DD TS 00 62 DD TS 00 63 DD TS 00 67 DD TS 00 77 77 TO T	CLR COM INC DEC NEG ADC SBC TST ROR ROL ASR ASL MARK SXT (unused) MOV CMP BIT BIC BIS ADD MUL DIV ASH ASH ASHC	07 50 0R 07 50 1R 07 50 2R 07 50 3R 07 50 40 07 67 77 07 7R NN 10 00 XXX 10 10 XXX 10 14 XXX 10 20 XXX 10 30 XXX 10 30 XXX 10 34 XXX 10 34 XXX 10 34 XXX 10 34 XXX	FADD FSUB FMUL FDIV (unused) SOB BPL BMI BHI BLOS BVC BVS BCC BHIS BCS BLO EMT	10 50 DD 10 51 DD 10 52 DD 10 53 DD 10 54 DD 10 55 DD 10 56 DD 10 56 DD 10 57 DD 10 60 DD 10 61 DD 10 62 DD 10 63 DD 10 64 SS 10 67 DD 11 SS DD 12 SS DD 13 SS DD 14 SS DD 15 SS DD 16 SS DD 17 00 00 17 77 77	CLRB COMB INCB DECB NEGB ADCB SBCB TSTB RORB ROLB ASRB ASLB MTPS MFPS MOVB CMPB BITB BICB BISB SUB RESERVED
00 4R DD JS	SR 07 4R DD	XOR				

C2.4.10 Alphabetic Listing of Instructions

ADC **ADCB**

add carry

8055DD

Operation:

(dst) \leftarrow (dst) + (C bit)

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise

V: set if (dst) was 077777 and (C)

was 1; cleared otherwise

C: set if (dst) was 177777 and (C)

was 1; cleared otherwise

Description:

Adds the contents of the C-bit into the destination. This permits the carry from the addition of the low-order words to be carried into

the high-order result.

Byte: Same

Example:

Double precision addition may be

done with the following

instruction sequence.

ADD A0, B0; add low-order parts

ADC B1 ; add carry into high-

order

ADD A1,B1; add high order

parts

ADD

add src to dst

06SSDD

Operation:

(dst) (src) + (dst)

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise

V: set if there was arithmetic overflow as a result of the operation; that is both operands were of the same sign and the result was of the opposite sign; cleared

otherwise.

C: set if there was a carry from the most significant bit of the result;

cleared otherwise.

Description:

Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not Two's complement affected.

addition is performed.

Note: There is no equivalent byte

mode.

Example:

Add to Register: ADD 20,R0

Add to memory: ADD R1, XXX

Add register to register: ADD

R1. R2

Add memory to memory: ADD @

17750.XXX

XXX is a programmer-defined

mnemonic for a memory location.

ASH

shift arithmetically

072RSS

Operation:

R-R Shifted arithmetically NN

places to right of left

Where NN = low order 6 bits of

source i.e. r-r x 2(SS)

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise

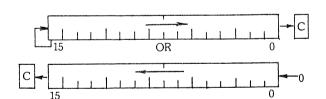
V: set if sign of register changed during shift; cleared otherwise. C: loaded from last bit shifted out

of register

Description:

The contents of the register are shifted right or left the number of times specified by the shift count. The shift count is taken as the low order 6 bits of the source operand. This number ranges from -32 to +31. Negative is a right shift and positive is a left

shift.



6 LSB of source Action is general register

shift left 31 places 011111 shift left 1 place 000001 shift right 1 place 111111 shift right 32 places 100000

Example: ASH R0, R3

After Before (R3) = 012340(R3) = 001234(R0) = 000003(R0) = 000003

C23

ASHC

arithmetic shift combined

073RSS

Operation:

R, Rv1 - R, Rv1 The double word is shifted NN places to the right or left, where NN = low order six bits of source.

i.e. $(rv1,r) \leftarrow (rv1,r) \times 2^{(SS)}$

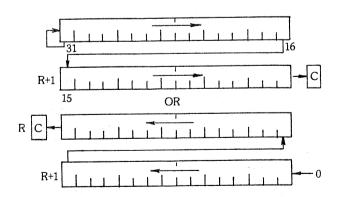
Condition Codes: N: set if result < 0; cleared otherwise

> Z: set if result = 0: cleared otherwise

> V: set if sign bit changes during the shift; cleared otherwise. C: loaded with high order bit when left shift; loaded with low order bit when right shift (loaded with the last bit shifted out of the 32-bit operand)

Description:

The contents of the register and the register OR'ed with one are treated as one 32 bit word, R + 1 (bits 0-15) and R (bits 16-31) are shifted right or left the number of times specified by the shift count. The shift count is taken as the low order 6 bits of the source operand. This number ranges from -32 to +31. Negative is a right shift and positive is a left shift. When the register chosen is an odd number the register and the register OR'ed with one are the same. In this case the right shift becomes a rotate (for up to a shift of 16). The 16 bit word is rotated right the number of bits specified by the shift count.



ASL ASIB

arithmetic shift left

■063DD

Operation:

(dst) - (dst) shifted one place to

the left.

Condition Codes: N: set if high-order bit of the result

is set (result < 0): cleared

otherwise.

Z: set if the result = 0; cleared

otherwise.

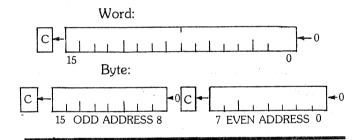
V: loaded with the exclusive OR of the N-bit and C-bit (as set by the completion of the shift operation).

C: loaded with the high-order bit

of the destination.

Description:

Word: Shifts all bits of the destination left one place. Bit 0 is loaded with a 0. The C-bit of the status word is loaded from the most significant bit of the destination. ASL performs a signed multiplication of the destination by 2 with overflow indication.



ASR ASRB

arithmetic shift right

■062DD

Operation:

 $(dst) \leftarrow (dst)$ shifted one place to

the right.

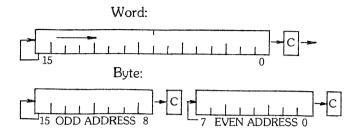
Condition Codes: N: set if the high-order bit of the result is set (result < 0); cleared otherwise.

> Z: set if the result = 0; cleared otherwise.

> V: loaded from the exclusive OR of the N-bit and C-bit (as set by completion of the shift operation) C: loaded from low-order bit of the destination.

Description:

Word: Shifts all bits of the destination right one place. Bit 15 is reproduced. The C-bit is loaded from bit 0 of the destination. ASR performs signed division of the destination by two.



BCC

branch if carry is clear

103000 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if C = 0

Condition Codes: Unaffected

Description:

Tests the state of the C-bit and causes a branch if C is clear. BCC is the complementary operation to BCS.

BCS

)

branch if carry is set

103400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if C = 1

Condition Codes: Unaffected

Description:

Tests the state of the C-bit and causes a branch if C is set. It is used to test for a carry in the result of a previous operation.

BEQ

branch if equal (to zero)

001400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if Z = 1

Condition Codes: Unaffected

Description:

Tests the state of the Z-bit and causes a branch if Z is set. As an example, it is used to test equality following a CMP operation, to test that no bits set in the destination were also set in the source following a BIT operation, and generally, to test that the result of the previous operation was zero.

Example:

CMP A.B BEQ C

; compare A and B ; branch if they are

egual

will branch to C if A=B (A-B=0)

and the sequence

ADD A.B ; add A to B

BEQ C ; branch if the result=0

will branch to C if A + B = 0.

BGF

branch if greater

002000 Plus offset

than or equal (to zero)

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if $N \forall V = 0$

Condition Codes: Unaffected

Description:

Causes a branch if N and V are either both clear or both set. BGF. is the complementary operation to BLT. Thus BGE will always cause a branch when it follows an operation that caused addition of two positive numbers. BGE will also cause a branch on a zero

result.

BGT

branch if greater than (zero)

003000 Plus offset

Operation:

 $PC \leftarrow PC + (2 \text{ x offset}) \text{ if } Zv(N \forall V)$

= 0

Condition Code: Unaffected

Description:

Operation of BGT is smilar to BGE, except BGT will not cause a

branch on a zero result.

BHI

branch if higher

101000 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if C = 0 and

Z = 0

Condition Codes: Unaffected

Description:

Causes a branch if the previous operation caused neither a carry nor a zero result. This will happen in comparison (CMP) operations as long as the source has a higher

unsigned value than the des-

tination.

BHIS

branch if higher or same

103000 Plus offset

Operation:

 $PC \leftarrow F + (2 \times \text{offset}) \text{ if } C = 0$

Condition Codes: Unaffected

Description:

BHIS is the same instruction as BCC. This mnemonic is included

only for convenience.

BIC RICB

bit clear

4SSDD

Operation:

 $(dst) \leftarrow \sim (src) \wedge (dst)$

Condition Codes: N: set if high order bit of result set;

cleared otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: not affected

Description:

Clears each bit in the destination that corresponds to a set bit in the source. The original contents of

the destination are lost. The contents of the source are

unaffected.

Example:

BIC R3, R4

Before After

(R3) = 001234(R3) = 001234(R4) = 000101(R4) = 001111

> NZVC NZVC 1 1 1 1 0001

Before: (R3) = 0 000 001 010 011 100

 $(R4) = 0\ 000\ 001\ 001\ 001\ 001$ After: (R4) = 0 000 000 001 000 001

BIS BISB

bit set

■5SSDD

Operation:

 $(dst) \leftarrow (src) v (dst)$

Condition Codes: N: set if high-order bit of result

set, cleared otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: not affected Description:

Performs "Inclusive OR" operation between the source and

destination operands and leaves the result at the destination address: that is, corresponding bits set in the source are set in the destination. The contents of the

destination are lost.

Example:

BIS RO, R1

Before After (R0) = 001234(R0) = 001234

(R1) = 001111(R1) = 001335

NZVC NZVC 0 0 0 0 0 0 0 0

Before: $(R0) = 0\ 000\ 001\ 010\ 011\ 100$

(R1) = 0 000 001 001 001 001

After: $(R1) = 0\ 000\ 001\ 011\ 011\ 101$

BIT RITR

bit test

■3SSDD

Operation:

(src) A (dst)

Condition Codes: N: set if high-order bit or result

set; cleared otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: not affected

Description:

Performs logical "and" comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination operands are affected. The BIT instruction may be used to test whether any of the corresponding bits that are set in the destination are also set in the source or whether all corresponding bits set in the destination are clear in the source.

Example:

BIT #30 R3

test bits 3 and 4 of R3 to see if both

are off.

R3 = 0 000 000 000 011 000 After Before NZVC NZVC

0 0 0 1 1 1 1 1

BLE

branch if less than or equal

(to zero)

003400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset) \text{ if } Z \vee (N \vee V)$

Condition Codes: Unaffected

Description:

Operation is similar to BLT but in addition will cause a branch if the result of the previous operation

was zero.

BLO

branch if lower

103400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if C = 1

Condition Codes: Unaffected

Description:

BLO is same instruction as BCS. This mnemonic is included only

for convenience.

BLOS

branch if lower or same

101400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset) \text{ if } C \vee Z = 1$

Condition Codes: Unaffected

Description:

Causes a branch if the previous operation caused either a carry or a zero result. BLOS is the complementary operation to BHI. The branch will occur in comparison operations as long as the source is equal to, or has a lower unsigned value than the destination.

BLT

branch if less than (zero)

002400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \text{ x offset}) \text{ if } N \neq V = 1$

Condition Codes: Unaffected

Description:

Causes a branch if the "Exclusive OR" of the N and V bits are 1. Thus BLT will always branch following an operation that added two negative numbers, even if overflow occurred. In particular, BLT will always cause a branch if it follows a CMP instruction operating on a negative source

and a positive destination (even if overflow occurred). Further, BLT will never cause a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT will not cause a branch if the result of the previous operation was zero (without overflow).

BMI

branch if minus

100400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset) \text{ if } N = 1$

Condition Codes: Unaffected

Description:

Tests the state of the N-bit and causes a branch if N is set. It is used to test the sign (most significant bit) of the result of the previous operation), branching if negative. BMI is the complementary function of BPL.

BNE

branch if not equal (to zero)

001000 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset) \text{ if } Z = 0$

Condition Codes: Unaffected

Description:

Tests the state of the Z-bit and causes a branch if the Z-bit is clear. BNE is the complementary operation to BEQ. It is used to test inequality following a CMP, to test that some bits set in the destination were also in the source, following a BIT operation, and generally, to test that the result of the previous operation was not zero.

Example:

CMP A.B

; compare A and B

BNE C

; branch if they are not equal

will branch to C if $A \neq B$

and the sequence

ADD A.B

; add A to B

BNE C

: Branch if the result is not

equal to 0

will branch to C if $A + B \neq 0$

BPL

branch if plus

100000 Plus offset

Operation:

 $PC \leftarrow PC + \sqrt{2} \times \text{ offset}$) if N = 0

Condition Codes: Unaffected

Description:

Tests the state of the N-bit and

causes a branch if N is clear, (positive result). BPL is the complementary operation of BMI.

BPT

breakpoint trap

000003

Operation:

(SP) -- PS

↓(SP) ← PC PC - (14)

PS - (16)

Condition Codes: N: loaded from trap vector

Z: loaded from trap vector

V: loaded from trap vector

C: loaded from trap vector

Description:

Performs a trap sequence with a trap vector address of 14. Used to call debugging aids. The user is

cautioned against employing code 000003 in programs run under

these debugging aids.

(No information is transmitted in

the low byte)

BR

branch (unconditional)

000400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$

Condition Codes: Unaffected

Description:

Provides a way of transferring program control within a range of

- 12810 to +12710 words with a one

word instruction.

New PC address = updated PC + $(2 \times offset)$

Updated PC = address of branch instruction + 2

Example:

With the Branch instruction at.

location 500, the following offsets

apply:

New PC Address Offset Code Offset (decimal) 474 375 -3 476 376 -2 500 377 -1 502 000 0 504 001 +1 506 002 +2

BVC

branch if overflow is clear

102000 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset)$ if V = 0

Condition Codes: Unaffected

Description:

Tests the state of the V-Bit and

causes a branch if the V bit is clear. BVC is complementary

operation to BVS.

BVS

branch if overflow is set

102400 Plus offset

Operation:

 $PC \leftarrow PC + (2 \times offset) \text{ if } V = 1$

Condition Codes: Unaffected

Description:

Tests the state of V-bit (overflow) and causes a branch if the V bit is set. BVS is used to detect arithmetic overflow in the

previous operation.

CCC

clear condition code bits

000257

Operation:

None

Condition Codes: N: cleared

Z: cleared V: cleared C: cleared

Description:

Clear all condition code bits in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3 if bit 4 is a 1. Clears corresponding bits if

bit 4 = 0.

CLC

clear carry

000241

Operation:

None

Condition Codes: N: unaffected

Z: unaffected V: unaffected C: cleared

Description:

)

Clear carry bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator: i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CLN

clear negative bit

000250

Operation:

Description:

None

Condition Codes: N: cleared

Z: unaffected V: unaffected C: unaffected

Clear negative condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CLR CLRB

clear destination

■050DD

Operation:

 $(dst) \leftarrow 0$

Condition Codes: N: cleared

Z: set V: cleared

C: cleared

Description:

Word: Contents of specified destinations are replace with zeros.

Byte: Same

Example:

CLR R1

Before

(R1) = 177777

After (R1) = 000000

NZVC 1 1 1 1

NZVC 0 1 0 0

Note:

CLR and CLRB perform a DATIO bus cycle as the last bus cycle during the instruction execution. The DATI portion of the DATIO cycle is a "don't care" condition. but the addressed memory or device must be capable of responding to the DATI cycle to avoid a bus timeout error.

CLV

clear overflow bit

000242

Operation:

None

Condition Codes: N: unaffected

Z: unaffected V: cleared C: unaffected

Description:

Clear overflow condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bit 4 = 0.

CLZ

clear zero bit

000244

Operation

None

Condition Codes: N: unaffected

Z: cleared V: unaffected C: unaffected Description:

Clear zero condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CMP **CMPB**

compare src to dst

2SSDD

Operation:

(src)-(dst)

Condition Codes: N: set if result <0; cleared otherwise

Z: set if result = 0; cleared

otherwise

V: set if there was arithmetic overflow; that is, operands were of opposite signs and the sign of the destination was the same as the sign of the result; cleared othewise.

C: cleared if there was a carry from the most significant bit of the result; set otherwise.

Description:

Compares the source and destination operands and sets the condition codes, which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Note that unlike the subtract instruction the order of operation is (src)-(dst), not (dst)-(src).

COM COMB

complement dst

™051DD

Operation:

(dst) \sim (dst)

Condition Codes: N: set if most significant bit of

result is set: cleared otherwise. Z: set if result is 0: cleared

otherwise. V: cleared C: set

Description:

Replaces the contents of the destination address by their logical complement (each bit equal to 0 is set and each bit equal to 1 is

cleared) Bute: Same

Example:

COM RO

After Before (R0) = 164444(R0) = 013333NZVC NZVC 0 1 1 0 1001

DEC **DECB**

decrement dst

■053DD

Operation:

 $(dst) \leftarrow (dst) -1$

Condition Codes: N: set if result is < 0; cleared

otherwise

Z: set if result is 0; cleared

otherwise

V: set if (dst) was 100000; cleared

otherwise C: not affected

Description:

Word: Subtract 1 from the con-

tents of the destination

Bute: Same

Example:

DEC R5

Before After (R5) = 000001(R5) = 000000NZVC NZVC 1000 0 1 0 0

DIV

divide

071RSS

Operation:

 $r \leftarrow (rv1,r) \text{ div (src)}$

 $rv1 \leftarrow (rv1, r) \mod (src) (r \text{ must be})$

even)

Condition Codes: N: set if quotient < 0; cleared

otherwise

Z: set if quotient = 0; cleared

otherwise

V: set if source = 0 or if the absolute value of the register is equal to or larger than the absolute value of the source. (In this case the instruction is aborted because the quotient would exceed 15

bits).

C: set if divide 0 attempted; cleared otherwise.

Description:

The 32-bit two's complement integer in R and Rv1 is divided by the source operand. The quotient is left in R: the remainder in Rv1. Division will be performed so that the remainder is of the same sign as the dividend. R must be even.

Example:

CLR R0

MOV #20001,R1 DIV #2,R0

Before

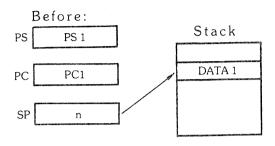
After

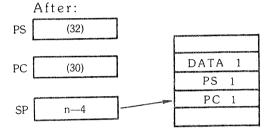
(R0) = 000000

(R0) = 010000Quotient

(R1) = 020001

(R1) = 000001Remainder Caution: EMT is used frequently by system software and is therefore not recommended for general use.





EMT

emulator trap

104000-104377

Operation:

↓(SP) ←PS 1(SP) ← PC PC - (30) $PS \leftarrow (32)$

Condition Codes: N: loaded from trap vector

Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector N: loaded from trap vector

Description:

All operation codes from 104000 to 104377 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30. The new PC is taken from the word at address 30; the new processor status (PS) is taken from the word at address 32.

FADD

floating add

07500R

Operation:

 $[(R) + 4, (R) + 6] \leftarrow [(R) + 4, (R) + 6]$

+[(R),(R) +2]

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: cleared

Description:

Adds the A argument to the B argument and stores the result in the A Argument position on the stack. General register R is used as the stack pointer for the operation.

 $A \leftarrow A + B$

FDIV

floating divide

07503R

Operation:

 $[(R)+4,(R)+6] \leftarrow [(R)+4,(R)+6]$

/[(R),(R)+2]

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: cleared

Description:

Divides the A Argument by the B Argument and stores the result in the A Argument position on the stack. If the divisor (B Argument) is equal to zero, the stack is left

untouched.

 $A \leftarrow A/B$

Note:

Unlike the PDP-11/40 (and PDP-11/35), the LSI-II processor pushes one word onto the stack during execution of FMUL and FDIV instructions and pops the word from the stack when completed. Thus, the SP (R6) must point to a read/write memory location; otherwise, a bus error (timeout) will occur.

FMUL

floating multiply

07502R

Operation:

 $[(R) +4, (R) +6] \leftarrow [(R) +4, (R) +6]$

x [(R), (R) + 2)]

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: cleared

Description:

Multiplies the A Argument by the B Argument and stores the result

in the A Argument position on the stack.

A - A x B

FSUB

floating subtract

07501R

Operation

 $[(R) + 4, (R) + 6] \leftarrow [(R) + 4, (R) + 6]$

-[(R),(R)+2]

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result = 0; cleared

otherwise V: cleared C: cleared

Description:

Subtracts the B Argument from the A Argument and stores the

result in the A Argument position

on the stack.

 $A \leftarrow A - B$

HALT

halt

000000

Condition Codes: Not affected

Description:

Causes the processor to leave RUN mode. The PC points to the next instruction to be executed. The processor goes into the control console mode. The contents of the PC are displayed on the console terminal and the console mode of operation is enabled, if the system is configured to support console mode.

INC **INCB**

increment dst

■052DD

Operation:

 $(dst) \leftarrow (dst) + 1$

Condition codes: N: set if result is < 0; cleared

otherwise

Z: set if result is 0; cleared

otherwise

V: set if (dst) held 077777; cleared

otherwise C: not affected

Description:

Word: Add one to contents of

destination Bute: Same

Example:

INC R2

Before

After

(R2) = 000333

(R2) = 000334

NZVC

NZVC

0 0 0 0

IOT

input/output trap

000004

jump to subroutine

004RDD

Operation:

↓ (SP) - PS $(SP) \leftarrow PC$ PC - (20)

PS -(22) Condition Codes: N: loaded from trap vector Z: loaded from trap vector

V: loaded from trap vector C: loaded from trap vector

Description:

Performs a trap sequence with a

trap vector address of 20.

(No information is transmitted in the low byte).

.IMP

iump

)

0001DD

PC ← (dst) **Operation:** Condition Codes: unaffected

Description:

JMP provides more flexible program branching than provided with the branch instructions. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes, with the exception of register mode 0. Execution of a jump with mode 0 will cause an "illegal instruction" condition, and will cause the CPU to trap to vector address 4. (Program control cannot be transferred to a register). Register deferred mode is legal and will cause program control to be transferred to the address held in the specified register. Note that instructions are word data and must therefore be fetched from an even-numbered address.

Deferred index mode JMP instructions permit transfer of control to the address contained in a selectable element of a table

of dispatch vectors.

Example:

JMP. FIRST; Transfers to First

First:

JMP @LIST; Transfers to location pointed

to at LIST.

List:

FIRST

; pointer to FIRST

JMP

@(SP)+; Transfer to location pointed to by the top of the stack, and remove the pointer from the stack.

Operation:

JSR

 $(tmp) \leftarrow (dst)$ (tmp) is an internal

processor register)

 \downarrow (SP) \leftarrow reg (push reg contents

onto processor stack)

reg - PC (PC holds location following JSR: this address now put

in reg)

PC - (dst) (PC now points to

subroutine destination)

Description:

In execution of the JSR, the old contents of the specified register (the "LINKAGE POINTER") are automatically pushed onto the processor stack and new linkage information placed in the register. Thus subroutines nested within subroutines to any depth may all be called with the same linkage register. There is no need either to plan the maximum depth at which any particular subroutine will be called or to include instructions in each routine to save and restore the linkage pointer. Further, since all linkages are saved in a reentrant manner on the processor stack execution of a subroutine may be interrupted, the same subroutine reentered and executed by an interrupt service routine. Execution of the initial subroutine can then be resumed when other requests are satisfied. This process (called nesting) can proceed to any level.

A subroutine called with a JSR reg, dst instruction can access the arguments following the call with either autoincrement addressing, (reg) +, (if arguments are accessed sequentially) or by indexed addressing, X(reg), (if accessed in random order). These addressing modes may also be deferred, @(reg) + and @X(reg) if the parameters are operand addresses rather than the operands themselves.

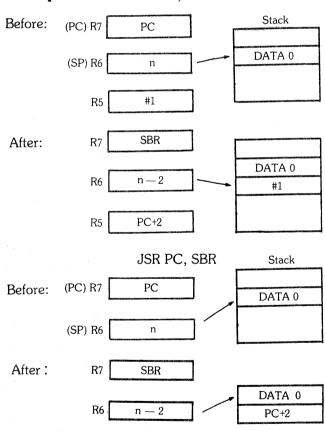
JSR PC, dst is a special case of the PDP-11 subroutine call suitable for subroutine calls that transmit parameters through the general registers. The SP and the PC are the only registers that may be modified by this call.

Another special case of the JSR instruction is JSR PC, @(SP)+ which exchanges the top element of the processor stack and the contents of the program counter. Use of this instruction allows two routines to swap program control and resume operation when recalled where they left off. Such routines are called "co-routines."

Return from a subroutine is done by the RTS instruction. RTS reg loads the contents of reg into the PC and pops the top element of the processor stack into the specified register.

Example:

JSR R5, SBR



MARK

mark

0064NN

Operation:

 $SP \leftarrow updated PC + 2 + 2n (n =$

number of parameters)

PC - R5 $R5 \leftarrow (SP) \uparrow$

Condition Codes: Unaffected

Description:

Used as part of the standard PDP-11 subroutine return convention. MARK facilitates the stack clean up procedures involved in subroutine exit. Assembler format is: MARK N.

Example:

MOV R5, -(SP)

; place old R5 on stack ; place N parameters

MOV P1, -(SP) MOV P2, -(SP)

; on the stack to be used there

by the subroutine.

MOV PN. -(SP)

MOV #MARK N, -(SP); places the instruction

: MARK N on the stack

MOV SP, R5

; set up address at MARK Nin-

struction.

JSR PC, SUB

; jump to subroutine

At this point the stack is as follows:

OLD R5
P1
PN
MARK N
OLD PC

And the program is at the address SUB which is the beginning of the subroutine.

SUB:

; execution of the subroutine itself.

RTS R5: the return begins: this causes the contents of R5 to be placed in the PC which then results in the execution of the instruction MARK N. The contents of old PC are placed in R5

MARK N causes: (1) the stack pointer to be adjusted to point to the old R5 value; (2) the value now is R5 (the old PC) to be placed in the PC; and (3) contents of the old R5 to be popped into R5 thus completing the return from subroutine.

MFPD MFPI

•

move from previous data space move from previous instruction space 0065SS

Operation: (temp) -

 $(temp) \leftarrow (src) - (SP) \leftarrow (temp)$

Condition Codes: N: set if the source < 0

Z: set if the source = 0

V: cleared C: unaffected

Description:

Pushes a word onto the current stack from an address in previous space. The source address is computed using the current registers and memory map. Since data space does not exist in the KDF11, MFPD executes the same as a MFPI. (LSI-11/23 only).

MFPS

move byte from processor status word

1067DD

Operation:

(dst) - PSW dst lower 8 bits

Condition Codes: N = set if PSW bit 7 = 1; cleared

otherwise

 $Z = \text{set if PS } \langle 0.7 \rangle = 0$; cleared

otherwise V = cleared C = not affected

Description:

The 8 bit contents of the PS are moved to the effective destination. If destination is mode 0, PS bit 7 is sign extended through upper byte of the register. The destination operand address is treated as a byte address.

Example:

MFPS R0

Before

After

R0 [0] PS [000014] R0 [000014] PS [000000]

MOV MOVB

move source to destination

■1SSDD

Operation:

(dst) - (src)

Condition Codes: N: set if (src) < 0; cleared

otherwise

Z: set if (src) = 0; cleared

otherwise V: cleared C: not affected

Description:

Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The contents of the source address are not

affected.

Byte: Same as MOV. The MOVB to a register (unique among byte instructions) extends the most significant bit of the low order byte (sign extension). Otherwise MOVB operates on bytes exactly as MOV operates on words.

Example:

MOV XXX,R1; loads Register 1 with the contents of memory location; XXX represents a programmer-defined mnemonic used to represent a memory

location.

MOV #20,R0; loads the number 20 into Register 0; "#" indicates that the value 20 is the operand. MOV@#20,-(R6); pushes the operand contained in location 20

onto the stack

MOV (R6)+,@#177566; pops the operand off a stack and moves it into memory location 177566

(terminal print buffer).

MOV R1,R3; performs an inter

register transfer.

MOVB @#177562, @#177566; moves a character from terminal keyboard buffer to terminal

printer buffer.

Note:

The MOVB instruction performs a DATIOB bus cycle as the last bus cycle during the instruction execution.

The DATI portion of the DATIOB bus cycle is a "don't care" condition, but the addressed memory or device must be capable of responding to the DATI cycle to avoid a bus timeout error. The MOV instruction performs only the DATO cycle as the last bus cycle.

MTPD MTPI

move to previous data space move to previous instruction space

10665 (0066SS)

 $(temp) \leftarrow (SP) + (dst) \leftarrow (temp)$

Condition Codes: N: set if the source <0

Z: set if the source = 0

V: cleared C: unaffected

Description:

This instruction pops a word off the current stack determined by PS (bits 15, 14) and stores that word into an address in previous space PS (bits 13, 12). The destination address is computed using the current registers and memory map.

Since data space does not exist in the KDF11, MTPD executes the same as MTPI. (LSI-11/23 only).

Note:

As a performance optimization, on the LSI-11/23 the lost bus cycle of a MTPD and MTPI is a DATO. This instruction was not implemented on LSI-11 and LSI-11/2 processors.

MTPS

move byte to processor status word

1064SS

Operation:

PSW ← (SRC)

Condition Codes: Set according to effective SRC

operand bits 0-3.

Description:

The 8 bits of the effective operand replaces the current contents of the PSW. The source operand address is treated as a byte address.

Note that the T bit (PSW bit 4) cannot be set with this instruction. The SRC operand remains unchanged. This instruction can be used to change the priority bit (PSW bit 7) in the PSW.

Note:

MTPS performs a DATIO bus cycle as the bus cycle during the instruction execution. The DATI portion of the DATIO cycle is a "don't care" condition, but the addressed memory or advice must be capable of responding to the DATI cycle to avoid a bus timeout error.

MUI.

multiply

070RSS

Operation:

If r even: $(rv1,r) \leftarrow r \times (src)$

If r odd: $r \leftarrow r \times (src)$

Condition code:

N: set if product is < 0: cleared

otherwise

Z: set if product is 0; cleared

otherwise V: cleared

C: set if the result is less than -215 or greater than or equal to 215 -1.

Description:

The contents of the destination register and source taken as two's complement integers are multiplied and stored in the destination register and the succeeding register (if R is even). If R is odd only the low order product is stored. Assembler syntax is: MUL S,R.

(Note that the actual destination is R, Rv1 which reduces to just R when R is odd).

Example:

CLC

16-bit product (R is odd) ; Clear carry condition code

MOV #400.R1 MUL #10,R1 **BCS ERROR**

: Carry will be set if product is less than -215 or greater than or equal to 215 no significance lost.

Before (R1) = 000400

After (R3) = 004000

NEG NEGB

negate dst

■054DD

Operation:

 $(dst) \leftarrow (dst)$

Condition Code: N: set if the result is < 0; cleared

otherwise

Z: set if result is 0: cleared

otherwise

V: set if the result is 100000

cleared otherwise

C: cleared if the result is 0; set

otherwise

Description:

Word: Replaces the contents of the destination address by its two's complement. Note that 100000 is replaced by itself -(in two's complement notation the most negative number has no positive counterpart).

Byte: Same

Example:

NEG R0

Before

(R0) = 000010

NZVC 0 0 0 0

After (R0) = 177770

NZVC 1 0 0 1

NOP

no operation

000240

Operation:

None

Condition Codes: N: unaffected

Z: unaffected

V: unaffected C: unaffected

Description:

No changes in PSW. NOP can be used to occupy time or memory space in lieu of an alternate instruction. The alternate instruction may be patched in, to

replace NOP.

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

RESET

reset external bus

000005

Condition Codes: Not affected

Description:

Sends INIT on the BUS for 10 μ sec. All devices on the BUS are reset to their state at power-up. The processor remains in an idle state for 90 µsec following issuance of INIT.

ROL ROLB

rotate left

■061DD

Operation:

 $(dst) \leftarrow (dst)$

rotate left one place

Condition Codes: N: set if the high-order bit of the result word is set (result < 0):

cleared otherwise.

Z: set if all bits of the result word =

0: cleared otherwise.

V: loaded with the Exclusive OR of the N-bit and C-bit (as set by the completion of the rotate

operation)

C: loaded with the high-order bit

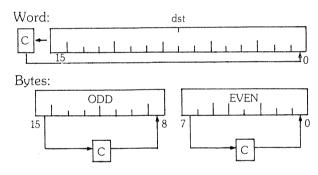
of the destination.

Description:

Word: Rotate all bits of the destination left one place. Bit 15 is loaded into the C-bit of the status word and the previous contents of the C-bit are loaded into Bit 0 of

the destination. Byte: Same

Example:



ROR RORB

rotate right

■060DD

Operation:

 $(dst) \leftarrow (dst)$

rotate right one place

Condition Codes: N: set if the high-order bit of the

result is set (result < 0); cleared

otherwise

Z: set if all bits of result = 0; cleared otherwise

V: loaded with the Exclusive OR of the N-bit and C-bit (as set by the completion of the rotate operation)

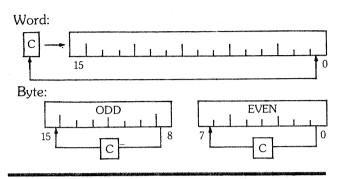
C: loaded with the low-order bit of

the destination

Description:

Rotates all bits of the destination right one place. Bit 0 is loaded into the C-bit and the previous contents of the C-bit are loaded in bit 15 of the destination. Bute: Same

Example:



RTI

return from interrupt

000002

Operation:

$$PC \leftarrow (SP) \uparrow$$

 $PS \leftarrow (SP) \uparrow$

Condition Codes: N: loaded from processor stack

Z: loaded from processor stack V: loaded from processor stack C: loaded from processor stack

Description:

Used to exit from an interrupt or TRAP service routine. The PC and PS are restored (popped) from the processor stack. If a trace trap is pending, the first instruction after RTI will not be executed prior to the next T trap.

RTS

return from subroutine

00020R

Operation:

$$PC \leftarrow (reg)$$

 $(reg) \leftarrow (SP)$

Description:

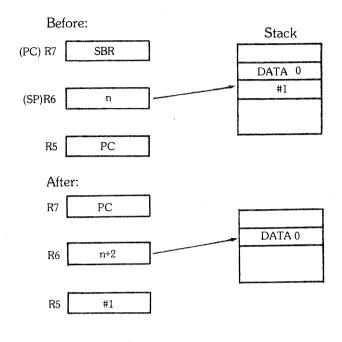
Loads contents of register into PC and pops the top element of the processor stack into the

specified register.

Return from a non-reentrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a JSR PC, dst exists with a RTS PC and a subroutine called with a JSR R5, dst, may pick up parameters with addressing modes (R5) +, X(R5), or @ X(R5) and finally exists, with an RTS R5.

Example:

RTS R5



RTT

return from interrupt

000006

Operation:

$$PC \leftarrow (SP) \uparrow$$

 $PS \leftarrow (SP) \uparrow$

Condition Codes: N: loaded from processor stack Z: loaded from processor stack

V: loaded from processor stack C: loaded from processor stack

Description:

Operation is the same as RTI except that it inhibits a trace trap while RTI permits trace trap. If new PS has T bit set, trap will occur after execution of first instruction after RTT.

SBC SBCB

subtract carry

■056DD

Operation:

 $(dst) \leftarrow (dst) - (C)$

Condition Codes: N: set if result < 0; cleared

otherwise

Z: set if result 0; cleared

otherwise

V: set if (dst) was 100000; cleared

otherwise

C: set if (dst) was 0 and C was 1;

cleared otherwise

Description:

Word: Subtracts the contents of the C-bit from the destination. This permits the carry from the subtraction of two low-order words to be subtracted from the high order part of the result.

Byte: Same

Example:

Double precision subtraction is

done by: SUB A0,B0 SBC B1 SUB A1,B1

SCC

set all condition code bits

000277

Operation:

None

Condition Codes: N: set

Z: set

V: set C: set

Description:

Set all condition code bits in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEC

set carry bit

000261

Operation:

None

Condition Codes: N: unaffected

Z: unaffected V- unaffected

C: set

Description:

Set carry condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bit 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEN

set negative bit

000262

Operation:

None

Condition Codes: N: set

Z: unaffected V: unaffected C: unaffected

Description:

Set negative condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator, i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEV

set overflow bit

000263

Operation:

None

Condition Codes: N: unaffected

Z: unaffected

V: set

C: unaffected

Description:

Set overflow condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bit 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SF7

set zero bit

000264

Operation:

None

Condition Codes: N: unaffected

Z: set

V: unaffected

C: unaffected

Description:

Set zero condition code bit in PSW. Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bit 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2 or 3 if bit 4 is a 1. Clears corresponding bits if 4 = 0.

SOB

subtract one and branch (if $\neq 0$)

077RNN

Operation:

 $(R) \leftarrow (R) - 1$; if this result $\neq 0$ then $PC \leftarrow PC - (2 \times offset)$ if (R)

= 0: PC ← PC.

Condition Codes: Unaffected

Description:

The register is decremented. If it is not equal to 0, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a six bit positive number. This instruction provides a fast, efficient method of loop control. Assembler syntax is:

SOB R,A

where A is the address to which transfer is to be made if the decremented R is not equal to 0. Note that the SOB instruction can not be used to transfer control in the forward direction.

SUB

subtract src from dst

16SSDD

Operation:

 $(dst) \leftarrow (dst) - (src)$

Condition Codes: N: set if result < 0; clear

otherwise

Z: set if result = 0: cleared

otherwise

V: set if there was arithmetic overflow as a result of the operation, that is if operands were of opposite signs and the sign of the source was the same as the sign of the result; cleared

otherwise.

C: cleared if there was a carry from the most significant bit of the

result; set otherwise.

Description:

Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double-precision arithmetic the C-bit, when set,

indicates a "borrow".

Example:

SUB R1.R2

Before (R1) = 011111(R1) = 011111

(R2) = 012345(R2) = 001234

> NZVC NZVC 1 1 1 1 0 0 0 0

SWAB

swap bytes

0003DD

After

Operation:

Byte 1/Byte 0 ← Byte 0/Byte 1

Condition Codes: N: set if high-order bit of loworder byte (bit 7) of result is set;

cleared otherwise

Z: set if low-order byte of result =

0; cleared otherwise

V: cleared C: cleared Description:

Exchanges high-order byte and low-order byte of the destinattion word (destination must be a word

address)

Example:

SWAB R1

Before After (R1) = 077777(R1) = 177577NZVC NZVC 1 1 1 1 0 0 0 0

SXT

sign extend

0067DD

Operation:

(dst) ← 0 if N-bit is clear $(dst) \leftarrow -1 \text{ N-bit t is set}$

Condition Codes: N: unaffected

Z: set if N-bit is clear

V: cleared C: unaffected

Description:

If the condition code bit N is set then a -1 is placed in the destination operand: if N bit is clear, then a 0 is placed in the destination operand. This instruction is particularly useful in multiple precision arithmetic because it permits the sign to be extended through multiple words.

Example:

SXT A

Before After (A) = 012345(A) = 177777NZVC NZVC 1 0 0 0 1 0 0 0

TRAP

trap

104400 - 104777

Operation:

(SP) ← PS (SP) ← PC PC ← (34) $PS \leftarrow (36)$

Condition Codes: N: loaded from

trap vector Z: loaded from trap vector V: loaded from trap vector loaded from trap vector

Description:

Operation codes from 104400 to 104777 are TRAP instructions. TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34.

Note: Since DEC software makes frequent use of EMT, the TRAP instruction is recommended for

general use.

TST **TSTB**

test dst

Operation:

 $(dst) \leftarrow (dst)$

Condition Codes: N: set if the result is < 0: cleared

otherwise

Z: set if result is 0; cleared

otherwise V: cleared C: cleared

Description:

Word: Sets the condition codes N and Z according to the contents of the destination address, contents

of dst remains unmodified

Byte: Same

Example:

TST_{R1}

Before After (R1) = 012340(R1) = 012340NZVC NZVC 0 0 1 1 0 0 0 0

WAIT

wait for interrupt

000001

Condition Codes: Not affected

Description:

Provides a way for the processor to relinquish use of the bus while it waits for an external interrupt request. Having been given a WAIT command, the processor will not compete for bus use by fetching instructions or operands from memory. This permits higher transfer rates between a device and memory, since no processor-induced latencies will be encountered by interrupt requests from devices. In WAIT, as in all instructions, the PC points to the next instruction following the WAIT instruction. Thus when an interrupt causes the PC and PS to be pushed onto the processor stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction

following the WAIT.

XOR

exclusive OR

074RDD

Operation:

 $(dst) \leftarrow R \forall (dst)$

Condition Codes: N: set if the result < 0; cleared

otherwise

Z: set if result = 0; cleared otherwise

V: cleared C: unaffected

Description:

The exclusive OR of the register and destination operand is stored in the destination address. Contents of register are unaffected. Assembler format is: XOR

R.D

Example:

XOR RO, R2

Before After (R0) = 001234(R0) = 001234(R2) = 000325(R2) = 001111NZVC NZVC

0 0 0 1 1 1 1 1 (R0) = 0 000 001 010 011 100Before:

(R2) = 0 000 001 001 001 001(R2) = 0 000 000 011 010 101After:

C2.5 INSTRUCTION TIMING

The execution time for an instruction depends on the instruction itself, the modes of addressing used, and the type of memory referenced. In most cases the instruction execution time is the sum of a Basic Time, a Source Address (SRC) Time, and a Destination Address (DST) Time.

INSTR. TIME = Basic Time + SRC Time + DST Time.

All timing information is in microseconds, unless otherwise noted. Times are typical; timing can vary ±20 percent. A 350ns microcycle is assumed.

Timing information for the EIS and FIS Instruction Sets is different from the Basic Instruction Set, and is therefore in a separate section.

C2.5.1 Basic Instruction Set Times

Source and Destination Time

MODE	SRC TIME (Word)	SRC TIME (Byte)	DST TIME (Word)	DST TIME (Byte)
0	0	0	0	0
1	$1.40\mu s$	$1.05 \mu s$	$2.10\mu s$	$1.75 \mu s$
2	1.40	1.05	2.10	1.75
3	3.50	3.15	4.20	4.20
4	2.10	1.75	2.80	2.45
5	4.20	3.85	4.90	4.90
6	4.20	3.85	4.90	4.55
7	6.30	5.95	6.65	7.00

NOTE: for MODE 2 and MODE 4 if R6 or R7 used with Byte operation, add $0.35 \mu s$ to SRC time and $0.70 \mu s$ to DST time.

Basic Time

DOPS (Double Operand Instructions)	DMO	DM1-7
MOV	$3.50 \mu \text{s}$	2.45μ s
ADD, SOR, SUB, BIC, BIS	3.50	4.20
CMP, BIT	3.50	3.15
MOVB	3.85	3.85
BICB, BISB	3.85	3.85
CMP, BITB	3.15	2.80

NOTE:

DMO = Destination Mode 0

DM1-7 = Destination Modes 1 through 7

SOPS (Single Operand Instructions)	DMO	DM1-7
·	0.05	1.00 -
CLR	$3.85 \mu s$	4.20 μs
INC, ADC, DEC, SBC	4.20	4.90
COM, NLG	4.20	4.55
ROL, ASL	3.85	4.55
TST	4.20	3.85
ROR	5.25	5.95
ASR	5.60	6.30
CLRB, COMB, NEGB	3.85	4.20
ROLB, ASLB	3.85	4.20
INCB, DECB, SBCB, ADC	3.85	4.55
TSTB	3.85	3.50
RORB	4.20	4.90
ASRB	4.55	5.95
SWAB	4.20	3.85
SXT	5.95	6.65
MFPS (1067DD)	4.90	6.65
MTPS (1064SS)	7.00	7.00*

^{*}For MTPS use Byte DST time not SRC time.

^{*}Add 0.35 μ s to instr. time if Bit 7 of effective OPR = 1.

JMP/JSR MODE	DST TIME
1	0.70 µs
2	1.40
3	1.75
4	1.40
5	2.45
6	2.45
7	4.20

1

INSTRUCTION	BASIC TIMES
JMP	3.50 μs
JSR (PC = LINK)	5.25
JSR (PC \neq LINK)	8.40
ALL BRANCHES	3.50 (condition met or not met)
SOB (BRANCH)	4.90
SOB (NO BRANCH)	4.20
SET CC	3.50
CLEAR CC	3.50
NOP	3.50
RTS	5.25
MARK	11.55
RTI	8.75*
RTT	8.75*+
TRAP, EMT	16.80* μ s
IOT, RPT	18.55*
WAIT	6.30
HALT	5.60
RESET	5.95+ 10.0 μs for INIT+ 90.0 μs

^{*}If new PS has bit 4 or bit 7 set add $0.35~\mu$ s for each +If new PS has bit 4 (T bit) set add $2.10~\mu$ s

C2.5.2 EIS Instruction Times

MODE	SRC TIME
0	$0.35~\mu \mathrm{s}$
1	2.10
2	2.80
3	3.15
4	2.80
5	3.85
6	3.85
7	5.60

INSTRUCTION BASIC TIME

MUL	24.0 to 37.0 μs64.0 μs Worst Case78.0 μs Worst Case	(If both numbers less than 256 in
DIV	78.0 u.s. Worst Case	16 bit multiply.
ASH (RIGHT)	10.1 + 1.75 per shift	(10 011 1111111111111111111111111111111
	10.8 + 2.45 per shift	
ASHC (RIGHT)	10.1 + 2.80 per shift	
ASHC (LEFT)	10.1 + 3.15 per shift	

C2.5.3 FIS Instruction Times

INST. TIME = BASIC TIME + SHIFT TIME TO ALIGN BINARY POINTS + SHIFT TIME FOR NORMALIZATION.

EXPONENT DIFFERENCE	ALIGN BINARY POINTS
0 - 7 8 - 15 16 - 23	2.45 μ s per shift 3.50 + 2.45 per shift over 8 7.00 + 2.45 per shift over 16
EXPONENT DIFFERENCE	NORMALIZATION
0 - 7 8 - 15 16 - 23	2.1 μ s per shift 2.1 + 2.1 per shift over 8 4.2 + 2.1 per shift over 16
INSTRUCTION	BASIC TIME
FADD FSUB FMUL	42.1 μ s 42.4 74.2 to 80.9 μ s if either argument has only 7 bits of precision, i.e., the second word of the 32 bit argument is 0.
FDIV	121.1 μ s worst case (i.e., arguments have more than 7 bits of precision). 151 μ s typical
	232μ s worst case

C2.5.4 DMA (Direct Memory Access) Latency

DMA latency, which is the time from request (BDMRL) to bus mastership for the first DMA device, is $6.45\,\mu\,\mathrm{s}$, maximum. This time is the longest processor DATIO cycle which occurs for an ASR instruction with destination modes of 1 through 7. DMA requests are honored during memory refresh by the processor.

C2.5.5 Interrupt Latency

If processor is performing memory refresh:

Time from interrupt request	
(BIRQ L) to acknowledgement	
(BIAK L)	118 μs max
Time from acknowledgement	
(BIAK L) to fetch of first service	
routine instruction	+ $16.5 \mu s$ max
Total time from request to first	
service routine instruction	$134.5 \mu s max$

If processor is not performing memory refresh:

Time from interrupt request (BIRQ L) to acknowledgement (BIAK L)

 $27.6\mu \text{ s max}$

Time from acknowledgement (BIAK L) to fetch of first service routine instruction

 $+16.5 \,\mu$ s max

Total time from request to first service routine instruction

 $44.1 \,\mu\text{s}$ max

Note:

During all EIS and FIS instructions, device and event interrupt requests are periodically scanned. If present, the instruction is aborted and all processor state information is backed up to the beginning of the instruction. After the interrupt is processed, the KEV11 instruction is re-executed from the beginning. Caution should be observed with the frequency of interrupts; if the frequency is too high, an EIS or FIS instruction will never complete. It is suggested a maximum frequency of 3300 interrupts per second be assured if the EIS or FIS instruction set is being executed.

APPENDIX D. VIDEO CONTROLLER MEMORY REFERENCE GUIDE

D1. MEMORY MODULE

1

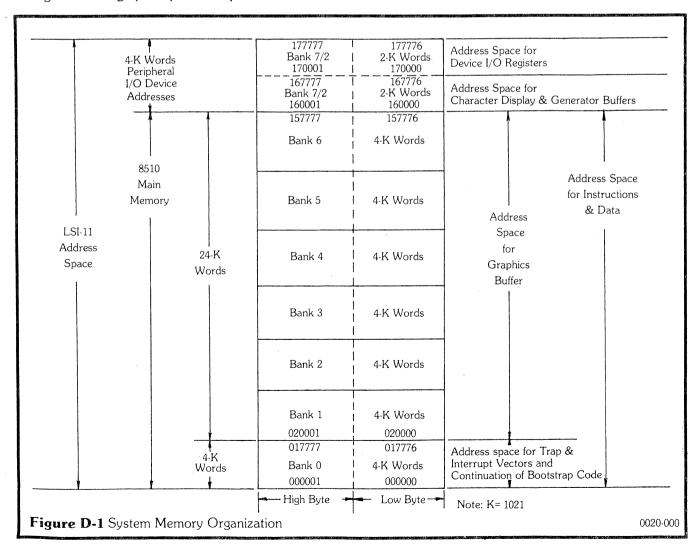
The 8510 contains 28-K words (56-K bytes) of read/write memory. This main memory is used for graphics display storage as well as data processing storage. The memory has a two-port structure; graphics display refresh is performed by a video controller. The system bus is not used for graphics display refresh; the result is that data processing throughput is only reduced by a maximum of 8% (worse case) during graphics display refresh. Typical throughput degradation is less than 2%.

The system memory organization is shown in Figure D-1. The processor can directly address 32-K words. Banks 1 through 6 of the address space are used for graphics storage. One full graphics picture requires 4800 words.

The graphics picture can be divided into 3 zones; each zone requires 1600 words.

Bank 7 of the address space is used for peripheral I/O addressing. A ROM, on the Floppy Disk Controller module, is addressed from a location in Bank 7; this ROM contains the beginning of a bootstrap program. The remainder of the bootstrap program resides on the operating system diskettes. When the Boot ROM is addressed, (during power-on, power fail/restart or reboot), it will cause the diskette portion of the bootstrap program to be loaded into Bank 0.

Bank 7 locations are also used to address a character generator RAM and a character page RAM on the Video Controller module. The character generator RAM is used to store 192 blocks of 8x10 dot patterns; these patterns are used to construct text characters. The character page RAM is used to store the text characters for the display. The page RAM can store 2000 characters.



Bank 0 is used for vector addresses and the continuation of the bootstrap program. The following is a list of reserved locations in Bank 0.

Reserved low memory, trap and interrupt vectors

000 Reserved

004 Bus timeout & illegal instructions (eg. JMP R0)

Note: Address & stack overflow traps not implemented.

010 Illegal & reserved instruction

014 BPT instruction & T bit

020 IOT instruction

024 Power Fail

030 EMT instruction

034 Trap instruction

040 through 056 Reserved for system communication

060 Console input Device

064 Console Output Device

074 Console-Emulator, Alt.

100 Vertical Retrace, Start

120, 124 Serial Interface Unit 1

130, 134 Serial Interface Unit 2

150, 154 Serial Interface Unit 3

164 Console Emulator

174 Console Emulator, Alt.

200 Line Printer 0

240 Line Printer 1

244 FIS Error Trap

250 QX Floppy Disk

300, 304 Parallel Interface Unit 0

310, 314 Parallel Interface Unit 1

320, 324 Serial Interface Unit 4

330, 334 Serial Interface Unit 5

340, 344 Serial Interface Unit 6

350, 354 Serial interface Unit 7

D2. VIDEO CONTROLLER

The Video Controller produces the display geometry shown in Figure D-2. Two distinct pictures are controlled: a 320 dot x 240 dot graphic picture and a 1920 (maximum) character picture. Both pictures can be displayed simultaneously on a 240 line, non-interlaced, raster scan monitor, typically the 8532-1 unit.

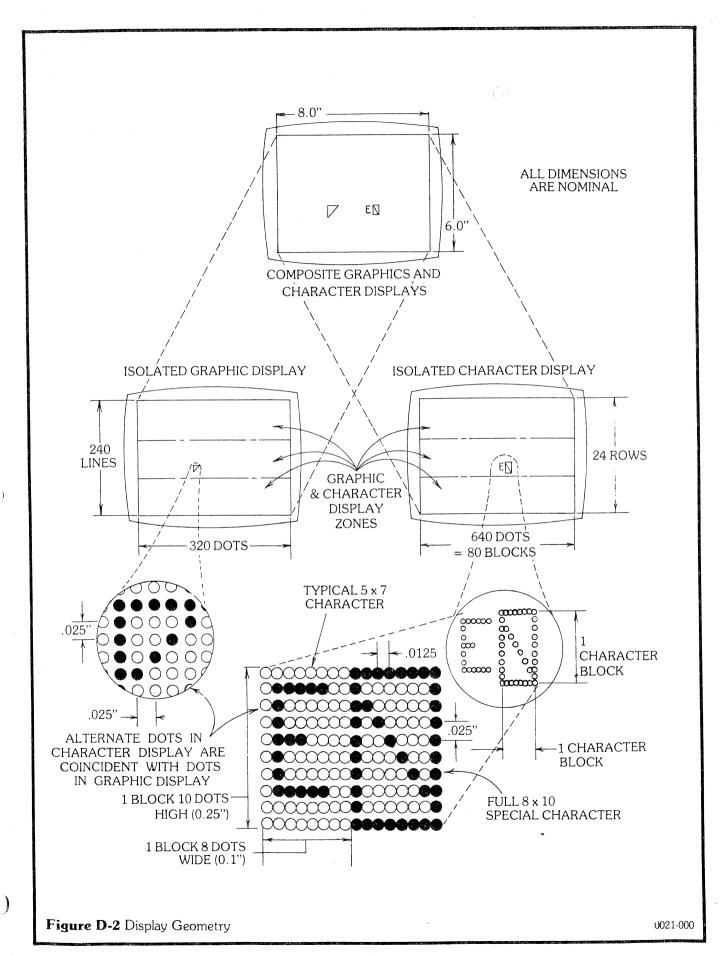
The dots in a graphics picture are equally spaced in the horizontal and vertical directions; the dots in a character picture are unequally spaced, as shown in Figure D-2. On a 12 in. monitor, the graphic picture dots are on 0.025 in. centers; the character picture dots are on 0.025 centers in the vertical direction and 0.0125 in. in the horizontal direction. This arrangement causes every other character dot to be coincident with a graphics dot.

Each line of a graphics picture requires 20 words of storage. Each bit of a word corresponds to a dot; the LSB of the word corresponds to the leftmost dot of the group. The 20 memory words of a line correspond to 320 dots; the 240 lines of a picture require 4800 words of storage. Each third of a picture can be blanked; this causes the minimum graphics picture memory allocation to be 1600 words (per picture) in main memory.

A character picture is organized as 640 dots wide x 240 dots high. Each character is formed in an 8 dot wide x 10 dot high matrix. This allows 1920 character positions in a 24 line, 80 character per line format. Note that the character examples in Figure D-2 show a 5 x 7 dot character and an 8 x 10 dot character. The 5 x 7 dot matrix is typically used to provide spacing between upper/lower case letters; the 8 x 10 dot matrix can be used to create special symbols such as cursors, line segments, etc. A 2-K byte character generator RAM, (on the Video Controller module), can store 192 blocks of 8 x 10 dot patterns.

A 2-K byte character page RAM (also on the Video Controller module) can store 2000 characters; 1920 characters will be displayed in the 24 line, 80 character per line format, the remaining 80 characters represent a hidden, 25th line. This 25th line is provided to allow continuous scrolling (moving the body of text up or down, one character line at a time) on the display. The Video Controller also supports panning (moving the body of text up or down one raster line at a time).

Both the graphics picture and character picture can be selectively blanked. For blanking control, the display is divided into three horizontal zones: 80 lines by 320 dots for the graphics picture or 8 rows of 80 character blocks each for the character picture. Each zone may be selectively blanked or displayed. Blanking of any one character picture zone is mutually independent of the blanking of any one graphics picture zone. When a graphics picture is blanked, the corresponding area of the graphics display buffer in main memory is not read during refresh, thus reducing the buffer storage size. If all zones of the graphics display buffer are blanked, no graphics refresh occurs.



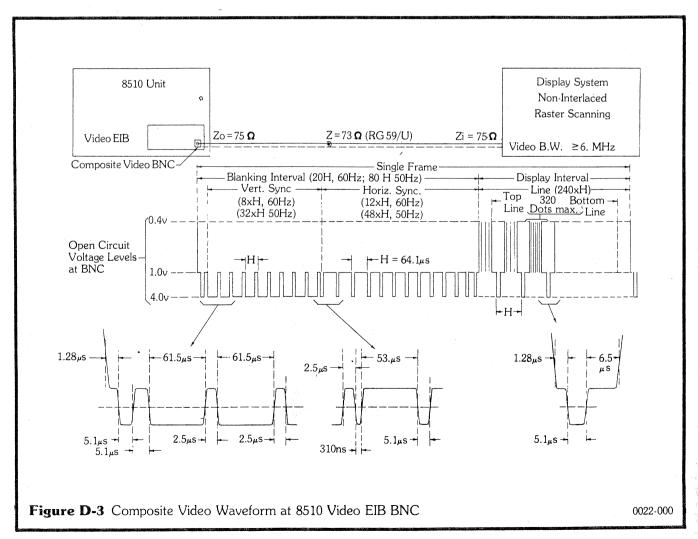
The Video Controller supports keyboard input from the 8532-2 unit, or similar devices. Programmable audio tones, produced by a speaker in the 8532-1 Monitor, are controlled by circuitry on the Video Controller module. This module also supports emulation of a serial interface. I/O registers, on the Controller, may be programmed exactly like a serial interface and, when supported by emulation software, will drive the character display like a terminal.

The 8510 unit is shipped with a Video External Interface Board (EIB) installed. This EIB connects the 8532-1 Monitor and the 8532-2 Keyboard to the 8510 Video Controller module. A switch, (KBD/EML), on the Video EIB, assigns the 8532-1 Monitor and 8532-2 Keyboard as the system console or as a distinct peripheral. Normally, this switch is in the OFF (STD) position to use the 8532-1 and 8532-2 as the system console. If another device, (typically a serial interface), is to be used as the system console, this switch must be set to ON (ALT). Switches on the Asynchronous Serial Interface EIB (see Appendix F) can then be programmed to allow the attached peripheral to function as the system console.

A composite signal (horizontal & vertical synchronization, graphics video and character video) is available at a BNC connector on the Video External Interface Board (EIB) on the 8510 unit. The waveform is shown in Figure D-3. This signal can be used to drive a composite video monitor other than the 8532-1 unit. The monitor should have a bandwidth of at least 6 MHz (@-3dB). The output impedance at the composite video BNC connector is 75 ohms; to preserve the waveform and minimize reflections, a matched impedance transmission connection should be used, as suggested in Figure D-3. The monitor must contain video, horizontal and vertical separation circuits.

The composite video signal is similar to EIA RS 170 requirements. The exception is that a non-interlaced, 240 visible line system is used. There are also slight differences in horizontal equalization timing during the vertical retrace period.

The Video Controller (quad height) module is attached to the Memory module through short ribbon cables. The Memory module plugs into the 8510 backplane.



D2.1 VIDEO CONTROLLER I/O CONTROL

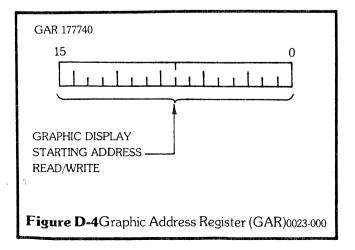
There are 5 status/control registers and 3 data buffers on the Video Controller that are used for I/O control. These registers and buffers are accessed via locations in bank 7 of the processor address space. The character page RAM and character generator RAM are also accessed via locations in bank 7. The 4800 word graphics buffer can be located anywhere in banks 1 through 6 of the address space. The KBD/EML switch on the Video EIB selects the address (and interrupt vectors) of certain registers and buffers. A listing of all addresses applicable to the Video Controller is as follows:

(NOTE: All addresses are in octal.)

KBD/EML Switch	OFF (STD)	ON (ALT)
Keyboard Status Register KSR	177560	177760
Keyboard Data Buffer KDB	177562	177762
Emulator Status Register ESR	177564	177764
Emulator Data Buffer EDB	177566	177766
KSR Interrupt Vectors	60	70
ESR Interrupt Vectors (Read)	64	74
ESR Interrupt Vectors (Write)	164	174
Graphics Address Register GAR	1777	740
Video Index Register VIR	1777	742
Video Control Register VCR	1777	744
Character Page Buffer (RAM)	160000 throu	ıgh 167777
Character Generator Buffer (RAM)	160000 throu	ıgh 167777
Graphics Buffer (Main Memory)	20000 throu	ıgh 157777

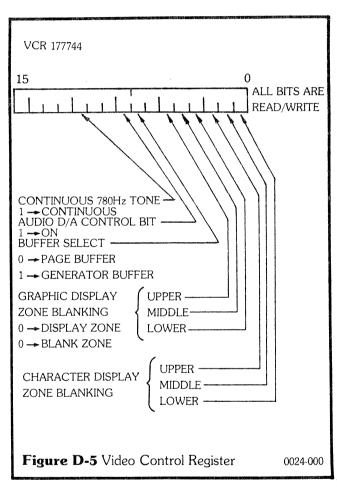
D2.1.1 Graphics Address Register

The format of the Graphics Address Register is shown in Figure D-4. This register must be loaded with the starting address of the graphics buffer. The starting address must be even-valued (on a word boundary). If any of the three 1600 word display zones have been blanked, (by certain bits in the VCR), the value loaded into the GAR must still be an address pointing to the beginning of the first zone, as if none were blanked. Unblanked zones must be in Banks 1 through 6; however, blanked zones can be in Banks 0 or 7.



D2.1.2 Video Control Register

The format of the Video Control Register is shown in Figure D-5. As indicated, bits in the low byte control graphic and character display blanking and Character Page-Character Generator Buffer selection. Bits 8 & 11 are the only other bits used; all other bits are "don't care" (can be "1"s or "0"s). The frequency of the tone controlled by bit 8 is a function of the rate at which it is written on ("1") or off ("0"); that is, bit 8 is used to create a pulse train whose envelope determines the resultant frequency.



D2. 1. 3 Video Controller Buffers

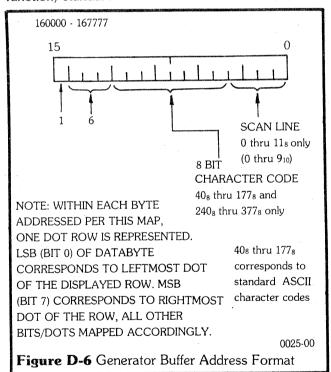
The Character Page and Character Generator Buffers are both mapped onto the same 4-K bytes of address space (locations 160000 through 167777). Bit 7 of the VCR controls which buffer is selected.

D2. 1. 3. 1 Character Generator Buffer

The address format for the Character Generator Buffer is shown in Figure D-6. There are 1920 addressable byte locations in this buffer. Each location is use to store an 8-dot pattern. Ten of these dot patterns are required for each 8-wide by 10-high dot matrix of a character; as a result, 192 distinct characters can be designated under software control.

The scan line bits of the Character Generator address correspond to the 10 rows of the dot matrix; because the monitor raster scanning is from the top, down (and left to right), the all-zeros state of these bits correspond to the top-most row of a dot matrix. The LSB of each output byte corresponds to the left-most dot of a matrix row; a "1" level illuminates a dot location.

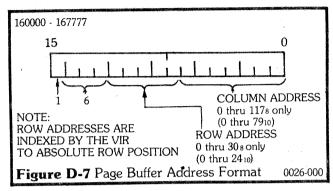
Bits 4 through 11 are for character coding; each allowed pattern of these bits, when combined with the 10 codes of the scan line bits 0 through 3, select the ten 8-dot output bytes that form a character. Bits 4 through 11 are restricted to the 192 codes contained in the 40 through 177 and 240 through 377 (octal) ranges. The codes 40 through 177 are the codes for the 96 (non-control function) standard ASCII character set.



D2. 1. 3. 2 Character Page Buffer

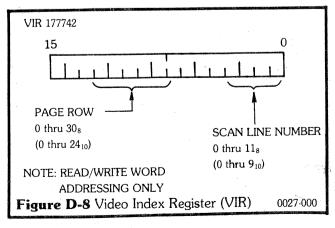
The address format for the Character Page Buffer is shown in Figure D-7. There are 2000 addressable byte locations in this buffer. Each location is used to store a code in the range 40 through 177 and 240 through 377 (octal). These output codes are used as partial addresses (the character codes) for the Character Generator Buffer.

A page of text on a monitor is organized as 24 lines (or rows) of 80 characters each. As shown in Figure D-7, bits 0 through 6 address the character position in a row of text; the all-zeroes state of these bits correspond to the left-most position (column) of a row. Bits 7 through 11 address the text row; the all-zeroes state of these bits correspond to the top-most row of text. The 30 (octal) state of these bits correspond to a 25th, hidden row of text.



D2. 1. 4 Video Index Register

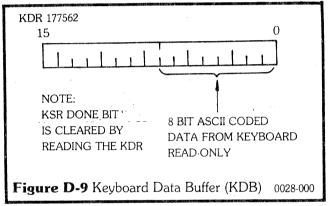
The format of the Video Index Register is shown in Figure D-8. The VIR is used to pan and scroll the text on a monitor. Loaded into the VIR are the initial values of the (character dot matrix) scan line number and (text page) row number. These values are used at the start of the visible raster scan. The VIR scan line bits (0 through 3) are used as the initial value for the scan line address bits of the Character Generator Buffer (see Figure D-6). The VIR page row bits 8 through 12 are used as the initial value for the Character Page Buffer row address bits (see Figure D-7). These initial addresses are then incremented as the visible raster progresses to the end of its frame; the column address of the Character Page Buffer is also cycled through its range every time the page row address is incremented. The result is that a page of text (1920 characters, if available) is displayed on the monitor. If the VIR is loaded with a lower-than-previous value for the page row number, the text will scroll downward on the display. Bits 4 through 7 and 13 through 15 of th VIR are "don't care" (can be "1" or "0").

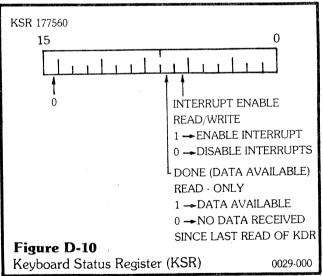


D2. 1. 5 Keyboard Interface

The Video Controller contains an interface to the 8532-2 Keyboard Unit. This interface consists of a Keyboard Data Buffer, a Keyboard Status Register and interrupt generation logic. The formats of the KDB and KSR are shown in Figures D-9 and D-10.

A keyboard output code (see Figure 9-1) is loaded into the KDB by the activation of a strobe signal from the Keyboard Unit, which in turn, was initiated by a keystroke. The DONE (Data available) bit in the KSR will be set. If the KSR Interrupt Enable bit is set, an interrupt will be requested. When acted upon, the request will cause the processor to vector to location 60 (or 70), the keyboard service routine. The processor will read the KDB and clear the DONE bit in the KSR.



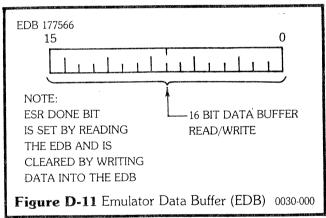


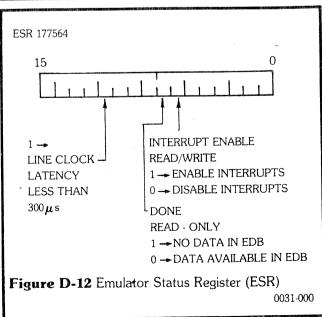
D2. 1. 6 Control Console Emulation

The Video Controller also contains hardware to support the emulation of a serial console device. An Emulation Data Buffer and Emulation Status Register are provided; their formats are shown in Figures D-11 and D-12. The EDB provides temporary word storage, that, in conjunction with software, can be used as a transfer device to emulate any desired control console function.

When a word is written into the EDB, the ESR DONE bit (bit 7) will be cleared, indicating that data is available. An interrupt to vector 164 (or 174) is also generated. This request, an EDB-read service routine, will be acted upon immediately, independent of the state of the ESR Interrupt Enable bit (bit 6). This allows the emulation software to read and transfer the contents of the EDB. Once the EDB is read, the ESR DONE bit will be set. If the ESR Interrupt Enable bit is set at this time, an interrupt to vector 64 (or 74) will be requested in approximately 450 microseconds (7 horizontal raster lines). This request alerts the emulation software that another word can be written into the EDB. The 450 microsecond delay regulates the transfer rate of the emulated serial channel to be consistent with the transfer rate of the emulated console.

A line clock latency bit (bit 11) is available in the ESR for timing reference. This bit is set during raster lines 240 through 247 (the beginning of the vertical retrace interval). This bit signifies that there will still be 12 lines (approximately 770 microseconds) left in the vertical retrace interval. This bit can also be used as a "done" bit for timing algorithms.





D2. 1. 7 Vertical Retrace Start Interrupt

The Video Controller continuously requests an interrupt through vector 100 at the start of the vertical retrace the raster. This interrupt may not be simultaneous with the leading edge of the line clock latency bit, as higher priority events (e.g. memory refresh), may delay recognition of the request. For critical applications requiring that all interrupt service activity be completed within vertical retrace, while the screen is inherently blanked, the interrupt latency bit must be used.

APPENDIX E. DISKETTE CONTROLLER REFERENCE GUIDE

E1. GENERAL DESCRIPTION

)

The 8510/a system utilizes flexible (floppy) 8 in. magnetic diskettes for mass memory. A floppy Diskette Controller module, in the 8510 unit, can control up to four diskette drives. The 8510, 8512 and 8515 units each contain a diskette drive assembly. The drives in the 8512 and 8515 units are connected to the diskette controller in the 8510 by means of External Interface Boards (EIBs) mounted on the rear of each unit. Sections 2 & 4 of this document describe the installation.

The diskettes are IBM compatible: data written on a diskette can be read by an IBM 3740 system; conversely, diskette data written by an IBM 3740 system can be read by a TERAK 8510/a system. The following file format is defined by IBM.

Reserved Track	:	1	
Index Tracks	:	1	
Data Tracks	:	73	
Spare Tracks	:	2	
Sectors per track		26	
Data bytes per sector	:	128	
Sectors per diskette	:	1924	246,72
			476.

The file format used by TERAK is slightly different and operating system dependent. Refer to TERAK software documentation for specifics.

The drive assemblies provide the following mechanical characteristics:

Tracks per inch: 48
Total no. of tracks: 77
Inside track radius: 2.029 in.
Read/write track width: 0.012 in.
Rotational speed: 360 RPM ±2.5%

Rotational direction (recording side): CCW

Average latency: 83.3 milliseconds

Access Times:

Track to track: 6 Milliseconds (step time, minimum)

Head settling time: 24 milliseconds (at last step, maximum)

Head load time: 50 milliseconds (from head down command, maximum).

The 8510/a diskette drive system recording mode is soft-sectored, double frequency modulation (FM). The read and write circuitry is designed to record and recover data interchangeably with an IBM 33FD, as applied in the IBM 3740 system. The interface provides write current

control to meet these IBM requirements. The 8510/a drive system is designed to the following recording characteristics:

Maximum bits/diskette: 2,650,048 (formatted)
Maximum bits/track: 26,624 (formatted)

Recording density: 6,621 bits/radian (3,248 bits/in. inside track)

Data Transfer rate: 250-K bits/second

Maximum error rates:

1 recoverable error per billion bits

1 non-recoverable error per trillion bits

These error rates exclude external causes, such as diskette defects or contamination. A non-recoverable error is defined as an error that persists after the error recovery procedure is performed. The error recovery procedure consists of 10 retries to read the record in error.

The 8510/a Diskette Controller also provides a means for automatic bootstrap of operating systems or user software from a diskette. This module supports a 64 word bootstrap. The Controller is addressed via two locations in Bank 7 of the LSI-11 address space.

The Controller provides the electronics necessary to interface the diskette drive to the LSI-11 data bus (Q bus) plus record and recover information under the IBM format.

Write Protect Option

If a disk drive is installed with the <u>Write Protect option</u> the write electronics are disabled only if a hole is provided on the jacket of a diskette. When the diskette is inserted in the 8510, the hole is positioned in a LED/photo transistor sensor assembly. If the hole is provided on the jacket, the light passes through it and activates the photo transistor circuit which disables the write electronics. If the hole is not provided, the photo transistor is not activated and the write operation can be performed on the diskette.

An interface status register bit indicates whether or not the diskette is write protected.

E2. PROCESSOR INTERFACE PROTOCOL

General

The Diskette drive Controller provides the electronics necessary to interface the diskette drive to the processor data bus plus record and recover information under the IBM format. In addition, the Controller contains 64 words (16 bits) of Read Only Memory for initial system bootstrap. Up to four disk drive units can be supported. Due to the daisy chain structure of the drive interconnect, only one drive may be addressed at a time.

Processor software support is required for the following functions:

Head initialization to Track 00, head positioning on each of the four possible drives (stepp) 3 head in and out) and head step settling timing.

Head down and settling.

Low write current selection when recording inside of track 43.

Location of the desired sector on a track.

Loading and unloading one sector (128 bytes) of information via a data buffer.

The interface consists of two device registers addressable on the processor data bus; the QX Control and Status Register (QXCS) and the QX Data Buffer (QXDB). QXDB is a 64 word (128 bytes) buffer; data is transferred to and from the disk by the QXDB. The processor writes data to be recorded into the buffer and reads recovered data from the buffer. Data is read or written as 64 consecutive 16 bit words.

Each reference to QXDB increments to the next word. In general, the buffer is initialized to the first word whenever QXCS is referenced. During the location of a sector on a track, QXDB is also used to transfer track address and sector address to the LSI-11. The following are the definitions of QXCS and QXDB and rules governing their use.

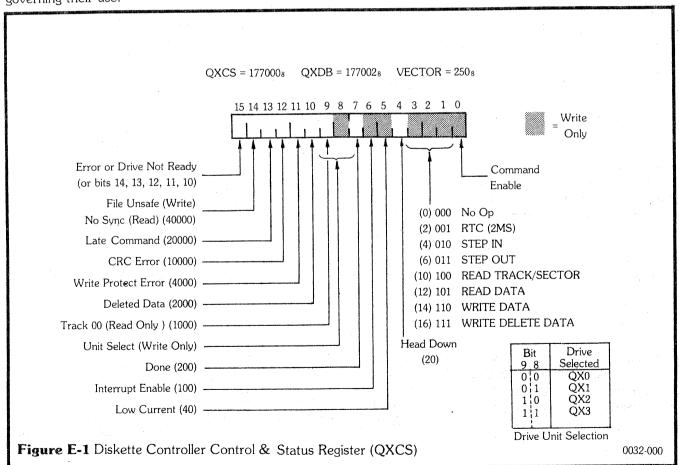
E2. 1 QX Control and Status Register (QXCS)

The format of this register is shown in Figure E-1; this register is located at 177000 (octal). While an operation is in progress, the controller will not respond to any other operations except QXCS-Read until "Done". This includes references to the resident bootstrap Read Only Memory. All attempts will result in a Bus Fault trap through location 4. If altering any bit, all bits must be reestablished even if no change is desired. QXCS is word oriented. Byte addressing will be treated as word addressing.

The bit functions of the QXCS are as follows;

QXCS Control Operations

- Bits 9 & 8: Drive unit Select (Binary coded drive select lines)
- Bit 6: Interrupt Enable (When set, will enable an interrupt to occur through location 250 (octal) upon completion of an operation or when an error occurs.)
- Bit 5: Low Current (Must be reset when writing on tracks 0-43 (decimal), (0-53 octal), and set for tracks 44-76 (decimal), (54-114 octal). Ignored during read operations.



Bit 4: Head Down

-

A. When set indicates the head is in contact with the diskette. If not previously loaded, allow 50mS for the head to load and settle before attempting read or write operations. During stepping operations, it is recommended the head be loaded approximately 4 tracks from the final destination. This is computed from: (50mS load settling 30mS

seek settling) = approx. 4 tracks
6mS track to track access

- B. The head will unload under processor control if the bit is reset. The head will unload automatically if: (1) the drive door is opened, or (2) within 500.0 to 666.8mS (3 to 4 disk revolutions) following the completion of an operation if a new operation is not started. Any reference to QXCS will reset the time out.

 NOTE: The drive door/head interlock is performed by the disk drive and will not be reflected in the QXCS register.
- C. Head control is a function of "Unit Select". Selection of a new drive will unload the head on the deselected drive.

Bits 3, 2 and 1: Function Command

Binary coded command lines. Completion of a command or occurence of an error sets "Done" (bit 7) and/or interrupts through location 250 base 8. The operation is performed once per "Command Enable" (bit 0). The codes are:

- 0) No Op: No disk drive operation. Command completes within 400mS.
- 1) Real Time Clock: No disk drive operation. Command completes within 1.8 to 2.2mS (2.0 nominal) regardless of drive ready status (Error Condition bit 15).
- 2/3) Step In/Step Out: Moves the head one track position toward/away from the center of the disk. If reexecuting, wait 6mS for track to track access. Allow an additional 24mS for track settling when the desired track is reached or if changing directions (6mS + 24mS = 30mS total on the last step). Verify track number before attempting to read or write. The real time clock function command is provided to support these timing requirements.

NOTE: When stepping after a write command, the step will not be performed until after the track trim erase has been turned off (approximately 600 μ s after "Done"). This time should be accounted for in head settling when stepping after write.

- 4) Read Track/Sector ID: Detect any address mark, then load the track and sector identity into QXDB at location 177002 (octal). If no address mark is found in 3 to 4 revolutions of the diskette, "No Sync" (Error Condition bit 14) will be set and the command terminated.
- 5) Read Data: Detect any data mark, then load the sector data into the data buffer. If the command is attempted at any time except within $300\,\mu\,\mathrm{S}$ of completion of "Read Track/Sector", "Late Command" (Error Condition bit 13) will be set and the command terminated. Detection of a Delete Data Mark will set "Delete Data" (Error Condition bit 10) and the command will continue to completion. If no Data Mark or Delete Data Mark is found within 1mS, "No Sync" (Error Condition bit 14) will be set and the command terminated.
- 6) Write Data: Sector data previously loaded into the data buffer, is written on the disk. If the command is attemped at any time except within 300 μ S of completion of "read Track/Sector". "Late Command" (Error Condition bit 13) will be set and the command terminated.
- 7) Write Delete Data: Same as "Write Data" except a Delete Data Mark will be written preceding the data instead of a Data Mark.

Bit 0: Command Enable

Setting this bit causes the command indicated by the Function Command code bits 3 through 1 to be executed on the selected drive. Once enabled, the Controller will not respond to any operation except QXCS-Read until "Done". Attempts to write QXCS, reference QXDB, or the Bootstrap Read Only Memory will result in a Bus Fault trap through location 4.

QXCS Error Conditions

Any error condition (bits 15:10), once set, will remain set until QXCS is written. When QXCS is written these bits are cleared.

Occurrence of any error (except "Delete Data") will set "Done" and interrupt through location 250 (octal) if "Interrupt Enable" (QXCS 6) is set.

Bit 15: Drive Not Ready/Lost Ready

- A. Indicates selected drive is not loaded and ready to use or selected disk drive was disturbed during a command. If awaiting a status change, QXCS must be periodically written until this bit is no longer set.
- B. Also set when any bit from 14 thru 10 is set.

Bit 14: File Unsafe/No Sync

A. File unsafe: During write operations. this bit indicates the occurrence of one of the following anditions which may jeopardize data integrity.

- (1) Write gate and no write data.
- (2) Write gate and no write current.
- (3) Write gate and no write enable (option)
- (4) Write gate and step.
- (5) Write gate and head not loaded.
- (6) Write gate and no erase gate.
- B. No Sync: During read operations, valid IBM format was not detected within 3 to 4 revolutions of the disk.

Bit 13: Late Command

Read or write data command did not arrive within 300 μ S after completion of the "Read Track/Sector ID" command.

Bit 12: CRC Error

Information read from the diskette did not generate the same check characters as were generated when data was written on the diskette.

Bit 11: Write Protect Error. Diskette file protect is enabled, (option).

Bit 10: Deleted Data. Delete data mark was detected during sector read command.

QXCS Status Operations

Bit 7: Done

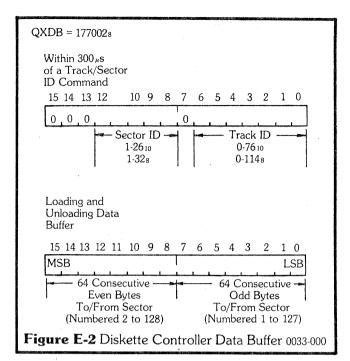
Indicates Controller is ready to receive new command. When clear, the Controller will not respond to any operation except QXCS read. Attempts to write QXCS, to read or write QXDB, or to read the bootstrap ROM, will result in a Bus Fault trap through location 4.

Bit 9: Track 00

Indicates the selected drive's read/write head is positioned over track 00.

E2.2 QX Data Buffer (QXDB)

The format of this register is shown in Figure E-2; this register is located at 177002 (octal).



QXDB Track/Sector Information

Track and Sector information is available in QXDB following a "Read Track/Sector" command; however, this information is only available during the 300 μ S interval following completion of a "Read Track/Sector" command, (the period during which a "Read Data" or "Write Data"/"Write Delete Data" must be issued to avoid a "Late Command" error condition). During this interval, QXDB is not available for transfers to/from the 64 word buffer. Also during this interval, the buffer is not initialized to the first word with a QXCS reference and referencing QXDB will not disturb the 643 word data buffer.

Bit 15 through 8: Sector Address

Indicates the sector passing beneath the read/write head (0-26 decimal or 1-32 octal).

Bits 7 through 0: Track Address

Indicates the track over which the read/write head is positioned (0-76 decimal or 0-114 octal).

QXDB Read/Write Data

Data is loaded/unloaded, to/from the data buffer through QXDB as 64 consecutive 16 bit words at any time except during the $300 \,\mu$ S following a "Read/Track/Sector" command or during any command.

Each reference to QXDB increments to the next word in the buffer. During buffer reading, however, the Diskette Controller reads one word ahead, in anticipation of the next sequential read request. Therefore, alternation of reads and writes will result in every third word being written.

QXDB is word oriented; byte addressing will be treated as word addressing.

The buffer is initialized to the first word whenever QXCS is referenced, except when "Track/Sector ID" is available.

The 128 bytes of the sector are mapped into the 64 words (16 bits each) of the buffer as follows:

1

Counting the 128 consecutive bytes of a sector from 1, the odd bytes are mapped into the lower consecutive bytes (bits 7 through 0); the even bytes are mapped into the upper consecutive bytes (bits 15 through 8). The effect of this mapping is to place the even (low order) byte of words transferred to/from memory, from/to the disk, as the physically leading byte on the disk.

Bits 15 through 8: Upper Read/Write Data Bytes. Sixty-four consecutive EVEN bytes to/from a sector, counting sector bytes from 2 to 128, decimal.

Bits 7 through 0: Lower Read/Write Data Bytes. Sixty-four consecutive ODD bytes to/from a sector, counting sector bytes from 1 to 127, decimal.

E3. INTERRUPT PRIORITY -

When the Diskette Controller is enabled for interrupts, (QXCS Bit 6), an interrupt request will be issued upon completion of termination of a command. Interrupt service is prioritized by passing the processor service grant from device to device along the data bus. The Controller is the electrically closest device and, therefore, has the highest device priority. To enable interrupt request servicing, the LSI-11 Processor Status Word priority level must be set to less than 4 (that is, Processor Status bit 7 must be 0).

E4. POWER UP

Initial position of the read/write head with respect to data tracks is indeterminate immediately after application of power. In order to assure proper positioning of the read/write head prior to any read/write operation, a Step Out operation should be performed until Track 00 (QXCS Bit 9) is set, or a single step (to align the head) followed by a "Read Track/Sector" command.

E5. ERROR RECOVERY

To guard against degradation from imperfection in the media, no more than 4 attempts to write a record should be made when read after write errors are encountered. In the event a record cannot be successfully written with 4 attempts, it is recommended that the sector or track be labled defective and an alternate sector or track be used. If more than 2 defective tracks are encountered, it is recommended the diskette be replaced.

In the event of a read error up to 10 attempts should be made to recover with re-reads. If, after 10 attempts, the data was not recovered, step the head one track away in the same direction and then re-position to recover the data.

Unloading the head when not transferring data will increase the data reliability and extend diskette life. The Controller will lift the head automatically if a new command is not received within 3 to 4 revolutions of the diskette.

E6. BOOTSTRAP SYSTEM

The Diskette Controller supports a read only memory for system bootstrapping and serialization. At absolute location 173000 and extending through location 173176 is the read only memory containing the 8510 bootstrap, (location 173000 thru 173174) and provisions for the system serial number (location 173176). A restriction on the use of this memory is that any reference to locations 173000 thru 173176 while a Controller command is in progress will result in a bus fault or loss of data. For those reasons, any reference to the Controller read only memory must be done carefully to preclude simultaneous disk I/O activity. In particular, location 173176 of the read only memory is reserved for the system serial number; it can contain a binary number unique to each 8510 processor, and can be used as required by the system software, provided the above precaution is observed. The 8510 bootstrap is initiated when:

Power is initially applied to the 8510

A power fail restart is triggered by a line voltage fluctuation

The 8510 power switch is pressed to the upward position and released

Software transfers control to location 173000

Any of the above will start the processor at absolute location 173000. A listing of the code in the read only memory is presented in Figure E-3. Since the execution of instructions from the read only memory and controller I/O activity are incompatible, the bootstrap first moves itself into low memory, at locations 10000 through 10174, and then enters this code. This bootstrap routine will then attempt to load sector 1 of track 1, from the diskette in Drive Unit QX0, into memory locations 0 thru 176. If the drive is not ready when the bootstrap is started, the boot will simply wait for a diskette to be inserted, and the drive door closed before proceeding. If I/O errors are encountered during this read activity, retrys will be attempted up to a finite limit. Each retry will jog the disk head. If excessive I/O errors are encountered, the bootstrap routine will hang the processor, and no error message will be printed on the console. Once sector 1 has been successfully loaded, the contents of location 0 (corresponding to the first word of sector 1) is verified against a fixed validation code (010001). If the correct validation code is not found, the bootstrap routine will hang the processor and no error message will print on the console. If the correct validation code is found, control will be transferred from the bootstrap routine to location 0.

At this stage of the bootstrap, any user software previously loaded into sector 1, track 1 of the disk will be executed. The first instruction (at location 0) must be a "MOV R0, R1", which is the validation code.

The bootstrap routine in locations 10000 through 10174 has been designed such that it may also be used as a read only driver by more advanced stages of the bootstrap. Table E-1 defines the entry and exit conditions for the use of this routine. Note that all processor registers are used, including the stack pointer (SP). Control is initially transferred to location 0 with all interrupts locked out. It is essential that interrupts and any other activity requiring the stack be locked until the bootstrap routine is no longer required (and the SP is correctly loaded). Note also that the bootstrap routine re-checks the validation code in location 0 each time it is called. The correct code must be left in location 0 until the bootstrap routine is no longer required. Last, note that locations 10000 thru 10174 must not be modified (e.g., by loading the advanced bootstrap) until the bootstrap routine is no longer required.

Table E-1. Entry and Exit Conditions for Bootstrap Routine

Entry: location 10110 (absolute)

Exit: location 0 (absolute)

Call: JMP @#10110

With: R 5 = Sector to be loaded (1 thru 31 octal)

R1 = Buffer address

Return: CLR PC

With: $R0 \rightarrow$ first word above end of buffer

R1 preserved R5 preserved Errors: Re-trys of any read request are automatic. If excessive I/O errors are encountered, the bootstrap routine will hang the processor. Errors are cumulative from the start of the bootstrap.

Note: All registers are used (including SP) and must be modified by user code with the exception of R0, R1 and R5. Location 0 must retain a 010001 code which is a MOV R0, R1 instruction. Thus, R1 will be updated to ascending sequential buffers unless the user code modifies R1. Sectors may be loaded in any order, but will be read only from the track 1 of unit 0.

Since most operating system bootstraps require more code that can be loaded into one sector, additional code is typically loaded into memory to continue the bootstrap. An example of such code is displayed in Figure E-4. This bootstrap will load memory locations 200 thru 1776 with the contents of sectors 3, 5, 7, 21, 23, 25 and 27. These correspond to logical blocks 0 and 2, which are the standard locations for the system bootstrap on disks operated under the RT-11 operating system. The space in sector 1 (locations 0 through 176) which is unused by the primitive bootstrap may be used for vector loading, subroutines or data for the advanced bootstrap. It should be noted that when the advanced bootstrap is entered, any I/O through the Diskette Controller to other than track 1, or using logical block addressing (requiring a mapping algorithm), must be performed by a read only handler loaded along with the advanced bootstrap; the bootstrap routine cannot be used for such purposes. For additional information on the RT-11/85 bootstrap, see the RT-11/85 System Release Notes and related software support bulletins.

Another type of bootstrap which is required by most systems is a "dummy" bootstrap that will inform the operator, by printing a message on the console, that the diskette which was mounted into Drive unit QX0, for bootstrap, does not contain a valid system bootstrap. This bootstrap is typically carried on data diskettes, or diskettes carrying files other than those required to complete the advanced bootstrap. An example of such codes is displayed in Figure E-5. This bootstrap is written into sector 1 only; additional code is not required. After the message is printed on the console terminal, the code will hang the processor, preventing operator access to micro-ODT. This precludes any potential for damaging data on a diskette by abuse of peripheral controller registry, especially, the Diskette Controller.

Some of the detail in these bootstraps is provided as a suggestion of the manner in which a user of the 8510/a system may bootstrap foreign software successfully. Details, such as the vector skipping, downward-going sector list, and filler characters need not be used in all circumstances.

E7. SERIALIZATION OF HARDWARE

Absolute address 173176 in the BOOT ROM is reserved for serialization of 8510 processors. Two other locations are also available for user installation identification. These are the low order bytes of the words at absolute locations 173064 and 173104.

The current contents of these locations are 173064/2310 and 173104/23734 but can be modified to 173064/2400 and 173104/24000 to meet requirements for additional serialization, at a very slight cost in the primary bootstrap's speed. The low order bytes of each of these locations can then be programmed to 0 thru 377. Including the standard serialization word (173176), this provides a total of 32 bits available for serialization. The same rules apply to reading the contents of these locations as apply to reading the standard serialization word.

E8. APPLICABLE DOCUMENTS

IBM Diskette OEM Information GA 21-9190-1 File No. Genl. 19

IBM 3740 Manual

RT-11/85 System Release Notes

RT-11/85 Software Support Manual

CXFROM RT-11 MACRO VM02-12 19-MAY-77 00:00:48 PAGE 1

1 .T	ITLE	CXPROM
2		
3		; COPYRIGHT (C) TERAK CORPORATION MCMLXXVII
4		; ALL RIGHTS RESERVED BY TERAK CORPORATION
5		
€		THIS SOFTWARE IS PROVIDED UNDER A LICENSE FOR USE ON A
7		;SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE
8		; INCLUSION OF THIS NOTICE. THIS SOFTWARE OR ANY COPIES
ç		;THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE
1Ø		; IN ANY WAY TO ANY PERSON EXCEPT FOR USE ON SUCH SYSTEM
11		; AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO
12		; AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
13		; REMAIN, IN TEPAK CORPORATION.
14		🛊 e de la companya del companya de la companya de la companya del companya de la companya del companya de la companya de la companya de la companya de la companya del companya de la com
15		;TERAK ASSUMES NO RESPONSIBILITY FOR THE RELIABILITY OR
16		; CONSEQUENCE OF USE OF ITS SOFTWARE ON EQUIPMENT WHICH
17		; IS NOT SUPPLIED BY TERAK. THE INFORMATION IN THIS
18		;SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD
19		; NOT PE CONSTRUED AS A COMMITMENT BY TERAK CORPORATION.
20		•

Figure E-3 Diskette Controller ROM Listing

```
CX CONTROLLER BOOTSTRAP ROM
2345€?
                              ;THIS CODE WILL RESIDE IN LOCATIONS 173000 THRU 173176.
                              OR LOCATIONS 173200 THRU 173376, DEPENDING
                              ; UPON A STRAP OPTION ON THE DISK CONTROLLER.
                              WHEN HIGHER ADDRESS IS SELECTED FOR BOOT ROM, THE
٤
                              CONTROLLER CONTROL & STATUS REGISTER ADDRESS
                              ; AND DATA REGISTER ADDRESS CHANGE ACCORDINGLY:
11
                                        BOOT PROM
                                                            CNTRL & STAT
13
                                                            177000
                                                                                250
14
15
                              ;IO:
                                        173000+
                                        173200+
                                                            177200
                                                                                254
                              ; H I :
16
                              ; POOT STARTS AT 173000 (AUTO) OR 173200 (VIA ODT); AND IMMEDIATELY MOVES ITSELF TO LOW MEMORY
17
18
                              ; (THIS IS NECESSARY DUE TO INTERLOCK OF QX CONTROLLER
19
                              ; AND POOT ROM). THE CONTROL AND STATUS REGISTER ADDRESS
20
                              ; WILL PE CALCULATED BASED UPON THE LOCATION OF THE BOOT
21
                              ; SUCH THAT A BOOT STARTED AT EITHER ADDRESS WILL
22
                               PROCEED USING THE CORRESPONDING CONTROLLER. THIS IS
23
 24
                              PROVIDED PRIMARILY FOR MANUFACTURING TEST.
25
                              THE MOVED BOOT ROUTINE WILL THEN WAIT FOR UNIT @ TO FECOME READY (DISK MOUNTED) AND THEN SEEK AND LOAD
2€
27
                              ; SECTOR 1, ON TRACK 1, ON UNIT Ø INTO LOCATIONS Ø THRU 1 ; THEN, IF LOCN Ø = Ø1ØØØ1, THE SECOND STAGE
 28
 29
                               OF POOT IS STARTED AT LOCN 0.
 30
 31
                               ; AFTER THE SECOND STAGE IS ENTERED. THE BOOTSTRAP ROUTIN ; IN LOCATIONS 10000 THRU 10176 MAY BE USED AS A READ-ONL
 32
33
                               DRIVER TO LOAD FURTHER CODE FROM SECTORS ON TRACK 1.
 34
 35
```

19-MAY-77 20:00:48 PAGE 2

Figure E-3 Diskette Controller ROM Listing

OXFROM RT-11 MACRO VM02-12

```
19-MAY 7 00:00:48 PAGE 3
        RT-11 MACRO VMØ2-12
                           .ASECT
         200000
23
                          CONTROLLER DEPENDENT EQUATES
4
56
          173000
                          OXPTLO
                                            173000
                                            177000
                          OXCSLO
          177000
                                   ==
                          BOOTLO
                                             10000
                                                     ;LOC'N FOR MOVED BOOT
7
                                   =
          010000
                                                     ;HEAD DWN/READ TRKSEC/UNIT Ø
;HEAD DWN/READ/UNIT Ø
٤
          000031
                          XRTS
                                   =
                                            31
                                            33
                          XPEAD
9
          000033
                                   =
                                            25
                                                     ; HEAD DOWN/STEP IN/UNIT 3
          000025
                          XSEEKN
10
                                            27
                                                     ;HEAD UP/STEP OUT/UNIT @
                          XSEEKO
11
          200007
                                   ==
          001000
                          TRKØØ
                                            1000
                                                     ;TRK Ø SENSE SWITCH BIT
12
13
                                                     ; ADDRESS TO "CALL" BOOT ROUTINE
                           .GLOBL BOOT7G
14
15
                                                     ; NEG DISPLACEMENT BETWEEN ROM &
          015000
                           DISPL = BOOTLO - BOOT
1€
                                                    ;USED TO CALCULATE ENTRANCE
                           BOOTTG = BOOTT + DISPL
17
          010110
18
                           .MCALL .REGDEF
19
                           .REGDEF
2Ø
21
   00000
                           ; CAUTION...ASSEMBLY REC'S QXBTLO = 173000
22
23
```

Figure E-3 Diskette Controller ROM Listing

CXBROM RT-11 MACRO VM02-12 19-MAY-77 00:00:48 PAGE 4

```
;START OF BOOT PROM
                         . = QXETLO
MTPS (PC)
         173000
                                                  ; A 12704 TO PS...NO INTRPTS
2 173000 106417 BOOT:
                         MOV #BOOTY + DISPL, R4 ;R4->POOTY IN MOVED CODE
3 173002 012704
         010010
                         MOV PC, R2
                                                  ;R4 = FOOTY
4 173006 010702
                        MOV (R2)+, (R4)+
                                                  ; MOVE A WORD
5 173010 012224 BOOTY:
                                                  ;10200 => NEG BYTE
                         TSTB R4
6 173012 105704
                                                  ;NEG BIT->64 WRDS
7 173014 100375
6 173016 000137
                         BPL BOOTY
                                                  ; JMP TO BOOTM IN MOVED CODE
                         JMP @#BOOTM + DISPL
         010022
                         FOLLOWING CODE IS ACTUALLY EXECUTED AS IF
10
                         ; A . = BOOTLO WERE INSERTED HERE
11
12
                                                   ;R4 = 10200 ON ENTRY
14 73022 062702 BCOTM: APD #CXCSLO-BOOT-200,R2 ;R2=173200 OR 173400 ON ENTRY
         ØØ3€ØØ
                                                   ; NOW R2 = C&S REG (PIC)
15
                                                  ;R5 = SECTOR TO BE FETCHED
                         MOVB (PC), R5
1€ 7302€ 111705
                                                   ;R1 = PUFFER START ADRS
17
   73030 005001
                         CLR R1
18 73032 005012 BCOTR:
                        CLR (R2)
                                                   ; NULL CMD TO CLEAR ERRORS
                                                   ; R4 BYTE = 200 = -128 DEC
19 73034 005712
                         TST (R2)
                                                   WAIT ON READY BIT
                         BLT BOOTR
20 7303€ 002775
                                                   ;R4 EYTE COUNTS # RETRIES
21 73040 105204
                         INCB R4
                                                   ; IF NOT NEG, HANG UP
22 73042 002377 HALT:
                         BGE HALT
23 73044 032712 BOOT1: BIT #TRK00, (R2)
                                                   ;TRACK Ø ???
          001000
                                                   ;YES..GO STEP IN
                          BNE BOOT4
 24 73050 001010
                                                   ; HEAD UP, STEP OUT
                          MOV #XSEEKO, (R2)
 25 73052 012712
          660067
                                                   ;WAIT FOR DONE BIT
26 73056 105712 BOOT2:
                          TSTP (R2)
 27 73060 100376
                          BPL BOOT2
 28 73062 012703
                          MOV #2310, R3
          002310
                         SOB R3, BOOT3
                                                   ; WAIT 6 MILLISECS
 29 73066 077301 EOOT3:
                                                   GO CHECK TRACK Ø SWITCH
                          BR BOOTR
 30 73070 000760
                                                   ;STEP IN, HEAD DOWN
                          MOV #XSEEKN. (R2)
 31 73072 012712 BCOT4:
          000025
                                                   ; DONE??
                          TSTB (R2)
 32 7307€ 105712 BOOT5:
 33 73100 100376
                          BPL FOOT5
                                                   ;WAIT 50 MILLISECS
                          MOV #23734, R3
 34 73102 012703
          023734
                          SOB R3, BOOT6
 35 73106 077301 BOOT6:
 36
                          FOOT? IS ENTRANCE FOR STAGE II
 37
                                                   DESTROYED
                                           RØ
                           ON ENTRY....
 38
                                                    PTS @ FIRST BUFFER WORD
                                           R1
 39
                                                    CNTRL&STATUS..DO NOT DESTROY
                                           R2
 40
                                           R3
                                                    DESTROYED
 41
                                                    RETRY COUNT..DO NOT DESTROY
                                           R4
 42
                                           R5
                                                    SECTOR NMBR TO FETCH
 43
                                           SP
                                                    DESTROYED
 44
                           RETURN IS TO LOCK Ø
```

Figure E-3 Diskette Controller ROM Listing

QXEROM RT-11 MARO VM02-12 19-MAY-77 00:00:48 PAGE 5

```
1 173110 010203 BCOT7: MOV R2, R3
                                                     ;SET UP DATA PNTR
2 173112 @12712
                          MOV #XRTS. (R2)
                                                    ; READ TPKSEC, HEAD DOWN
         000031
                                                     ; WAIT ON DONE BIT
3 17311€ 1Ø5712 BCOT8:
                          TSTB (R2)
                          BPL BOOTS
4 173120 100376
5 173122 005723
                          TST (R3)+
                                                     ; ERRORS??
                          BMI BOOTR
                                                     ; RETRY IF SO (=> STEP OUT & IN)
6 173124 100742
                                                     ;STILL TRACK 1 ??
                          CMPB (R3)+, #1
7 17312€ 122327
          000001
                          BNE BOOTR
                                                     ; RETRY IF NOT
8 173132 001337
                                                     ; CORRECT SECTOR ??
                          CMPB R5, (R3)
BNE BOOT?
9 173134 120513 BCOTB:
                                                     ; NO. . KEEP LOOKING
10 7313€ 0013€4
                                                     ; READ THEM BITS
                          MOV #XREAD, (R2)
11 73140 012712
          000033
12 73144 005303
                          DEC R3
                                                     ; ADJ TO WORD BOUNDARY
                                                     ;PUT A 100 INTO SP .
13 73146 111706
                          MOVE (PC), SP
                                                     ; PUT BUF ADR INTO
                          MOV R1, RØ
14 73150 010100
15 73152 105712 BOOT9:
                          TSTB (R2)
                                                     ; WORKING REGISTER
                                                     ;DONE??
1€ 73154 100376
                          BPL BOOT9
                          TST (R2)
BMI BOOTR
17 73156 005712
                                                     ; ERRORS ... RETRY
18 73160 100724
                          MOV (R3), (R@)+
                                                     SPREAD THEM BITS
19 73162 @11320 BOOTA:
                          SOB SP. BOOTA
CMP (SP), (PC)+
20 73164 077602
                                                     ; MOVE 64 WORDS
                                                     ; CHECK FIRST WORD
21 73166 021627
                          MOV RØ, R1
                                                     ; VALIDATION CODE
22 73170 Ø10001
                          BNE ALTO
                                                     ; IF NOT, HANG UP ; JMP TO LOCATION Ø
23 73172 001377 ALTO:
24 73174 005007
 25
          0000011
                           .END
 2€
```

Figure E-3 Diskette Controller ROM Listing

```
QXFROM RT-11 MACRO VMØ2-12
                               19-MAY-77 00:00:48 PAGE 5+
SYMBOL TABLE
        173172
                         BOOT
                                  173000
                                                   BOOTA
                                                           173162
ALTC
ECOTE
        173134
                         BOOTLO= @1@@@@
                                                   BOOTM
                                                           173022
                                                           173044
        173032
                                                   POOT1
                         POOTY
                                  173010
POOTR
                         BOOT3
                                  173066
                                                   BOOT4
                                                           173072
BOCT2
        173056
                                                   POOT?
                                                           173110
FOCT5
        17307€
                         BOOT€
                                  1731∅€
                                                           173152
                         BOOTE
                                  173116
                                                   FOOT9
POCT7G= 010110 G
                         HALT
DISPL = 015000
                                  173042
                                                   РC
                                                         =%0000007
                         QXCSL0= 177000
                                                         =%0.00000
QXETLO= 173000
                                                   PQ
                         R2
                                =%000002
                                                         =%0000003
      =2000001
                                                   R3
R1
                                =%000005
                         R5
                                                   SP
                                                         =%0000006
      =%000004
R4
                                                   XRTS
TRKØØ = ØØ1ØØØ
                         XREAD = 000033
                                                        = 000331
XSEEKN= 000025
                         XSEEKO= 000007
        17317€
                    000
. APS.
         000000
                    001
ERRCRS DETECTED: 0
FREE COFE: 17733. WORDS
.LP:=QX1:QXFROM.MAC
```

Figure E-3 Diskette Controller ROM Listing

.TITLE STANDARD SYSTEM BOOT 2 ; COPYRIGHT (C) TERAK CORPORATION MCMLXXVII ; ALL RIGHTS RESERVED BY TERAK CORPORATION 456789 THIS SOFTWARE IS PROVIDED UNDER A LICENSE FOR USE ON A ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE ; INCLUSION OF THIS NOTICE. THIS SOFTWARE OR ANY COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE ; IN ANY WAY TO ANY PERSON EXCEPT FOR USE ON SUCH SYSTEM 10 ; AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO ; AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES 11 12 ; REMAIN IN TERAK CORPORATION. 13

RT-11 MACRO VM02-12

STANDARD SYSTEM BOOT

14

15

16 17

18

; TERAK ASSUMES NO RESPONSIBILITY FOR THE RELIABILITY OR ; CONSEQUENCE OF USE OF ITS SOFTWARE ON EQUIPMENT WHICH ; IS NOT SUPPLIED BY TERAK. THE INFORMATION IN THIS ; SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD ; NOT BE CONSTRUED AS A COMMITMENT BY TERAK CORPORATION

19-MAY-77 00:03:31 PAGE 1

Figure E-4 Diskette Controller Sample System Bootstrap Listing

```
19-MAY-77 72:03:31 PAGE 2
                         RT-11 MACRO VM02-12
STANDARD SYSTEM BOOT
                                                    SAMPLE "SYSTEM" BOOTSTRAP
                          8510 DATA PROCESSOR
23
                          THE FOLLOWING CODE WILL RESIDE ON SECTOR 1, TRACK 1 OF
                                          THE REMAINING SPACE IN SECTOR 1 CAN BE
                          A SYSTEM DISK.
                          USED TO LOAD VECTORS, AND LOAD OTHER CODE, ETC.
THE LIST OF SECTORS IS IN DECENDING ORDER TO SAVE SPACE.
5€
                          THE SECTOR LIST SEOWN CORRESPONDS TO THE RT-11 LOCATION
                          FOR THE COMPLETE BOOTSTRAP, BUT MAY BE MODIFIED FOR
                          ANY OR ALL OF THE SECTORS ON TRACK 1 (@ TERU 31 OCTAL).
                          THE SECTORS WILL BE LOADED INTO ASCENDING MEMORY SPACE
12
                          FROM LOCATION 200 UP.
11
12
                          .MCALL .REGDEF
13
                          .REGDEF
14
   00000
15
16
          000000
                          .ASECT
17
18
                                                    ;FIXED ADDRESS OF BOOT ROUTINE
                          BOOTRIN = 10110
          010110
19
                                                    ; ADDRESS OF NEXT STAGE OF BOOT
                          BOOT = 200
20
          000200
                          DEFINED HERE ONLY FOR SAMPLE CODE SANS ERROR MESSAGE
21
22
23
          0.02000
                           = 0
                                                    ; VALIDATION CODE, AND
                          MOV RØ,R1
24 00000 010001
                                                    :UPDATE BUFFER POINTER
25
                                                    ; // SKIP OVER VECTORS
26
                                                     ;// FROM LOC'N 4 THRU 34
                          PR POOT1
27 00002 000416
                                                         ALLOWS LOADING OF VECTORS
                           . = 40
28
          202042
                                                     ; \\ CONCURRENT WITH BOOTING
29
30
                                                    ; NEXT SECTOR INTO R5
31 00040 113705 BOOT1: MOVB @(PC)+,F5
   20042 200177 POINTR: .WORD SECLST
                                                     ;LIST OF BYTES
                                                     ; NON-ZERO SECTOR -> CONT LOADING
33 00044 001455
                          BEQ BOOT
                                  LOAD COMPLETE, BR TO NEXT STAGE OF THE BOOTSTRAP
                  ; @ SECTOR ->
34
   00046 005367
 35
                          DEC POINTR
                                                    ; ADVANCE POINTER
          177770
 3€
                                                     ; "CALL" BOOT ROUTINE,
    00052 000137
                           JMP @#BCOTRIN
          010110
 38
                                                     ; IT RETURNS TO LOC'N Ø
 39
 40
 41
 42
                                            ; ADJUST THIS LOC'N TO REFLECT LENGTH OF
          000170
                    = 170
 43
                           .FYTE @
                                            ;LIST TERMINATOR
 44 00170
              000
                           .FYTE 27
 45
   00171
              Ø27
                           .PYTE 25
                                                             LOGICAL BLOCK 2
 46
   00172
              Ø25
                                                             UNDER RT-11
 47 00173
                           .PYTE 23
              Ø23
 48 00174
              021
                           .BYTE 21
                                                             REMAINING THREE SECTORS
 49 00175
              007
                           .EYTE 7
                           .BYTE
                                                             OF LOGICAL PLOCK &
 50 00176
              005
                                                             UNDER RT-11
              ØØ3 SECLST: .BYTE 3
 51 00177
                           ; LAST BYTE IS AT LOCATION 177 TO MAXIMIZE
 52
                           ;PROGRAM SPACE AVAILABLE IN SECTOR 1
 53
 5.4
                           .END
           000001
 55
```

Figure E-4 Diskette Controller Sample System Bootstrap Listing

```
RT-11 MACRO VMØ2-12
                                                19-MAY-77 00:03:31 PAGE 2+
STANDARD SYSTEM BOOT
SYMBOL TABLE
                        BOOTRI = 010110
                                                  BOOT1 000040
FOCT
     = 000200
                                                  RØ
R3
                        POINTR 000042
                                                        =%000000
      =20000007
PC
                        R2
                               =%0000002
                                                        =%000003
R1
      =2000001
                                                  SECLST 000177
                        R5
                               =%0000005
      =%000004
      =2000000€
SP
                   222
. AES.
        000200
                   001
        000000
ERRCRS DETECTED: @
FREE CORE: 17827. WORDS
,LP:=QX1:SBOOT.MAC
```

Figure E-4 Diskette Controller Sample System Bootstrap Listing

Figure E-5 Diskette Controller Listing of Dummy Boot Example

```
RT-11 MACRO VMØ2-12 19-MAY-77 ØØ:00:27 PAGE 2
STANDARD DUMMY BOCT
                                                              SAMPLE "DUMMY" BOOT
                                   8510 DATA PROCESSOR
23
                           THIS BOOT STRAP WHEN IN PLACE ON SECTOR 1, TRACK & OF A
                          FLOPPY DISK, WILL CAUSE THE ERROR MESSAGE TO BE PRINTED ONTO THE CONSOLE TERMINAL IF POOTSTRAP IS ATTEMPTED
                           FROM THIS DISK. THIS IS TYPICALLY USED TO PROTECT DATA
                           DISKS
                                   .REGDEF
                           .MCALL
                           .REGEFF
9 000000
10
11
                           .ASECT
          222222
12
13
14
                           TPS = 1775€4
                                                      ; CONSOLE
          177564
15
                                                      ; ADDRESS
                           TPP = 177566
          177566
16
17
18
          000000
                            . = 2
19
                                                      ; VALIDATION CODE
20 00000 010001
                           MOV RØ,R1
                                                      ;SKIP OVER PUS FAULT VECTOR
                           BR DUMMY
21 00002 000402
22
23
                                                      ;SET UP BUS FAULT VECTOR
                            .WORD HANG
24 00004 000030
                                                      ; IN CASE THERE'S NO CONSOLE
 25 00006 000340
                            .WORD 340
 2€
 27
 28 00010 012700 DUMMY: MOV
                                    #MSG.RØ
           000032
 29
 30 00014 105737 1$:
                                    @#TPS
                                                       :CONSOLE READY???
                            TSTE
           177564
 31
                                                       ; NC ... LOOP ; PRINT A BYTE
 32 00020 100375
                            BPL 1$
                                     (RØ)+,@#TPB
                            MOVB
 33 00022 112037 2$:
           177566
 34
                                                       ;LOOP TO END OF MESSAGE
 35 00026 100372
36 00030 000777 HANG:
                            BPL 1$
                                              ; KEEP OPERATOR AWAY FROM MICRO-ODT
                            BR HANG
 38
                            .NLIST BEX
 39
                            .BYTE 0,0,15,12,0,0,0,0,0
                                                               ; CR-LF AND FILLERS
 40 00032
              002 MSG:
                            .ASCII /?B-NO BOOT ON VOLUME/<15><12><200>
 41 00043
              Ø77
                            .END
           222221
 42
```

Figure E-5 Diskette Controller Listing of Dummy Boot Example

```
STANDARD DUMMY FOOT
                        RT-11 MACRO VM02-17
                                               19-MAY-77 00:00:27 PAGE 2+
SYMFOL TAPLE
                                                 MSG
                                 000030
                                                         000032
DUMMY
        000010
                        HANG
                        RØ
                                                 R1
PC
R2
     =%000007
                              =%020000
                                                       =%000001
      =%0000002
                        R3
                              =%000003
                                                 R4
                                                       =2000004
                                                 TPP
                                                       = 177566
R5
                        SP
                              =%000006
      =2000005
     = 177564
TPS
                   200
. AES. 000072
        000000
                   001
ERRCRS DETECTED: @
FREE CORE: 17735. WORDS
,LP:=QX1:DBOOT.MAC
```

Figure E-5 Diskette Controller Listing of Dummy Boot Example

APPENDIX F. ASYNCHRONOUS SERIAL INTERFACE REFERENCE GUIDE

F1. GENERAL DESCRIPTION

The 8510 unit contains an Asynchronous Serial Interface module that performs serial to parallel data conversions. The module is designed to interface peripheral devices that transmit and receive asynchronous serial data over EIA RS 232C lines or a 20 ma current loop.

This interface is configured by means of 16 switches on an External Interface Board on the rear of the 8510 unit. These switches allow the user to select the character length (7 or 8 bits), odd or even parity (or none), data rate (50 baud through 19,200 baud), device identification and device I/O address assignment. A switch also controls connection of a noise filter for suppresion of transients while driving KSR 33, KSR 35 teletypewriters using the current loop interface. The EIB switches and connectors are shown in Figure F1.

A maximum of <u>eight</u> serially interfaced peripherials can be attached to an 8510/a system (by means of the module expansion capability of the 8515 unit and additional asynchronous serial interface modules). Installation procedures are described in Section 4 of this document.

F2. SERIAL INTERFACE ADDRESSING

ş

The serial interfaces are accessed via locations in bank 7

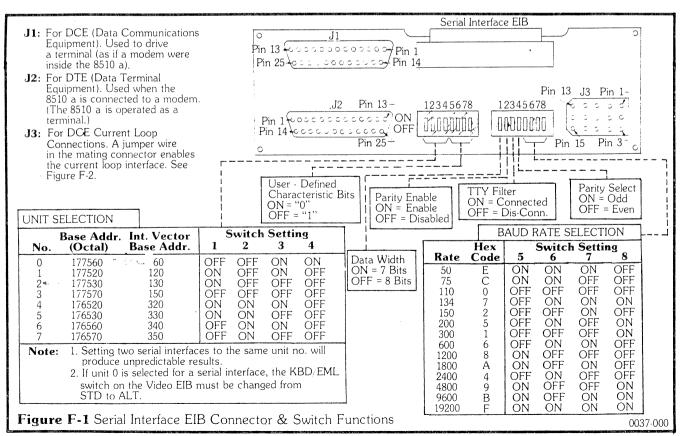
of the address space; the base addresses for the eight allowable serial interfaces are as follows:

(NOTE: All addresses are in octal)

Unit Number		Interrupt Vector Base Address
0	177560	60
1	177520	120
2	177530	130
3	177570	150
4	176520	320
5	176530	330
-6	176560	340
7	176570	350

A serial interface is programmed by means of four internal I/O registers. Eight addresses have been assigned to each serial interface: four addresses are for data & control; four others are interrupt vectors. These two groups begin at the base addresses listed above. For example, Unit 1 utilizes the following addresses:

	I	ス
Receiver Status Register (RSR)	177520	530
Receiver Data Buffer (RDB)	177522	592
Transmitter Status Register (XSR)	177524	5 34
Transmitter Data Buffer (XDB)	177526	5:16
Receiver vector & status address	120/122	
Transmitter vector & status address	124/126	



F3. RECEIVER STATUS REGISTER

The Receiver Status Register contains error information, the setting of the EIB characteristics switches, and "Done", "Interrupt Enable", and "Reader Run" bits. All bits are read-only, unless otherwise noted.

Receiver Status Register (RSR) Bit Map

- Bit 0: READER RUN (write only)
- Bit 1: Not used
- Bit 2: Not used
- Bit 3: Not used
- Bit 4: Not used
- Bit 5: Not used
- Bit 6: INTERRUPT ENABLE (read and write)
- Bit 7: DONE
- Bit 8: Controlled by EIB Switches (See Part F5.2)
- Bit 9: Controlled by EIB Switches (See Part F5.2)
- Bit 10: Controlled by EIB Switches (See Part F5.2)
- Bit 11: Controlled by EIB Switches (See Part F5.2)
- Bit 12: OVERRUN ERROR
- Bit 13: BREAK DETECTED
- Bit 14: PARITY ERROR DETECTED
- Bit 15: ERROR (Inclusive -OR of bits 14, 13, 12)
- Bit 0:READER RUN When a character is received, the Reader Run bit is cleared. When the Reader Run bit is set ON, a line in the current loop interface to KSR33,35 type teletyperwriters will enable the built-in paper tape reader to read one character. To read paper tape continuously, the Reader Run bit must be set for each character.
- Bit 7:DONE When a character is received, the Done bit will be set, indicating that the data is available in the Receiver Data Buffer. If a seven bit character is selected (see Part F5.3), only the low order seven bits of the Data Buffer are valid. If an eight bit character is selected, all eight bits of the Data Buffer are valid. If the Interrupt Enable bit has been previously written ON, an interrupt through the receiver vector address will be requested. If an error condition is detected, the Error bit and the Done bit will be set and an interrupt requested, if enabled. Three error conditions are detected.

When a system reset, power cycle, or reset instruction occurs, the Done bit will be cleared. If the Interrupt Enable bit is set while the Done bit is set, an immediate interrupt will be requested. If the Interrupt Enable bit is set while the Done bit is cleared, no immediate interrupt will occur.

- Bit 12:OVERRUN New data has been received and placed into the Data Buffer before the processor accepted the previous data. Current data is valid.
- Bit 13: BREAK The serial input line was in a continuous "MARK" condition. This is sometimes used as a reverse channel to request data transmission; it can also indicate an open line.

Bit 14: PARITY - A Parity Error was detected on the received data.

F4. TRANSMITTER STATUS REGISTER

The Transmitter Status Register contains error information, the setting of the EIB characteristics switches, and "Done", "Interrupt Enable" and Break-Transmission control bits. All bits are read-only unless otherwise noted.

- Bit 0: TRANSMIT CONTINUOUS BREAK (write only)
- Bit 1: Not used
- Bit 2: Not used
- Bit 3: Not used
- Bit 4: Not used
- Bit 5: Not used
- Bit 6: INTERRUPT ENABLE (read and write)
- Bit 7: DONE
- Bit 8: Controlled by EIB Switches (See Part F5.2)
- Bit 9: Controlled by EIB Switches (See Part F5.2)
- Bit 10: Controlled by EIB Switches (See Part F5.2)
- Bit 11: Controlled by EIB Switches (See Part F5.2)
- Bit 12: Not used
- Bit 13: Not used
- Bit 14: Not used
- Bit 15: MODEM READY (ON)
- Bit 0: TRANSMIT CONTINUOUS BREAK The Transmit Continuous Break bit, when set ON, will do just that. This is used to indicate an inactive state to the receiver of the transmitted data. The break state will be continuous until the bit is cleared off.
- Bit 7:DONE A character is transmitted when the Transmitter Data Buffer is written. Transmission is considered complete when the serial bit stream has been completely generated; then, the Done bit will be set, indicating that the transmitter can accept another character. If a seven bit character is selected (see Part F5.3), only the low order seven bits of the Data Buffer are transmitted. If an eight bit character is selected, all eight bits of the Data Buffer are transmitted. If the Interrupt Enable bit had been previously written ON, an interrupt through the transmitter vector address will be requested. Note that the Modem Ready bit has the reverse logic of the Receiver Error bits: ON implies the modem is ready; since it is the static state of the RS-232c signal, its interpretation may depend upon the specific peripheral driving the interface. This bit does not generate an interrupt; it is generally a static indication that a device is present and connected to the Serial Interface.

When a system reset, power cycle or reset instruction occurs, the Done bit will be set. If the Interrupt Enable bit is set while the Done bit is cleared, no immediate interupt will occur; if the interrupt Enable bit is set while the Done bit is set, an immediate interrupt will be requested.

F5. SERIAL EIB SWITCH FUNCTIONS

)

Sixteen switches on the Serial Interface EIBs (see Figure F1) are used for peripheral characterization. Depress the ON side of a switch to turn it ON; depress the OFF side to turn it OFF. A red marker will appear on the side opposite the selected function when the switch has completed its travel.

A maximum of eight serially interfaced peripherals (and their eight Serial Interface EIBs) can be attached to an 8510/a system. In an 8510/a system, the Video Controller module (see Appendix D), is normally assigned to be the system console (Unit 0). If a serial interface is assigned as Unit 0, the KBD/EML switch on the Video EIB must be changed from STD to ALT.

User interactive control of the 8510/a operating system is accomplished via the unit 0 device addresses; the peripheral assigned as Unit 0 must be the user's console. Assignment of the same unit number/device address to two different peripherals will produce unpredictable results.

Four switches on the Serial interface EIB are for the selection of a unit number. Other switches control character data width and parity checking, a noise filter for KSR 33, 35 TTYs and data rate selection. The TTY filter switch should be OFF except when driving KSR 33, 35 TTYs at 110 baud, or lower. The data rate (baud) switches select any of 14 rates from 50 to 19,200 baud. Two rates, 2400 and 19,200, are redundant at hex switch codes 3 and D, respectively.

F5.1 Unit Selection Switches (S1 through S4)

These switches select one-of-eight I/O addresses for the Control, Status and Data Registers of the Serial Interface. Unit 0 occupies the addresses normally used as the console terminal for control of the operating system. This should be selected (normally) only if the Serial module is to be used to interface the console terminal. The following table covers the switch patterns.

S	51) ∫ ∫ (S2	S3	S4	Unit No.	I/O Register <u>Base</u> Add <u>re</u> ss	Interrupt Vector Base Address	
0	FF	OFF	ON	ON	0	177560	60 Viáeo	Coutra
С	N	OFF	ON	OFF	1	177520	120 -	
- C	N	OFF	OFF	OFF	21-	177530	130 —	
0	FF	OFF	OFF	OFF	3	177570	150	
С	N	ON	ON	OFF	4	176520 🔨	320	
С	N	ON	OFF	OFF	5	176530	330	
0	FF	ON	ON	OFF	6	176560/	340	4.4
Ο	FF	ON	OFF	OFF	7	176570	350	

F5.2 Characteristics Switches (S5 through S8)

These may be set to any pattern. The setting of these switches will be available in both the Receiver Status Register and the Transmitter Status Register of the Serial Interface, to be used as required by the I/O software. Typically, these switches could be used to identify the type of terminal or modem attached, of alternatively, could, be used to encode a station number for clusters of 8510 systems.

Note: A switch set to the "OFF" position generates a "ONE" bit in both Status Registers (an "ON" generates "ZERO").

Switch Number Bit No. (in either Status Register)

5	11
6	10
7	9
8	8

F5. 3 Data Width (S9)

This Switch selects the number of bits used in the serial code.

Switch Function	Data Bits
ON	7 bits
OFF	8 bits

F5. 4 Parity Enable (S10)

This switch enables the transmission of parity, or the transmission of a "space" for the parity bit. On input, this switch also enables the verification of parity, setting an error condition if enabled.

Switch Function	Parity
ON	Active
OFF	Disabled

F5. 5 TTY Filter (S11)

This switch connects a noise filter for suppresion of transients while driving KSR 33, KSR 35 type teletypewriters using the current loop interface. This switch must be off except when driving such a terminal at low data speeds (typically 110 baud).

Switch Function	Filter
ON	Connected
OFF	Disconnected

F5. 6 Parity Select (S12)

This switch is active only if parity is enabled by switch 10. Even or odd parity, for both transmission and reception of data is selected.

Switch Function	Parity
ON	Odd
OFF	Even

F5. 7 Data Rate (S13 through S16)

These switches select one of the following fourteen data rates for both transmission and reception of data. The two redundant cases are included.

S14	S15	616	D 4 D 4	TO 1
	010	S16	Data Rate	, Baud
ON	ON	OFF	50	
ON	OFF	OFF	75	
OFF	OFF	OFF	110	
ON	ON	ON	134	
OFF	ON	OFF	150	
ON	OFF	ON	200	
OFF	OFF	ON	300	
ON	ON	OFF	600	
OFF	OFF	OFF	1200	
OFF	ON	OFF	1800	
ON	OFF	OFF	2400	
OFF	ON	ON	2400	(Redundant)
OFF	OFF	ON	4800	
OFF	ON	ON	9600	
ON	ON	ON	19200	
ON	OFF	ON	19200	(Redundant)
	ON OFF ON OFF ON OFF OFF OFF OFF	ON OFF OFF OFF ON ON OFF OFF OFF OFF ON ON OFF OFF OFF ON ON OFF OFF ON ON OFF OFF ON OFF OFF ON OFF OFF OFF ON ON OFF OFF ON ON OFF OFF	ON OFF OFF OFF OFF ON ON OFF ON OFF ON OFF ON OFF OFF ON OFF OFF OFF OFF OFF OFF OFF ON OFF OFF ON OFF OFF ON ON OFF OFF ON ON ON ON	ON OFF OFF 75 OFF OFF OFF 110 ON ON ON 134 OFF ON OFF 150 ON OFF ON 200 OFF OFF ON 300 ON ON OFF 600 OFF OFF OFF 1200 OFF ON OFF 1800 ON OFF OFF 2400 OFF ON ON 2400 OFF OFF ON 4800 OFF ON ON 9600 ON ON ON 19200

F6. SERIAL EIB CONNECTOR FUNCTIONS

The locations of the Serial Interface EIB connectors are shown in Figure F1. Peripheral interconnections are made via J1, J2 or J3. Suggested mating connectors for J1, J2 and J3 are as follows:

: Pins, male, crimp (MOLEX #02-06

2103) (or equivalent) : Strain relief (MOLEX #15-04-0201) (or equivalent)

F6. 1 Connector J1

The following is the J1 pin list for the RS-232C connection as Data Communications Equipment (DCE). No connections are allowed to J2 or J3.

Pin No.	Signal Name	Input/ Output
J1- 1	Frame Ground	
J1- 7	Signal Ground	
J1- 8	Carrier Detect	Output
J1- 4	Request To Send	Input
J1- 3	Receive Data	Output
J1-20	Data Terminal Ready	Input
J1- 5	Clear To Send	Output
J1- 6	Data Set Ready	Output
J1- 2	Transmit Data	Input

F6. 2 Connector J2

The following is the J2 pin list for the RS-232C connection as Data Terminal Equipment (DTE). No connections are allowed to J1 or J3.

Pin No.	Signal Name	Input/ Output
J2- 1	Frame Ground	
J2- 7	Signal Ground	
J2-20	Data Terminal Ready	Output
J2- 5	Clear To Send	Input
J2- 2	Transmit Data	Output
J2- 8	Carrier Detect	Input
J2- 6	Data Set Ready	Input
J2- 4	Request To Send	Output
J2- 3	Receive Data	Input

F6. 3 Connector J3

1

J3 can be used for RS-232-C DCE connections or for current loop connections. The following is the J3 pin list for RS-232C connection as Data Communications Equipment (DCE). No connections are allowed to J1 or J2.

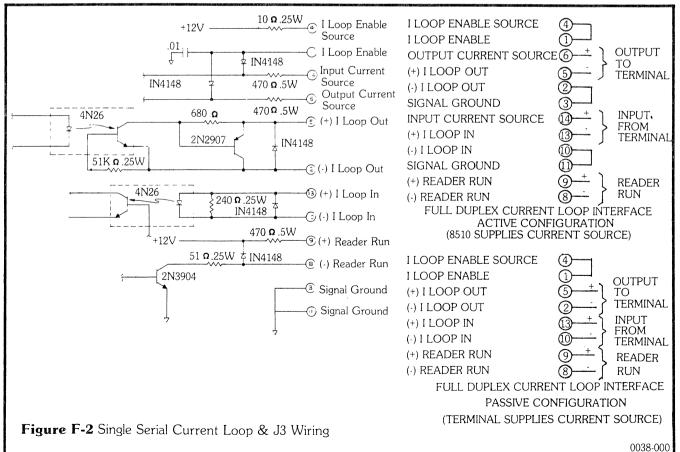
Pin No.	Signal Name	Input/ Output
J3- 3	Frame Ground	
J3-11	Signal Ground	
J3- 9	Clear To Send	Output
J3-12	Data Set Ready	Output
J3-15	Carrier Detect	Output
J3- 7	Receive Data	Output
J3- 6	Data Terminal Ready	Input
J3-14	Transmit Data	Input

The following is the J3 pin list for current loop connections. Input and output current levels (including Reader Run signal) are 20ma. No connections are allowed to J1 or J2. Either the input or output loop may be active or passive; the 20ma Enable Source pin must always be connected to the 20ma Enable pin by a jumper in the external connector. To connect either an input or output loop pair in a passive configuration: use the (+) and (-) signals, leaving the associated current source

open. To connect either the input or output loop pair in an active configuration: connect the (-) signal to ground with a jumper in the external connector; use the (+) signal as the (-) signal in the external cable; and use the associated current source as the (+) G and in the external cable. A schematic of the current loop interface is shown in Figure F2.

Note: the input and output current loops are independent. Either or none of the two current loops may be active or passive. (The Reader Run signal is always active (20ma =>Run)).

Pin No.	Signal Name	(Passive Mode)	(Active Mode)
	Current loop Enable Enable source	Jump to pin 4 Jump to pin 1	Jump to pin 4 Jump to pin 1
J3-5	(-) Current loop out (+) Current loop out Output current source	(+) in cable	Jump to pin 3 (-) in cable (+) in cable
J3-13	(-) Current loop in (+) Current loop in Input current source	(-) in cable (+) in cable No connection	Jump to pin 11 (-) in cable (+) in cable
	(+) Reader Run (-) Reader Run		
	Signal ground Signal ground		



()

APPENDIX G. PARALLEL INTERFACE REFERENCE GUIDE

G1. GENERAL DESCRIPTION

A

Sixteen-bit parallel interfaces, each consisting of a dual-height module and an External Interface Board (EIB), can be added to an 8510/a system. This TTL/DTL-compatible, word or byte oriented interface allows program-controlled data transfer rates of up to 40 K-words per second. The module contains LSI-11 bus interface and control logic for interrupt processing and vector generation. Data is handled by 16 diode-clamped input lines and 16 latched output lines. Device address is user-assigned; a 16-bit Control/Status register (DRCS) and 16-bit input/output buffers (DRIB/DROB) are compatible with PDP-11 software routines. Interconnect cables between a peripheral and an 8510 or 8515 unit can be up to 25 feet (7.6M) in length.

G2. PARALLEL INTERFACE ADDRESSING

The parallel interfaces are accessed via locations in bank 7 of the address space. The base addresses for two (optional) TERAK-installed parallel interfaces are as follows:

(NOTE: All addresses are in octal)

Unit No.	I/O Register Base Address	Interrupt Vector Base Address
0	177540	300/304
1	177440	310/314

Address assignment of additional parallel interface modules (or reassignment of the TERAK-installed modules) is at the user's discretion. The serial and parallel unit vector address assignments are suggested; only the non-serial and non-parallel address assignments in low memory are mandatory. See the assigned low memory address listing in Appendix D.

G2.1 Parallel Interface Unit Assigned Addresses

The following assignments are normally used for the two (optional) TERAK-installed parallel interface modules:

:	Control & Status Register (Output Buffer (DROB) Input Buffer (DRIB)	(DRCS)	177540 177542 177544
	Interrupt Vector (A) Interrupt Vector (B)	([.:	300 304
	DRCS DROB DRIB		177440 177442 177444
	Interrupt Vector (A) Interrupt Vector(B)		310 314

G2. 2 Jumper-Selected Addressing & Vectors

The addresses for a parallel interface module are jumper-selectable, as shown in Figure G1. An installed jumper represents a logical "0". Jumpers A3 through A12 are for I/O address decoding; jumpers V3 through V7 are for interrupt vector decoding.

If the bits controlled by jumpers A3 through A12 match the bits of an address word that the processor places on the bus, the parallel interface controlled by these jumpers will be selected. As shown by the device address control format (in Figure G1) the three least significant bits of the address word provide I/O register and highlow byte selection.

In a similar manner, when the bits controlled by V3 through V7 are matched, an interrupt will be serviced. Bit 2 of the vector address is controlled by the peripheral device; a logic "0" signifies an output data (REQ A) service request, and a logic "1" signifies an input data (REQ B) service request. These requests are prioritized; REQ A has a higher priority than REQ B.

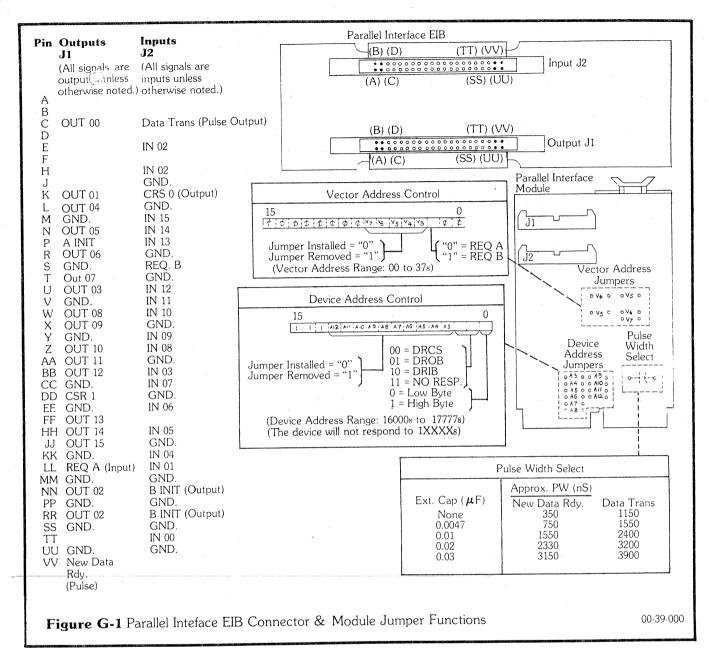
The two (optional) TERAK-installed parallel interface modules will have the following jumpers installed:

Unit 0: A7, A4, A3, V5, V4, V3. Unit 1: A7, A6, A4, A3, V5, V4.

G3. PERIPHERAL INTERFACING

Connections to a peripheral are made via J1 & J2 of the EIB. A pin number-name listing for J1 & J2 is shown in Figure G1. All interface output pins are capable of driving eight standard TTL loads, except for the following:

NEW DATA READY 10 unit loads DATA TRANS 30 unit loads A, B INIT 30 unit loads



All interface input pins are one standard TTL unit load; the IN00 through IN15 pins are protected by diode clamps.

The module contains two 16-bit buffers that transfer data between the peripheral and the data address lines (BDAL 0-15) of the system bus. User/peripheral generated request signals (REQ A, REQ B) and system generated achnowledgement pulses (NEW DATA READY, DATA TRANS) provide the handshaking for data transfers. A system generated signal (BINIT L) is provided at J1 & J2 (as A INIT H, B INIT H) for user-implemented peripheral initialization. (This signal also initializes the parallel interface modules.) Two additional program-controlled signals (CSR0, CSR1) are available for the user's application.

As indicated in Figure G1, the widths of the NEW DATA READY and DATA TRANS pulses are capacitor-

variable to accomodate different-speed peripherals. No capacitor is included on the TERAK-installed modules.

G4. OUTPUT DATA BUFFER (DROB)

This buffer can be loaded or read under program control. When DROB is loaded by the processor, a NEW DATA READY pulse is generated at J1. This signal can be used to inform the peripheral that data is available in the output buffer. To allow the data to settle on the peripheral interface cable, the trailing edge of this positive-going pulse should be used to strobe the data into the peripheral.

The 16 output lines of the output buffer are connected to the OUT00 through OUT15 pins of J1; the 16 input lines of DROB are connected to BDAL0 through BDAL15. OUT00 corresponds to BDAL0; OUT15 corresponds to BDAL15.

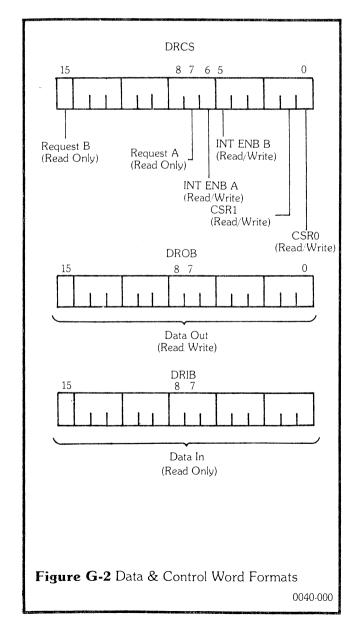
G5. INPUT DATA BUFFER (DRIB)

A CORNEL

DRIB is a read-only buffer, consisting of gated bus drivers that transfer data from the peripheral to the system bus under program control. DRIB cannot store the data, therefore, the data must remain at the inputs to this buffer until it is read by the system processor. Once read, a positive-going DATA TRANS pulse is generated; the trailing edge of this pulse signifies data transfer completion.

The 16 input lines of DRIB are connected to the IN00 through IN15 pins of J2; the 16 output lines of DRIB are connected to BDAL0 through BDAL15. IN00 corresponds to BDAL0; IN15 corresponds to BDAL15.

G6. CONTROL AND STATUS REGISTER (DRCS)



The format of DRCS is shown in Figure G2. A description of its bit functions follows:

Bit 15: REQUEST B — This bit is under control of the user's device and may be used to initiate an interrupt sequence or to generate a flag that may be tested by the program.

When used as an interrupt request, it is asserted by the external device and initiates an interrupt provided the INT ENB B (bit 05) is also set. When used as a flag, this bit can be read by the program to monitor external device status.

When the maintenance test cable is used, the state of this bit is dependent on the state of CSR1 (bit 01). This permits checking interface operation by loading a 0 or 1 into CSR1 and then verifying that REQUEST B is the same value.

Read-only bit. Cleared by INIT when in maintenance mode.

Bits 14-08: Not used, read as 0.

Bit 07: REQUEST A — Performs the same function as REQUEST B (bit 15) except that an interrupt is generated only if INT ENB A (bit 06) is also set.

When the maintenance test cable is used, the state of REQUEST A is identical to that of CSR0 (bit 00).

Read-only bit. Cleared by BINIT when in maintenance mode.

Bit 06: INT ENB A — Interrupt enable bit. When set, allows an interrupt request to be generated, provided REQUEST A (bit 07) becomes set.

Can be loaded or read by the program (read/write bit). Cleared by BINIT.

Bit 05: INT ENB B — Interrupt enable bit. When set, allows an interrupt sequence to be initiated, provided REQUEST B (bit 15) becomes set.

Bit 04 - 02: Not used, read as 0.

Bit 01: CSR1 — This bit can be loaded or read (under program control) and can be used for a user defined command to the device (appears only on Connector J1).

When the maintenance test cable is used, setting or clearing this bit causes an identical state in bit 15 which cannot be loaded by the program.

Can be loaded or read by the program (read/write bit). Cleared by INIT.

Bit 00: CSR0 — Performs the same functions as CSR1 (bit 01) but appears only on Connector J2.

When the maintenance test cable is used, the state of this bit controls the state of bit 07. (REQUEST A).

Read/write bit. Cleared by INIT.

G7. REQUEST FLAGS

The two signal lines, REQ A and REQ B, can be used by the peripheral as service request flags. REQ A is read as bit 7 in the DRCS; REQ B is read as bit 15. These signals are used in conjunction with the program-controlled INT ENB A and INI ENB B bits (6 and 5, respectively) in the DRCS to initiate interrupt request. When REQ A and INT ENB A are asserted (high), a jump to vector A (output data service) is requested. When REQ B and INT ENB B are asserted (high), a jump to vector B (input data service) is requested. The REQ A, REQ B lines must remain asserted until the interrupt processing is complete. At that time, the NEW DATA READY, DATA TRANS pulses can be used to cancel the request. The NEW DATA READY pulse is used to cancel a "A" request; DATA TRANS is used to cancel a "B" request.

G8.INITIALIZATION

The processor-generated initialize signal (BINIT L) is applied to the parallel interface modules for initialization purposes. This signal will clear bits 6, 5, 1 and 0 of the DRCS and the module's Interrupt Request and Interrupt Acknowledge flip flops. (When the parallel interface maintenance test cable is connected, BINIT L will also clear DRCS bits 15 and 7.)

The BINIT L signal is also available at the peripheral interface connectors as: A INIT H (J1, pin P) and B INIT H (J2, pins RR & NN). These signals can be used for peripheral intialization.

G9. I/O TIMING

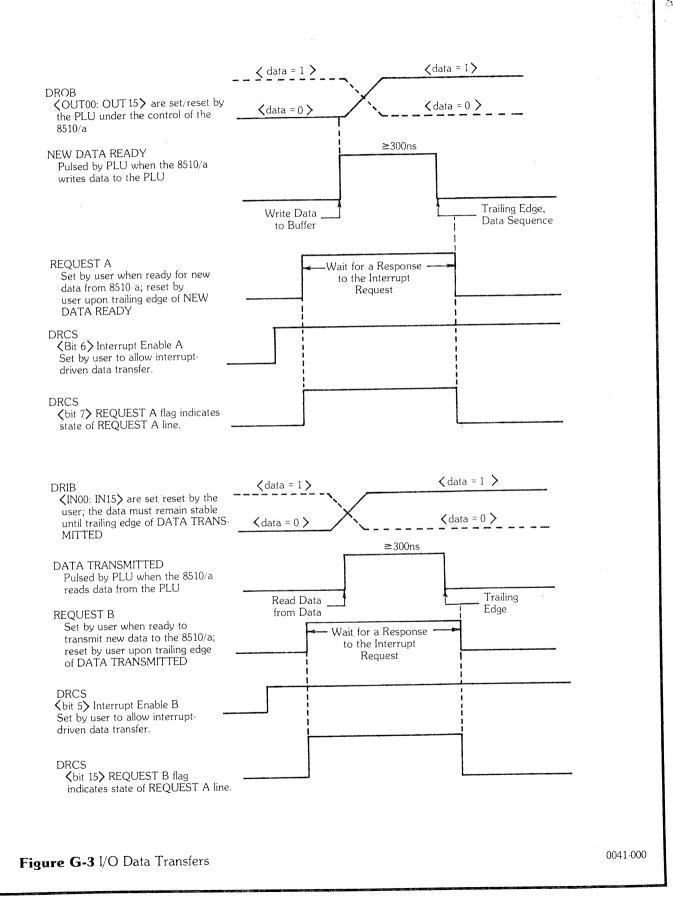
Data transfers through the parallel interface are illustrated in Figure G3.

G10. PERIPHERAL INTERCONNECTIONS

Connections to a peripheral are made via J1 and J2 of the Parallel Interface EIB. Suggested mating connectors for J1 & J2 are as follows:

J1: A female 40-pin connector (3M 3417-7000)

J2: Same as above.



1

APPENDIX H. USE OF CONTROL CONSOLE MODE (MICRO-ODT)

H1. GENERAL

de la constitución de la constit

The microcode ROMs (microms) on the processor module contain routines for on-line debugging techniques (ODT). The processor enters Control Console Mode when it is halted (executes a HALT instruction or encounters certain bus errors). The Micro-ODT firmware supports a Control Console Mode, through which all processor and device registers, and memory, may be inspected and changed, and execution resumed. This Control Console Mode is used for hardware failure diagnosis and is also useful during program development and troubleshooting. Since this mode can only be accessed when the processor is in the HALT state, the 8532 display cannot be used as the console terminal. When the processor is in the HALT state, interrupt processing stops, therefore, the terminal emulator cannot pass characters to the display screen.

A serial terminal device is required to access Micro-ODT. This terminal may be a very simple type; only standard ASCII codes are used by Micro-ODT. The serial interface must be configured for console addresses 177560 through 177566. These values are fixed in the microm firmware and address the serial interface used as the console terminal.

To use a serial interface as the console device, (Unit 0), the KBD/EML switch on the 8510 Video EIB must be changed from STD to ALT (see Appendix D) and a Serial Interface module must be selected as Unit 0 (see Appendix F) by means of its EIB switches. These changes cause the serial interface to appear at addresses 177560 through 177566, and the keyboard and emulator registers to appear at addresses 177760 through 177766.

The alternate console device must be capable of transmitting and receiving asynchronous serial data (ASCII characters) over EIA RS 232C lines or 20ma current loops. Appendix F describes the functions of the Serial EIB switches and connectors, and the attachment of a terminal to the serial interface.

H. 2 MICRO-ODT ACCESS

Micro-ODT is entered in one of the following ways:

Executing a HALT instruction

The BHALT L bus signal is asserted (HALT SW see Sect. H3.13)

A double Bus Error (Bus Error trap with SP [R6] pointing to nonexistent memory)

A Bus Error (timeout) during memory refresh

A Bus Error (timeout) when the processor is attempting to input a vector from an interrupting device.

Upon entering Micro-ODT, the processor outputs the following ASCII nonprinting and printing characters to the terminal:

<CR><LF>

nnnnnn<CR><LF>



Note: In these diagrams, all underlined symbols are typed by the processor; nonunderlined symbols are typed by the operator.

The nnnnnn is the location of the next instruction to be executed, and is always the contents of the PC (R7). The $\langle CR \rangle$ and $\langle LF \rangle$ are carriage return and line feed codes. The @ symbol is displayed as the prompt character for the operator; Micro-ODT will accept any of the commands described in this section at this point.

H3. MICRO-ODT COMMANDS

The following is a list of ODT commands and how they are used with the console terminal. Note that in the examples provided, characters output by the processor are shown underlined. Characters input by the operator are not underlined.

Note also that all commands and characters are echoed by the processor and that illegal commands will be echoed and followed by a? (ASCII 077) followed by CR (ASCII 015) followed by LF (ASCII 012) followed by @ (ASCII 100). If a valid command character is received when no location is open, the valid command character will be echoed and followed by a?, CR, LF, @. Opening nonexistent locations will have the same response. The console always prints six numeric characters as addresses or data; however, the user is not required to type leading zeros for either address or data. If a bus error (timeout) occurs during memory refresh while in the console ODT mode, a?, CR, LF, @ will be typed.

H3. 1 "/" Slash (ASCII 057)

This command is used to open a memory location, general-purpose register or the processor status word.

The / command is normally preceded by a location identifier. Before the contents of a location are typed, the console prints a space (ASCII 40) character.

example:

@ 001000/021525

where:

@ = ODT prompt character (ASCII 100)

001000 = octal location in address space to be opened / = command to open and exhibit contents of location 012525 = contents of octal location 1000

Note:

If / is used without a preceding location identifier, the address of the last opened location is used. This feature can be used to verify data just entered in a location.

H3. 2 "CR" Carriage Return (ASCII 015)

This command is used to close an open location. If the contents of the open location are to be changed, CR should be preceded by the new value. If no change to the location is necessary, CR will not alter its contents.

example:

@ 01000/<u>012525</u> <CR><<u>LF</u>>

@ /012525

or

example:

@ 001000/012525 15126421 (CR) (LF)

@ /126421

where:

CR = (ASCII 015) is used to close location 1000 in both examples. Note that in the second example, the contents of location 1000 were changed and that only the last 6 digits entered were placed in location 1000.

H3.3 "LF" Line Feed (ASCII 021)

This command is used to close an open location or GPR (general-purpose register). If entered after a location has been opened, it will close the open location or GPR and open location +2 or GPR +1. If the contents of the open location or GPR are to be modified, the new contents should precede the LF operator.

example:

@ 1000/012525 <LF> <CR> 001002/005525 <CR> <LF>

@

where:

LF = (ASCII 012) used to close location 1000 and open location 1002, if used on the PS, the LF will modify the PS if new data has been typed and close it; then, a CR, LF, @ is issued. If LF is used to advance beyond R7, the register name printed is meaningless, but the contents printed are those of R0.

H3. 4 "†" Up Arrow (ASCII 135)

The " \uparrow " command is also used to close an open location or GPR. If entered after a location or GPR has been opened, it will close the open location or GPR and open location — 2, or GPR — 1. If the contents of the open location or GPR are to be modified, the new contents should precede the " \uparrow " operator.

example:

@ 1000/012525 **† <CR> <LF>**000776/010101 **<CR> <LF>**

where:

"†" = (ASCII 135) used to close location 1000 and open location 776.

If used on the PS, the \uparrow will modify the PS if new data has been typed and close it; then CR,LF,@ is issued. If \uparrow is used to decrement below R0, the register name printed is meaningless but the contents are that of R7.

H3. 5 "@" At Sign (ASCII 100)

Once a location has been opened, the @ command is used to close that location and open a second location, using the contents of the first location as an indirect address to the second location. That is, the contents of the first location points to the second location to be opened. The contents of the first location can be modified before the @ command is used. This command is useful for stack operations.

example:

@ 1000/000200 @ <CR> <LF>
000200/000137 <CR> <LF>

where:

 $@ = (ASCII\ 100)$ used to close location 1000 and open location 200.

Note that the @ command may be used with either GPRs or memory contents.

If used on the PS, the command will modify the PS if new data is typed and close it; however, the last GPR or memory location contents will be used as a pointer.

H3. 6 "_" Underscore (ASCII 137)

This command is used once a location has been opened. ODT interprets the contents of the currently open word as an address indexed by the PC and opens the addressed location. This is useful for relative instructions where it is desired to determine the effective address.

example:

where:

"_" = (ASCII 137) used to close location 1000 and open location 1202 (sum of contents of location 1000 which is 200, 1000 and 2). Note that this command cannot be used if a GPR or PS is the open location and, if attempted, the command will modify the GPR or PS if data has been typed, and close the GPR or PS; then a CR, LF,@ will be issued.

H3. 7 "\$" Dollar Sign (ASCII 044) OR R (ASCII 122) Internal Register Designator

Either command if followe (a) a register value 0-7 (ASCII 060-067) will allow that specific general-purpose register to be opened if followed by the / (ASCII 057) command.

example:

@\$n/<u>012345</u> <CR> <<u>LF</u>>

where:

\$ = register designator. This could also be R. n = octal register 0-7.

012345 = contents of GPR n.

Note that the GPRs once opened can be closed with either the CR, LF, "1", or @ commands. The "_" command will also close a GPR but will not perform the relative mode operation.

H3. 8 "\$S" (ASCII 123) Processor Status Word

By replacing "n" in the above example with the letter S (ASCII 123) the processor status word will be opened. Again, either \$ or R (ASCII 122) is a legal command.

example:

@\$S/000200 <CR> <<u>LF</u>>

where

it from GPR's.

\$ = GPR or processor status word designator S = specifies processor status register and differentiates

000200 = eight bit contents of PS; bit 7 = 1, all other bits = 0

Note that the contents of the PS can be changed using the CR command, but bit 4 (the T bit) cannot be modified using any of the commands.

H3.9 "G" (ASCII 107)

The "G" (GO) command is used to start execution of a program at the memory location typed immediately before the "G".

example:

@100 G or 100;G

The PC (R7) will be loaded with 100, the PS is cleared, and execution will begin at that location. Immediately after echoing the "G", two null (000) characters are sent to the console terminal serial line unit (SLU) to act as fill characters in case the bus BINIT L signal clears the SLU. Before starting execution, a BUS INIT is issued for 10μ s idle time. Note that a semicolon character (ASCII 037)

can be used to separate the address from the G and this is done for PDP-11 ODT compatibility. Since the console is a character-by-character processor, as soon as the "G" is typed, the command is processed and a RUBOUT cannot be issued to cancel the command. If the B HALT L line is asserted, execution does not take place and only the BUS INIT sequence is done. The machine returns to console mode and prints the PC followed by CR, LF, @.

Note:

When program execution begins, the serial line unit is still busy processing the two null characters. Thus, the program should not assume the done bit (bit 7) is set in the output status register at 177564.

H3. 10 "P" (ASCII 120)

The "P" (Proceed) command is used to continue or resume execution at the location pointed to by the current contents of the PC (R7).

example:

@ P or ;P

If the B HALT L line is asserted, a single instruction will be executed, and the machine will return to console mode. It will print the contents of the PC followed by a CR, LF @. In this fashion, it is possible to single instruction step through a user program. However, since the B HALT L line has higher priority than device interrupts, device interrupts will not be recognized in the single step mode.

The semicolon is accepted for compatibility with other ODT programs. If the semicolon character is received during any character sequence, the console ignores it.

H3. 11 "M" (ASCII 115)

The "M" (Maintenance) command is used for maintenance purposes and prints the contents of an internal CPU register. This data reflects how the machine got to the console mode.

example:

@ M 00213 <CR> <LF>

The console prints six characters and then returns to command mode by printing CR, LF, @.

The last octal digit is the only number of significance and is encoded as follows. The value specifies how the machine got to the console mode.

Last Octal Digital Value Function

- 0 or 4 Halt instruction or B Halt line
- 1 or 5 Bus error occurred while getting device interrupt vector. This error probably indicates that the priority chain (BIAKI/O L signal) is broken in the system and that an open slot exists between modules, or a device asserting BIRQ L did not latch its request. Modules must be inserted in a contiguous fashion according to the priority daisy chain.
- 2 or 6 Bus Error occurred while doing memory refresh.
- 3 Double Bus Error occurred (stack contains nonexistent address).
- Reserved instruction trap occurred (nonexistent Micro-PD address occurred on internal CPU bus).
- 7 A combination of 1, 2 and 4, which implies that all three conditions occurred.

In the above example, the last octal digit is a "3", which indicates a Double Bus Error occurred.

The codes listed above are valid only when the console mode is entered, and the code is immediately displayed. This information is lost when a "G" command is issued; the code reflects what happened in the program since the last "G" command was issued.

H3. 12 Rubout (ASCII 177)

While RUBOUT is not truly a command, the console does support this character. When typing in either address or data, the user can type RUBOUT to erase the previously typed character and console will respond with a "Y" (Backslash - ASCII 134) for every typed RUBOUT.

example:

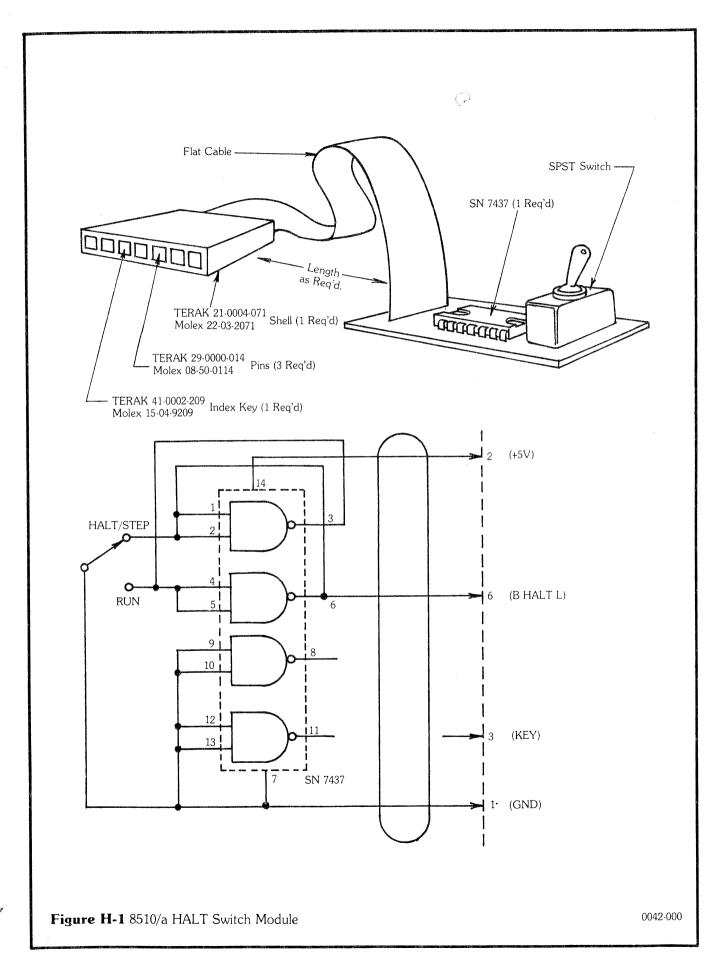
@000100/077777 123457 (RUBOUT)\6 \CR\ \LF\
@000100/123456

In the above example, the user typed a "7" while entering new data and then typed RUBOUT. The console responded with a "\" and then the user typed a "6" and CR. Then the user opened the same location and the new data reflects the RUBOUT. Note that if RUBOUT is issued repeatedly, only numerical characters are erased and it is not possible to terminate the present mode the console is in. If more than six RUBOUTS are consecutively typed, the open location will be modified with all zeros.

The RUBOUT command cannot be used while entering a register number. R2 (RUBOUT) 4 / 012345 will not open register R4; however, the RUBOUT command will cause ODT to revert to memory mode and open location

H3. 13 HALT Switch Module Details

A schematic description of the HALT switch module is shown in Figure H-1. This module connects the 8510/a maintenance connector on either the backplane (early models) or power supply boards of the 8510 unit. The module permits manual assertion of B HALT L, causing the processor to halt, and be single-instruction stepped by use of the Micro-ODT 'P' and 'G' commands. When the switch is reset to 'RUN', these commands will resume full-speed execution. The flat cable, shown in the Figure, may be dressed between the top rear surface of the chassis and the cover.



A^T QV