M8063 Falcon SBC-11/21 Single-Board Computer User's Guide

PRELIMINARY

Prepared by Educational Services
of
Digital Equipment Corporation

1st Preliminary, November 1981
2nd Preliminary, December 1981

Copyright © 1981 by Digital Equipment Corporation

All Rights Reserved

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	TOPS-20	MASSBUS
DEC	DECsystem-10	OMNIBUS
PDP	DECSYSTEM-20	OS/8
DECUS	DIBOL	RSTS
UNIBUS	EDUSYSTEM	RSX
DECnet	VAX	IAS
TOPS-10	VMS	MINC-11

CONTENTS

		Page
CHAPTER 1	**INTRODUCTION**	
1.0	GENERAL	1-1
1.1	SPECIFICATIONS	1-2
1.1.1	Physical	1-2
1.1.2	Power Requirements	1-2
1.1.3	Bus Loading	1-4
1.1.4	Environmental	1-4
1.2	BACKPLANE PIN UTILIZATION	1-5
1.3	RELATED HARDWARE MANUALS	1-6
CHAPTER 2	**INSTALLATION**	
2.0	GENERAL	2-1
2.1	SELECTING OPERATIONAL FEATURES	2-1
2.1.1	Battery Backup	2-1
2.1.2	Wake Up Circuit	2-6
2.1.3	Starting Address	2-8
2.1.4	Interrupts	2-8
2.1.5	Parallel I/O	2-10
2.1.6	Serial I/O	2-13
2.1.7	Memories	2-13
2.1.7.1	Memory Maps	2-15
2.1.7.2	PROMs/EPROMs	2-17
2.1.7.3	RAMs	2-19
2.2	SELECTING BACKPLANES AND OPTIONS	2-20
2.3	POWER SUPPLY	2-21
2.4	EXTERNAL CABLES	2-21
2.4.1	Parallel I/O Interface (J3)	2-21
2.4.2	Serial Line Interfaces (J1 and J2)	2-23
2.5	VERIFYING OPERATION	2-26
2.5.1	Macro-ODT Option	2-26
2.5.2	Loopback Connectors	2-26
2.5.3	Verification Procedure	2-26
CHAPTER 3	**OPTIONS**	
3.0	GENERAL	3-1
3.1	SUPPORTED OPTIONS	3-1
3.2	UNSUPPORTED OPTIONS	3-5
CHAPTER 4	**MACRO-ODT**	
4.0	GENERAL	4-1
4.1	INSTALLATION AND CONFIGURATION	4-1
4.2	ENTRY CONDITIONS	4-1
4.2.1	Macro-ODT Input Sequence	4-1

4.2.2 Macro-ODT Output Sequence....4-2
4.3 MACRO-ODT COMMANDS....4-2
4.3.1 /(ASCII 057) Slash....4-3
4.3.2 <CR> (ASCII 15) Carriage Return....4-4
4.3.3 <LF> (ASCII 12) Line Feed....4-4
4.3.4 R (ASCII 122) Internal Register Designator....4-6
4.3.5 S (ASCII 123) Processor Status Word....4-6
4.3.6 G (ASCII 107) Go....4-6
4.3.7 P (ASCII 120) Proceed....4-7
4.3.8 DD, DX, DY Bootstraps....4-7
4.3.9 X (ASCII 130) Diagnostics....4-9
4.4 INITIALIZATION....4-9
4.5 WARNINGS AND PROGRAMMING HINTS....4-9
4.5.1 Error Decoding....4-9
4.5.2 ODT Stack Warning....4-9
4.5.3 Addresses to Avoid....4-9
4.5.4 CPU Priority....4-9
4.5.5 Terminal Related Problems....4-10
4.5.6 Spurious HALTs....4-10
4.5.7 Serial I/O Protocol....4-10
4.5.8 Interrupt Vector Initialization....4-10

CHAPTER 5 SYSTEM ARCHITECTURE

5.0 GENERAL....5-1
5.1 MICROPROCESSOR ARCHITECTURE....5-1
5.1.1 Registers....5-1
5.1.1.1 General Registers....5-1
5.1.1.2 Status Register....5-1
5.1.2 Hardware Stack....5-2
5.1.3 Interrupts....5-4
5.2 DMA (DIRECT MEMORY ACCESS)....5-5
5.3 MEMORY ORGANIZATION....5-5
5.4 POWER UP/DOWN FACILITY....5-7

CHAPTER 6 PROGRAMMING INFORMATION

6.0 GENERAL....6-1
6.1 ASYNCHRONOUS SERIAL LINE UNITS....6-1
6.1.1 Data Baud Rates....6-3
6.1.2 Interrupts....6-8
6.2 PROGRAMMING THE PARALLEL I/O INTERFACE....6-8
6.2.1 How To Use This Section of The Manual....6-10
6.2.2 Modes of Operation....6-10
6.2.2.1 Port C Register....6-10
6.2.2.2 Mode 0 Basic Input/Output....6-10
6.2.2.3 Port A and B Registers....6-10
6.2.2.4 Port C Register in Mode 0....6-13
6.2.2.5 Mode 1 (Strobed Input/Output)....6-14

6.2.2.6 Mode 2 (Strobed Bidirectional I/O).....6-14
6.2.3 Control Word Register.....6-21
6.2.3.1 Mode Selection.....6-25
6.2.3.2 Setting Bits in Port C.....6-25
6.2.4 Parallel I/O Initialization.....6-27
6.2.5 Parallel I/O Handshaking.....6-27

CHAPTER 7 ADDRESSING MODES AND INSTRUCTION SET

7.0 GENERAL.....7-1
7.1 ADDRESSING MODES.....7-1
7.1.1 Single Operand Addressing.....7-3
7.1.2 Double Operand Addressing.....7-4
7.1.3 Direct Addressing.....7-5
7.1.3.1 Register Mode.....7-6
7.1.3.2 Autoincrement Mode.....7-8
7.1.3.3 Autodecrement Mode (Mode 4).....7-10
7.1.3.4 Index Mode (Mode 6).....7-11
7.1.4 Deferred (Indirect) Addressing.....7-13
7.1.5 Use of the PC as a General Register.....7-17
7.1.5.1 Immediate Mode.....7-18
7.1.5.2 Absolute Addressing.....7-19
7.1.5.3 Relative Addressing.....7-21
7.1.5.4 Relative Deferred Addressing.....7-22
7.1.6 Use of Stack Pointer as General Register.....7-22
7.2 INSTRUCTION SET.....7-23
7.2.1 Instruction Formats.....7-24
7.2.1.1 Byte Instructions.....7-26
7.2.2 List of Instructions.....7-27
7.2.3 Single Operand Instructions.....7-29
7.2.3.1 General.....7-29
7.2.3.2 Shifts & Rotates.....7-33
7.2.3.3 Multiple Precision.....7-38
7.2.3.4 PS Word Operators.....7-41
7.2.4 Double Operand Instructions.....7-42
7.2.4.1 General.....7-42
7.2.4.2 Logical.....7-46
7.2.5 Program Control Instructions.....7-49
7.2.5.1 Branches.....7-49
7.2.5.2 Signed Conditional Branches.....7-54
7.2.5.3 Unsigned Conditional Branches.....7-57
7.2.5.4 Jump & Subroutine Instructions.....7-59
7.2.5.5 Traps.....7-64
7.2.5.6 Reserved Instruction Traps.....7-68
7.2.5.7 Halt Interrupt.....7-68
7.2.5.8 Trace Trap.....7-68
7.2.5.9 Power Failure Interrupt.....7-69
7.2.5.10 Interrupts.....7-69

7.2.5.11 Special Cases T-bit....7-69
7.2.6 Miscellaneous Instructions....7-70
7.2.7 Condition Code Operators....7-71

CHAPTER 8 THEORY OF OPERATION

8.0 GENERAL....8-1
8.1 MICROPROCESSOR....8-1
8.1.1 Microprocessor Initialization....8-4
8.1.1.1 RESET....8-4
8.1.1.2 Power Up....8-5
8.1.2 Clock Input....8-5
8.1.3 Ready Input....8-5
8.1.4 Microprocessor Control Signals....8-5
8.1.4.1 Row Address Strobe (RAS)....8-6
8.1.4.2 Column Address Strobe (CAS)....8-6
8.1.4.3 Priority In (PI)....8-6
8.1.4.4 Read/Write (R/-WHB and R/-WLB)....8-6
8.1.4.5 Select Output Flags (SEL0 and SEL1)....8-6
8.1.4.6 Bus Clear (BCLR)....8-6
8.1.4.7 Clock Out (COUT)....8-6
8.1.5 Microprocessor Transactions....8-6
8.1.5.1 Fetch/Read....8-7
8.1.5.2 Write....8-7
8.1.5.3 IAK....8-10
8.1.5.4 DMA....8-11
8.1.5.5 ASPI....8-12
8.1.5.6 NOP....8-12
8.2 MODE REGISTER CONTROL....8-13
8.3 INTERRUPT CONTROL....8-15
8.3.1 Details of Interrupt Control Logic....8-17
8.3.2 READY....8-19
8.3.3 IAK DATA IN (IAKDIN)....8-21
8.3.4 HALT Interrupt....8-21
8.3.5 Power Fail....8-24
8.3.6 Local....8-24
8.3.7 External....8-25
8.3.8 DMA Interrupt....8-25
8.4 DC004 PROTOCOL....8-25
8.5 ADDRESS LATCH....8-26
8.6 MEMORY ADDRESS DECODE....8-26
8.7 RAM MEMORY....8-26
8.8 ROM/RAM MEMORY SOCKETS....8-27
8.9 SERIAL LINE INTERFACE UNITS....8-29
8.10 PARALLEL I/O INTERFACE....8-31
8.11 POWER UP....8-33
8.12 CLOCK....8-34
8.13 CLOCK CONTROL....8-35
8.14 DMA....8-37

8.15 TSYNC....8-38
8.16 READ/WRITE....8-39
8.17 REPLY TIMEOUT....8-41
8.18 BUS CONTROL....8-42

CHAPTER 9 LSI-11 BUS

9.0 GENERAL....9-1
9.1 SBC-11/21 SINGLE BOARD COMPUTER....9-2
9.2 MASTER/SLAVE RELATIONSHIP....9-2
9.3 DATA TRANSFER BUS CYCLES....9-4
9.3.1 Bus Cycle Protocol....9-5
9.3.1.1 Device Addressing....9-5
9.3.2 Direct Memory Access....9-11
9.4 INTERRUPTS....9-12
9.4.1 Device Priority....9-13
9.4.2 Interrupt Protocol....9-13
9.5 CONTROL FUNCTIONS....9-15
9.5.1 Halt....9-16
9.5.2 Initialization....9-16
9.5.3 Power Status....9-16
9.5.4 Power-Up/Down Protocol....9-16
9.6 LSI-11 BUS ELECTRICAL CHARACTERISTICS....9-17
9.7 MODULE CONTACT FINGER IDENTIFICATION....9-17

APPENDIX A INSTRUCTION TIMING

APPENDIX B PROGRAMMING DIFFERENCE LIST

APPENDIX C SOFTWARE DEVELOPMENT

APPENDIX D MACRO-ODT ROM

APPENDIX E SBC-11/21 SCHEMATICS

FIGURES

Figure No.	Title	Page
1-1	KXT11-AA (M8063) SBC-11/23 Module	1-3
2-1	SBC-11/21 Module Layout	2-2
2-2	Interrupt Configurations	2-9
2-3	Time-out During LSI-11 Bus Interrupt Acknowledge	2-9
2-4	Parallel I/O Configuration	2-11
2-5	Socket Sets A and B Interconnection	2-15
2-6	Memory Configuration	2-16
2-7	Memory Maps	2-17
2-8	30 Pin Parallel I/O Connector	2-22
2-9	10 Pin Serial Line Unit Connector	2-24
2-10	BC20N-05 "Null Modem" Cable	2-25
2-11	BC21B-05 Modem Cable	2-25
5-1	Registers and Processor Status Word	5-2
5-2	Memory Maps	5-6
6-1	Serial Line Unit Interface (SLU)	6-2
6-2	Serial Line Unit Register Bit Maps	6-4
6-3	Parallel I/O Interface	6-9
6-4	Parallel I/O Flowchart Guide	6-11
6-5	Mode 0 Port A or B, Bit Assignments	6-12
6-6	Mode 0 Port C Bit Assignments	6-13
6-7	Mode 1 Port C Bit Assignments	6-14
6-8	Mode 2 Port C Bit Assignments	6-22
6-9	Mode 1 Input Data Handshaking Sequence	6-28
6-10	Mode 1 Strobed Input Timing	6-28
6-11	Mode 1 Output Data Handshaking Sequence	6-29
6-12	Mode 1 Strobed Output Timing	6-30
6-13	Mode 1 Port B Strobed Output Timing	6-30
6-14	Mode 2 Port A Bidirectional Timing	6-32
7-1	Single Operand Addressing	7-3
7-2	Double Operand Addressing	7-4
7-3	Mode 0 Register	7-5
7-4	Mode 2 Autoincrement	7-5
7-5	Mode 4 Autodecrement	7-6
7-6	Mode 6 Index	7-6
7-7	INC R3 Increment	7-7
7-8	ADD R2, R4 Add	7-7
7-9	COMB R4 Complement Byte	7-8
7-10	CLR (R5)+ Clear	7-8
7-11	CLRB (R5)+ Clear Byte	7-9
7-12	ADD (R2)+ R4 Add	7-9
7-13	INC -(R0) Increment	7-10
7-14	INCB -(R0) Increment Byte	7-10
7-15	ADD -(R3), R0 Add	7-11
7-16	CLR 200 (R4) Clear	7-12
7-17	COMB 200 (R1) Complement Byte	7-12

7-18 ADD 30 (R2, 20 (R5) Add....7-13
7-19 Mode 1 Register Deferred....7-14
7-20 Mode 3 Autoincrement Deferred....7-14
7-21 Mode 5 Autodecrement Deferred....7-14
7-22 Mode 7 Index Deferred....7-15
7-23 CLR @ R5 Clear....7-15
7-24 INC @ (R2)+ Increment....7-16
7-25 COM @ -(R0) Complement....7-16
7-26 ADD @ 100 (R2), R1 Add....7-17
7-27 ADD # 10, R0 Add....7-19
7-28 CLR @ # 1100 Clear....7-20
7-29 ADD @ # 2000 Add....7-20
7-30 INC A Increment....7-21
7-31 CLR @ A Clear....7-22
7-32 Single Operand Group....7-24
7-33 Double Operand Group....7-24
7-34 Program Control Group Branch....7-24
7-35 Program Control Group JSR....7-24
7-36 Program Control Group (RTS)....7-25
7-37 Program Control Group Traps....7-25
7-38 Program Control Group Subtract....7-25
7-39 Operate Group....7-25
7-40 Condition Group....7-25
7-41 Byte Instructions....7-26
7-42 CLR....7-29
7-43 COM....7-30
7-44 INC....7-31
7-45 DEC....7-31
7-46 NEG....7-32
7-47 TST....7-33
7-48 ASR....7-34
7-49 ASR Description....7-34
7-50 ASL....7-34
7-51 ASL Description....7-35
7-52 ROR....7-35
7-53 ROR Description....7-36
7-54 ROL....7-36
7-55 ROL Description....7-37
7-56 SWAB....7-37
7-57 Multiple Precision....7-38
7-58 ADC....7-39
7-59 SBC....7-40
7-60 SXT....7-40
7-61 MFPS....7-41
7-62 MTPS....7-42
7-63 MOV....7-42
7-64 CMP....7-44
7-65 ADD....7-44
7-66 SUB....7-45

7-67 BIT....7-46
7-68 BIC....7-47
7-69 BIS....7-48
7-70 XOR....7-48
7-71 BR....7-50
7-72 BNE....7-51
7-73 BEQ....7-51
7-74 BPL....7-52
7-75 BMI....7-52
7-76 BVC....7-53
7-77 BVS....7-53
7-78 BCC....7-54
7-79 BCS....7-54
7-80 BGE....7-55
7-81 BLT....7-56
7-82 BGT....7-56
7-83 BLE....7-57
7-84 BHI....7-57
7-85 BLOS....7-57
7-86 BHIS....7-58
7-87 BLO....7-58
7-88 JMP....7-59
7-89 JSR....7-60
7-90 JSR Example....7-62
7-91 RTS....7-63
7-92 RTS Example....7-63
7-93 SOB....7-64
7-94 EMT....7-64
7-95 EMT Example....7-65
7-96 TRAP....7-66
7-97 BPT....7-66
7-98 IOT....7-67
7-99 RTI....7-67
7-100 RTT....7-68
7-101 HALT....7-70
7-102 WAIT....7-70
7-103 RESET....7-71
7-104 MFPT....7-71
7-105 Condition Code Operators....7-71
8-1 SBC-11/21 Functional Block Diagram....8-2
8-2 SBC-11/21 Microprocessor....8-4
8-3 Fetch/Read Transaction....8-8
8-4 Write Transaction....8-9
8-5 IACK Transaction....8-10
8-6 DMA Transaction....8-11
8-7 ASPI Transaction....8-12
8-8 BUS NOP Transaction....8-13
8-9 Mode Register Control....8-14
8-10 SBC-11/21 Interrupt Logic....8-16

8-11 Interrupt Control....8-17
8-12 Ready....8-20
8-13 IAKDIN....8-22
8-14 HALT Interrupt....8-23
8-15 Memory Maps....8-27
8-16 RAM Memory....8-28
8-17 ROM/RAM Memory Sockets....8-29
8-18 Serial Line Interface Units....8-30
8-19 Parallel I/O Interface....8-32
8-20 Power Up....8-33
8-21 Clock....8-35
8-22 Clock Control....8-36
8-23 DMA....8-37
8-24 TSYNC....8-39
8-25 Read/Write....8-40
8-26 Reply Timeout....8-41
8-27 Bus Control....8-42
9-1 DATI Bus Cycle....9-6
9-2 DATI Bus Cycle Timing....9-8
9-3 DATO or DATOB Bus Cycle....9-9
9-4 DATO or DATOB Bus Cycle Timing....9-10
9-5 DMA Protocol....9-12
9-6 DMA Request/Grant Timing....9-13
9-7 Interrupt Request/Acknowledge Sequence....9-14
9-8 Power-Up/Power-Down Timing....9-17
9-9 Double-Height Module Contact Finger Identification....9-18
C-1 Overview of Software Development....C-4
C-2 Application Overview....C-7
C-3 Monitor Program....C-8
C-4 Load Map....C-10
C-5 Power Up Task....C-10
C-6 Power Fail Recovery....C-10
C-7 SLU Diagnostic Task....C-11
C-8 RAM Diagnostic Task....C-12
C-9 ROM Diagnostic Task....C-12
C-10 Parallel I/O Diagnostic Task....C-13
C-11 Control Task....C-14
C-12 Power Fail Task....C-15

TABLES

Table No. Title Page

1-1 SBC-11/21 Module Backplane Pin Utilization....1-5
2-1 Configuration Pin Definitions....2-3
2-2 Standard Factory Configuration....2-7
2-3 Mode Register Configuration....2-8

2-4 Mode 0 Buffer Configuration (No Handshake).....2-12
2-5 Mode 1 Buffer Configuration (Strobed I/O).....2-13
2-6 Mode 2 Buffer Configuration and Handshake.....2-14
2-7 Jumper Connection and BREAK Response.....2-14
2-8 Memory Map Configurations.....2-16
2-9 Socket Set A Configuration for EPROM/PROM.....2-18
2-10 Socket Set B Configuration for EPROM/PROM.....2-19
2-11 Socket Set A Configuration for RAM.....2-20
2-12 Socket Set B Configuration for RAM.....2-20
2-13 EIA Slew Rate Resistor Values.....2-23
2-14 Diagnostic Fault Indicators.....2-28
3-1 Unsupported LSI-11 Options.....3-5
4-1 Macro-ODT Commands.....4-3
4-2 Macro-ODT States and Valid Input Characters.....4-5
5-1 Processor Status Word Bit Descriptions.....5-3
5-2 PSW Interrupt Levels.....5-4
5-3 SBC-11/21 Interrupts.....5-7
6-1 Serial Line Unit Register Addresses.....6-1
6-2 Receiver Control and Status Bit Descriptions.....6-3
6-3 Receiver Data Buffer Bit Descriptions.....6-5
6-4 Transmitter Control and Status Bit Description.....6-6
6-5 Transmitter Data Buffer Bit Descriptions.....6-7
6-6 Parallel I/O Register Addresses.....6-8
6-7 Mode 0 Configuration.....6-12
6-8 Mode 0 Port A or B Bit Descriptions.....6-12
6-9 Mode 0 Port C Bit Descriptions.....6-13
6-10 Port C Control Signals in Mode 1.....6-15
6-11 Combinations of Mode 1.....6-16
6-12 Mode 1 Port C Bit Descriptions.....6-17
6-13 Mode 1 Configuration.....6-21
6-14 Port C Control Signals in Mode 2.....6-22
6-15 Mode 2 Port C Bit Descriptions.....6-23
6-16 Mode 2 Configuration.....6-24
6-17 Control Register Mode Selection Bit Functions.....6-24
6-18 Control Words for Mode Selection.....6-25
6-19 Control Register Bit Set/Reset Bit Functions.....6-26
6-20 Interrupt Set/Reset Control Words.....6-26
6-21 Mode 1 Input Handshaking Signals.....6-27
6-22 Mode 1 Output Handshaking Signals.....6-29
6-23 Mode 2 Bidirectional Handshaking Signals.....6-31
8-1 Start Address Configurations.....8-15
8-2 Designated Interrupts.....8-18
8-3 Serial Line Unit Registers.....8-31
8-4 PPI Addressable Registers.....8-32
9-1 Signal Assignments.....9-3
9-2 Data Transfer Operations.....9-4
9-3 Bus Signals Used in Data Transfer Operations.....9-4
9-4 Bus Pin Identifiers.....9-19

PREFACE

This manual provides the user with configuration, system architecture and programming information for the SBC-11/21. The configuration requirements are described in Chapter 2 and the system architecture is presented in Chapter 5. The programming techniques are described in Chapter 6 and the instruction set is listed in Chapter 7. The operational theory is presented in Chapter 8 and the schematics are in Appendix E. The Macro-ODT option is described in Chapter 4 and the listing is in Appendix D. Options for use on the LSI-11 bus are listed in Chapter 3 and the module bus requirements are described in Chapter 9. An example of software development is covered by Appendix C. Appendix A summarizes the instruction timing and Appendix B compares the SBC-11/21 to other LSI-11 microprocessors.

NOTE

This manual is to be used only for the SBC-11/21 module, M8063 Revision C. This revision can be identified by the circuit board #501448C located on the module as described in Figure 1-1.

CHAPTER 1
INTRODUCTION

1.0 GENERAL

The KXT11-AA (M8063) module is shown in Figure 1-1 and designated as the SBC-11/21 single board computer. It is a complete computer system on an 8.5 X 5.2 inch printed circuit board that executes the well known PDP-11 instruction set (see Appendix B). The SBC-11/21 module contains 4 Kb of RAM, sockets for up to 32 Kb of PROM or additional RAM, two serial I/O lines, 24 lines of parallel I/O and a 50 Hz, 60 Hz or 800 Hz real time clock. In addition, the SBC-11/21 is equipped with a complete LSI-11 bus interface which enables it to communicate with most of DIGITAL's large family of modules (see Chapter 3) as described in the Microcomputer Interfaces and Microcomputers and Memories handbooks.

The SBC-11/21 features the following:

- A powerful processor running the PDP-11 instruction set.
- Direct addressing of 32K, 16-bit words, or 64K 8-bit bytes (K = 1024).
- Efficient processing of 8-bit characters without the need to rotate, swap or mask.
- On-board 4K bytes of static read/write memory.
- Sockets for up to 32K bytes of PROM, jumper configurable for a wide range of memory types from several vendors. Additional RAM can also be installed in these sockets.
- Hardware memory stack for handling structured data, subroutines, and interrupts.
- Direct-memory access for high-data-rate devices inherent in the bus architecture.
- Eight general-purpose registers that are available for data storage, pointers and accumulators; two are dedicated: SP and PC.
- Fast on-board bus for high throughput when external memory access is not required.
- LSI-11 bus structure that provides position-dependent priority as peripheral device interfaces are connected to the I/O bus.
- Fast interrupt response without device polling.
- A powerful and convenient set of programming

instructions.

- Two serial I/O interfaces, compatible with EIA RS-232C and EIA RS-423, with software programmable baud rates over the range of 300 to 38,400 baud.

- One parallel I/O interface, with two bidirectional 8-bit input/output ports, and one 8-bit control port.

- Real-time clock which can be set by the user to 50 Hz, 60 Hz or 800 Hz.

- Many jumper-configurable operating modes, including four different memory maps, exception handling, start and restart address, parallel I/O configurations, and real-time clock frequency.

- Optional PROM resident Macro-ODT containing module diagnostics, bootstrap programs for mass storage devices (TU58, RX02 and RX02), console communications and on-line debugging facility.

1.1 SPECIFICATIONS

The SBC-11/21 module specifications are described below:

1.1.1 Physical

Height	13.2 cm (5.2 in)
Length (includes module handle)	22.8 cm (8.9 in)
Width	1.27 cm (0.5 in)
Weight	360 gm (12 oz) max

1.1.2 Power Requirements

Power Supply:

+5.0 V $\pm$ 5%	2.5 A (Typical) 2.8 A (Maximum)
+12.0 V $\pm$ 5%	60 mA (typical) used by on-board circuitry 1.1 A (Maximum) includes current supplied to outside world through pin 10 of the serial I/O connector

Battery Backup:

+5.0 V $\pm$ 5%	170 mA (Typical) 260 mA (Maximum)

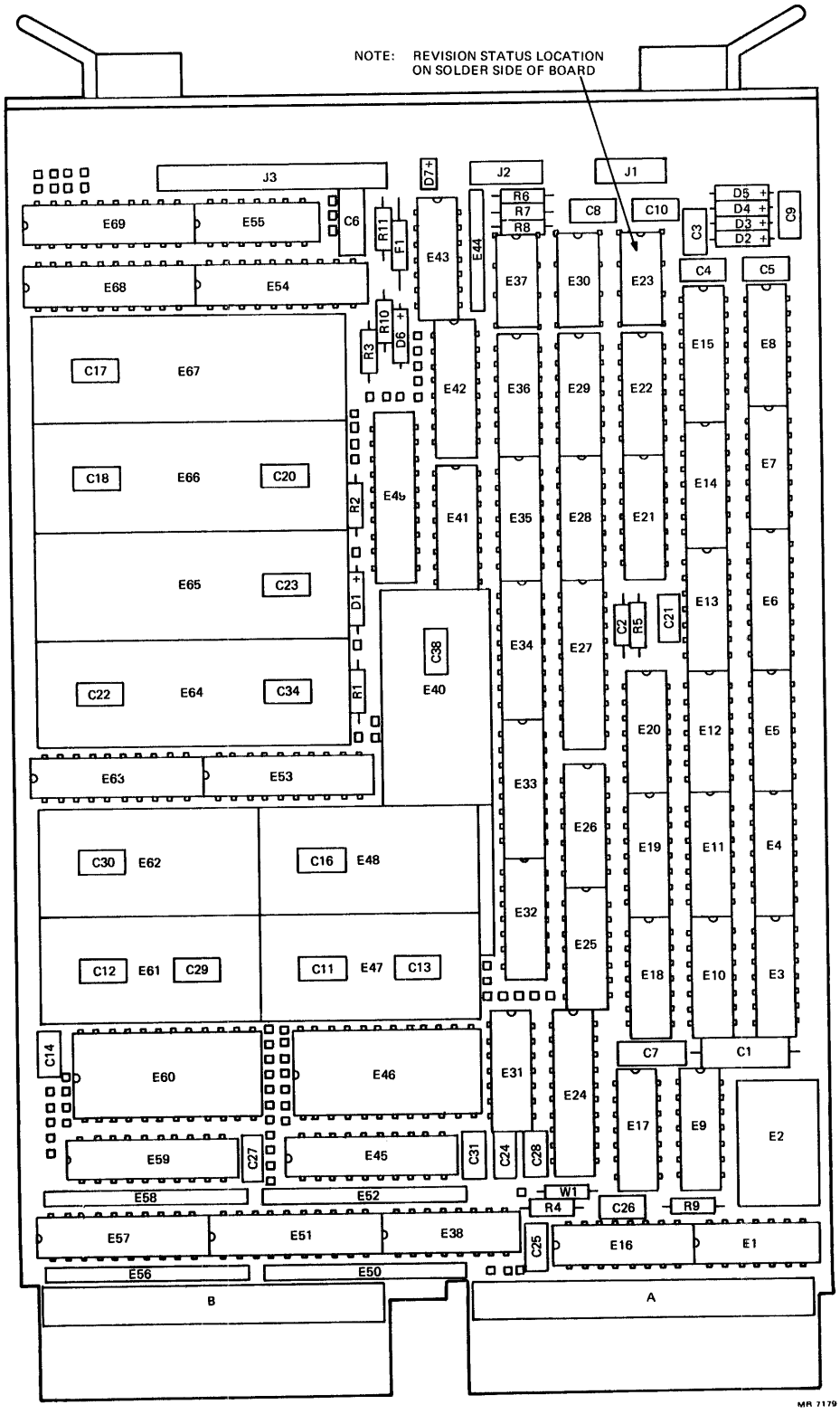


Figure 1-1 KXT11-AA (M8063) SBC-11/21 Module

NOTE

The +12.0 V typical current is measured with no connections at pin 10 of the serial I/O connectors (fused line).

1.1.3 Bus Loading

A.C. Loads	2.4
D.C. Loads	1.0

1.1.4 Environmental

Temperature:

Storage	-40° to 65° C (-40 to 150° F)
Operating	5° to 60° C (41° to 140° F)

NOTE

The module must be brought into the operating temperature environment and allowed to stabilize before operating.

Relative Humidity:

Storage	10% to 90% (no condensation)
Operating	10% to 90% (no condensation)

Altitude:

Storage	Up to 15 Km (50,000 ft)
Operating	Up to 15 Km (50,000 ft) (90 mm mercury minimum)

NOTE

Derate the maximum operating temperature by 1° C (1.8° F) for each 300 m (1000 ft) of altitude above 2.4 Km (8000 ft).

Environment: Air must be non-caustic.

Airflow: (operating)

Adequate airflow must be provided to limit the inlet to outlet temperature rise across the module to 5° C (9° F) when the inlet temperature is 60° C (140° F). For operation below 55° C (131° F), airflow must be provided to limit the inlet to outlet temperature rise across the module to 10° C (18° F) maximum.

NOTE

These are design limits. Lower temperature limits will serve to increase the life of the product.

1.2 BACKPLANE PIN UTILIZATION

Backplane pin connections for the SBC-11/21 module are listed in Table 1-1. The table also includes pin utilization and signal names unique to the SBC-11/21 module and a list of standard LSI-11 bus backplane names associated with each pin. Note that although the signal names may differ, the module is completely LSI-11 bus compatible, with the exception of bus refresh transaction (BREF), not performed by the SBC-11/21. Signals STOP L, SRUN L, and START L are not used on the LSI-11 bus. These are TTL level signals unique to the SBC-11/21.

Table 1-1 SBC-11/21 Module Backplane Pin Utilization

Side 1 (Component Side)

Backplane Pin	SBC-11/21 Signal Function	LSI-11 Bus Signal Name
AA1	Bus terminator	BIRQ5 L
AB1	Bus terminator	BIRQ6 L
AC1	Bus terminator	BDAL16 L
AD1	Bus terminator	BDAL17 L
AE1	STOP L	SSPARE1
AF1	SRUN L	SSPARE2
AH1	Not connected	SSPARE3
AJ1	GND	GND
AK1	Not connected	MSPAREA
AL1	GND	MSPAREA
AM1	GND	GND
AN1	BDMR L	BDMR L
AP1	BHALT L	BHALT L
AR1	Bus terminator	BREF L
AS1	Not connected	+12B
AT1	GND	GND
AU1	Not connected	PSPARE1
AV1	+5 VB (battery)	+5B
BA1	BDCOK H	BDCOK H
BB1	BPOK H	BPOK H
BC1	Bus terminator	SSPARE4
BD1	Bus terminator	SSPARE5
BE1	Bus terminator	SSPARE6
BF1	Bus terminator	SSPARE7
BH1	START L	SSPARE8
BJ1	GND	
BK1	Not connected	MSPAREB
BL1	Not connected	MSPAREB
BM1	GND	
BN1	BSACK L	BSACK L
BP1	Bus terminator	BIRQ7 L
BR1	BEVNT L	BEVNT L
BS1	Not connected	+12B
BT1	GND	GND
BU1	Not connected	PSPARE2
BV1	+5 V	+5 V

Table 1-1 SBC-11/21 Module Backplane Pin Utilization (Cont)

Side 2 (Solder Side)

Backplane Pin	SBC-11/21 Signal Function	LSI-11 Bus Signal Name
AA2	+5 V	+5 V
AB2	Not connected	-12 V
AC2	GND	GND
AD2	+12 V	+12 V
AE2	BDOUT L	BDOUT L
AF2	BRPLY L	BRPLY L
AH2	BDIN L	BDIN L
AJ2	BSYNC L	BSYNC L
AK2	BWTBT L	BWTBT L
AL2	BIRQ4 L	BIRQ4 L
AM2	Not connected	BIRKI L
AN2	BIAKO L	BIAKO L
AP2	BBS7 L	BBS7 L
AR2	Not connected	BDMGI L
AS2	BDMGO L	BDMGO L
AT2	BINIT L	BINIT L
AU2	BDAL0 L	BDAL0 L
AV2	BDAL1 L	BDAL1 L
BA2	+5 V	+5 V
BB2	Not connected	-12 V
BC2	GND	GND
BD2	+12 V	+12 V
BE2	BDAL2 L	BDAL2 L
BF2	BDAL3 L	BDAL3 L
BH2	BDAL4 L	BDAL4 L
BJ2	BDAL5 L	BDAL5 L
BK2	BDAL6 L	BDAL6 L
BL2	BDAL7 L	BDAL7 L
BM2	BDAL8 L	BDAL8 L
BN2	BDAL9 L	BDAL9 L
BP2	BDAL10 L	BDAL10 L
BR2	BDAL11 L	BDAL11 L
BS2	BDAL12 L	BDAL12 L
BT2	BDAL13 L	BDAL13 L
BU2	BDAL14 L	BDAL14 L
BV2	BDAL15 L	BDAL15 L

1.3 RELATED HARDWARE MANUALS

The primary reference document for the SBC-11/21 is this volume. A considerable amount of useful information about other LSI-11 bus compatible products may be found in the following publications.

Title	Document No.
Microcomputers and Memories Handbook, 1981 Edition	EB-18451-20
Microcomputer Interfaces Handbook, 1980 Edition	EB-20175-20
PDP-11 Bus Handbook, 1979 Edition	EB-17525-20

These documents can be ordered from:

Digital Equipment Corporation
444 Whitney St.
Northboro, MA 01532

Attention: Printing and Circulation
Mail Station NR-2/W3

CHAPTER 2
INSTALLATION

2.0 GENERAL

The installation procedure for the SBC-11/21 single-board computer module must include the following.

NOTE

It is best to leave the factory configuration undisturbed until module performance has been verified.

- Install jumpers to select operational features.
- Select and mount an LSI-11 bus-structured backplane and add any required LSI-11 bus options.
- Select and connect an appropriate power supply.
- Provide appropriate cables to connect external devices to the serial and parallel I/O interfaces.
- Verify operation of the module.

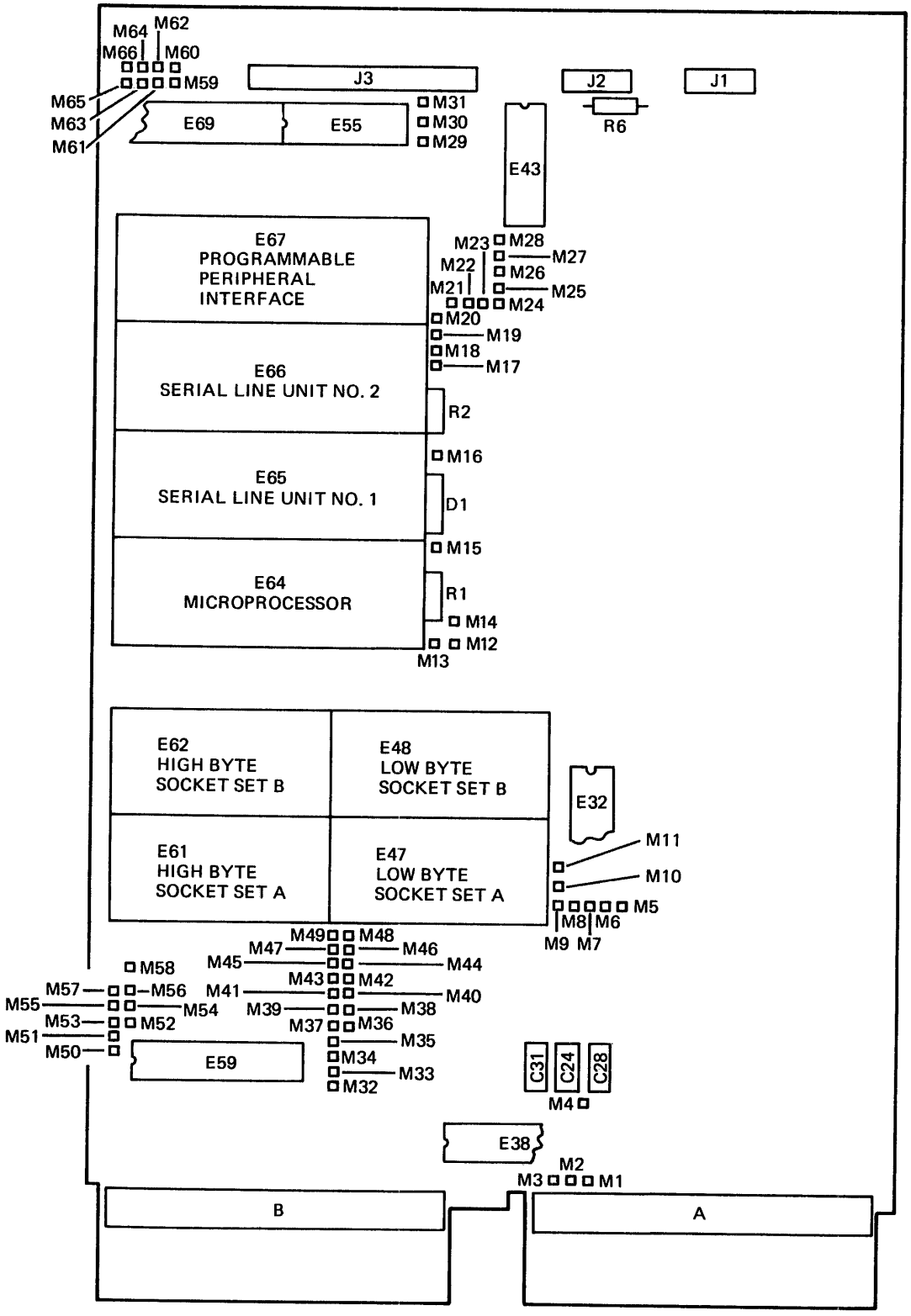
These items are discussed in detail in this chapter.

2.1 SELECTING OPERATIONAL FEATURES

The module has 66 wirewrap pins for the users to configure the module for the operating modes necessary to meet their requirements. This is accomplished by either installing or removing jumper wires between the wirewrap pins. The locations and numerical identification of the wirewrap pins are shown in Figure 2-1. The wirewrap pins and their functions are listed in Table 2-1, by the features they support. The selectable features are Battery Backup, Power Up, Starting Address, Interrupts, Parallel I/O Buffering and Memory maps. Detailed requirements for each of these configurations are described in the following paragraphs. The standard factory configuration is described in Table 2-2.

2.1.1 Battery Backup

The user can select the Battery Backup mode to maintain a +5 Vdc battery supply to the 4KB of static RAM and, if desired, to the two 28-pin sockets that are designated as socket set A. The +5 Vdc battery supply is provided through the LSI-11 bus via pin AV1 and a maximum of 260 mA is required. This supply is connected to wirewrap pin M3. To enable battery backup of 4KB of static RAM, remove the jumper wire between M1 and M2, and install a jumper wire between M3 and M2. To provide backup for socket set A, remove the jumper wire between M4 and M1, and install a jumper wire between M2 and M4.



MR-6691

Figure 2-1 SBC-11/21 Module Layout

Table 2-1 Configuration Pin Definitions

Pin	Function	Description
	Battery Backup	
M1		System +5 V power
M2		+5 Vdc power distribution to support static RAM
M3		Battery backup +5 V power source
M4		Socket set A, pin 26, high and low byte.
	Nonmaskable Interrupt and Trap to the Restart Address	
M5		+5 Vdc voltage level
M6		-CTMER interrupt request input (edge sensitive)
M7		Timeout error (TMER) output
M8		-CTMER interrupt enable
M9		Interrupt Acknowledge (-IAK) output
M10		System GND
M11		High logic level (+3 Vdc)
	Serial Line Unit #1	
M12		System GND
M13		Transmit side of BHALT line transceiver
M14		Serial Line unit #1 Break Detect, Interrupt request output
	Power Up	
M15		System +5 V power, wake up circuit diode, cathode side.
M16		Wake up circuit diode, anode side.

Table 2-1 Configuration Pin Definitions (Cont)

Pin	Function	Description
	Serial Line Unit 2	
M17		Transmit side of BEVNT line transceiver
M18		50 Hz real time clock output
M19		60 Hz real time clock output
M20		800 Hz real time clock output
	Memory Map Decoder	
M21		High logic level (+3 Vdc)
M22		Memory map select (LSB)
M23		Memory map select (MSB)
M24		System GND
	Start Address (Mode Register)	
M25		Start address control
M26		Start address control
M27		Start address control
M28		High logic level (+3 Vdc)
	BHALT Interrupt (Level 7, maskable)	
M29		System GND
M30		Receive side of BHALT line transceiver
M31		BHALT interrupt request input (edge sensitive)

Table 2-1 Configuration Pin Definitions (Cont)

Pin	Function	Description
	Memory	
M32		Address line 11
M33		High logic level, for PROMs
M34		Socket set A, high byte, pin 23
M35		Socket set B, high and low byte, pin 27
M36		Socket set A, high and low byte, pin 20
M37		Socket set B, high byte, pin 23
M38		Socket set B, high and low byte, pin 20
M39		Socket set B, high byte, pin 22
M40		Socket set B, low byte, pin 22
M41		Socket set A, low byte, pin 22
M42		Socket set B, low byte, pin 23
M43		Socket set A, low byte, pin 23
M44		Socket set A, high and low byte, pin 2
M45		Socket Set B, high and low byte, pin 2
M46		Address line 13
M47		Socket Set B Chip Select (-CSKTB)
M48		Socket Set A, high and low byte, pin 27
M49		Address line 12
M50		System GND
M51		Read Strobe (-Read)
M52		Write low byte strobe (-WLB)

Table 2-1 Configuration Pin Definitions (Cont)

Pin	Function	Description
M53		Static RAM high byte Output Enable (OE)
M54		Static RAM low byte Output Enable (OE)
M55		High byte write strobe (-WHB)
M56		Socket set A Chip Select (-CSKTA)
M57		Socket set A, high byte, pin 22
M58		Socket set A and B, high and low byte, pin 21
	Parallel Input/Output	
M59		Port B Buffer direction control
M60		System GND
M61		Port C buffered output, to J3 pin 5
M62		Port C buffered output, to J3 pin 7
M63		Port C PC4 output (8255A-5 pin 13)
M64		Port C PC6 output (8255A-5 pin 11)
M65		High logic level (+3 Vdc)
M66		Port A buffer direction control

2.1.2 Wake Up Circuit

The module has an on-board power wake up circuit designed to be used in systems without the LSI-11 bus power sequencing protocol. This circuit holds the BDCOK line negated until one second after +5 V power is applied. When the module is being used in an LSI-11 backplane, which has a power sequencing routine, the module wake up circuit must be disabled. This is accomplished by installing a jumper wire between M15 and M16. The jumper wire is removed when using power supplies without power sequencing. The user should note that the module requires the +5 Vdc and +12 Vdc power supplies to have a rise time of less than 50 ms.

Table 2-2 Standard Factory Configuration

Function	Jumpers Installed Between
Standard LSI-11 Bus Power	M1 and M2
(No Battery Backup)	M1 and M4
Wake up Circuit Enabled	No jumpers
Start Address*	M25 and M26
Start address 10000	M26 and M24
Restart address 10004	M27 and M28
Memories:	
Memory Map 0	M22 and M23
	M23 and M24
2K X 8 INTEL EPROM	M34 and M43
	M5 and M34
	M41 and M57
	M56 and M36
	M51 and M57
	M32 and M58
	M37 and M42
	M5 and M37
	M41 and M39
	M47 and M38
	M51 and M40
Interrupts:	
Timeout traps to restart address except during LSI-11 bus IAK.	M9 and M8 M7 and M6
SLU#1 Break asserts BHALT and BHALT is received as level 7 interrupt (vector 140)	M13 and M14 M30 and M31
SLU#2 60 Hz Real Time Clock asserts LSI-11 BEVNT	M19 and M17
Parallel I/O in Mode 1:	M59 and M60
Port A Receive data, with STROBEA on PC4	M65 and M66
Port B Transmit data	M61 and M63

No Connection to pins
M3, M10, M11, M12, M15, M16, M18,
M20, M21, M33, M35, M44, M45, M46, M48,
M49, M50, M52, M53, M54, M55,
M62, M64

*Before using with Macro-ODT the start address must be changed to 172000, as described in Table 2-3.

2.1.3 Starting Address

The starting address for the microprocessor is selected by the user via wire wrap pins. When the module is powered up, the microprocessor loads this value into R7 (program counter) as the first fetch address. The wirewrap pins are M24, M25, M26, M27, and M28, defined in Table 2-1. The user has eight starting addresses available to choose from. Table 2-3 lists the available addresses and the jumper connections required for each address. The restart address is always the start address incremented by four. The wirewrap pin locations are shown in Figure 2-1.

Table 2-3 Mode Register Configuration

Start Address	Restart Address	Connect M27 to	Connect M26 to	Connect M25 to
000000	000004	M28	M24	M28
010000*	010004	M28	M24	M24
020000	020004	M24	M28	M28
040000	040004	M24	M28	M24
100000	100004	M24	M24	M28
140000	140004	M24	M24	M24
172000	172004	M28	M28	M28
173000	173004	M28	M28	M24

*Factory setting. The start address should be selected in conjunction with the memory map configuration. Figure 2-6 shows how the available start addresses fit into the memory maps.

2.1.4 Interrupts

The SBC-11/21 implements a multilevel interrupt system that consists of eleven separate interrupts. A complete listing of system interrupts will be found in Table 5-3. Three of these interrupts, CTMER, BKRQ, and REVNT, are user-configurable by means of jumper wires as shown in Figure 2-2 and will be discussed here.

The CTMER interrupt is at the highest level (non-maskable). It is caused by a timeout, that is, failure to detect RRPLY during a FETCH/READ, WRITE, or IACK transaction. For the factory configuration, -IAK is connected to the D input of flip-flop E4 via M9 to M8 jumper. This prevents setting that flip-flop and inhibits CTMER for timeouts occurring during IACK transactions. Such a condition could only occur if the peripheral which caused the interrupt failed to return BRPLY during the vector reading operation. Refer to Chapter 8 for a discussion of External Interrupts. To help the user evaluate the advantages and disadvantages of this jumper option, the sequence of events which takes place during the IAK timeout is described in Figure 2-3.

Observe that a timeout during IAK causes a zero vector to be read in by the microprocessor. This occurs in both cases described in Figure 2-3. The difference is in the setting of CTMER, which causes the second stacking of PC and PSW, followed by jump to RESTART.

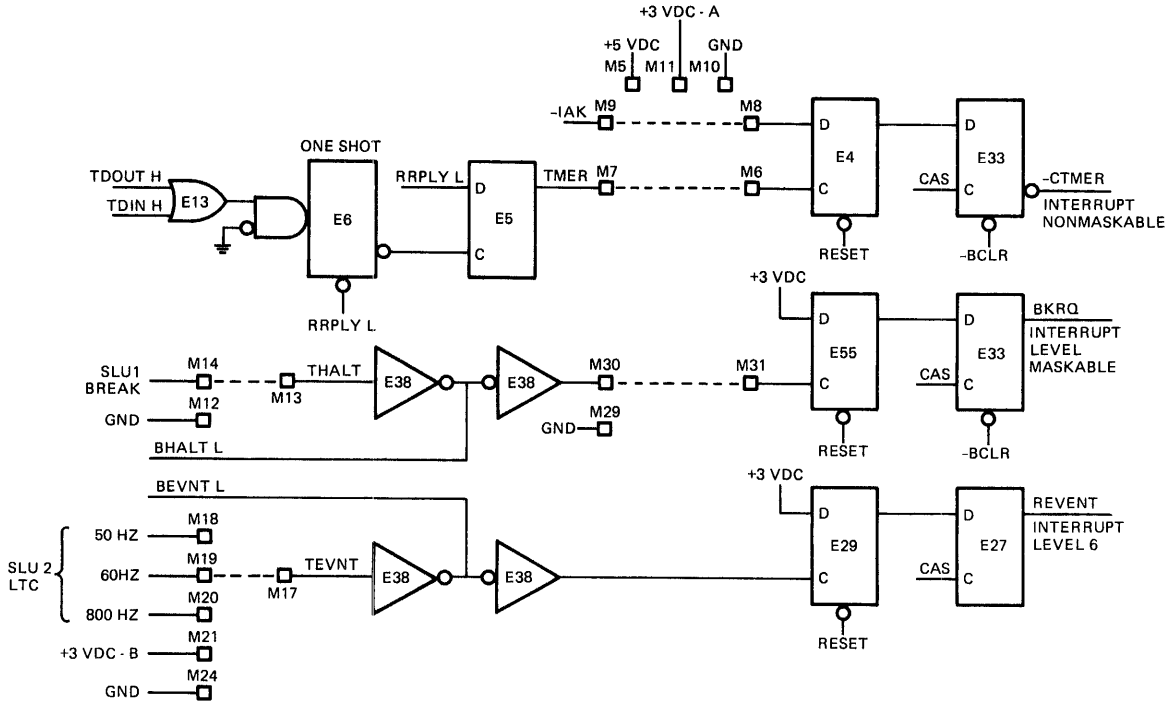


Figure 2-2 Interrupt Configurations

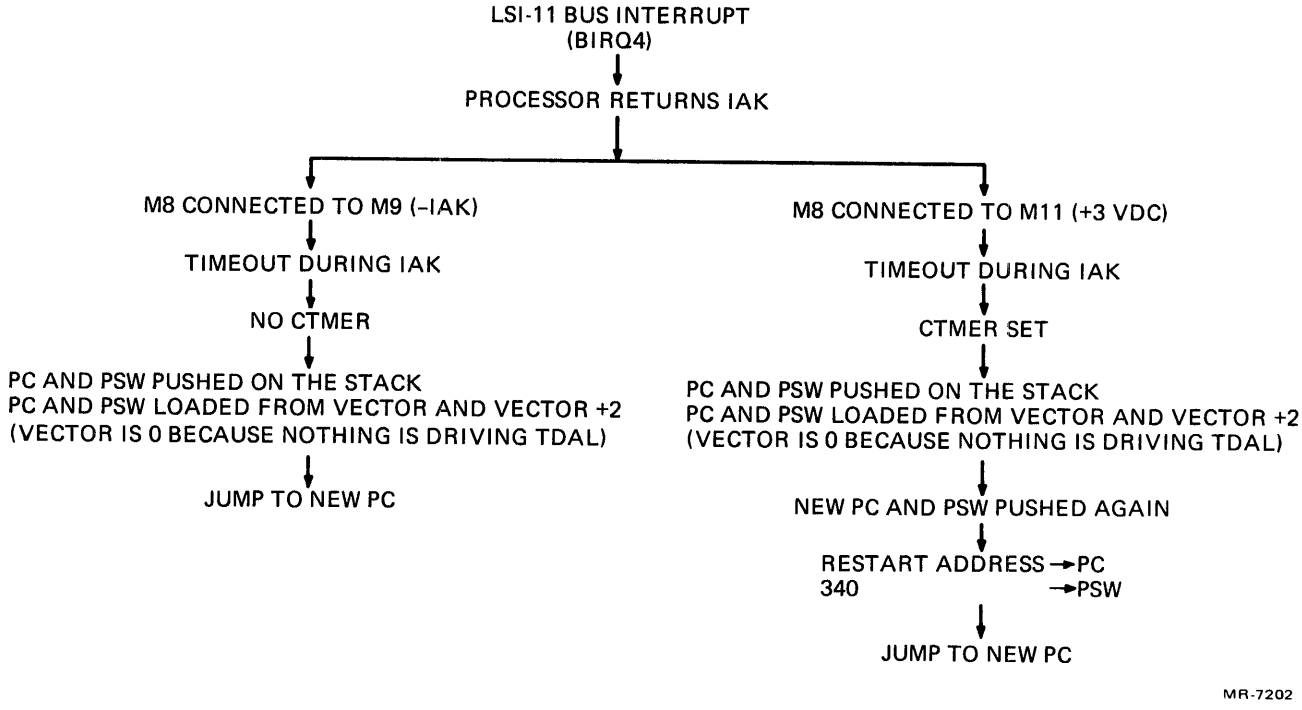


Figure 2-3 Time-out During LSI-11 Bus Interrupt Acknowledge

The other two interrupts configurable by the user are BKRQ and REVNT. Their vectors and priorities are described in Table 5-3. All jumper combinations which are "electrically correct" as described in Figure 2-2 are legal.

Some typical configurations are described below to familiarize the user with the various combinations available.

Install jumpers between M7 and M31
M8 and M11
M6 and M14
M12 and M13
M17 and M20

This arrangment allows the SLU 1 BREAK input to set the -CTMER nonmaskable interrupt and trap to the restart address. The timeout (TMER) input sets the BKRQ level 7 maskable interrupt. The BHALT L bus signal is ignored. The SLU 2 800 Hz line time clock and the BEVNT L bus signal enable the REVNT interrupt.

Install jumpers between M6 and M30
M14 and M31
M8 and M11
M13 and M12
M17 and M24

This arrangment allows the BHALT L bus signal to set the -CTMER nonmaskable interrupt and trap to the restart address. The SLU 1 BREAK input sets the BKRQ level 7 maskable interrupt and only the BEVENT L bus signal enables the REVNT interrupt.

Install jumpers between M7 and M6
M8 and M11
M14 and M13
M30 and M31
M17 and M21

This arrangement allows the timeout (TMER) to set the -CTMER nonmaskable interrupt for all timeouts. The SLU 1 BREAK or the BHALT bus signal set the BKRQ level 7 maskable interrupt and the BEVNT L bus line is clamped low and therefore no interrupts can be generated by BEVNT L.

2.1.5 Parallel I/O

The Parallel I/O is implemented with the 8255A-5 Programmable Peripheral Interface (PPI) and connects to the user's interface through the J3 connector. Wirewrap pins used for the configuration of the Parallel I/O are shown in Figure 2-4 and are defined in Table 2-1. Dash lines in Figure 2-4 represent the factory configuration jumpers installed. The wirewrap pin locations are

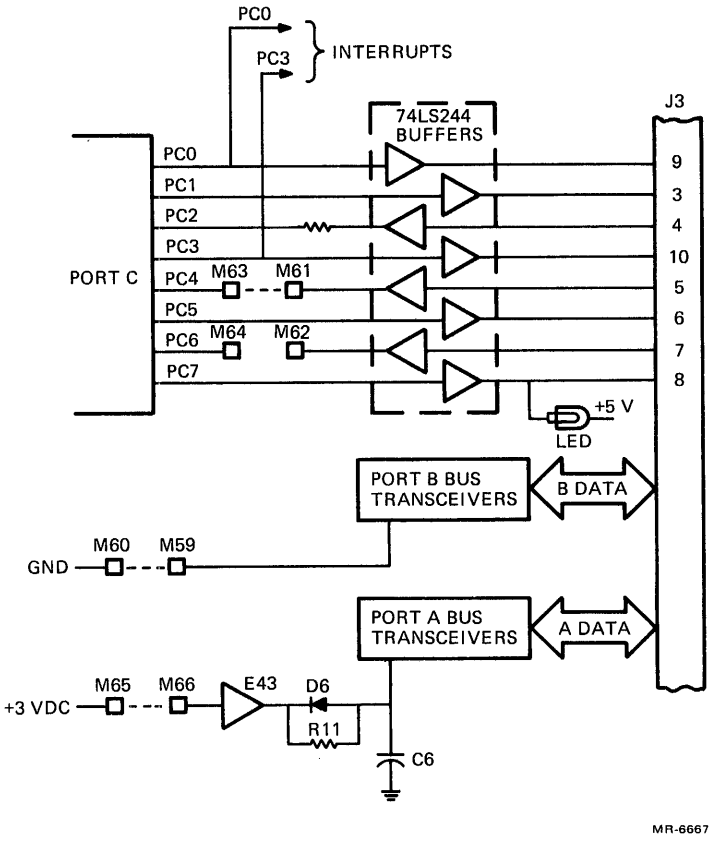


Figure 2-4 Parallel I/O Configuration

shown in Figure 2-1. The directions of port A and port B transceivers are dependent upon the logic level connected to M59 and M66. Wirewrap pin 66 connects to port A through a 200 ns minimum rising edge time delay circuit. When M65 (+3 Vdc) is jumpered to pins M59 and M66, port A and port B buffers act as inputs to the Programmable Peripheral Interface from the J3 connectors. When M60 (GND) is jumpered to pins M59 and M66, port A and port B buffers act as outputs from the Programmable Peripheral Interface to the J3 connector.

The direction of port A and port B can also be controlled by the user's program. To make this possible, M63 and M64 must be jumpered to M59 and M66. Then data outputs via port C will control the voltage levels at the direction control inputs to ports A and B. The software required to accomplish this control is discussed in Chapter 6.

Wirewrap pins M61 and M62 can be jumpered to M59 and M66 to allow the user to control the direction of the transceivers via J3 connector pins 5 and 7. When not using wirewrap pins M63 and M61 or M64 and M62 to control the direction of ports A and B, jumpers connected between M63 and M61 and between M64 and M62 allow PC4 and PC6 to be used as inputs to the PPI from the J3 connector.

NOTE

If pins M61, M62, M63 or M64 are used for program control of ports A or B, the user must ensure that the PPI and the buffer do not contend as driver output to driver output. If this condition is allowed to occur, damage to both drivers may result.

The Programmable Peripheral Interface can function in three modes selected by software. The jumper configurations and the handshake signals for each of these modes are shown in Tables 2-4, 2-5, and 2-6. For programming information refer to Chapter 6.

Table 2-4 Mode 0 Buffer Configuration (No Handshake)

PPI Element	To Act as Input	To Act as Output	Program Control via Port C
Port A	M66 to M65	M66 to M60	M66 to M64 or M63
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	Never an Input	Always an Output	
PC6	M64 to M62	Never an external Output	
PC5	Never an Input	Always an Output	
PC4	M63 to M61	Never an External Output	
PC3	Never an Input	Interrupt A (Vector 134) Always an Output	
PC2	Always an Input	Never an Output	
PC1	Never an Input	Always an Output	
PC0	Never an Input	Interrupt B (Vector 130) Always an Output	

Table 2-5 Mode 1 Buffer Configuration (Strobed I/O)

PPI Element	To Act As Input	To Act As Output	Program Control via Port C
Port A	M66 to M65	M66 to M60	N/A
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	Never an Input	Indicates Buffer A Full	
PC6	M62 to M64 (Acknowledge A)*	Never an External Output	
PC5	Never an Input	Indicates Buffer A Full	
PC4	M61 to M63 (Strobe A)	Never an External Output	
PC3	Never an Input	Interrupt A	
PC2	Strobe B in Input Mode Acknowledge B in Output Mode	Never an Output	
PC1	Never an Input	Buffer B Full on Input or Output	
PC0	Never an Input	Interrupt B (Vector 130)	

*User's hardware acknowledges receipt of data output by port A.

2.1.6 Serial I/O

The jumper options relating to the serial I/O determine the interrupt response of the system, and have been thoroughly explained in Paragraph 2.1.4. For the sake of completeness, a tabulation of responses to the BREAK detection by the SLU1 is given in Table 2-7.

2.1.7 Memories

The memory system for the module consists of the LSI-11 bus, 4K bytes of local RAM and four 28-pin sockets that will accept either 24-pin or 28-pin industry standard +5 V memory chips. These chips are provided by the user and can be either EPROMs, PROMs, ROMs or static RAM. The sockets will accept 1K X 8, 2K X 8, 4K X 8, and 8K X 8 PROMs/EPROM or 2K X 8 static RAMs.

Table 2-6 Mode 2 Buffer Configuration and Handshake

PPI Element	Input Signal	Output Signal
Port A	Bidirectional bus	If M66 to M64 to M62
Port B	Not used in Mode 2	Not used in Mode 2
PC7	Never an Input	Output Buffer A Full
PC6	Acknowledge A	Never an Output
PC5	Never an Input	Input Buffer A Full
PC4	Strobe A (if M61 to M63)	Never an Output
PC3	Never an Input	Interrupt A
PC2	Always on Input	Never an Output
PC1	Never an Input	Always an Output
PC0	Never an Input	Always an Output

Table 2-7 SLU1 BREAK Detection

Jumper Connection	BREAK Response
M14 to M13 M30 to M31	BHALT L Signal to the LSI-11 Bus and BKRQ interrupt (vector 140)
M13 to M12 M30 to M31 M14 NC	no response
M14 to M31 M13 to M12	BKRQ interrupt (vector 140) (no BHALT L to bus)
M14 to M6 M13 to M12 M8 to M11	CTMER interrupt (HALT trap) through Restart

There are two socket sets, one designated as SET A which is controlled by -CSKTA and the other designated as SET B which is controlled by -CSKTB. Each set consists of a high byte socket and a low byte socket which are interconnected as shown in Figure 2-5. The wirewrap pins used to configure the memory are shown in Figure 2-6 and described in Table 2-1. The standard factory configuration of the jumper wires installed is represented by the dash lines in Figure 2-6. In addition to configuring the sockets, the user must configure the Decode Memory Address chip to select one of the four memory maps available.

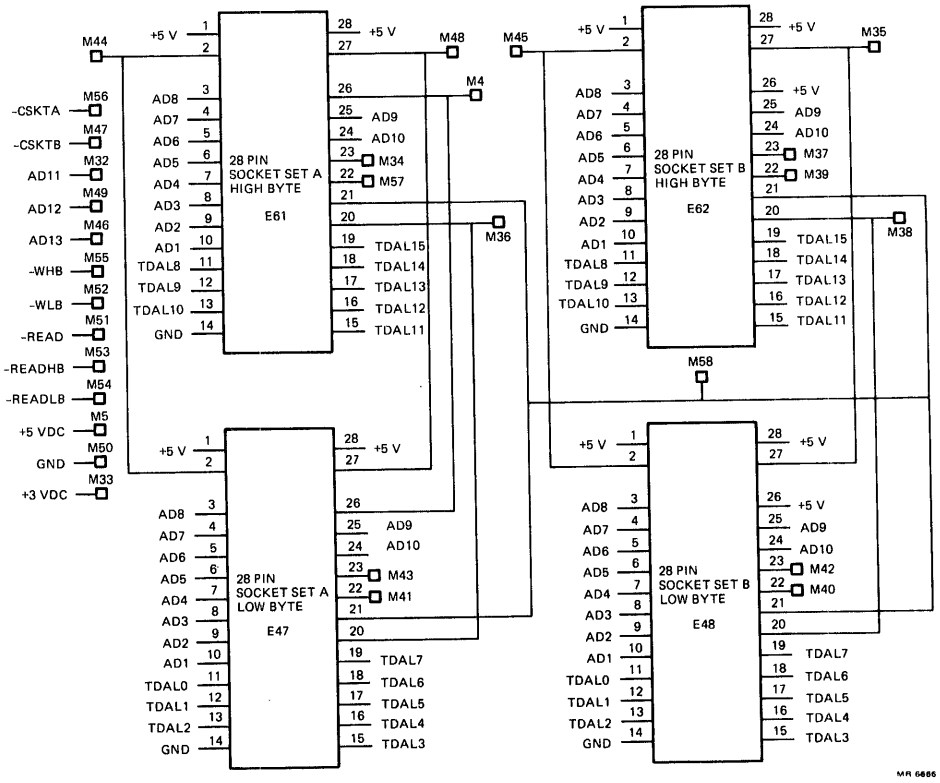


Figure 2-5 Socket Sets A and B Interconnection

NOTE

When configuring pins M32 to M58 the RAM chips in sockets E60 and E46 must be removed. The RAM chips are replaced into the sockets when the configuration is completed.

2.1.7.1 Memory Maps -- There are four memory maps available as shown in Figure 2-7 and the module can be configured to select one that meets the user's requirements. Wirewrap pins M21, M22, M23, and M24 are used to select the memory map and the jumper requirements are listed in Table 2-8.

Table 2-8 Memory Map Configurations

Map Selection	Jumper M22 to	Jumper M23 to
Map 0	M23	M24
Map 1	M21	M24
Map 2	M24	M21
Map 3	M23	M21

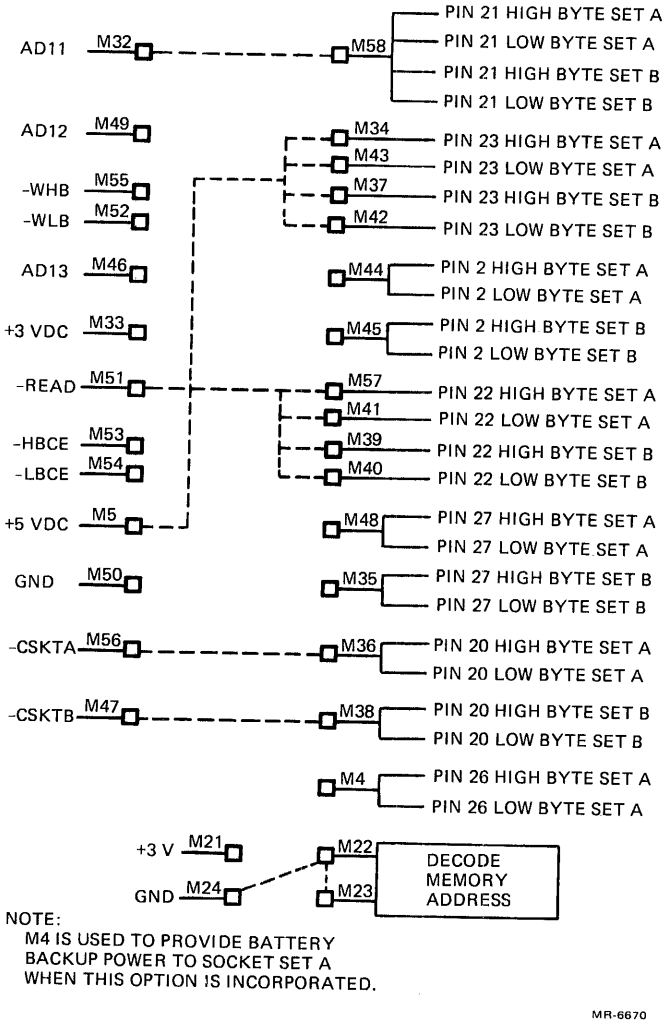
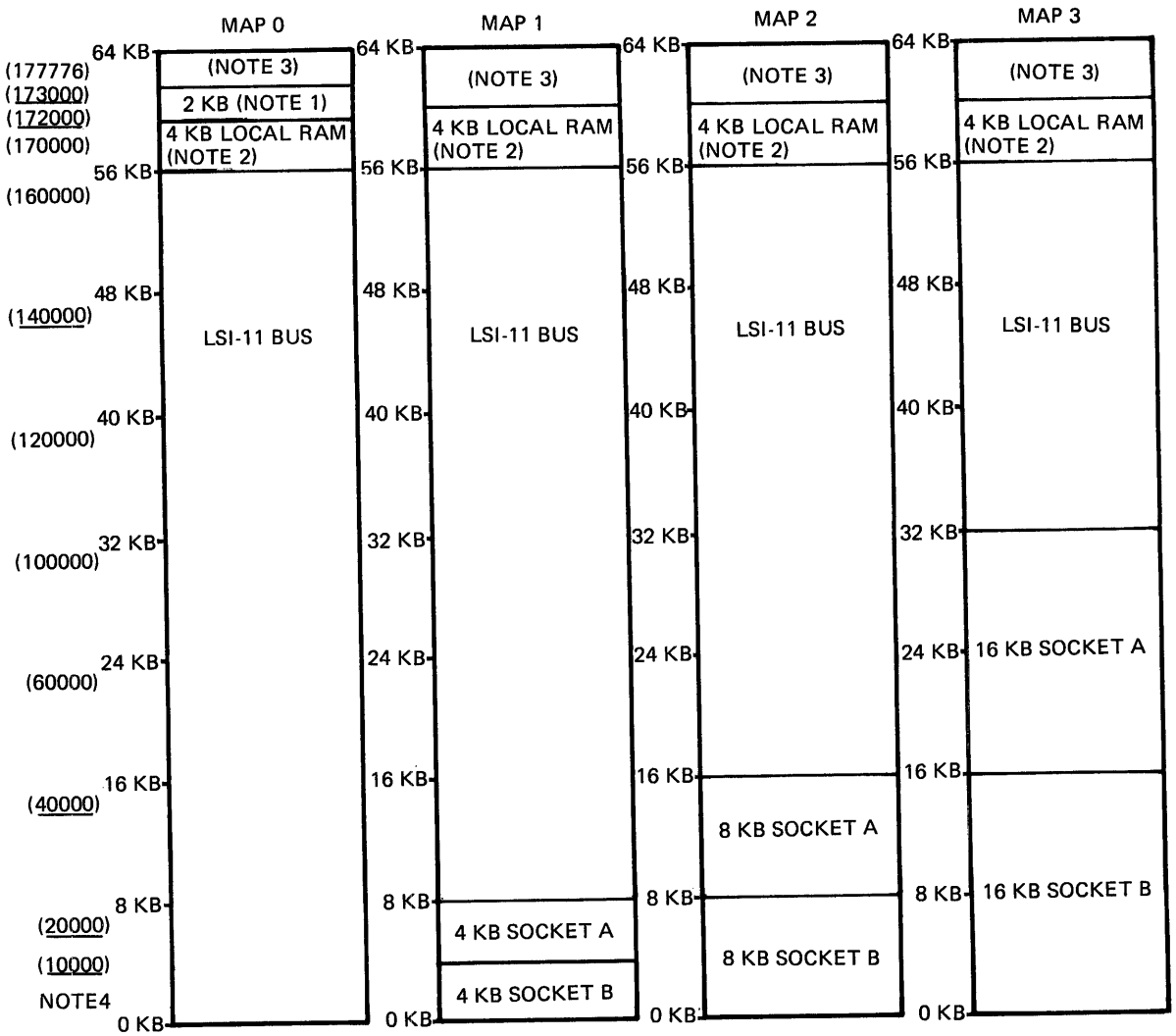


Figure 2-6 Memory Configuration



NOTES:

1. SOCKET SET A IS MAPPED OVER SOCKET SET B AND IS THEREFORE LIMITED TO USING EITHER SOCKET A OR SOCKET B, BUT NOT BOTH TOGETHER.
2. ADDRESSES 160000 THROUGH 160007 ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
3. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUs AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
4. UNDERLINED ADDRESSES ARE JUMPER-SELECTABLE START ADDRESSES, ACCORDING TO TABLE 2-3.

Figure 2-7 Memory Maps

MR-7243

2.1.7.2 PROMs/EPROMs -- The 28-pin sockets accept 24-pin and 28-pin PROMs or EPROMs. If 24-pin chips are selected, caution must be observed to ensure pin 1 of the chip is placed into socket hole 3. The configuration requirements of some industry compatible PROMs/EPROMs are described by Tables 2-9 and 2-10. The user may select chips from other vendors but the pin configuration must be compatible with the sockets provided. A 250 ns maximum output

Table 2-9 Socket Set A Configuration for EPROM/PROM

Vendor	Part	Pins	Size	Connect Referenced Pin to Socket A Pin M43	M48	M44	M34	M57	M36	M41	M58
EPROMS											
INTEL	2758	24	1K X 8	M5	NC	NC	M5	M51	M56	M51	M50
INTEL	2716	24	2K X 8	M5	NC	NC	M5	M51	M56	M51	M32
	2716-1	24	2K X 8								
	2716-2	24	2K X 8								
INTEL	2732	24	4K X 8	M49	NC	NC	M49	M51	M56	M51	M32
	2732A	24	4K X 8								
INTEL	2764	28	8K X 8	M49	M33	M46	M49	M51	M56	M51	M32
TI	TMS2508	24	1K X 8	M5	NC	NC	M5	M51	M56	M51	M33
TI	TMS2516	24	2K X 8	M5	NC	NC	M5	M51	M56	M51	M32
	TMS2516-35	24	2K X 8								
TI	TMS2564	28	8K X 8	M46	M50	M51	M46	M56	M49	M51	M32
Mostek	MK2716	24	2K X 8	M5	NC	NC	M5	M51	M56	M51	M32
Mostek	MK2764	28	8K X 8	M49	NC	M46	M49	M51	M56	M51	M32
PROMS											
INTEL	3628	24	1K X 8	M56	NC	NC	M56	M51	M33	M51	M33
Signetics	82LS181	24	1K X 8	M56	NC	NC	M56	M51	M33	M51	M33

NC requires NO connection.

enable time is also required and the maximum access time for compatible PROMs/EPROMs is 450 ns. The maximum output enable time is defined as the time from the assertion of TDIN or TDOUT by a bus master to the time the module asserts valid data onto the bus.

The user installs a jumper wire from the pin referenced by the chip type to the socket pin described in the tables. Figure 2-6 provides a reference for all signals and the socket pins associated with the wirewrap pins. These interconnections are listed separately under socket set A and socket set B, and some jumper wires are common to both socket sets. Some devices may not require a connection or a jumper wire installed and these are designated by an "NC" in the tables. The wirewrap pin locations are shown in Figure 2-1.

Table 2-10 Socket Set B Configuration for EPROM/PROM

Vendor	Part	Pins	Size	Connect Referenced Pin to Socket B Pin M42	M35	M45	M37	M39	M38	M40	M58
EPROMS											
INTEL	2758	24	1K X 8	M5	NC	NC	M5	M51	M47	M51	M50
INTEL	2716	24	2K X 8	M5	NC	NC	M5	M51	M47	M51	M32
	2716-1	24	2K X 8								
	2716-2	24	2K X 8								
INTEL	2732	24	4K X 8	M49	NC	NC	M49	M51	M47	M51	M32
	2732A	24	4K X 8								
INTEL	2764	28	8K X 8	M49	M33	M46	M49	M51	M47	M51	M32
TI	TMS2508	24	1K X 8	M5	NC	NC	M5	M51	M47	M51	M33
TI	TMS2516	24	2K X 8	M5	NC	NC	M5	M51	M47	M51	M32
	TMS2516-35	24	2K X 8								
TI	TMS2564	28	8K X 8	M46	M50	M51	M46	M47	M49	M51	M32
Mostek	MK2716	24	2K X 8	M5	NC	NC	M5	M51	M47	M51	M32
Mostek	MK2764	28	8K X 8	M49	NC	M46	M49	M51	M47	M51	M32
PROMS											
INTEL	3628	24	1K X 8	M47	NC	NC	M47	M51	M33	M51	M33
SIGNETICS	82LS181	24	1K X 8	M47	NC	NC	M47	M51	M33	M51	M33

NC requires NO connection.

2.1.7.3 RAMs -- The 28-pin sockets can also accept 24-pin static RAM chips and caution must be observed to ensure pin 1 of the chip is installed into socket hole 3. The configuration requirements of some industry compatible RAMs are described in Tables 2-11 and 2-12. The user may select chips from other vendors but the pin configuration must be compatible with the sockets provided. The selected RAMs are required to meet the maximum output enable time and the maximum access time specified for the PROMs.

The user installs a jumper wire from the pin referenced by the chip type to the socket pin described in the tables. Figure 2-6 provides a reference for all signals and the socket pins associated with the wirewrap pins. The interconnections are listed

separately under socket A and socket B, and some jumper wires are common to both socket sets. Some devices may not require a connection or a jumper wire installed and these are designated by "NC" in the tables. The wirewrap locations are shown in Figure 2-1.

Table 2-11 Socket Set A Configuration for RAM

Vendor	Part	Pins	Size	Connect Referenced Pin to Socket A Pin M43	M48	M44	M34	M57	M36	M41	M58
MOSTEK	MK4802	24	2K X 8	M52	NC	NC	M55	M53	M56	M54	M32
TOSHIBA	TMM2016P	24	2K X 8	M52	NC	NC	M55	M53	M56	M54	M32
	TMM2016P-1	24	2K X 8								
HITACHI	HM6116P	24	2K X 8	M52	NC	NC	M55	M53	M56	M54	M32

NC requires NO connection.

Table 2-12 Socket Set B Configuration for RAM

Vendor	Part	Pins	Size	Connect Referenced Pin to Socket B Pin M42	M35	M45	M37	M39	M38	M40	M58
MOSTEK	MK4802	24	2K X 8	M52	NC	NC	M55	M53	M47	M54	M32
TOSHIBA	TMM2016P	24	2K X 8	M52	NC	NC	M55	M53	M47	M54	M32
	TMM2016P-1	24	2K X 8								
HITACHI	HM6116P	24	2K X 8	M52	NC	NC	M55	M53	M47	M54	M32

NC requires NO connection.

2.2 SELECTING BACKPLANES AND OPTIONS

A number of different LSI-11 bus compatible backplanes and boxes are available from Digital. The choice must be dictated by the system requirements such as the number and type of options (described in Chapter 3), environmental conditions and packaging considerations. A list of all available backplanes and boxes is described in the Microcomputer Interfaces Handbook referenced in Chapter 1.

2.3 POWER SUPPLY

The choice of power supply is governed by the size of the system and packaging requirements. An important consideration is the behavior of the supply during power up and power down. All Digital power supplies listed in the Microcomputer Interfaces Handbook (see Chapter 1) are compatible with the LSI-11 bus protocol. This assures reliable operation with no loss of data when using battery-supported memories. Any user-designed power supply must conform to this protocol.

2.4 EXTERNAL CABLES

The module has a 30-pin connector (J3) for external interfacing with the Programmable I/O Interface and two 10-pin connectors (J1 and J2) for external interfacing with the Serial Line Units (SLU). The location of these connectors on the module is shown in Figure 2-1. The user's requirements to interface with these connectors is defined below.

2.4.1 Parallel I/O Interface (J3)

The module connector is a 30-pin AMP MODU connector with the I/O signals defined by Figure 2-8. The I/O signals are buffered and are capable of driving up to 50 feet (maximum) of unshielded flat ribbon or round cable with a 30-pin AMP contact housing at each end. The following list of connectors is compatible with the module connector.

AMP MODU polarized or nonpolarized contact housings for crimp snap-in pin and receptacle contacts:

Latching, polarized housings:	2-87631-6 no strain relief 87733-6 strain relief
Nonlatching, polarized housings:	1-87977-3 no strain relief 1-102184-3 strain relief
Nonlatching, nonpolarized housings:	2-87456-6 no strain relief 2-87832-7 strain relief
Receptacle contacts:	87045-3 for 30 to 26 AWG 102098-3 for 32 to 27 AWG

Mass termination connectors for flat cables

Separate parts: (nonpolarized)	1-88378-1 connector 1-86873-2 cover 1-88340-1 strain relief cover
Separate parts: (polarized)	1-88392-1 connector 1-86373-2 cover 1-88340-1 strain relief cover
Connector and cover kits: (nonpolarized)	1-88379-1 no strain relief 1-88476-1 with strain relief
Connector and cover kits: (polarized)	1-88393-1 no strain relief 1-88478-1 with strain relief

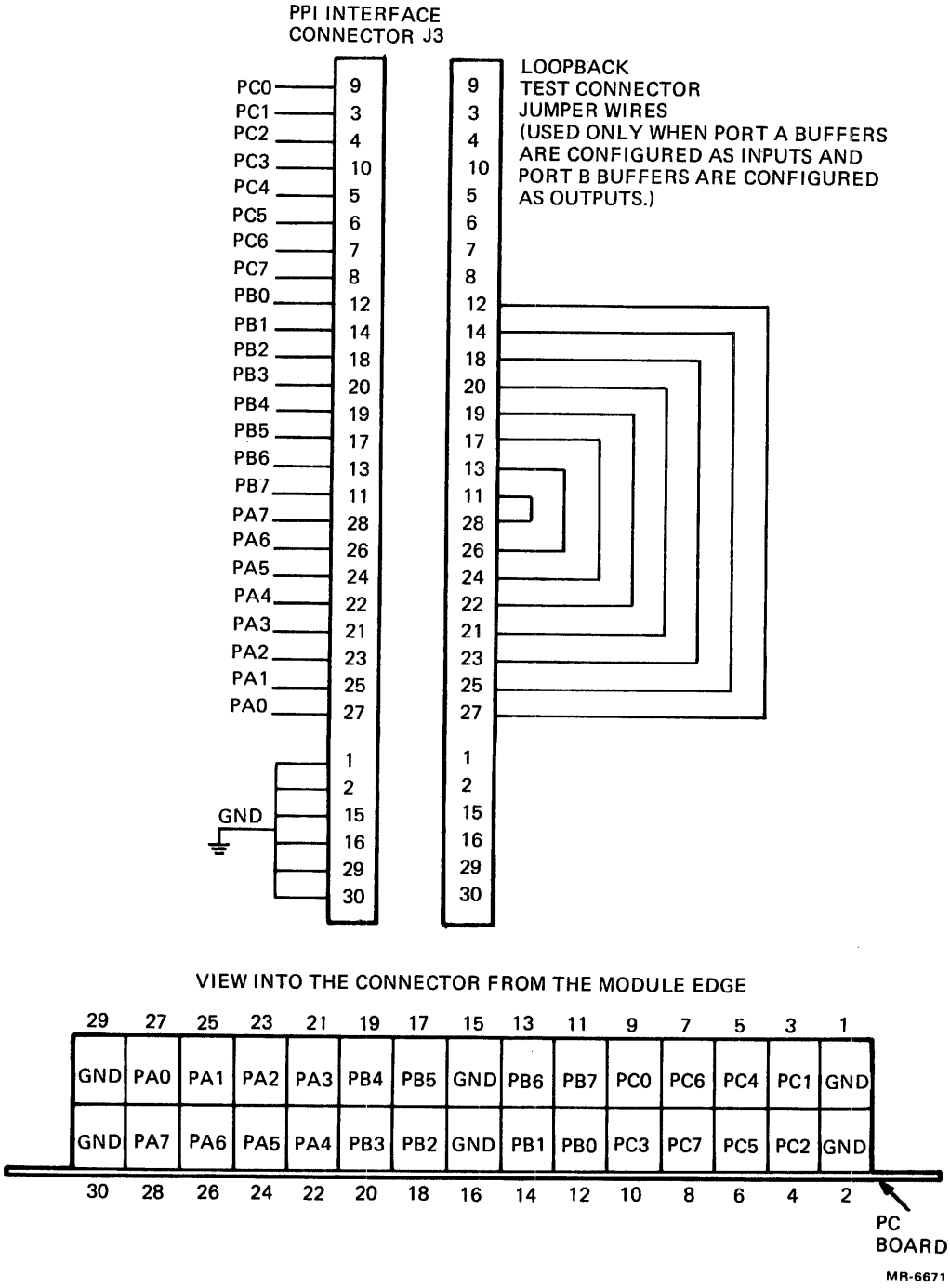


Figure 2-8 30 Pin Parallel I/O Connector

Separate parts:	1-88392-1 connector 1-86873-2 cover 1-88340-1 strain relief cover
Latching connectors and covers: (polarized)	1-88423-1 no strain relief 1-88479-1 with strain relief
Mass Modular Connector System:	1-102393-3 housing for 30-26 AWG 1-102396-3 cover 1-102392-3 kit 1-102398-3 housing for 26-22 AWG 1-102396-3 cover 1-102397-3 kit

Connectors can be terminated to discrete wire in sizes 30-26 AWG, 26-24 AWG, as well as jacketed cable and bonded ribbon cable.

2.4.2 Serial Line Interfaces (J1 and J2)

Each of the Serial Line Units is compabible with EIA RS-232 C and EIA RS-423 serial type interfaces. Serial line unit #1 interfaces through J1 and serial line unit #2 interfaces through J2. When a 20 mA current loop device is desired, then the DLV11-KA option must be used. The option has an EIA cable (BC21A-03) that connects the converter box to the module and the box mates with the standard 20 mA cable using the 8-pin Mate-N-Lock connector. Note that the option does not support the Reader Run strobe and the 110 baud rate so that LA-33 or similar devices cannot be used.

The user is required to install a slew rate resistor determined by the operating baud rate as defined in Table 2-13. The slew rate resistor is designated as R6 and its location on the module is shown in Figure 2-1.

Table 2-13 EIA Slew Rate Resistor Values

Baud Rate	Resistor R6 (ohms)
38400	22K*
19200	51K
9600	120K
4800	200K
2400	430K
1200	820K
600	1M
300	1M

* Factory Installed Value.

The serial line unit connectors showing the signals assigned to the connector pins are shown in Figure 2-9. The user is required to provide the interconnecting cables. The following list describes some standard DIGITAL cables and also some information to assist the user in designing cables.

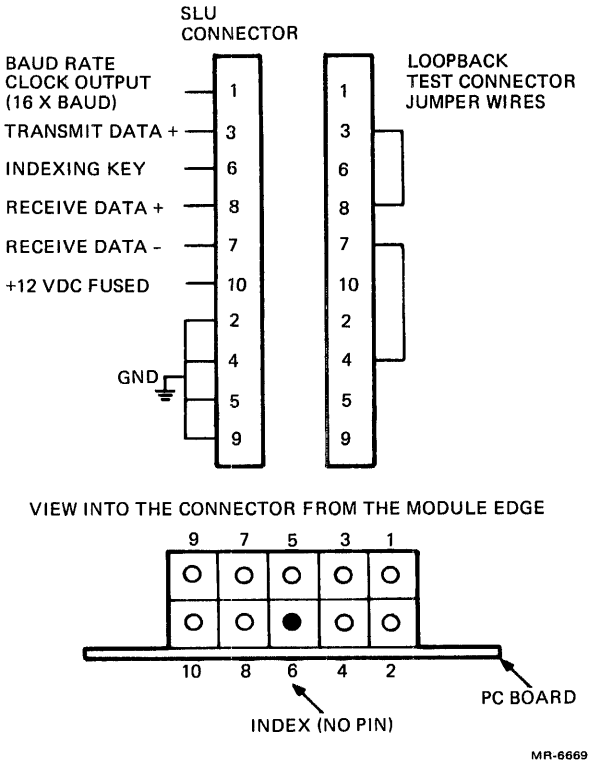


Figure 2-9 10 Pin Serial Line Unit Connector

DIGITAL cables for the SBC-11/21

BC20N-05 5 foot EIA RS-232C null modem cable to directly interface with the EIA RS-232C terminal (2 X 5 pin Amp female to RS-232C female; see Figure 2-10).

BC21B-05 5 foot EIA RS-232C modem cable to interface with modems and acoustic couplers (2 X 5 pin Amp female to RS-232 C male; see Figure 2-11).

BC20M-50 50 foot EIA RS-422 or RS-423 cable for highspeed transmission (19.2K baud) between two SBC-11/21's (2 X 5 pin Amp female to 2 X 5 pin Amp female).

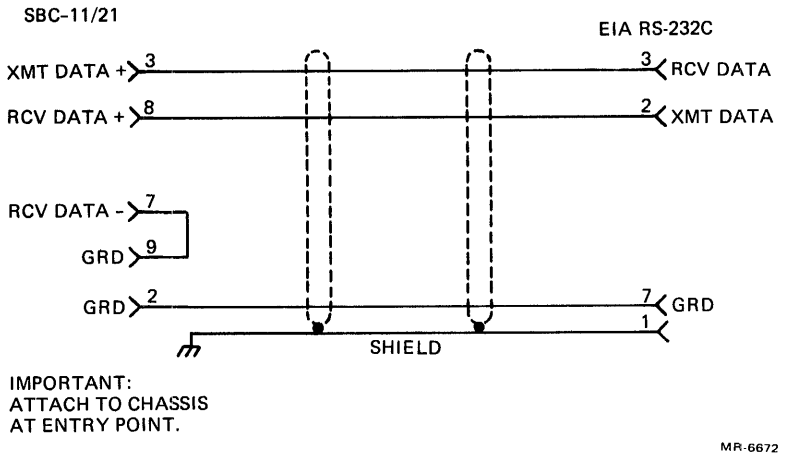


Figure 2-10 BC20N-05 "Null Modem" Cable

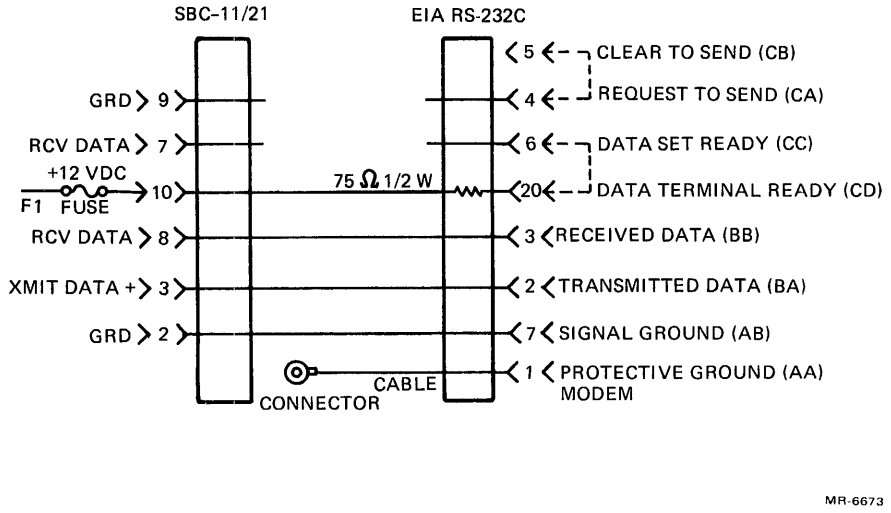


Figure 2-11 BC21B-05 Modem Cable

When designing a cable for the SBC-11/21, here are several points to consider:

1. The receivers on the SBC-11/21 have differential inputs. Therefore, when designing an RS-232C or RS-423 cable, RECEIVE DATA- (pin 7 on the 2 X 5 pin Amp connector) must be tied to signal ground (pins 2, 5, or 9) in order to maintain proper EIA levels. RS-422 is balanced and uses both RECEIVE DATA+ and RECEIVE DATA-.

2. To directly connect to a local EIA RS-232C terminal, it is necessary to use a null modem. To design the null modem into the cable, one must switch RECEIVE DATA (pin 2) with TRANSMITTED DATA (pin 3) on the RS-232C male connector as shown in Figure 2-10.

3. To mate to the 2 X 5 pin connector block, the following parts are needed.

Cable Receptacle	AMP PN 87133-5 DEC PN 12-14268-02
Locking Clip Contacts	AMP PN 87124-1 DEC PN 12-14267-00
Key Pin (pin 6)	AMP PN 87179-1 DEC PN 12-15418-00

2.5 VERIFYING OPERATION

The SBC-11/21 can be field tested to verify its functional operation. The Macro-ODT option and the loopback connectors are used to support the testing of the module.

2.5.1 Macro-ODT Option

The Macro-ODT option (part # KXT11-A2) consists of two, 24 pin, 2K X 8 PROM chips that contain the Macro-ODT code and module diagnostic programs. The Macro-ODT code is used to establish communication between the module and the user via console commands. The use of ODT commands is detailed in Chapter 4. The module diagnostic programs verify that the Parallel I/O and Serial Line Unit interfaces will function with commands from the microprocessor.

2.5.2 Loopback Connectors

The loopback connectors can be fabricated by the user for the module diagnostic tests. The 30-pin connector with the loopback jumper wires installed is shown in Figure 2-8, and is used with the Parallel I/O connector J3. The Serial Line Unit connector with the loopback jumper wires installed is shown in Figure 2-9, and is used with the serial line unit #2 connector J2.

2.5.3 Verification Procedure

The module must be restored to the standard factory configuration for the test to be valid, except that the start address must be 172000 instead of 10000. The module can be verified by using the following procedure.

1. Set the start address to 172000 as shown in Table 2-3.

2. Insert the high byte ODT ROM into socket set A, high byte socket E61. Ensure pin 1 is inserted into socket hole 3.

3. Insert the low byte ODT ROM into socket set A, low byte socket E47. Ensure pin 1 is inserted into socket hole 3.

4. Insert the 30-pin loopback connector (see Paragraph 2.5.2) into the module Parallel I/O connector J3.

5. Insert the 10-pin loopback connector (see Paragraph 2.5.2) into the serial line unit #2 connector J2.

6. Install the module into the LSI-11 backplane with the power turned off. An external power supply may be used to provide +5 Vdc to finger pins BV1, BA2, and AA2, +12 Vdc to finger pin BD2 and Ground to finger pins BJ1, AJ1, AT1, AC2, BC2, AM1, and BM1.

7. Connect an external terminal (printer or video). The terminal must be capable of generating a 7-bit ASCII code with odd parity or 8-bit ASCII code with no parity, and baud rates of 300, 600, 1200, 2400, 4800, or 9600. The terminal is connected to the serial line unit #1 connector J1 using a Digital BC20N-05 cable or equivalent. Turn the terminal on and on line.

8. Turn on the backplane power or enable the +5 Vdc and +12 Vdc sources. Monitor the module LED. The LED should illuminate and then return to the normal OFF state. If the LED remains illuminated, there is a fault in the serial line unit #1 circuits or the on-board RAM memory.

9. After the backplane power is turned on, press the "RETURN" key (carriage return) on the terminal in order for the module to synchronize its baud rate to that of the terminal. The module responds with the prompt character "@".

10 To initiate the module diagnostic programs press the "X" key. The diagnostic test will exercise the module including the Parallel I/O and Serial line unit #2. The results of the test are printed out on the terminal. The error results are listed in Table 2-14 and indicate what area of the module contains a fault. The error code "000000" indicates a good module.

Table 2-14 Diagnostic Fault Indicators

Printout	Parallel I/O Loopback Test	Internal Serial* I/O Loopback Test	External Serial** I/O Loopback Test
000000	Passed	Passed	Passed
000001	Failed	Passed	Passed
000010	Passed	Failed	Not Performed
000011	Failed	Failed	Not Performed
000100	Passed	Passed	Failed
000101	Failed	Passed	Failed
000110	Not Used	Not Used	Not Used
000111	Not Used	Not Used	Not Used

*The Internal Serial I/O Loopback Test exercises the parallel-to-serial conversion, the serial-to-parallel conversion, and the baud rate. This test can be performed without the loopback connector.

**The External Serial I/O Loopback Test exercises the above functions as well as the drivers, receivers, and the external signal paths.

CHAPTER 3
OPTIONS

3.0 GENERAL

The SBC-11/21 is a complete single board microcomputer that will operate on the LSI-11 bus or in a standalone configuration. In some applications it could be advantageous to add optional modules to the SBC-11/21 to extend its functionality beyond that provided by the module itself. A listing of all such options is given in the following sections. More information may be found in hardware manuals listed in Chapter 1 of this manual.

3.1 SUPPORTED OPTIONS

The following options are functionally compatible with the SBC-11/21. Software diagnostics for these options will run on the SBC-11/21 equipped with a mass storage device (TU58, RX01 or RX02) and the Macro-ODT option. To order diagnostics, contact your DEC sales representative.

TU58 — This low cost mass memory device can be used with the SBC-11/21 by attaching it to one of the serial I/O lines. TU58 offers random access to block formatted data on pocket-size cartridge media. It is ideal as a small computer systems device, as inexpensive archive mass storage, or as a software update distribution medium. A dual drive TU58 offers 512 Kb of storage space, making it one of the lowest cost complete mass storage subsystems available. For mounting flexibility, the TU58 is offered both as a component level subsystem and as a fully powered 5-1/2 inch rack-mount subsystem. The TU58 interfaces with the microprocessor over an RS-423 serial line interface.

DLV11-E — The DLV11-E is an asynchronous line interface module that interconnects the LSI-11 bus to standard serial communications lines. The module receives serial data, converts it to parallel data, and transfers it to the LSI-11 bus. Also, it accepts parallel data from the LSI-11 bus, converts it to serial data, and transmits it to the peripheral device. The module has jumper-selectable or software-selectable baud rates (50-19,200), and jumper-selectable data bit formats. The DLV11-E offers full modem control for EIA/CCITT interfaces.

DLV11-F — The DLV11-F is an asynchronous line interface module that interconnects the LSI-11 bus to several types of standard serial communications lines. The module receives serial data, converts it to parallel data, and transfers it to the LSI-11 bus. It also accepts parallel data from the LSI-11 bus, converts it to serial data, and transmits it to the peripheral

device. The module has jumper-selectable or software-selectable baud rates (50-19,200) and jumper-selectable data bits. THe DLV11-F supports either 20 mA current loop or EIA standard lines, but does not include modem control.

DLV11-J

The DLV11-J contains four independent asynchronous serial line channels used to interface peripheral devices to the LSI-11 bus. Each channel transmits and receives data from the peripheral device over EIA data leads (lines that do not use a control line). The module can be used with 20 mA current loop devices if a DLV11-KA adapter is used. The DLV11-J has jumper-selectable baud rates from 150 to 39.4 K baud.

DPV11-DA

The DPV11-DA is a single-line program-controlled, double-buffered communication device designed to interface the LSI-11 Bus to a serial synchronous line. This self-contained unit is capable of handling a wide variety of protocols including bit-oriented protocols such as SDLC, HDLC, ADCCP, and X.25, and byte-oriented protocols such as DDCMP and BISYNC.

The module is used for high speed synchronous lines such as remote batch, remote data collection, remote concentration, and communication networking. In addition to being compatible with EIA RS-232 and CCITT V.28 interface standards, the module is compatible with EIA RS-423 and 422 electrical standards, permitting low cost, local communications capability.

DRV11

The DRV11 is a parallel interface module that is used to interconnect the LSI-11 bus with general-purpose parallel line TTL or DTL devices. It allows program-controlled data transfers at rates up to 40K words per second and uses LSI-11 bus interface and control logic to generate interrupts and process vector handling. The data is handled by 16 diode-clamped input lines and 16 latched output lines. There are two 40-pin connectors on the module for user interface applications.

DRV11-B

The DRV11-B is an interface module that uses direct memory access (DMA) to transfer data directly between the system memory and an I/O device. The interface is programmed by the processor to move variable length blocks of 8- or 16-bit data words to or from specified locations in the system memory. Once programmed, there is no processor intervention required. The module can transfer up to 250K 16-bit words per second in the single-cycle mode and up to 500K 16-bit words per second in the burst mode. It also allows read-modify-restore operations.

DRV11-J — The DRV11-J provides sixty-four input/outut data lines on a double-height module for the LSI-11 bus. The DRV11-J also includes an advanced interrupt structure with bit interruptability up to 16 lines, programmable interrupt vectors, and program selection of fixed or rotating interrupt priority within the DRV11-J. The DRV11-J's bit interrupts for real-time response make it especially useful for sensor I/O applications. It can also be used as a general-purpose interface to custom devices, and two DRV11-Js can be connected back-to-back as a link between two LSI-11 buses.

DUV11-DA — The DUV11-DA synchronous line interface module establishes a data communication line between the LSI-11 bus and a Bell 201 synchronous modem or equivalent. The module is fully programmable with respect to sync characters, character length (up to 8 bits), and parity selection. The receiver logic accepts serial data for the LSI-11 bus. The transmitter logic converts the parallel LSI-11 bus data into serial data for the transmission line. The interface logic converts the TTL logic levels to the EIA voltage levels required by the Bell 201 modems and also controls the modem for half-duplex or full-duplex operation.

DZV11-B — The DZV11-B is an asynchronous multiplexer interface module that interconnects the LSI-11 bus with up to four asynchronous serial data communications channels. The module provides EIA interface voltage levels and data set control to permit dial-up (auto-answer) options with full-duplex modems such as Bell models 103, 113, 212, or equivalent. The DZV11-B does not support half-duplex operations or the secondary transmit and receive operations available in some modems such as Bell 202. The module has applications in data concentration and collection systems where front-end systems interface to a host computer and for use in a cluster controller for terminal applications.

IBV11-A — The IBV11-A is an interface module that interconnects the LSI-11 bus with the instrument bus described in IEEE standard 488 1975, "Digital Interface for Programmable Instrumentation." The IBV11-A makes a processor-controlled programmable instrument system possible. The module can accomodate up to 15 IEEE-488 devices.

MRV11-C — The MRV11-C is a flexible, high-density ROM module used with the LSI-11 bus. The module contains sixteen 24-pin sockets which accept a variety of user-supplied ROM chips. It will accept masked ROMS, fusible link PROMs, and ultraviolet erasable PROMs. It accepts several densities of ROM chips up to and including 4K

X 8 chips. Using these high-density chips gives the module a total capacity of 64K bytes. The contents of the module can be accessed in one of two ways -- either directly or window-mapped. Direct access provides total random access to all ROM locations on the module. Window-mapping provides two 2K-byte windows of memory address space to access 2K-byte segments of the ROM array. The segments that are viewed through each window can be varied under program control.

MSV11-D

The MSV11-D module has either 8K, 16K, or 32K X 16 bits of MOS memory. The module has an on-board memory refresh and performs the necessary LSI-11 bus cycles. The memory addressing is selectable by the user by configuring switch settings. The module can use a battery backup system to preserve data when primary power is lost.

MXV11-A

The MXV11-A is a dual height multifunction option module for the LSI-11 bus. It contains a read/write memory, provisions for read-only memory, two asynchronous serial line interfaces and a 60 Hz clock signal derived from a crystal oscillator. Read/write memory is supplied with either 8K or 32K bytes (4K or 16K words). Two 24-pin sockets are provided for +5 V read-only memories. 1K X 8, 2K X 8, or 4K X 8 ROMS may be used. The sockets may also be used for 256 words of bootstrap code. The two asynchronous serial lines transmit and receive EIA-423 signal levels from 150 baud to 38.4K baud. 20 mA active or passive current loop operation at 110 baud may be obtained with the DLV11-KA EIA to 20 mA converter option. The serial lines will not support the reader run function of the DLV11-KA option. The serial lines provide error indicator bits for overrun error, frame error, and parity error, but do not have modem controls. Serial line 1 may be configured to respond to a break signal. The serial lines have signal level interrupt logic. Serial line 1 along with serial line 0, may be used with any of several standard types of serial communication devices. The 60 Hz clock signal can be selected by a wirewrap jumper to provide line-time clock interrupts on the bus.

RXV21

The RXV21 floppy disk option is a random access mass memory device that stores data in fixed-length blocks on a preformatted, flexible diskette. Each diskette can store and retrieve up to 512K 8-bit bytes of data. The RXV21 system is rack-mountable and consists of an interface module, an interface cable, and either a single or dual RX02 floppy disk drive. The interface module converts the RX02 I/O bus to the LSI-11 bus structure. It controls the RX02 interrupts to the

processor, decodes device addresses for register selection, and handles the data interchange between the RX02 and the processor via DMA transfers. Power for the interface module is supplied by the LSI-11 bus.

RXV11 — The RXV11 option consists of an interface module, cable assembly, and either a single or dual drive RX01 floppy disk. This option is a random access, mass storage device that stores data in fixed-length blocks on a preformatted flexible diskette. Each diskette can store and retrieve up to 256K, 8-bit bytes of data. The RXV11 system is rack mountable in the standard 48.3 cm (19 in) cabinet.

3.2 UNSUPPORTED OPTIONS

The following LSI-11 bus options listed in Table 3-1 are not guaranteed to be functionally compatible with the SBC-11/21, and are termed "unsupported." Their diagnostics are not available.

Table 3-1 Unsupported LSI-11 Options

AAV11-A	FEPTC-BA	LPV11	TRV11
ADV11-A	FPF11	MRV11-AA/BA/VU	TSV11
BDV11-AA/BA	IPV12	MSV11-E/P	VMV66-A
DA11-MS/QQ/QU	KD11-F	NCV11-A	VK170
DAV11-A/B	KD11-HA	REV11	VSV11
DRL11-SN	KDF11-AB/AC/BB	RKV11	VTV01-A
DUV11-E/F	KDF11-BC/P	RLV11	VTV30-H
DUV25	KPV11-A	RLV12	
DW11	KWV11-A	TEV11	
DWV11-A	LAV11		

CHAPTER 4
MACRO-ODT

4.0 GENERAL

The Macro-ODT is the KXT11-A2 option that is available for users of the SBC-11/21 single board computer. The option consists of two 24-pin, 2K X 8 ROM chips which contain the Macro-ODT firmware, and a complete listing of the firmware. The chips are installed on the module using the PROM sockets.

Macro-ODT allows the user to do the following:

- Examine and deposit data in memory or general registers.
- Examine or alter the Processor Status Word (PSW).
- Start the execution of the program.
- Resume the execution of a halted program.
- Bootstrap programs from a mass storage device (TU58 cassette, RX01 or RX02 floppy disks).
- Run a confidence test for on-board devices.

4.1 INSTALLATION AND CONFIGURATION

This is described in detail in Chapter 2 and the user is referred to it for installation and startup instructions.

4.2 ENTRY CONDITIONS

Macro-ODT is entered:

1. Upon power up.
2. Via the "BREAK" key on the console terminal.
3. Execution of a HALT instruction.
4. Assertion of the BHALT L signal on the LSI-11 Bus.
5. Accessing nonexistent memory (i.e., a bus timeout).

4.2.1 Macro-ODT Input Sequence

Upon entry to Macro-ODT, the RBUF register is read using a DATI and the character present in the buffer is ignored. This is done so that erroneous characters or user program characters are not interpreted by Macro-ODT as commands, especially when a program is halted.

The input sequence for Macro-ODT is as follows.

1. Read and ignore character in RBUF.

2. Output a <CR> <LF> to terminal.

3. Output contents of PC (program counter R7) in six digits to terminal if ODT is entered via a BREAK, BHALT, HALT instruction or an attempt to fetch an instruction from nonexistent memory. Output a "?" to the terminal if ODT is entered via a bus timeout.

4. Output a <CR> <LF> to terminal.

5. Output the prompt character, @, to terminal.

6. Enter a wait loop for terminal input. The Done flag, bit 7 in RCSR, is tested using a DATI. If it is 0, the test continues.

7. If RCSR bit 7 is a 1, then low byte of RBUF is read using a DATI.

4.2.2 Macro-ODT Output Sequence

The output sequence for ODT is as follows.

1. Test XCSR byte 7 (Done flag) using a DATI and if a 0, continue testing.

2. If XCSR bit 7 is 1, write character to low byte of XBUF using a DATI followed by a DATO (high byte is ignored by interface).

4.3 MACRO-ODT COMMANDS

The Macro-ODT commands are listed in Table 4-1 and described in the following paragraphs. The commands are a subset of ODT-11 and use the same command character. The Macro-ODT internal states are listed in Table 4-2. For each state only specific characters are recognized as valid inputs; other inputs invoke a "?" response.

The parity bit, bit 7, on all input characters is ignored by Macro-ODT, and if the input character is echoed, the state of the parity is copied to the output buffer (XBUF). Output characters internally generated by ODT (e.g., <CR>) have the parity bit equal to 0. All input characters are echoed. Only uppercase command characters are recognized.

NOTE

The use of ODT commands establishes a dialog between the user and the microcomputer. Therefore, all the characters typed by the user are underlined and the system response is not underlined.

Table 4-1 Macro-ODT Commands

Command	Symbol	Use
Slash	/	Prints the contents of a specified location.
Carriage Return	<CR>	Closes an open location.
Line Feed	<LF>	Closes an open location and then opens the next location. This command cannot be used with the general registers.
Internal Register Designator	R	Opens a specific processor register.
Processor Status Word Designator	S	Opens the PSW -- must follow R command.
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Boot from Device	D	Loads and runs programs from floppy diskettes or TU58 cassettes.
Execute Diagnostics	X	Runs SBC-11/21 module verification diagnostic.

4.3.1 /(ASCII 057) Slash

This command is used to open an on board module address, LSI-11 Bus address, processor register, or processor status word and must be normally preceded by other characters which specify a location. In response to /, Macro-ODT prints the contents of the location (i.e., six characters) and then a space (ASCII 40). After printing is complete, Macro-ODT waits for either new data for that location or a valid close command (<CR> or <LF>). The space character is issued so that the location's contents and possible new contents entered by the user are legible on the terminal.

Example: @001000/12525<SPACE>

where:

@ = Macro-ODT prompt character

001000 = octal location in the LSI-11 Bus address space desired by the user (leading 0s are not required)

/ = command to open and print contents of location

012525 = contents of octal location 1000

<SPACE> = space character generated by Macro-ODT

A issued immediately after a prompt character causes a ? <CR> <LF> to be printed because a location is not open.

4.3.2 <CR>(ASCII 15) Carriage Return

This command is used to close an open location. If a location's contents are to be changed, the user should precede the <CR> with the new data. If no change is desired, <CR> closes the location without altering its contents.

Example: @R1/004321<SPACE> <CR> <CR> <LF>
@

Processor register R1 was opened and no change was desired so the user issued <CR>. In response to the <CR>, Macro-ODT printed <CR> <LF>@.

Example: @R1/004321<SPACE> 1234 <CR> <CR> <LF>
@

In this case the user desired to change R1, so new data, 1234, were entered before issuing the <CR>. Macro-ODT deposited the new data in the open location and then printed <CR> <LF> @.

Macro-ODT echoes the <CR> entered by the user and then prints an additional <CR>, followed by a <LF>, and @.

Example: @1000/012525<SPACE> 1234 <CR> <CR> <LF>

where:

first line = new data of 1234 entered into location 1000 and the location is closed with <CR>

4.3.3 <LF> (ASCII 12) Line Feed

This command is used to close an open location and then open the next contiguous location. LSI-11 Bus addresses are incremented by 2. If a processor register is open and a <LF> command is issued, the register is closed and any data that was typed in prior to <LF> will not enter the register. ODT prints the error message <CR> ? <CR> <LF>. If the open location's contents are to be changed, the new data should precede the <LF>. If no data are entered, the location is closed without being altered.

Example: @1000/123456<SPACE> <LF> <CR> <LF>
@1002/054321<SPACE>

Table 4-2 Macro-ODT States and Valid Input Characters

State	Example of Terminal Output	Valid Input
1	@	0--7 P X D
2	@R	0--7 S
3	@1000/ 123456	0--7 <CR>
4	@R1/123456	0--7 <CR> <LF>
5	@1000	0--7 / G
6	@R1 or @RS	/
7	@1000/ 123456 1000	0--7 <CR> <LF>
8	@R1/ 123456 1000	0--7 <CR>
9*	@DY	0 1 <CR>
10*	@DX	0 1 <CR>
11*	@DD	<CR> 0 1

*NOTE: Do not enter 0 or 1 followed by <CR>.

In this case, the user entered <LF> with no data preceding it. In response, Macro-ODT closed location 1000 and then opened location 1002.

4.3.4 R(ASCII 122) Internal Register Designator

The "R" character when followed by a register number, 0 to 7, or PS designator, S, will open that specific processor register.

Example: @R0/054321<SPACE>

or

@R7/000123<SPACE> 456 <CR> <CR> <LF>

@

If more than one character is typed (digit or S) after the R, Macro-ODT uses all the characters as the register designator.

Example: @R00007/000123<SPACE> <CR> <CR> <LF>

@

4.3.5 S (ASCII 123) Processor Status Word

This designator is for opening the PSW (processor status word) and must be employed after the user has entered the R register designator.

Example: @RS/100377<SPACE> 0 <CR> <CR> <LF>
@RS/100317<SPACE>

Note that the T-BIT FILTER prevents the user from setting the T-BIT via Macro-ODT. The T-BIT can be cleared by any write to the PSW. When the filter is disabled, the T-BIT can be set by loading the PSW to set bit 4 to a one. This is normally not considered desirable. The T-BIT FILTER can be disabled by setting bit 15 of location 167772 to a one.

The PRIORITY 7 FILTER prevents the user from setting a priority level of 7 via Macro-ODT. Operation at priority level 7 masks out (disables) the BREAK interrupt and makes it impossible to return to Macro-ODT. This type of operation is normally undesirable. If required, the PRIORITY 7 FILTER can be disabled by setting bit 7 of location 167772 to a one. With the filter disabled, a priority level of 7 is established by writing 340 into the PSW.

4.3.6 G (ASCII 107) Go

This command is used to start program execution at a location entered immediately before the G.

Example: @200G

The Macro-ODT sequence for a G, after echoing the command character, is as follows.

1. Load R7 (PC) with the entered data. (In the above example, R7 is equal to 200 and that is where program execution begins.)

2. The PS is cleared to 0.

3. The LSI-11 Bus is initialized by the processor's asserting BINIT L for 17 microseconds minimum and then negates BINIT L.

4. The user program begins execution at the location specified.

The user is warned that the command clears the PSW, which will permit clock interrupts to be acknowledged. Failure to load the address of the clock service routine into the clock vector address (100) may lead to unpredictable results.

4.3.7 P (ASCII 120) Proceed

This command is used to resume execution of a program. No programmer-visible machine state is altered using this command.

Example: @P

Program execution resumes at the address pointed to by R7. After P is echoed, Macro-ODT exits and the program resumes execution.

4.3.8 DD, DX, DY Bootstraps

This command is used to bootstrap a standalone program or XXDP+ diagnostics from an RX01, RX02 floppy diskette or a TU58 tape cartridge. The next character after the D command determines the type of device being booted. The numerical character, either 0 or 1, is optionally used to specify a particular drive or unit of the device being booted. If <CR> is typed instead of 0 or 1, then unit 0 is assumed.

Examples: Boot unit 0 of TU58 device:

@DD<CR>

Boot unit 1 of RX01 device:

@DX1

Boot unit 0 of RX02 device:

@DY0

NOTE
Do not type both unit number and <CR>.

To boot a diskette drive, ODT expects the RXV11 or RXV21 controller CSR address to be configured for 177170. To boot the TU58, it must be connected to SLU2 and the baud rate set for 38,400.

Any error detected during the execution of a boot command will cause a halt at one of several addresses in the boot portion of the ROM, with the PC contents printed on the console. The actual addresses, and the specific error each signifies, are given in the listing supplied with the option.

Some errors, however, are not reported. If no TU58 is connected to SLU2, or if baud rates are incompatible, no error indication is given after using the "DD" command and the program simply waits forever. This is also true when booting from floppies when the drive power if off. In either case, the user can use <BREAK> to return to ODT prompt level "@".

The D command will perform the following operations.

1. If there is no RAM memory at address zero, it will cause a halt.

2. It will initialize the LSI-11 Bus by asserting BINIT L for 17 microseconds minimum.

3. It will read Block 0 (the first 512 bytes) from the selected mass storage device into memory locations 000--777.

4. It will read location zero and if it is 240, it will load R1 register with the CSR address of the booted device, load R0 register with the selected unit or drive number, and jump to location zero.

5. If the contents of location zero is 260, then the mass storage device contains a "standalone program". Macro-ODT interprets the contents of locations 2, 4, and 6 as a RADIX-50 encoded six character file name. Macro-ODT assumes that the mass storage device is an RT-11 file structured volume and searches the directory of the volume for the file name provided by locations 2, 4, and 6. When the file is found, the entire file is loaded into contiguous memory starting at location zero. The R0 register is loaded with the number of the unit or drive and the R1 register is loaded with the CSR address of the booted device. The Stack Pointer (SP) is loaded with the contents of location 42 and the Program Counter (PC) is loaded with the contents of location 40. The program begins execution.

6. If the contents of location zero is not 240 or 260, then the device does not contain a valid boot block. The boot command is aborted and the SBC-11/21 is initialized as if a power up occurred.

4.3.9 X (ASCII 130) Diagnostics

After typing the letter "X", a 3-second delay will occur, then an octal number will be displayed. This command is described in detail in Chapter 2.

4.4 INITIALIZATION

When it is desired to re-initialize the system without removing power, enter 173000G from the console in response to the "@". Note that a carriage return will have to be typed, after a pause, to resynchronize the terminal as described in the following example.

Example: @173000G
After a pause of at least one second, type <CR> to resynchronize.

4.5 WARNINGS AND PROGRAMMING HINTS

The following warnings and programming hints are provided to assist the user in operating Macro-ODT.

4.5.1 Error Decoding

In the event of an unexpected appearance of "@", it is a good practice to examine the word at 167774. This is an error word that indicates the cause of entry to ODT. A HALT instruction, BREAK, or an attempt to fetch from nonexistent memory will appear as 100000. Other attempted bus transactions to nonexistent memory will appear as 000200, or 000201 if accessed by the stack pointer R6.

4.5.2 ODT Stack Warning

While performing its various functions, Macro-ODT requires two words of user stack. It will transparently push and pop internal information there. It is imperative then, that the user always provide two more words than those actually necessary for the proper execution of the application program. If desired, these two words can be given back when the program is completely debugged and operating within its own ROMs without ODT.

For proper program operation, R6 should always contain a valid even RAM memory address. Failure to observe this rule will cause unpredictable results.

4.5.3 Addresses to Avoid

Since the firmware uses the top of the SBC-11/21 on-board RAM as its scratchpad, the user should not write to any address above 167642 unless specifically designated in this manual.

The vector at 140 governs the BREAK interrupt. Altering locations 140 and 142 could result in the inability to suspend program execution.

4.5.4 CPU Priority

When the PSW is set to 340, the BREAK key will have no effect, and will not invoke Macro-ODT. Running at a level 6 priority (PSW set to 300) is adequate for most programming needs. This will disable all interrupts except for BREAK.

4.5.5 Terminal Related Problems

Macro-ODT echoes every character typed in response to the "@" prompt. Some intelligent terminals also respond to control characters as commands. The results may include loss of communication.

4.5.6 Spurious HALTs

When the last word of an instruction is all zeros and causes a bus timeout, Macro-ODT will interpret it as a HALT instruction. It will then print the contents of PC on the terminal, before issuing the "@" prompt.

4.5.7 Serial I/O Protocol

The Macro-ODT operates the serial line interface in full duplex mode and each character is echoed by the microprocessor to the terminal. Programmed I/O techniques are used rather than interrupts. When the Macro-ODT firmware is busy printing a multi-character message using the transmit side of the interface, the firmware is not monitoring the receive side for incoming characters. Any characters coming in at this time are lost. The interface may set the overrun error bit, but the Macro-ODT does not check this bit and those characters are not recognized. All peripherals communicating with the Macro-ODT through this interface must observe this protocol.

4.5.8 Interrupt Vector Initialization

Upon power-up, Macro-ODT initializes the LTC interrupt vector (REVNT at 100) and the BREAK interrupt vector (BKRQ at 140). No other vectors are initialized and may contain spurious data.

CHAPTER 5
SYSTEM ARCHITECTURE

5.0 GENERAL

This chapter describes the architecture of the microprocessor, memory organization and power up method. The microprocessor architecture describes the registers, hardware stack, interrupts and DMA mechanism. The memory organization describes byte or word addressing and memory mapping. The power up procedure and initialization are also briefly described.

5.1 MICROPROCESSOR ARCHITECTURE

The SBC-11/21 microprocessor executes a subset of the PDP-11 instruction set. It has eight, high speed, general purpose registers that are used as accumulators, address pointers, index registers and for other specialized functions. The microprocessor executes single and double operand instructions using either 16-bit words or 8-bit bytes. The Direct Memory Access (DMA) function transfers data directly from the LSI-11 bus to the on board I/O devices and memory, while the program continues to run.

5.1.1 Registers

With reference to Figure 5-1, the microprocessor contains a number of internal registers which are used for various purposes. The registers are broken up into two groups:

- o General
- o Status

5.1.1.1 General Registers -- The microprocessor contains 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 performed from one general register to another, from one memory location or device register to another, or between memory locations or a device register and a general register.

Registers R6 and R7 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. It is normally used for addressing purposes only and not as an accumulator.

5.1.1.2 Status Register -- The Processor Status Word (PSW) 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 PSW format is shown in Figure 5-1, and Table 5-1 lists status word bit descriptions. Certain instructions allow programmed manipulation

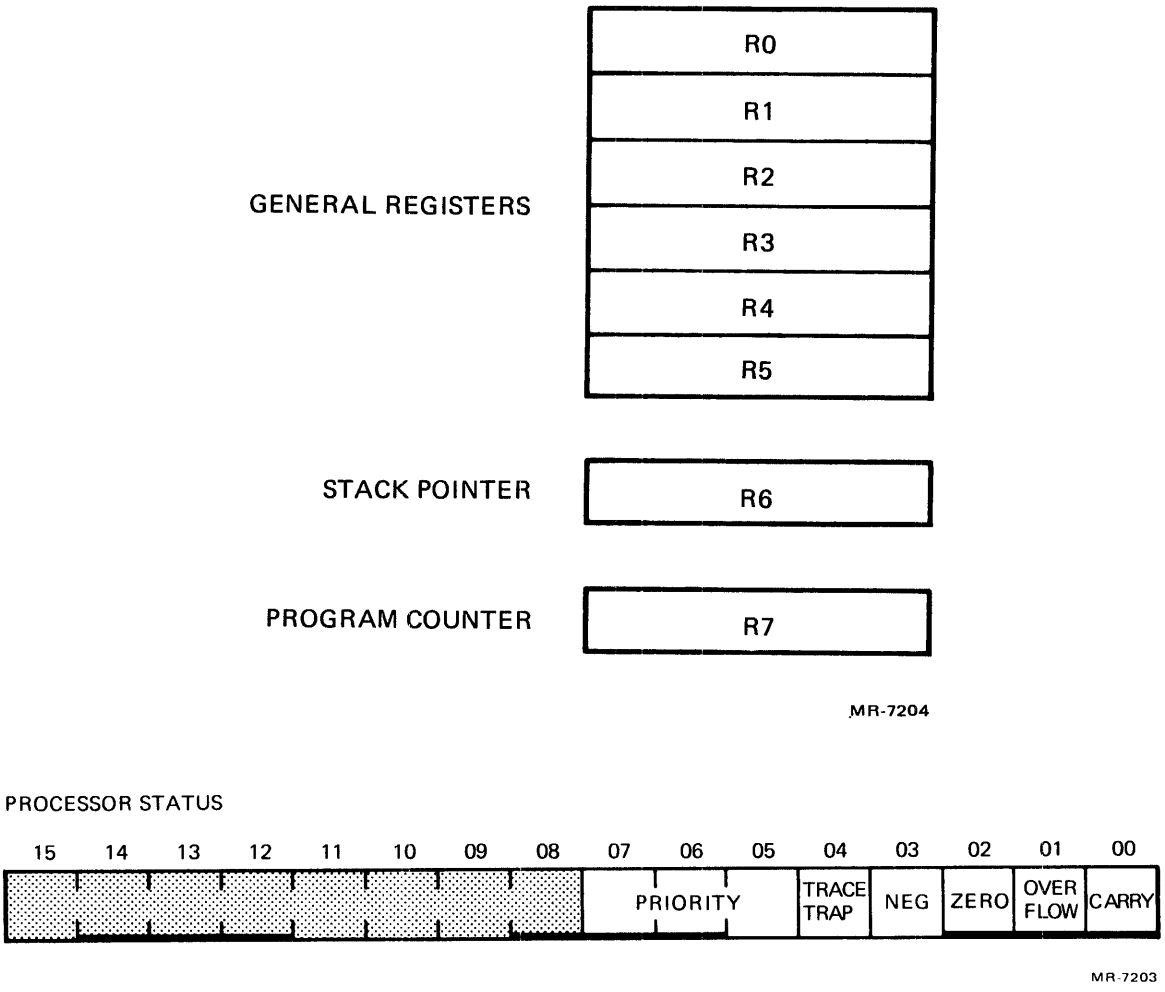


Figure 5-1 Registers and Processor Status Word

of condition code bits and loading and storing (moving) the processor status. Not all instructions affect the condition codes in an obvious manner. For details of specific instructions refer to Chapter 7.

5.1.2 Hardware Stack

The hardware stack is part of the basic design architecture of the SBC-11/21. It is an area of memory set aside by the programmer or by the operating system for temporary storage and linkage. It is handled on a LIFO (last in/first out) basis, where items are retrieved in the reverse of the order they were stored. The stack starts at the highest location reserved for it (376 octal at power up) and expands linearly downward to a lower address as items are added to the stack.

Table 5-1 Processor Status Word Bit Descriptions

Bit	Name	Description
15--8	N/A	These bits are not accessible to the programmer and contain no valid information.
7--5	Priority	These bits define the current priority level of the microprocessor program and only interrupts with a higher priority are recognized by the microprocessor. Table 5-2 describes the microprocessor interrupt levels as functions of bits 5-7.
4	Trace	When set this bit allows the microprocessor to trap to locations 14 and 16 after an instruction is executed. It can only be set by executing an RTI or RTT instruction with the desired PSW already on the stack. The trace bit is useful in debugging programs by allowing them to be single stepped.
3	Condition Code N	This bit is set when an instruction causes the result to be negative.
2	Condition Code Z	This bit is set when an instruction causes the result to be zero.
1	Condition Code V	This bit is set when an instruction causes an overflow condition.
0	Condition Code C	This bit is set when an instruction causes a carry out of the most significant bit.

It is not necessary to keep track of the actual locations into which data is being stacked. This is done automatically through the use of the Stack Pointer (SP). Register six (R6) always contains the memory address where the last item is stored in the stack. Instructions associated with subroutine linkage and interrupt service automatically use register six as the hardware stack pointer. For this reason, R6 is frequently referred to as the system SP. The hardware stack is organized in full word units only.

5.1.3 Interrupts

Interrupts are requests, made by peripheral devices, which cause the processor to temporarily suspend its present program execution to service the requesting device. A device can interrupt the processor only when its priority is higher than the processor priority indicated by PSW<7:5>, as shown in Table 5-2.

SBC-11/21 supports a vectored interrupt structure with priority on four levels. In addition, it supports two nonmaskable interrupts: Power Fail and -HALT.

Every interrupt except HALT is associated with an interrupt vector. The interrupt vector is a pair of words, next PC (address of that device's service routine) and next PSW (priority with which the routine must be executed). Upon interrupt the current PC and PSW are saved on the stack and the new PC and PSW are loaded from the vector address.

Up to 64 vectors may reside in the first 256 memory locations (octal 374 is the highest vector location). The vector address is provided by the interrupting device (external vector address) or generated internally by the microprocessor.

NOTE

The Power Fail interrupt uses interrupt vector address 24. HALT interrupt is not associated with a vector. It pushes the PC and PSW onto the stack and immediately goes to the restart address with PSW 340.

Table 5-2 PSW Interrupt Levels

Microprocessor Priority	Interrupt Levels Acknowledged	PSW Bits 7	6	5
level 7	Unmaskable Interrupt	1	1	1
level 6	7	1	1	0
level 5	7,6	1	0	1
level 4	7,6,5	1	0	0
level 0-3	7,6,5,4	0	X	X

The SBC-11/21 has eleven interrupt sources of which nine are maskable and two are nonmaskable. The interrupt request can occur at any time but is not acknowledged until the completion of the current instruction. This allows the microprocessor to execute a program until the interrupt occurs and then the microprocessor vectors to the service routine for the interrupt. After the service routine is completed, a Return From Interrupt (RTI) instruction is executed. The microprocessor then pops the top two words from the system stack, which were the original PC and PSW, and the interrupted program is resumed.

The eleven interrupt sources with their respective priorities are listed in Table 5-3. For a device to be serviced, its priority level must be higher than the current microprocessor level. When two devices with equal priority numbers simultaneously request an interrupt, the device listed closest to the top of the table will be serviced first.

When the interrupt is requested by several LSI-11 bus devices simultaneously, the device electrically nearest to the SBC-11/21 is serviced first.

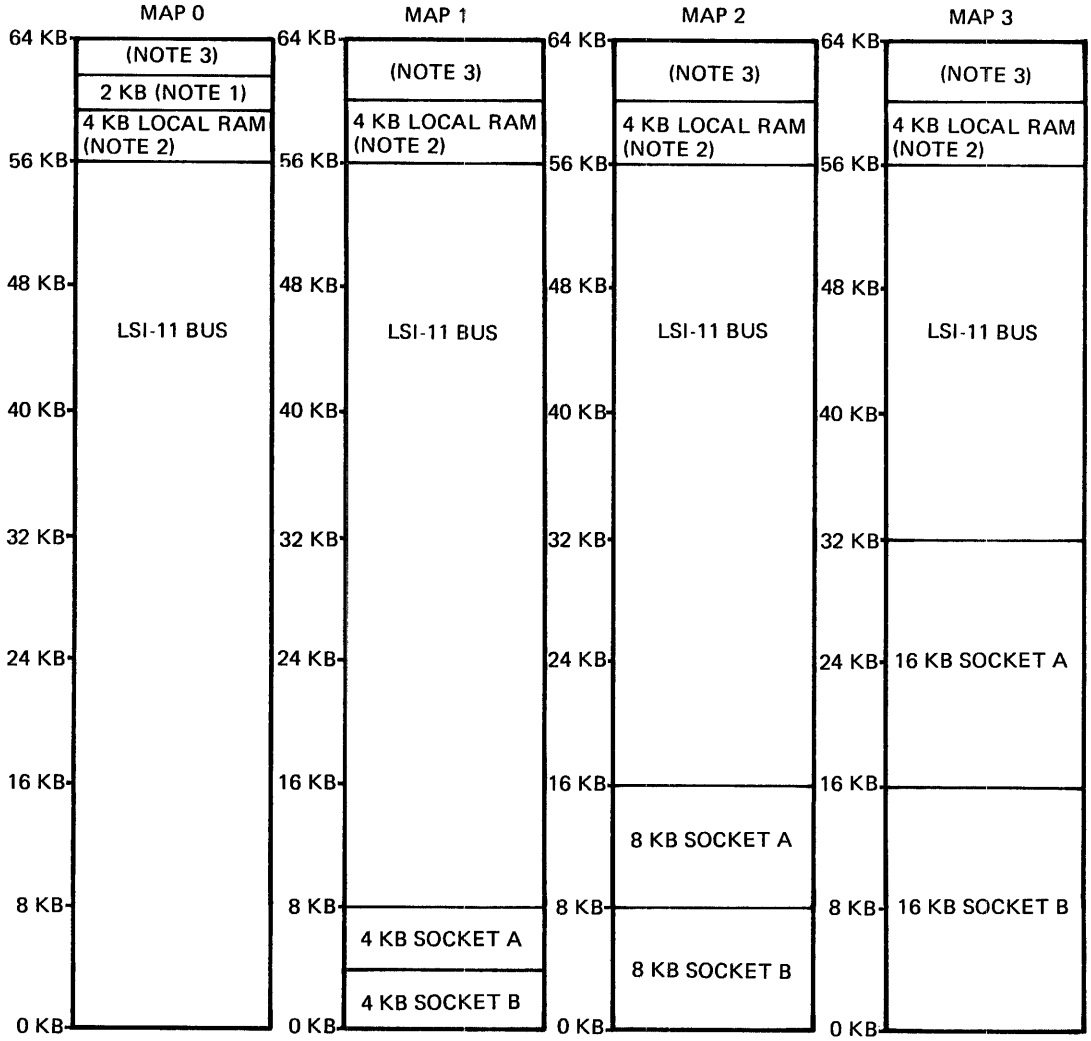
5.2 DMA (DIRECT MEMORY ACCESS)

DMA allows the programmer to implement block transfers by specifying the direction of transfer, the starting address in memory, the number of words and any additional parameters that a particular external device requires. SBC-11/21 does not have an on-board DMA interface, but can support DMA transfers for external devices via the LSI-11 bus interface. A typical device utilizing the DMA technique is the RX02 double-density floppy. User designed devices can also be connected to the SBC-11/21 DMA facility. For a more detailed treatment the reader is referred to Chapter 9.

5.3 MEMORY ORGANIZATION

The SBC-11/21 memory consists of onboard memory and LSI-11 bus memory. The memory map configurations and the types of onboard memory chips are described in Chapter 2. The memory maps are described in Figure 5-2. Addresses from 0 to 376 octal are reserved for vector locations and addresses from 60 Kb to 64 Kb are reserved for I/O devices.

The address space of the SBC-11/21 module is 64 Kbytes. A 16-bit word is composed of two 8-bit bytes with bits 0--7 representing the low byte and bits 8--15 representing the high byte. Words are always addressed by even numbers. The bytes are addressed by either even or odd numbers. The high bytes are stored in the odd numbered locations and the low bytes are stored in the even numbered locations.



NOTES:

1. SOCKET SET A IS MAPPED OVER SOCKET SET B AND IS THEREFORE LIMITED TO USING EITHER SOCKET A OR SOCKET B, BUT NOT BOTH TOGETHER.
2. ADDRESSES 160000 THROUGH 160007 ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
3. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUs AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.

MR-6643

Figure 5-2 Memory Maps

Table 5-3 SBC-11/21 Interrupts

Interrupt Source	Control Signal	Priority Level	Vector Address**
HALT	-CTMER	nonmaskable	*
POWER FAIL	-PFAIL	nonmaskable	24
LSI-11 Bus Signal BHALT	BKRQ	7	140
LSI-11 Bus Signal BEVNT	REVNT	6	100
SLU2 REC	RDL2	5	120
SLU2 XMIT	XDL2	5	124
PARALLEL I/O B	PBRQST	5	130
PARALLEL I/O A	PARQST	5	134
SLU1 REC	RDL1	4	60
SLU1 XMIT	XDL1	4	64
LSI-11 Bus Signal BIRQ4	IRQ4	4	Read from LSI-11 Bus

*The microprocessor jumps directly to the restart address with a PSW priority level 7. (RESTART is loaded into PC and 340 into PSW).

**All vectors defined in this table are internal vectors supplied by the microprocessor, except for the BIRQ4 interrupt which is read from the bus.

5.4 POWER UP/DOWN FACILITY

SBC-11/21 has facilities for assuring an automatic program start-up when power is turned on and for orderly shutdown, without loss of data, when power is turned off or lost. This is accomplished by a combination of hardware features and software.

Hardware features:

- Two signal lines in the LSI-11 bus called BDCOK H and BPOK H used only for power up/down protocol. These signals are usually generated by the power supply.
- One signal line in the LSI-11 bus, called BINIT L, which resets the system.
- The vectoring on interrupt facility of the SBC-11/21.
- Battery backup connections

Software features:

The programmer must provide power up and power down routines, and store their addresses at the jumper selected start address for power up, and at location 24 for the power down routine.

The detailed description of the power up/down protocol will be found in Chapter 9.

6.0 GENERAL

The SBC-11/21 has three on-board interfaces, one parallel I/O and two serial I/O lines. These interfaces contain a number of programmable features that allow the user to change their operating characteristics. This chapter explains how this can be accomplished.

SBC-11/21 is also equipped with hardware that enables the microprocessor to behave in a controlled manner when the power is turned on and off. This specialized hardware requires software to make it work and an example in Appendix C describes the basic principles of this programming.

6.1 ASYNCHRONOUS SERIAL LINE UNITS

The two Serial Line Units (SLUs) are described by Figure 6-1 and provide the means of transferring data between the microprocessor and two user connectors J1 or J2. The user interfaces support the RS232C EIA standard and RS423 protocol, at baud rates ranging from 300 to 38400.

Each SLU is equipped with four addressable registers that are listed in Table 6-1, described by Figure 6-2, and functionally described by Tables 6-2, 6-3, 6-4, and 6-5. The registers can be accessed by the microprocessor or any DMA bus master. SLU1, with the proper software handling, can be used as a system console and is capable of initiating a hardware interrupt when BREAK is detected. SBC-11/21 can be configured for the BREAK to cause a level 7 interrupt with an internal vector of 140, enable the BHALT interrupt or request a HALT trap to the restart address. SLU2 provides three line time clocks at 50 Hz, 60 Hz and 800 Hz, which can be wire-jumper configured to enable the BEVNT level 6 interrupt. Refer to Chapter 2 for details on how to configure the SLUs.

Table 6-1 Serial Line Unit Register Addresses

Register	Description	Address	AD2	AD1
SLU1				
RCSR	Receiver Control and Status	177560	0	0
RDBR	Receiver Data Buffer	177562	0	1
TCSR	Transmitter Control and Status	177564	1	0
TDBR	Transmitter Data Buffer	177566	1	1
SLU2				
RCSR	Receiver Control and Status	176540	0	0
RDBR	Receiver Data Buffer	176542	0	1
TCSR	Transmitter Control and Status	176544	1	0
TDBR	Transmitter Data Buffer	176546	1	1

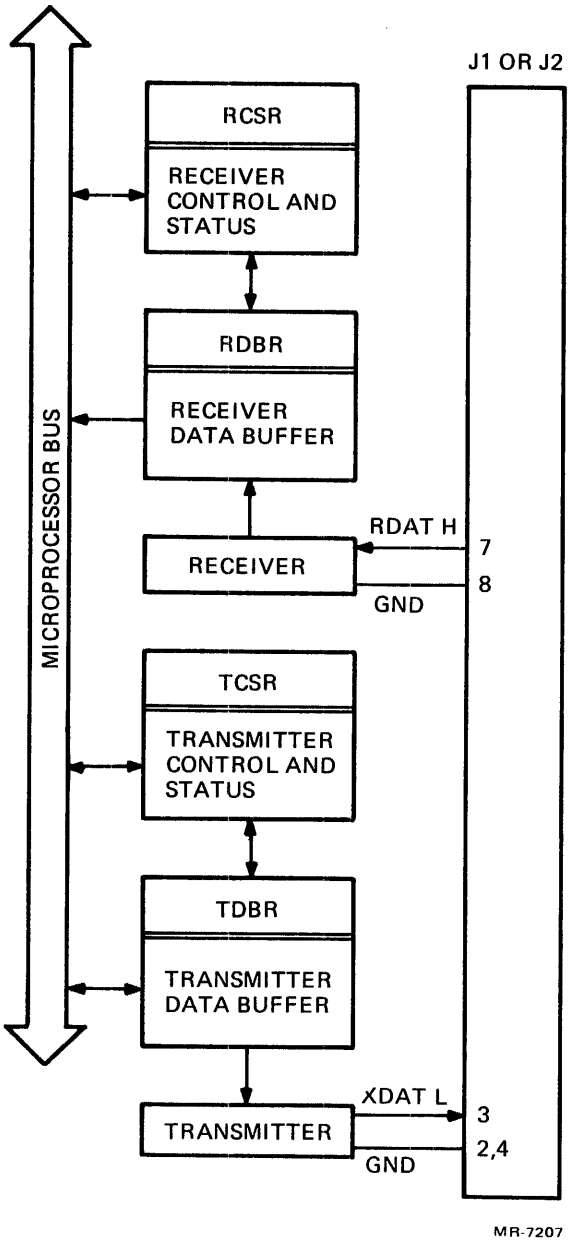


Figure 6-1 Serial Line Unit Interface (SLU)

Table 6-2 Receiver Control and Status Bit Descriptions

Bit	Name	Direction	Function
12-15	Not Used	Read Only	Reserved for future use.
11	Receiver Active	Read Only	This bit is set to a "one" by the start bit and is cleared to a "zero" by the stop bit at the end of each byte. It is also cleared to a "zero" on power-up.
08-10	Not Used	Read Only	Reserved for future use.
07	Receiver Done	Read Only	This bit is set to a "one" when the byte received is transferred into the RCV Data Buffer. It is cleared to a "zero" when the RCV Data Buffer is read. It is also cleared to "zero" on power-up.
06	Receiver Interrupt Enable	Read Write	This bit is set to a "one" under program control. When set, it allows an Interrupt Request to be initiated whenever the Receiver Done bit is set. It is cleared to a "zero" by RESET, power-up or under program control. Refer to Chapter 2 for interrupt jumper configuration.
00-05	Not Used	Read Only	Reserved for future use.

6.1.1 Data Baud Rates

The serial line units transmit or receive data serially by bit and by character. Each character consists of ten bits; a start bit, 8 bits of data, and the stop bit. Split speed operation of the receiver and transmitter for the SLU is not supported and the user cannot supply an external baud rate clock to the SLU. During Power Up or RESET, the outputs are disabled and later the baud rate defaults to 300.

The baud rates are programmable for 300, 600, 1200, 2400, 4800, 9600, 19200 or 38400 when bit 01 of the Transmitter Control and Status register (TCSR) is set to a one. The baud rate is then selected by programming bits 05--03 of the TCSR.

These four bits used for the baud rate selection are level sensitive, and do not latch. Therefore, the software in control of the TCSR must use bit set and bit reset type instructions after the baud rate is written into the SLU. Each SLU provides an output at TTL levels to pin 1 of its connector (J1 or J2) at 16 times the baud rate selected for that SLU.

RECEIVER CONTROL AND STATUS REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	RCV ACT	0	0	0	RCV DONE	RCV IE	0	0	0	0	0	0

SLU 1 ADDRESS 177560
SLU 2 ADDRESS 176540

RECEIVER DATA BUFFER REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ERR	OR ERR	FR ERR	0	REC BRK	0	0	0	RECEIVED DATA BUFFER							

SLU 1 ADDRESS 177562
SLU 2 ADDRESS 176542

TRANSMITTER CONTROL AND STATUS REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	XMIT RDY	XMIT IE	PBR SEL2	PBR SEL1	PBR SEL0	MAINT	PBR ENB	XMT BRK

SLU 1 ADDRESS 177564
SLU 2 ADDRESS 176544

TRANSMITTER DATA BUFFER REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	TRANSMIT DATA BUFFER							

SLU 1 ADDRESS 177566
SLU 2 ADDRESS 176546

MR-7208

Figure 6-2 Serial Line Unit Register Bit Maps

Table 6-3 Receiver Data Buffer Bit Descriptions

Bit	Name	Direction	Function
15	Error	Read Only	The bit is set to a "one" when the Overrun Error or the Framing Error bit is set. It is cleared to a "zero" when the error producing condition is removed.
14	Overrun Error	Read Only	The bit is set to a "one" when the received byte is transferred into the RCV Data Buffer before the RCV DONE bit is cleared. The Overrun Error indicates that the previous byte in the RCV Data Buffer was not cleared prior to receiving a new byte. The bit is updated when a byte is transferred into the RCV Data Buffer and cleared to a "zero" on power-up.
13	Framing Error	Read Only	The bit is set to a "one" when the received character does not have a valid stop bit and is transferred into the RCV Data Buffer. The bit is cleared to a "zero" when a character with a valid stop bit is received and is transferred into the RCV Data Buffer or on power-up.
12	Not Used	Read Only	Reserved for future use.
11	Received Break	Read Only	The bit is set to a "one" when the received signal goes from a MARK to a SPACE and stays in the SPACE condition for 11 bit times after serial reception starts. The bit is cleared to a "zero" when the received signal returns to the MARK condition or on power-up.
08-10	Not Used	Read Only	Reserved for future use.
00-07	Received Data Buffer	Read Only	These 8 bits represent the most recent byte received. These bits are cleared to "zero" on power-up.

Table 6-4 Transmitter Control and Status Bit Description

Bit	Name	Direction	Function
08-15	Not Used	Read Only	Reserved for future use.
07	Transmitter Ready	Read Only	The bit is set to a "one" when the XMIT DATA BUFFER is ready to accept a byte. The bit is cleared to a "zero" by writing into the XMIT DATA BUFFER. The bit is also set to a "one" on power-up.
06	Transmitter Interrupt Enable	Read Write	This bit is set to a "one" under program control. When set, it allows an Interrupt Request to be initiated whenever the Transmitter Ready bit is set. The bit is cleared to a "zero" by RESET, power-up or under program control.
03-05	Programmable Baud Rate Select	Read Write	The condition of these bits selects the baud rate under program control, provided the Programmable Baud Rate Select Enable bit is set. The baud rates are selectable by setting these bits as follows. 05 04 03 BAUD RATE 0 0 0 300 0 0 1 600 0 1 0 1200 0 1 1 2400 1 0 0 4800 1 0 1 9600 1 1 0 19200 1 1 1 38400 When the Programmable Baud Rate Select Enable bit is not set baud rate defaults to 300.

Table 6-4 Transmitter Control and Status Bit Description (Cont)

Bit	Name	Direction	Function
			NOTE The transmitter Programmable Baud Rate Select and Enable bits are level sensitive and not latched. This requires that software in control of the TCSR must use bit set and clear instructions to access the TCSR once the baud rate has been written into the SLU.
02	Maintenance	Read Write	This bit is controlled by the program. When set to a "one" the transmitter serial output is connected to the receiver serial input and disconnects the external serial input. This bit is cleared to a "zero" by INIT, on power-up, or the program.
01	Programmable Baud Rate Enable	Read Write	This bit is controlled by the program. When set to a "one", bits 03-05 are used to determine the baud rate. When cleared to a "zero" the baud rate will be 300 baud. This bit is cleared to a "zero" by INIT, power-up or the program.
00	Transmit Break	Read Write	This bit is controlled by the program. When set to a "one", the serial output is forced into the SPACE condition. This bit is cleared by INIT, power-up or the program.

Table 6-5 Transmitter Data Buffer Bit Descriptions

Bit	Name	Direction	Function
08-15	Not Used	Read Only	Reserved for future use.
00-07	Transmit Data Buffer	Read Write	These 8 bits represent the next data byte to be transmitted. These bits are cleared by power-up.

It should be noted that the Macro-ODT option is equipped with the autobaud feature that enables SLU1 to adjust itself to the terminal's baud rate between 300 and 9600 baud. The autobaud feature is operating only when Macro-ODT is running on the system.

6.1.2 Interrupts

Each SLU provides both a Receiver Interrupt and a Transmitter Interrupt to request service from the on-board microprocessor. The Receiver and Transmitter requests can be independently enabled by software. The Receiver Interrupt request is enabled when the RCV Interrupt Enable (bit 06) of the Receiver Control and Status register (RCSR) is set to a one.

SLU2 has a higher interrupt priority, level 5, than SLU1 which has a level 4 interrupt priority. Within each unit the receiver has higher priority than the transmitter. SLU1 uses vector address 60 for the receiver and 64 for the transmitter. SLU2 uses 120 for the receiver and 124 for the transmitter. These relationships are summarized in Table 5-3.

6.2 PROGRAMMING THE PARALLEL I/O INTERFACE

The parallel I/O interface, described by Figure 6-3, provides the means of transferring data between the microprocessor bus and the user interface connector J3. The interface is equipped with four addressable registers for data and control which are summarized in Table 6-6.

Table 6-6 Parallel I/O Register Addresses

Register	Address	Status
Port A	176200	Read/Write
Port B	176202	Read/Write
Port C	176204	Read/Write
Control Word	176206	Write Only

Port A and B registers are used only for data transfer to and from the user interface. Port C is used for both, data transfer and control. The control word register is used exclusively for control of the Parallel I/O interface. The interface owes its programmability to this register.

In addition to software programming, the parallel interface can also be programmed by hardware. This is covered in Chapter 2.

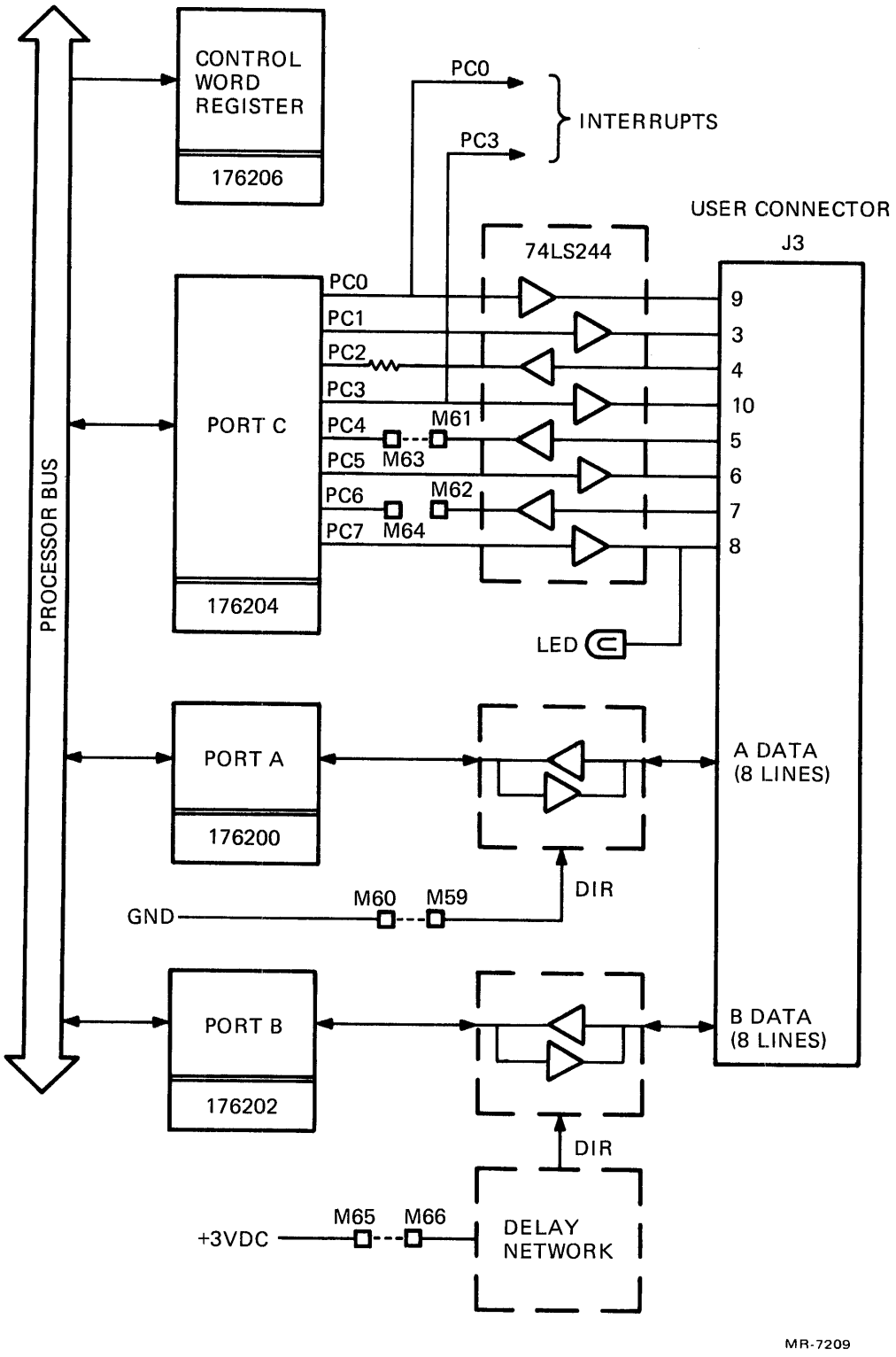


Figure 6-3 Parallel I/O Interface

6.2.1 How To Use This Section of The Manual

The parallel I/O interface is quite complex and a thorough understanding of all its capabilities will require a considerable effort. However, a very efficient use can be made of the parallel I/O by using a subset of these capabilities. This section has been organized to make the task of finding the needed information easier. The flowchart shown in Figure 6-4 provides an overview of the contents of this section and a few minutes reviewing it will aid the user.

After examining the flowchart the reader will be in a better position to decide whether to read in detail the rest of this chapter or if a specific I/O configuration is of immediate interest. If this is the case, the reader is encouraged to refer directly to that section.

6.2.2 Modes of Operation

The interface ports can operate in three basic modes that are selected by system software setting bits in the Control Word register. The modes are designated as Mode 0, 1 and 2 and define how the data is routed through ports A and B.

NOTE

If the bidirectional buffers are being hardwired, care must be taken to insure that the wired direction conforms to the programmed directions of ports A and B. This is necessary to avoid driver output to driver output connections, which could damage the integrated circuits.

6.2.2.1 Port C Register -- The bit assignments for the port C register are dependent upon the mode selected and the direction of ports A and B. This register provides the handshake controls to interface between the 8255A-5 and the output connector. The handshake control bits are set/reset by using the Control Word register which is described in the Control Word section. The port C condition for the various modes will be described in the sections dealing with those modes.

6.2.2.2 Mode 0 Basic Input/Output -- This mode provides simple input and output of either port A or port B or both as described in Table 6-7. The data is simply read from the port if programmed as an input or written to the port if programmed as an output with no handshaking requirements. The port A and port B bidirectional buffers may be hardwired as described in Chapter 2. They may also be program controlled by port C bits 4 and 6 if dynamic change of the port direction is desired. In this mode the outputs are latched but the inputs are not.

6.2.2.3 Port A and B Registers -- The bit assignments for the port A and B registers are shown in Figure 6-5 and described in Table 6-8. These registers are used as data buffers for all modes of operation.

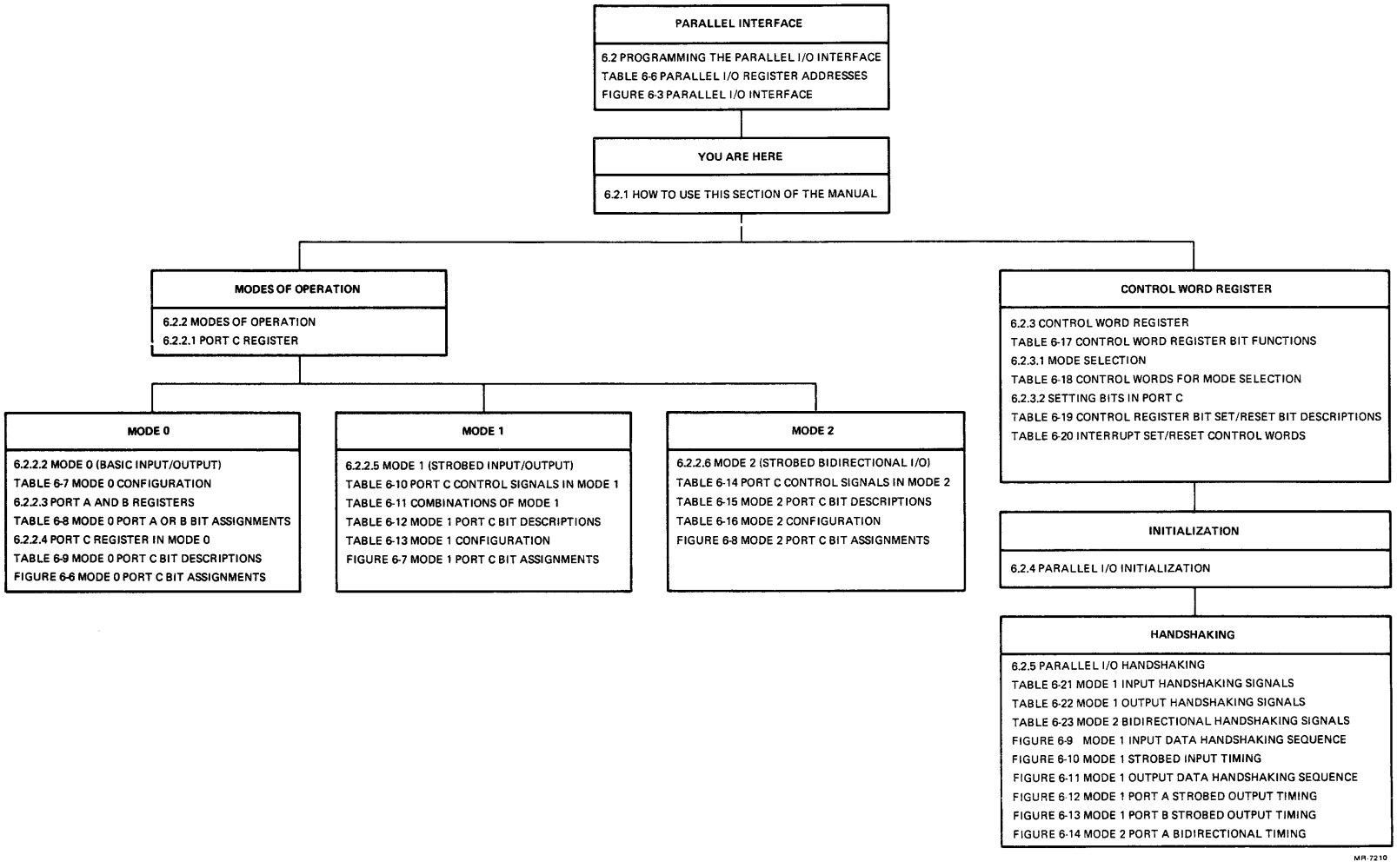


Figure 6-4 Parallel I/O Flowchart Guide

Table 6-7 Mode 0 Configuration

PPI Element	To Act as Input	To Act as Ouput	Direction Control via Port C
Port A	M66 to M65	M66 to M60	M66 to M64 or M63
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	Never an Input	Always an Output	
PC6	M64 to M62	Never an Output	
PC5	Never an Input	Always an Output	
PC4	M63 to M61	Never an External Output	
PC3	Never an Input	Interrupt A (Vector 134) Always an Output	
PC2	Always an Input	Never an Output	
PC1	Never an Input	Always an Output	
PC0	Never an Input	Interrupt B (Vector 130) Always an Output	

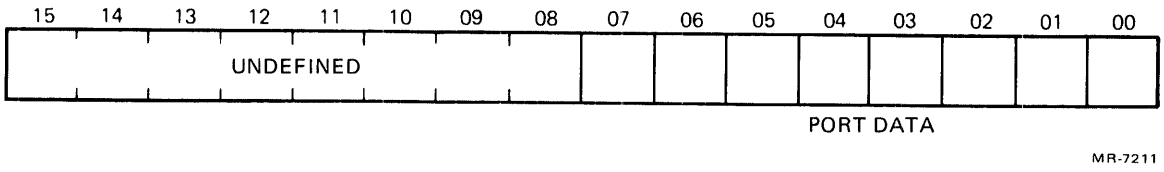


Figure 6-5 Mode 0, Port A or B, Bit Assignments

Table 6-8 Mode 0 Port A or B Bit Descriptions

Bit	Name	Direction	Function
8-15	Undefined	-	Not valid if a read is performed on the entire word.
0-7	Port Data	Read/Write	Data to output or input data to be read, depending on the port direction.

6.2.2.4 **Port C Register in Mode 0** -- Ports A and B use no handshaking signals and some of port C lines can be used as Input/Output data lines. The bit assignments are shown in Figure 6-6 and described in Table 6-9. When PC0 and PC3 lines are not being used as interrupt requests, these bits should be cleared by the Control Word to prevent erroneous interrupts.

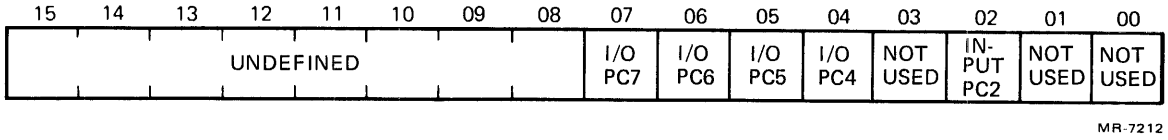


Figure 6-6 Mode 0 Port C Bit Assignments

Table 6-9 Mode 0 Port C Bit Descriptions

Bit	Name	Direction	Function
8-15	Undefined	-	Not valid if a read is performed on the entire word.
7	PC7	Read/Write*	Output bit. Drives the LED
6	PC6	Read/Write*	If Port C Upper is defined as input and M64 is connected to M62 then it is an input bit. If Port C Upper is defined as output and M64 is connected to M66 (M59) then it is output which controls the buffer direction for Port A (Port B). A "1" sets the buffer for input and a "0" for output.
5	PC5	Read/Write*	Same as PC7. No LED.
4	PC4	Read/Write*	If Port C Upper is defined as input and M63 is connected to M61 then it is an input bit. If Port C Upper is defined as output and M63 is connected to M59 (M66) then it is output which controls the buffer direction for Port B (Port A). A "1" sets the buffer for input and a "0" for output.
3	PC3	Not used	Not valid.
2	PC2	Read Only	Input bit.
0-1	PC0-PC1	Not used	Not valid.

*Bit is written by using the Control Word bit set/reset function explained in the Control Word section.

6.2.2.5 **Mode 1 (Strobed Input/Output)** -- In this mode, the lines on port C generate or accept signals from the user interface which control the transfer of data through ports A and B. Port C bits 0--3 (lower nibble) are used in conjunction with port B and bits 4--7 (upper nibble) are used with port A. These signals are known as "handshaking" signals. Before describing the details of the handshaking protocol, the basic functions of these control signals are defined in Table 6-10.

Table 6-11 describes the four input/output combinations of ports A and B usable in mode 1. The port C bit assignments as used in mode 1 are described by Figure 6-7 and tabulated in Table 6-12. Finally, Table 6-13 links operation of mode 1 to the jumper configurations discussed in Chapter 2.

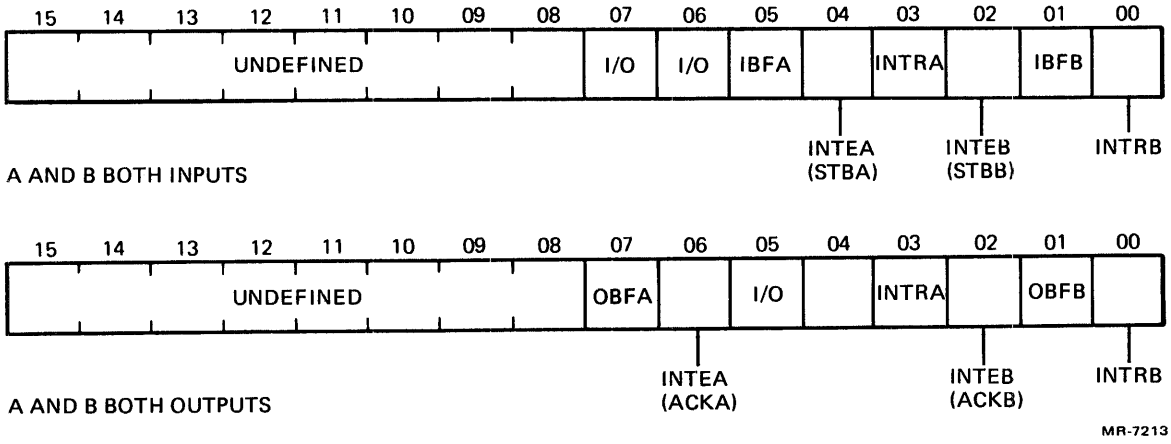


Figure 6-7 Mode 1 Port C Bit Assignments

6.2.2.6 **Mode 2 (Strobed Bidirectional I/O)** -- This mode implements a means of communication with a user device over a single 8-bit bus for both transmitting and receiving data. Handshaking and interrupt signals are used in a manner similar to mode 1.

This mode is used with port A only and five control lines on port C. Both inputs and outputs are latched. When port A is operating in this mode, the port B bidirectional buffers cannot be operated under program control since PC4 and PC6 are being used. Port B can still operate in either mode 0 or mode 1, but the buffers must be hardwired. PC0--PC2 are defined by port B usage for mode 1 and are available as I/O lines when port B is in mode 0.

Table 6-10 Port C Control Signals in Mode 1

Signal	Abbrev/Port C Bit	Function
Strobe Input	$-STB_A$/PC4 $-STB_B$/PC2	A low on this input loads user data into the input latch.
Input Buffer Full	IBF_A/PC5 IBF_B/PC1	A high on this output acknowledges that the data has been loaded into the input latch. Set by STB and reset by the program reading the input latch.
Interrupt Request (Input Mode)	$INTR_A$/PC3 $INTR_B$/PC0	A high on this output can interrupt the CPU when an input device strobes its data into the port.
Interrupt Enable (Input Mode)	$INTE_A$/PC4 $INTE_B$/PC2	Enables setting of $INTR_A$ and $INTR_B$. Program controlled by PC4 or PC2.
Output Buffer Full	$-OBF_A$/PC7 $-OBF_B$/PC1	This output goes low to tell the user interface that CPU has written data to the port. Reset by ACK input being low.
Acknowledge Input	$-ACK_A$/PC6 $-ACK_B$/PC2	A low on this input tells the processor that the user's device accepted the data from A or B.
Interrupt Request (Output Mode)	$INTR_A$/PC3 $INTR_B$/PC0	A high on this output can interrupt the CPU when an output device has accepted data transmitted by the CPU. Set by ACK. Reset when new data is written to the port.
Interrupt Enable (Output Mode)	$INTE_A$/PC6 $INTE_B$/PC2	Enables setting of INTR. Program controlled by PC6 or PC2.

Table 6-11 Combinations of Mode 1

Port C Bit Functions	Port A Input with Port B Output	Port A Output with Port B Input	Ports A & B Output	Ports A & B Input
STB_A	PC4	N/A	N/A	PC4
STB_B	N/A	PC2	N/A	PC2
IBF_A	PC5	N/A	N/A	PC5
IBF_B	N/A	PC1	N/A	PC1
$INTR_A$	PC3	PC3	PC3	PC3
$INTR_B$	PC0	PC0	PC0	PC0
OBF_A	N/A	PC7	PC7	N/A
OBF_B	PC1	N/A	PC1	N/A
ACK_A	N/A	PC6	PC6	N/A
ACK_B	PC2	N/A	PC2	N/A
Other Port C outputs	PC7 (Controls LED)	N/A	PC5	N/A
Other Port C inputs	N/A	PC4	N/A	PC6,7
Control Word				
D0 (Direction of PC0-3)	X	X	X	X
D1 (Direction of Port B)	0	1	0	1
D2 (Mode of Port B)	1	1	1	1
D3 (Direction of PC4-7)	0	1	1	0
D4 (Direction of Port A)	1	0	0	1
D5 Port A Mode	1	1	1	1
D6 Port A Mode	0	0	0	0
D7 Mode set enable	1	1	1	1

Table 6-12 Mode 1 Port C Bit Descriptions

Bit	Name	Direction	Function
8-15	Undefined	-	Not valid if a read is performed on the entire word.
7	PC7	Read/Write*	If Port A Mode 1 Input: If Port C bits 4-7, defined as output this bit is an output bit and controls the LED. A "0" turns the LED on and a "1" turns it off. Unused if PORT C bits 4-7 defined as input
	OBFA	Read Only	If Port A Mode 1 Output: OBFA goes low to indicate that data has been written into the output buffer by the processor. This bit is set when the ACKA (PC6, M64 to M62) input goes low indicating that the external device has accepted the output data. OBFA is present on PC7 to the external device.
6	PC6	Read/Write*	If Port A Mode 1 Input: If Port C bits 4-7 defined as input and M64 is connected to M62 then it is an input bit. If Port C bits 4-7 defined as output and M59 is connected to M64 it is an output which controls the buffer direction for Port B. A "1" sets the buffer for input and a "0" sets the buffer for output.
	INTEA	Read/Write*	If Port A Mode 1 Output: INTEA when set enables INTRA to interrupt the SBC-11/21 when output data has been accepted by the external device.
	ACKA		When M64 is connected to M62, an external signal acknowledging the receipt of data acts as INTEA.

Table 6-12 Mode 1 Port C Bit Descriptions (Cont)

Bit	Name	Direction	Function
5	IBFA	Read Only	If Port A Mode 1 Input: IBFA indicates that the input data has been latched for Port A. It is set by the STBA input (PC4, M63 to M61) being low and is reset by the processor reading the port data. This signal is present on PC5 to the external ldevice.
	PC5	Read/Write*	If Port A Mode 1 Output: If Port C Upper defined as output then it is an output bit. If Port C Upper defined as input it is unused.
4	INTEA	Read/Write*	If Port A Mode 1 Input: INTEA if set will allow INTRA to interrupt the SBC-11/21 whenever the input buffer is full.
	PC4	Read/Write*	If Port A Mode 1 Output: If Port C bits 4-7 is defined as output and M59 is connected to M63 then this bit is output which controls the direction of the Port B buffer. A "1" sets the buffer for input and a "0" sets it for output. If Port C bits 4-7 is defined as input and M63 is connected to M61 then it is an input bit and is interpreted as STBA (input strobe).
3	INTRA	Read Only	If Port A Mode 1 Input: A "1" indicates that Port A has valid input data. It is set by STBA (PC4, M63 to M61) being pulsed low and is reset by the processor reading the Port data. INTRA is enabled by INTEA being a "1" and disabled by INTEA being a "0".

Table 6-12 Mode 1 Port C Bit Descriptions (Cont)

Bit	Name	Direction	Function
			If Port A Mode 1 Output: A "1" indicates that Port A is ready to accept new output data. It is set by ACKA (PC6, M64 to M62) being pulsed low and reset by the processor writing new output data to the port. Enabled and disabled as above. When enabled INTRA interrupts the processor and has a vector of 134. This signal is also an output to the external device on line PC3.
2	INTEB	Read/Write*	INTEB when set will allow INTRB to interrupt the SBC-11/21 to request service.
1	IBFB	Read Only	If Port B Mode 1 Input: IBFB indicates input data has been latched for Port B when a "1". It is set by the STBB (PC2) being low and is reset by the processor reading the port data. This signal is present on PC1 to the external device.
	OBFB	Read Only	If Port B Mode 1 Output: OBFB goes low to indicate that the processor has written data to the port. This bit is set by ACKB (PC2) going low indicatinig the external device has accepted the output data. This signal is present on PC1 to the external device.

Table 6-12 Mode 1 Port C Bit Descriptions (Cont)

Bit	Name	Direction	Function
0	INTRB	Read Only	If Port B Mode 1 Input: A "1" indicates Port B has valid input data. It is set by STBB (PC2 being pulsed low and is reset by the processor reading the port data. INTRB is enabled when INTEB is "1" and disabled when it is "0". If Port B Mode 1 Output: A "1" indicates the port is ready to accept new output data. It is set by ACKB (PC2) being pulsed low and reset by the processor writing new output data to the port. Enabled and disabled as above. This signal is also an output to the external device on PC0.

*Bit is written by using the Control Word Bit set/reset function described in the Control Word section.

NOTE

If OBF is asserted low and a read or write access is made to the port by the processor before an ACK strobe is sent by the external device, the OBF line for the accessed port will negate during the assertion of the read or write to the port and become reasserted when the read or write operation is complete.

Table 6-13 Mode 1 Configuration

PPI Element	Input Conditions	Output Conditions	Program Control via Port C
Port A	M66 to M65	M66 to M60	N/A
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	Never an Input	Output Buffer A Full	
PC6	M62 to M64 (Acknowledge A)*	Never an External Output	
PC5	Never an Input	Input Buffer A Full	
PC4	M61 to M63 (Strobe A)	Never an External Output	
PC3	Never an Input	Interrupt A (Vector 134)	
PC2	Strobe B in Input Mode. Acknowledge B in Output Mode	Never an Output	
PC1	Never an Input	Buffer B Full on Input or Output	
PC0	Never an Input	Interrupt B (Vector 130)	

*User's hardware acknowledges receipt of data output by port A.

Control signals are defined in Table 6-14. The port C bit assignments as used in mode 2 are described by Figure 6-8 and tabulated in Table 6-15. Finally, Table 6-16 links operation of mode 2 to the jumper configurations discussed in Chapter 2.

6.2.3 Control Word Register

The Control Word register controls the operation of the parallel interface. If bit 7 is set (active high), the contents of the register determines the mode of operation and the Input/Output direction of the ports. If bit 7 is cleared (active low), the contents of the register will set/reset the Port C register bits. The functions of the register bits are described in Table 6-17, and determined by the state (active high or active low) of the bit.

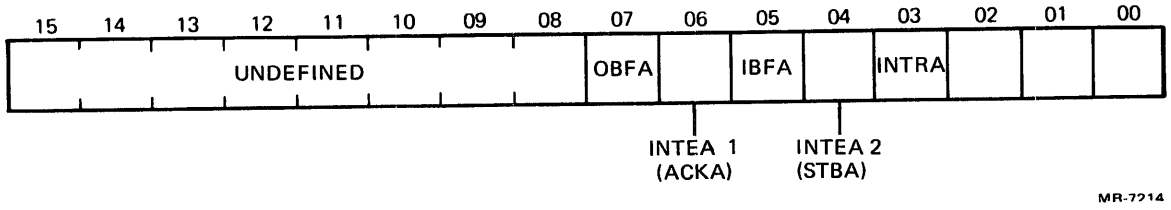


Figure 6-8 Mode 2 Port C Bit Assignments

Table 6-14 Port C Control Signals in Mode 2

Signal	Abbrev/Port C Bit	Function
Interrupt Request	$INTR_A$/PC3	A high on this output can interrupt the CPU for both input and output operations.
Output Buffer Full	OBF_A/PC7	This output goes low to indicate that the CPU has written data to port A.
Acknowledge	ACK_A/PC6	A low on this input enables the output tristate buffers of port A to send out the data. Otherwise that buffer is in the high impedance state.
Interrupt Enable	INTEA1/PC6	Enables INTR when OBF is true. Controlled by bit set/reset of PC6.
Strobe Input	STB_A/PC4	A low on this input loads data into the input latch.
Input Buffer Full	IBF_A/PC5	A high on this output indicates that data has been loaded into the input latch.
Interrupt Enable	INTEA2/PC4	Enables INTR when IBF is true. Controlled by bit set/reset of PC4.

Table 6-15 Mode 2 Port C Bit Descriptions

Bit	Name	Direction	Function
8-15	Undefined	-	Not valid. If a read is done on the entire word.
7	OBFA	Read Only	Will go low to indicate that the processor has written output data to the port. It is set when ACKA (PC6 M64 to M62) goes low indicating the external device has accepted the data. This signal is output on PC7 to the external device.
6	INTEA1	Read/Write*	When this bit is set it allows an interrupt INTRA when the output buffer is ready to accept new data.
5	IBFA	Read Only	IBFA indicates that input data has been latched when a "1". This bit is reset when the processor reads the input data. This signal is output on PC5 to the external device.
4	INTEA2	Read/Write*	When this bit is set it allows an interrupt INTRA when the input buffer is full.
3	INTRA	Read Only	A high on this bit indicates that the port is requesting service of the processor. This signal is output on PC3 to the external device.
2-0	PC0-PC2	-	These bits are defined by Port B mode selection.

NOTE

When using Port A in mode 2 operation the software must clear the input buffer of Port A if the input buffer full flag (IBFA) is set before it performs the read during an intended write to ensure that the handshake lines and port flags are not set out of sequence.

Table 6-16 Mode 2 Configuration

PPI Element	Input Conditions	Output Conditions
Port A	Bidirectional bus	M66 to M64 to M62
Port B	Hardwired only	Hardwired only
PC7	Never an Input	Output Buffer A Full
PC6	Acknowledge A	Never an Output
PC5	Never an Input	Input Buffer A Full
PC4	Strobe A (M61 to M63)	Never an Output
PC3	Never an Input	Interrupt A (Vector 134)
PC2	Always an Input	Never an Output
PC1	Never an Input	Always an Output
PC0	Never an Input	Always an Output

Table 6-17 Control Register Mode Selection Bit Functions

Bit	Bit Set	Bit Reset
8-15	Unused	Unused
7	Always set	Always set
6	Port A Mode 2	Port A mode 0 or 1
5	Port A Mode 1	Port A Mode 0
4	Port A input	Port A output
3	Port C bits 4 and 6 are inputs	Port C 4--7 are outputs
2	Port B Mode 1	Port B Mode 0
1	Port B input	Port B output
0	Port C bit 2 input	Port C bits 0, 1 and 3 outputs

6.2.3.1 **Mode Selection** -- The user determines the mode of operation for the ports and defines them as inputs, outputs or bidirectional. The user then must insure that the bidirectional buffers are configured (See Chapter 2) to match the software requirements. Table 6-18 lists all the control words available for the Control Word register. The user selects the control word that matches the requirements and loads it into the register. The register is defined as "write only" and any attempts to read the register results in erroneous data.

6.2.3.2 **Setting Bits in Port C** -- The control word register is also used to set or reset the Port C register bits. The control word bit functions are described in Table 6-19. To set a bit the

Table 6-18 Control Words for Mode Selection

	Port B Mode 0 IN	Port B Mode 0 OUT	Port B Mode 1 IN	Port B Mode 1 OUT	Port C PC4,PC6	Port C PC5,PC7
Port A Mode 0 IN	233	231	237	235	INPUT	
	233	221	227	225		OUTPUT
Port A Mode 0 OUT	213	211	217	215	INPUT	
	203	201	207	205		OUTPUT
Port A Mode 1 IN	273	271	277	275	INPUT	
	263	261	267	265		OUTPUT
Port A Mode 1 OUT	253	251	257	255	INPUT	
	243	241	247	245		OUTPUT
Port A Mode 2	3X3	3X1	3X7	3X5	*	

* Port C Unavailable, Used for handshaking

X = Don't care condition

Table 6-19 Control Register Bit Set/Reset Bit Functions

Bit	Function
8-15	Not used
7	Always reset
6,5,4	Not used
3,2,1	These bits select the Port C bit that is to be set or reset as follows.

Bit	03	02	01
PC0	0	0	0
PC1	0	0	1
PC2	0	1	0
PC3	0	1	1
PC4	1	0	0
PC5	1	0	1
PC6	1	1	0
PC7	1	1	1

0	This bit is set to set the selected bit or is cleared to reset the selected bit of Port C.

register is loaded with bit 7 cleared, bits 1--3 equal to the bit number being set and bit 0 set. To reset the same bit, bit 0 would be cleared. The bit set/reset can be used to enable or disable the Port A and B interrupts for the SBC-11/21. The control words used to enable or disable the interrupts are listed in Table 6-20.

Table 6-20 Interrupt Set/Reset Control Words

Mode	Direction	INTRA		INTRB	
		Enable	Disable	Enable	Disable
1	Input	011	010	005	004
1	Output	015	014	005	004
2	Input	011	010	None*	None*
2	Output	015	014	None*	None*

*Port B does not function in the bidirectional Mode 2.

6.2.4 Parallel I/O Initialization

During Power up or the execution of a RESET instruction, the Port C data lines are driven high and the LED (driven by bit 7 of port C) is turned off. If Ports A and B bidirectional buffers are hardwired, the directions are not altered and the data lines are driven high if the buffer is configured as an output. If Ports A and B bidirectional buffers are program controlled by Port C, the data lines will go to the input state.

6.2.5 Parallel I/O Handshaking

The Parallel I/O can operate in either mode 0, 1 or 2 to transfer data into or out of the SBC-11/21. The mode 0 data transfers do not require any handshaking control signals. The mode 0 input data is not latched and data should be available on the I/O connector at the same time as the read strobe enables the 8255A-5. The mode 0 output data is latched and data is valid at the I/O connector 362 nsec after the trailing edge of the WRITE strobe to the 8255A-5.

The handshaking signals which pass across the user interface are detailed as follows. Mode 1 operation requires the handshaking control signals and these are dependent upon defining the ports as inputs or outputs. Mode 1 input signals are listed in Table 6-21 and the handshaking function is shown in Figure 6-9. Mode 1 input timing is described in Figure 6-10. Mode 1 output signals are listed in Table 6-22 and the handshaking function is shown in Figure 6-11. Mode 1 output timing is described for Port A in Figure 6-12 and Port B in Figure 6-13. Mode 2 operation allows Port A to be bidirectional and the handshaking signals are listed in Table 6-23. Mode 2 timing is described in Figure 6-14. When Port A operates in mode 2, Port B can only operate in mode 0 or mode 1.

Table 6-21 Mode 1 Input Handshaking Signals

Signal Name	Function
STB (A or B) Port A - PC4 Port B - PC2	Strobe Input - This signal is asserted low by the external device and loads data into the SBC-11/21 input port latch. It must be asserted low for 525 ns minimum.
IBF (A or B) Port A - PC5 Port B - PC1	Input Buffer Full - This signal is asserted by the SBC-11/21 in response to an assertion of STB to notify the interface that data was loaded into the input latch.
INTR (A or B) Port A - PC3 Port B - PC0	Interrupt Request - This signal can be used to generate an interrupt to the microprocessor. The bitset/bitreset commands must be used to enable/disable the INTE bit for each port. Interrupts will be generated either when STB is negated with IBF asserted.

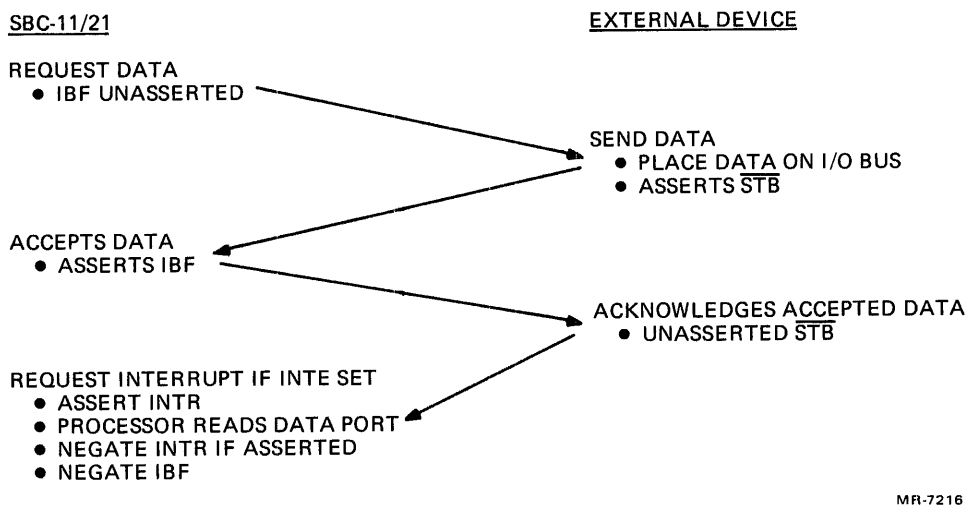


Figure 6-9 Mode 1 Input Data Handshaking Sequence

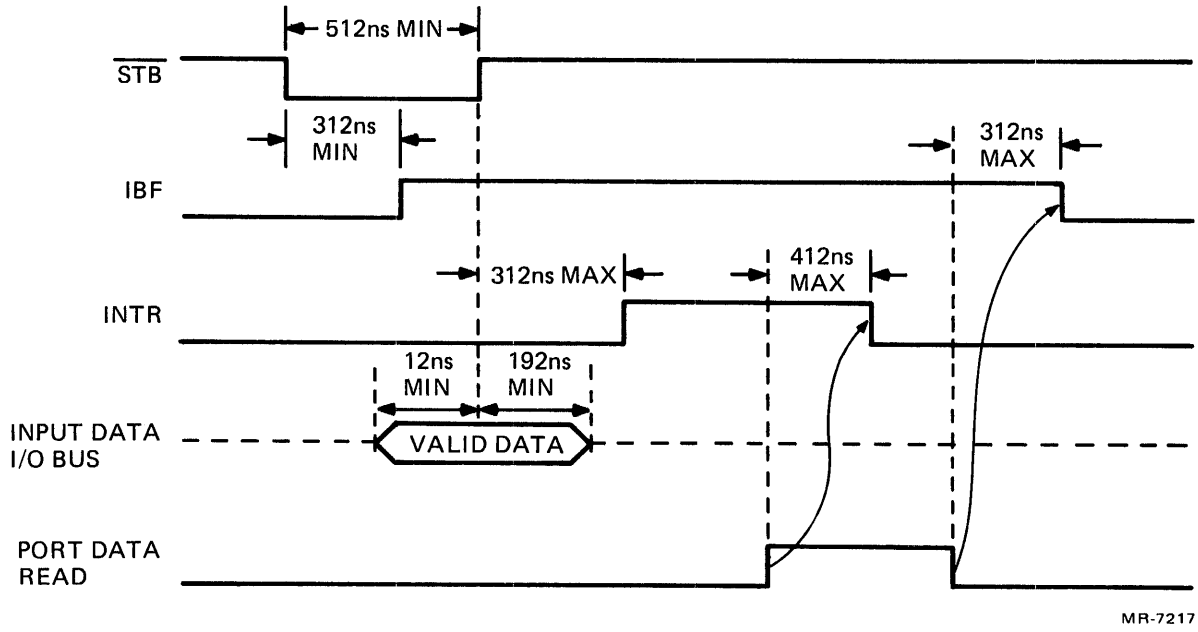
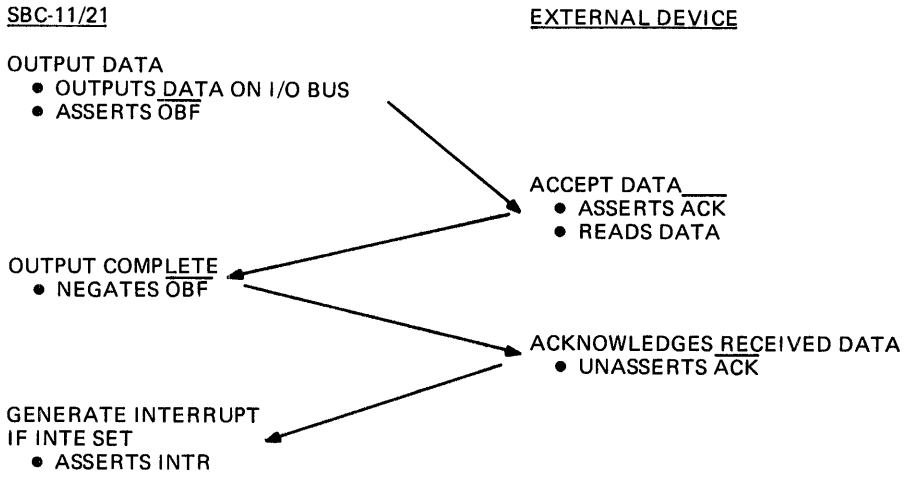


Figure 6-10 Mode 1 Strobed Input Timing

Table 6-22 Mode 1 Output Handshaking Signals

Signal Name	Function
$\overline{OBF}$ (A or B) Port A - PC7 Port B - PC1	Output Buffer Full - This output is asserted low to indicate that the microprocessor has written data into the specified port latches.
$\overline{ACK}$ (A or B) Port A - PC6 Port B - PC2	Acknowledge Input - This signal is asserted low by the external device to indicate it has accepted the latched output data from the specified port.
INTR (A or B) Port A - PC3 Port B - PC0	Interrupt Request - This signal can be used to generate an interrupt to the microprocessor when the external device has received the data and INTE is set and ACK is negated.



NOTE 1: IF $\overline{OBF}$ IS ASSERTED LOW AND A READ OR WRITE TO THE PORT BY THE SBC-11/21 PROCESSOR OCCURS BEFORE AN $\overline{ACK}$ STROBE IS SENT BY THE EXTERNAL DEVICE, THE $\overline{OBF}$ LINE FOR THE ACCESSED PORT WILL NEGATE DURING THE ASSERTION OF THE READ OR WRITE TO THE PORT AND THEN BECOME REASSERTED.

NOTE 2: $\overline{OBF}$ WILL ASSERT ON THE READ PORTION OF EVERY READ BEFORE INTENDED WRITE TO PORT B AND THE $\overline{OBF}_B$ WILL NEGATE AND REASSERT ON THE WRITE STROBE. IF $INTE_B$ IS SET AND $INTR_B$ IS ASSERTED, $INTR_B$ WILL NEGATE ON THE READ BEFORE THE INTENDED WRITE TO PORT B.

NOTE 3: $\overline{OBF}$ WILL ASSERT ON THE WRITE PORTION OF EVERY READ BEFORE INTENDED WRITE TO PORT A. IF $INTE_A$ IS SET AND $INTR_A$ IS ASSERTED, $INTR_A$ WILL NEGATE ON THE WRITE PORTION OF THE READ BEFORE INTENDED WRITE TO PORT A.

MR-7218

Figure 6-11 Mode 1 Output Data Handshaking Sequence

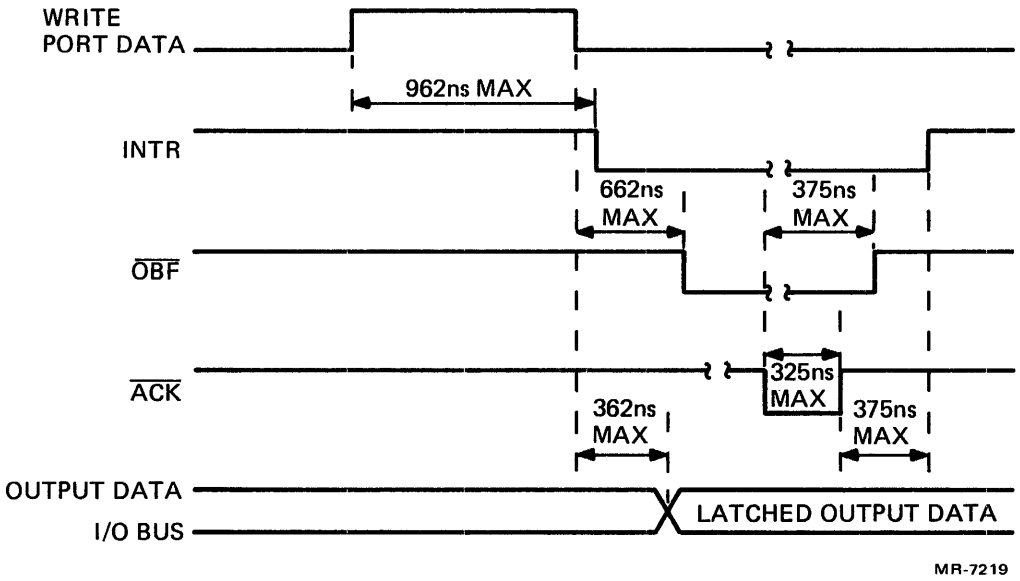


Figure 6-12 Mode 1 Strobed Output Timing

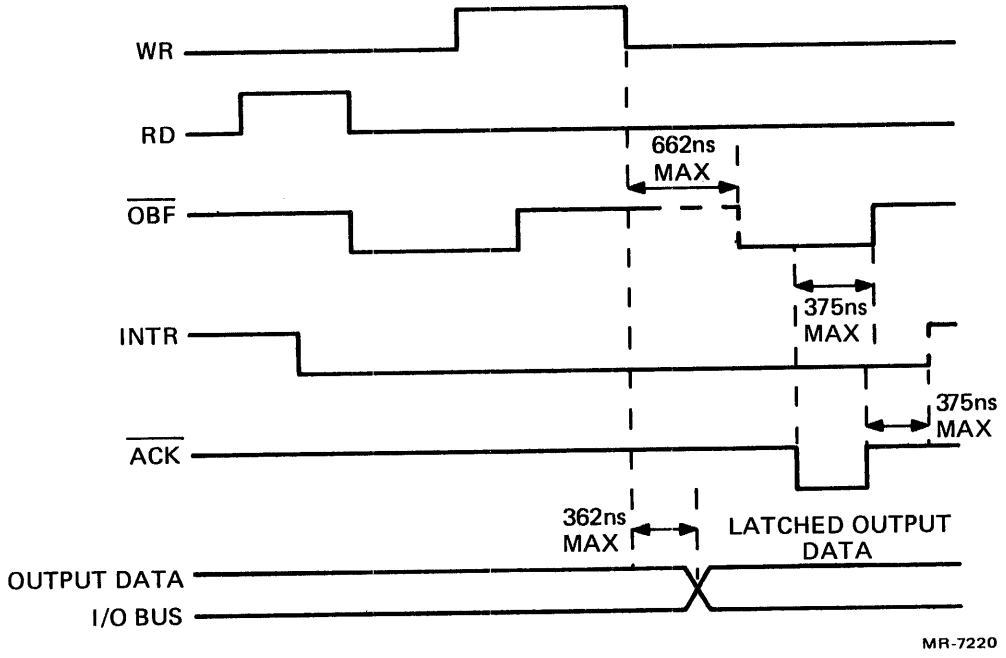


Figure 6-13 Mode 1 Port B Strobed Output Timing

Table 6-23 Mode 2 Bidirectional Handshaking Signals

Signal Name	Function
$\overline{\text{STB}}$ (PC4)	Strobe Input - This signal is asserted low by the external device and strobes data into Port A.
IBF (PC5)	Input Buffer Full - This signal is asserted when the microprocessor has accepted STB strobe.
INTR (PC3)	Interrupt Request - This signal can be used to generate an interrupt to the microprocessor when the external device is demanding service.
OBF (PC7)	Output Buffer Full - This output is asserted to indicate that the microprocessor has written data into the output port latches.
$\overline{\text{ACK}}$ (PC6)	Acknowledge Input - This signal is asserted low by the external device to indicate it has taken data from the output port latches. It controls the DIR pin of the port A buffer.

NOTE

When mode 2 is configured, PC6 (ACK) is jumpered to the port A direction control pin through a rising edge delay circuit. Hence, when PC6 is negated, the rising edge is delayed by 250 ns minimum, which means that the buffer will be driving data out the connector 250 ns minimum after the user interface negates ACK.

Since every write is preceded by a read, the contents of the input buffer should be saved if IBF_A is asserted prior to writing port A mode 2 data.

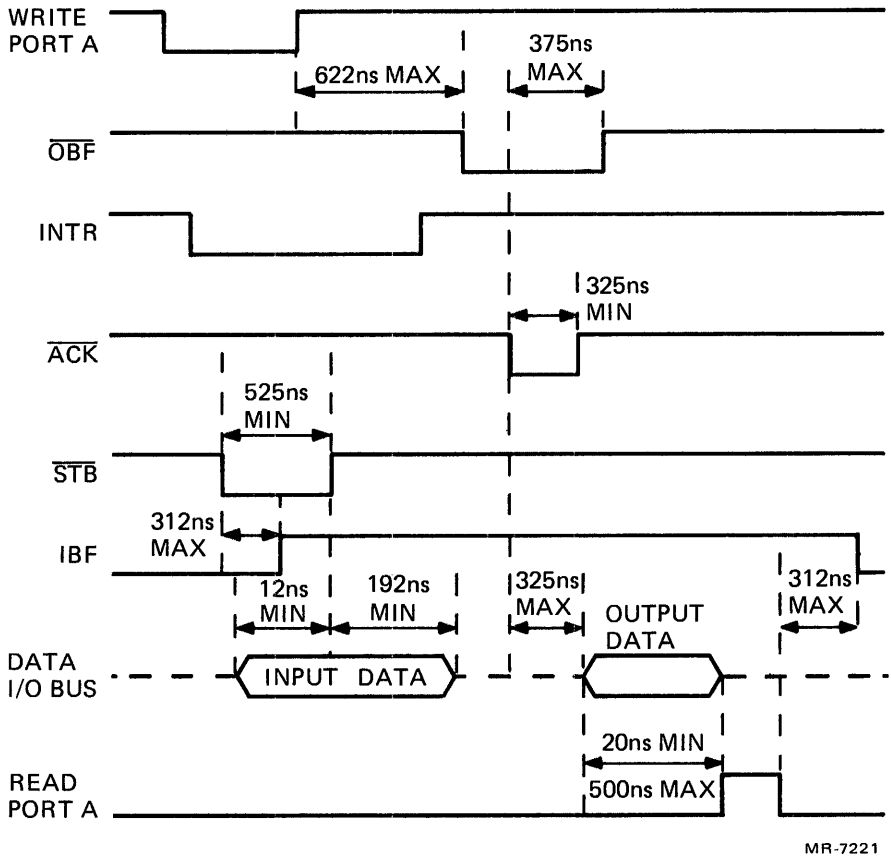


Figure 6-14 Mode 2 Port A Bidirectional Timing

CHAPTER 7
ADDRESSING MODES AND INSTRUCTION SET

7.0 GENERAL

The discussion of addressing modes is divided into six major topics:

- o **Single Operand Addressing** -- One part of the instruction word specifies the registers; the remaining part provides information for locating the operand.
- o **Double Operand Addressing** -- Part of the instruction word specifies the registers; the remaining parts provide information for locating two operands.
- o **Direct Addressing** -- The operand is the content of the selected register.
- o **Deferred (Indirect) Addressing** -- The contents of the selected register is the address of the operand.
- o **Use of the PC as a General Purpose Register** -- The PC is unique from other general-purpose registers in one important respect. Whenever the processor retrieves an instruction, it automatically advances the PC by 2. By combining this automatic advancement of the PC with four of the basic addressing modes, we produce the four special PC modes -- immediate, absolute, relative, and relative deferred.
- o **Use of the Stack Pointer as a General Purpose Register** -- Can be used for stack operations.

These addressing modes will now be discussed in detail, followed by descriptions of individual instructions.

> NOTE
> **Instruction mnemonics and address mode symbols are sufficient for writing assembly language programs. The programmer need not be concerned about conversion to binary digits; this is accomplished automatically by the assembler program.**

7.1 ADDRESSING MODES

Data stored in memory must be accessed and manipulated. Data handling is specified by an SBC-11/21 instruction (MOV, ADD, etc.), which usually indicates:

- o The function (operation code).

- A general-purpose register 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.). SBC-11/21 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 backward 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 microprocessor 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 SBC-11/21 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 of 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 accumulator. Whenever an instruction is fetched from memory, the program counter is automatically incremented by two to point to the next instruction word.

7.1.1 Single Operand Addressing

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

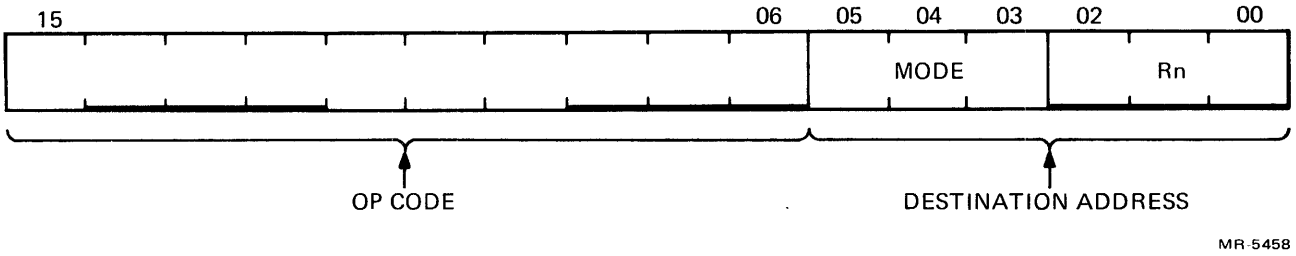


Figure 7-1 Single Operand Addressing

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

Bits five through zero form a six-bit field called the destination address field. This consists of two subfields.

- Bits zero through two specify which of the eight general-purpose registers is to be referenced by this instruction word.

- Bits three through five specify how the selected register will be used (address mode). Bit three is set to indicate deferred (indirect) addressing.

7.1.2 Double Operand Addressing

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 the double operand instruction is:

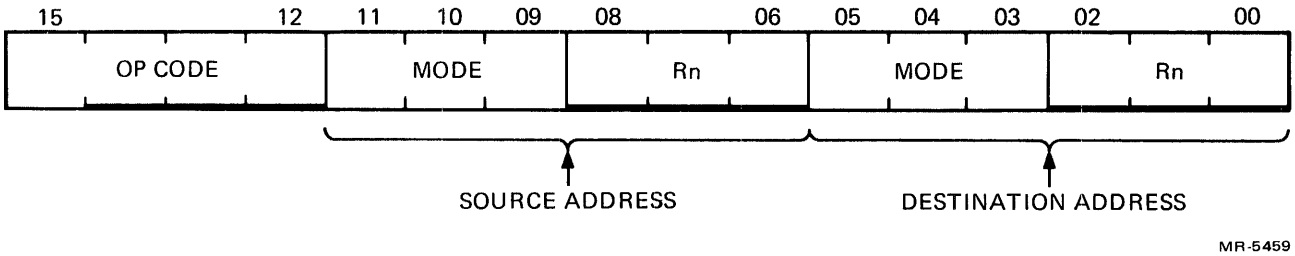


Figure 7-2 Double Operand Addressing

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.

Examples in this paragraph and chapter use the following sample SBC-11/21 instructions. A complete listing of the SBC-11/21 instructions is located in Paragraph 7.2.

Mnemonic	Description	Octal Code
CLR	Clear (zero the specified destination)	0050DD
CLRB	Clear byte (zero the byte in the specified destination)	1050DD
INC	Increment (add one to contents of destination)	0052DD
INCB	Increment byte (add one to the contents of destination byte)	1052DD

COM	Complement (replace the contents of the destination by its logical complement; each zero bit is set and each one bit is cleared)	0051DD
COMB	Complement byte (replace the contents of the destination byte by its 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

DD = destination field (six bits)

SS = source field (six bits)

( ) = contents of

7.1.3 Direct Addressing

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

DIRECT MODES

Mode	Name	Assembler Syntax	Function
0	Register	Rn	Register contains operand

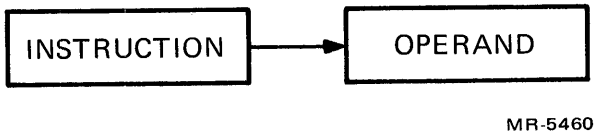


Figure 7-3 Mode 0 Register

2	Autoincrement	(Rn)+	Register is used as a pointer to sequential data, then incremented

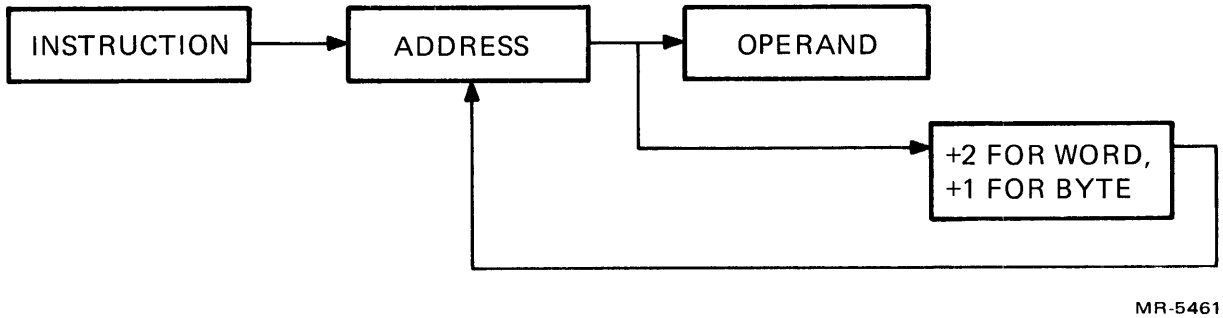


Figure 7-4 Mode 2 Autoincrement

4	Autodecrement	-(Rn)	Register is decremented and then used as a pointer.

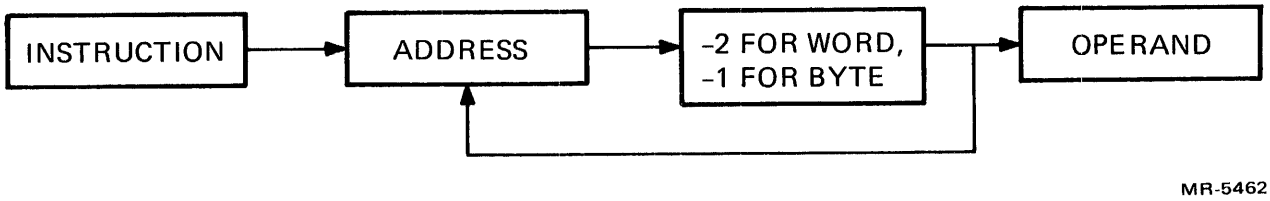


Figure 7-5 Mode 4 Autodecrement

6	Index	X(Rn)	Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

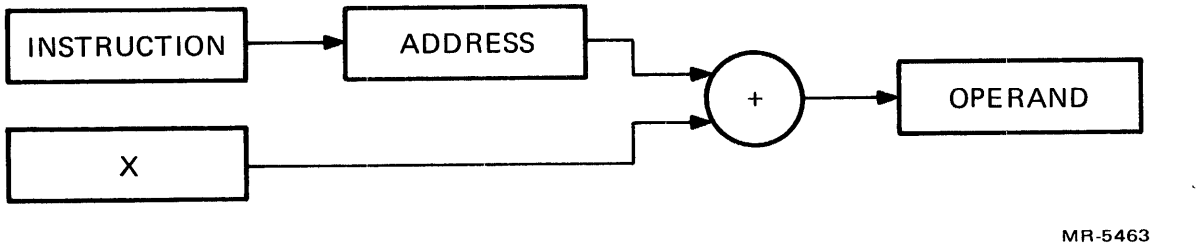


Figure 7-6 Mode 6 Index

7.1.3.1 Register Mode -- 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. The assembler interprets and assembles instructions of the form OPR Rn as register mode operations. Rn represents a general register name or number and OPR is used to represent a general instruction mnemonic. Assembler syntax requires that a general register be defined as follows.

```
R0 = %0 (% sign indicates register definition)
R1 = %1
R2 = %2, etc.
```

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.

OPR Rn

Register Mode Examples
(all numbers in octal)

	Symbolic	Octal Code	Instruction Name
1.	INC R3	005203	Increment

Operation: Add one to the contents of general purpose register R3.

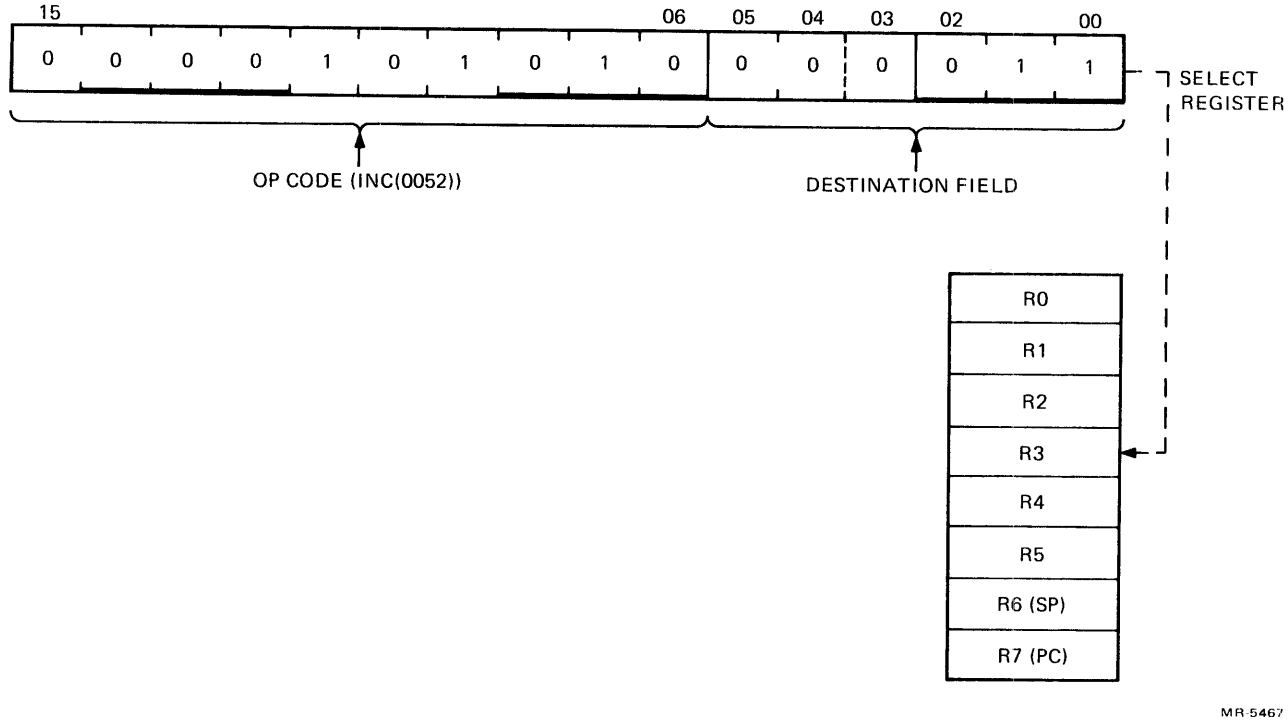


Figure 7-7 INC R3 Increment

2.	ADD R2, R4	060204	Add

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

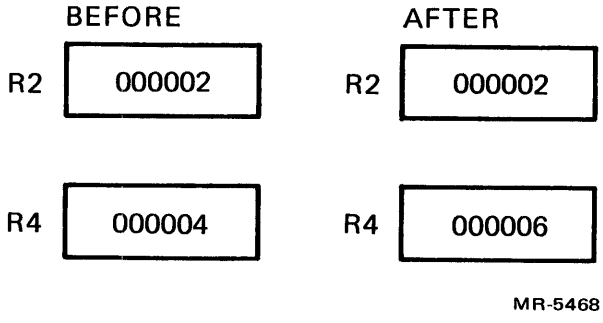


Figure 7-8 ADD R2, R4 Add

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.)

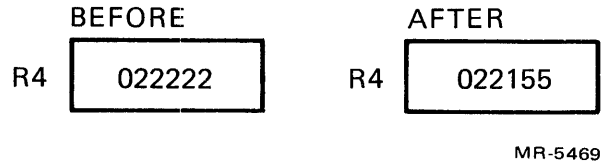


Figure 7-9 COMB R4 Complement Byte

7.1.3.2 Autoincrement Mode -- 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 purpose 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.

OPR (Rn)+

Autoincrement Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	CLR (R5)+	005025	Clear

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

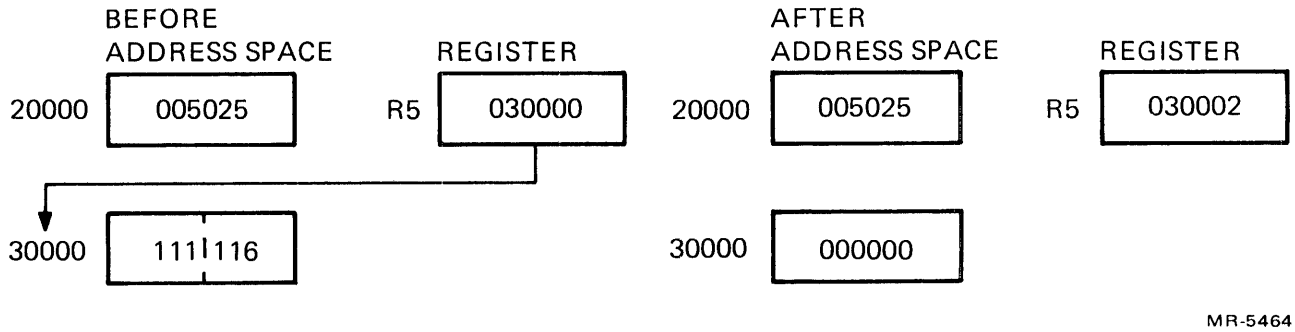


Figure 7-10 CLR (R5)+ Clear

2. CLRB (R5)+ 105025 Clear Byte

Operation: Use contents of R5 as the address of the operand. Clear selected byte operand and then increment the contents of R5 by one.

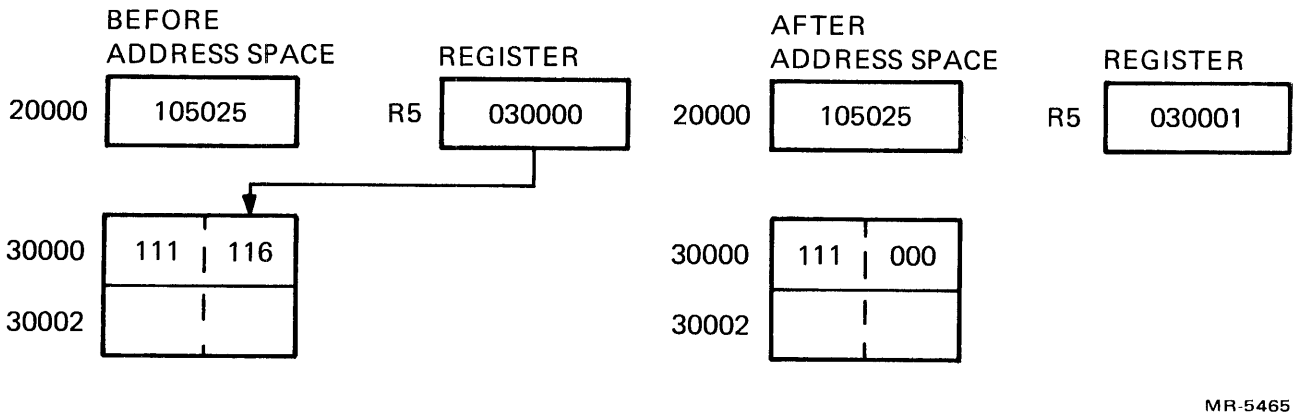


Figure 7-11 CLRB (R5)+ Clear Byte

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.

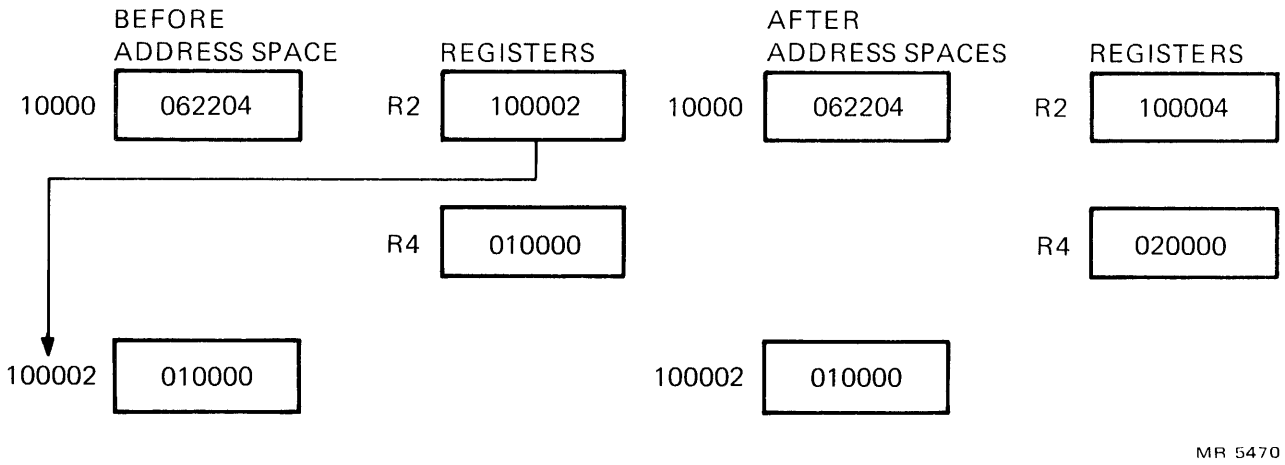


Figure 7-12 ADD (R2) + R4 Add

7.1.3.3 Autodecrement Mode (Mode 4) -- This mode is useful for processing data in a list in reverse direction. The contents of the selected general purpose register are decremented (by two for word instructions, by one for byte instructions) and then used as the address of the operand. The choice of postincrement, predecrement features for the SBC-11/21 were not arbitrary decisions, but were intended to facilitate hardware/software stack operations.

OPR-(Rn)

Autodecrement Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	INC -(R0)	005240	Increment

Operation: The contents of R0 are decremented by two and used as the address of the operand. The operand is incremented by one.

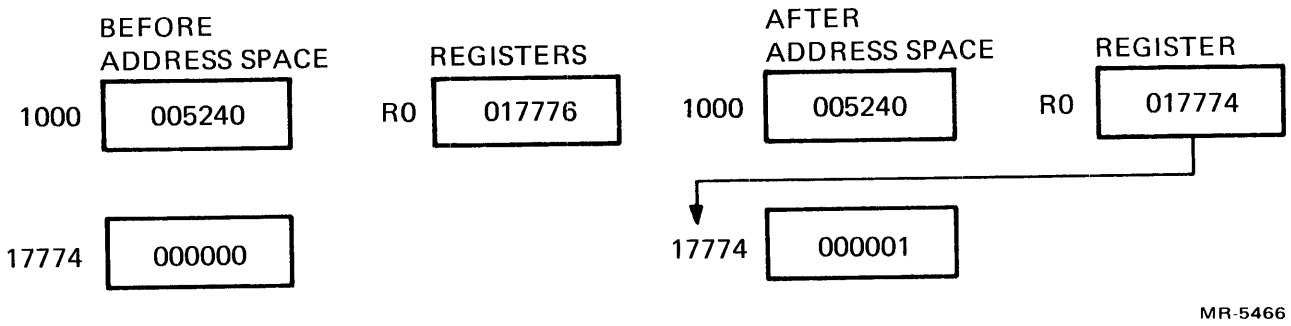


Figure 7-13 INC -(R0) Increment

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.

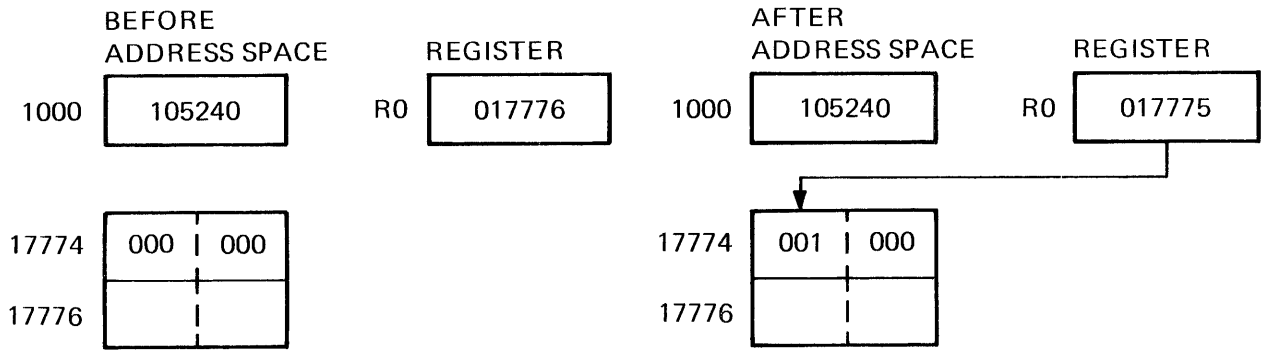


Figure 7-14 INCB -(R0) Increment Byte

3. ADD -(R3),R0 064300 Add

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

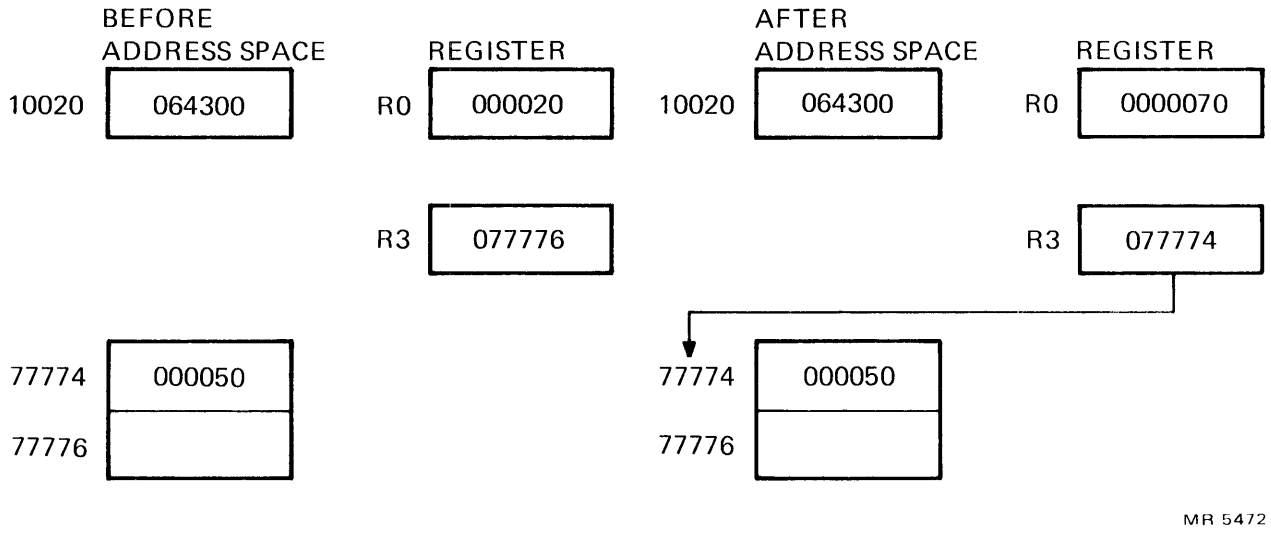


Figure 7-15 ADD -(R3), R0 Add

7.1.3.4 Index Mode (Mode 6) -- The contents of the selected general purpose 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 purpose register.

OPR X(Rn)

Index Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	CLR 200(R4)	005064 000200	Clear

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

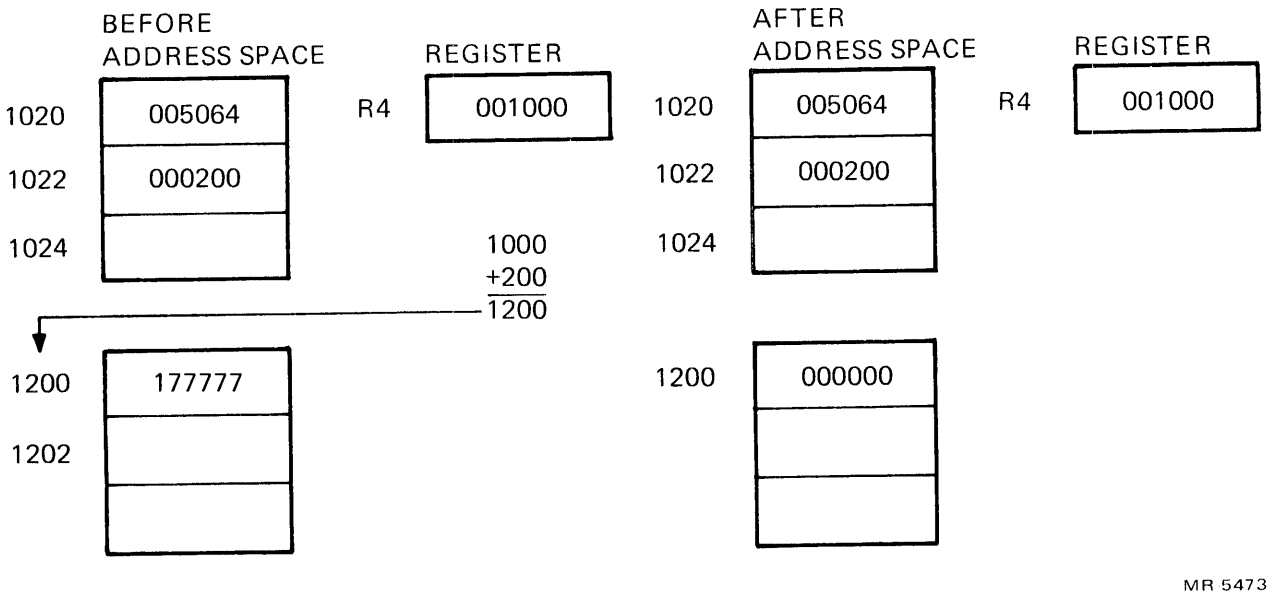


Figure 7-16 CLR 200 (R4) Clear

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).

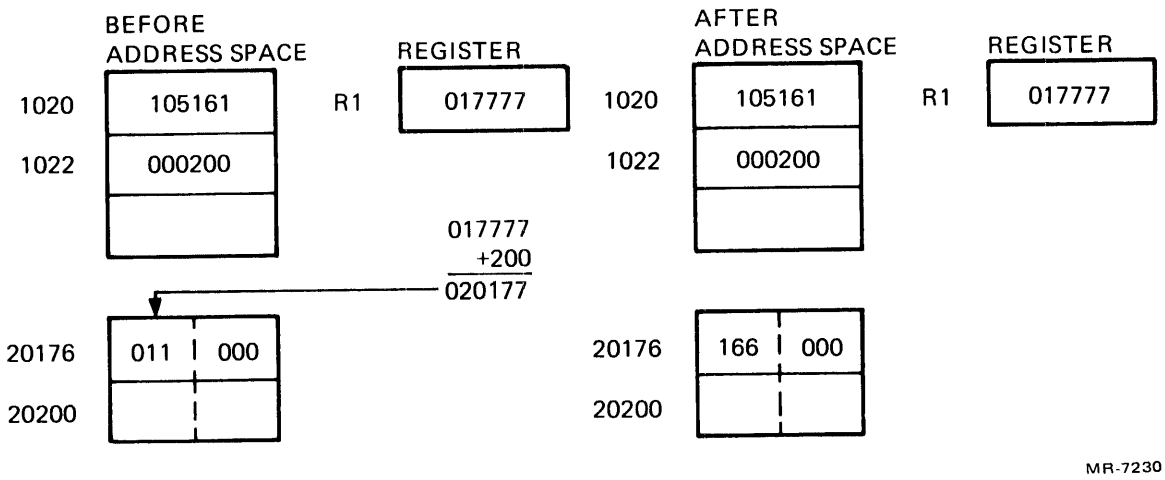


Figure 7-17 COMB 200 (R1) Complement Byte

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

```
066265   Add
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).

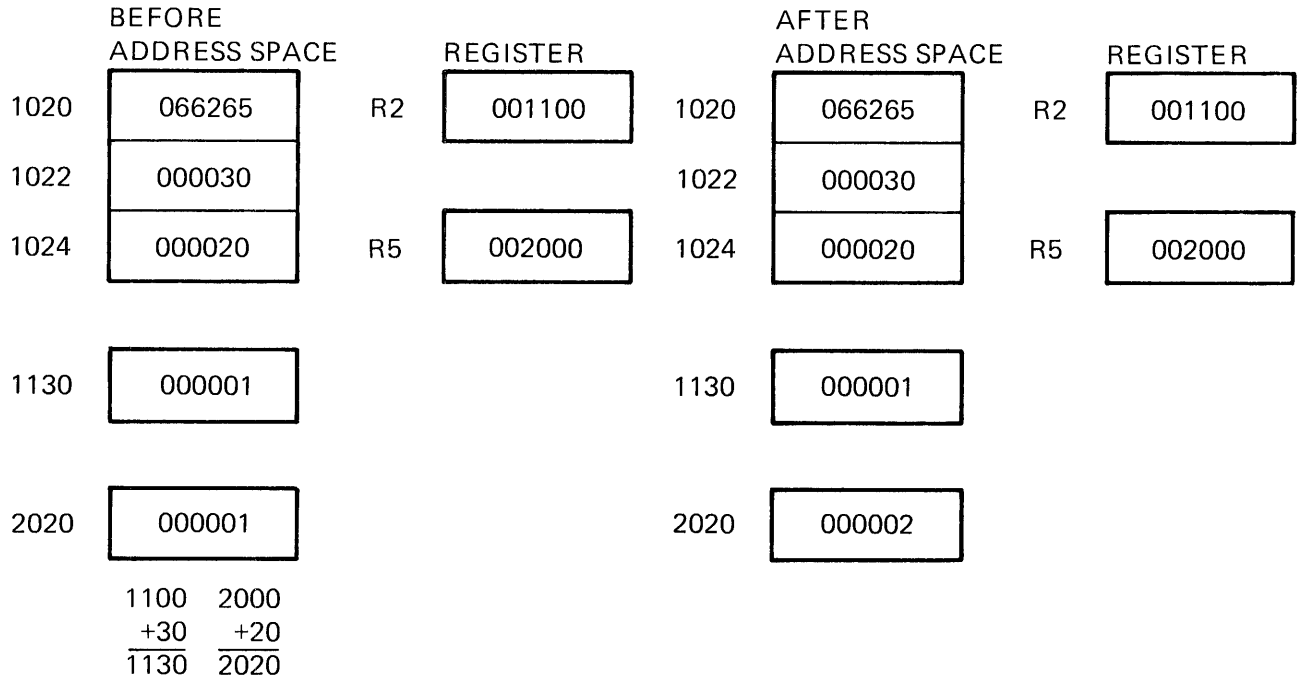


Figure 7-18 ADD 30 (R2), 20 (R5) Add

7.1.4 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.

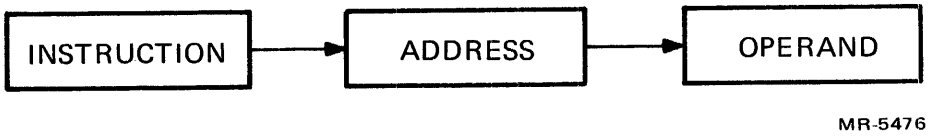


Figure 7-19 Mode 1 Register Deferred

3	Autoincrement Deferred	@(Rn) +	Register is first used as a pointer to a word containing the address of the operand, then incremented (always by two; even for byte instructions).

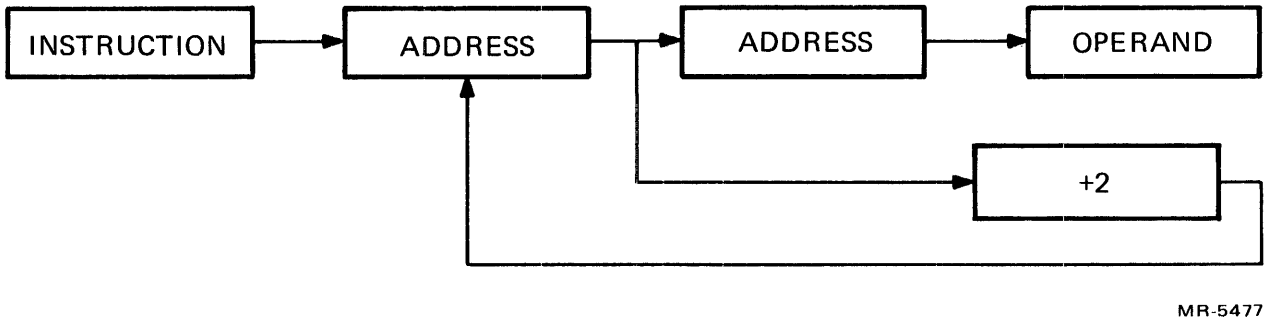


Figure 7-20 Mode 3 Autoincrement Deferred

5	Autodecrement Deferred	@-(Rn)	Register is decremented (always by two; even for byte instructions) and then used as a pointer to a word containing the address of the operand.

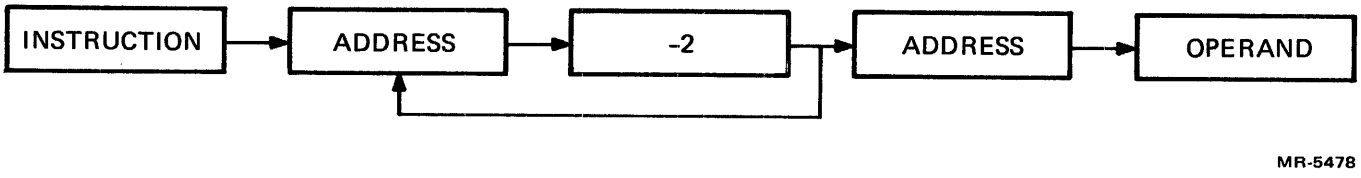


Figure 7-21 Mode 5 Autodecrement Deferred

7	Index Deferred	@X(Rn)	Value X (stored in a word following the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

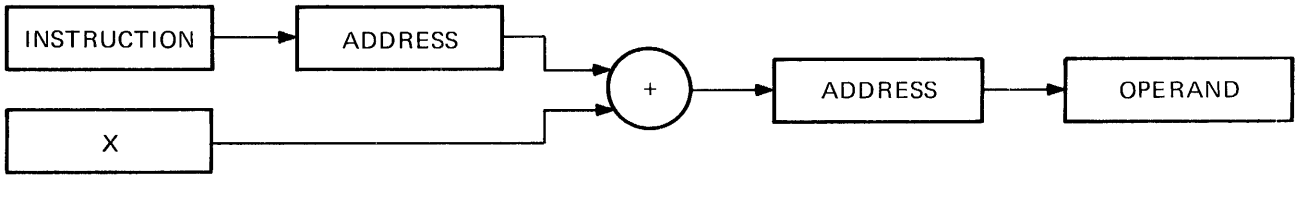


Figure 7-22 Mode 7 Index Deferred

The following examples illustrate the deferred modes.

Register Deferred Mode Example

Symbolic	Octal Code	Instruction Name
CLR @R5	005015	Clear

Operation: The contents of location specified in R5 are cleared.

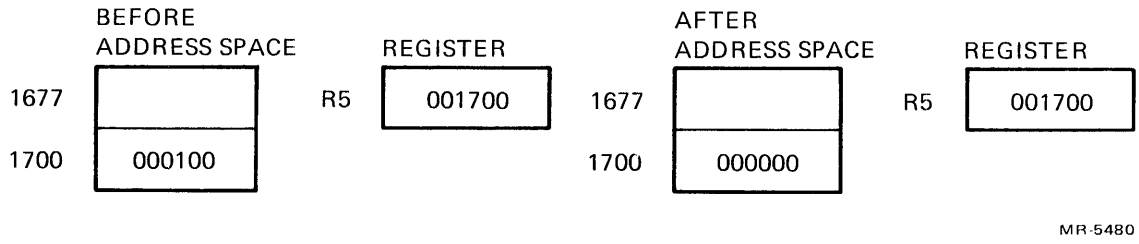


Figure 7-23 CLR @ R5 Clear

Autoincrement Deferred Mode Example (Mode 3)

Symbolic	Octal Code	Instruction Name
INC@(R2)+	005232	Increment

Operation: The contents of R2 are used as the address of the address of the operand.
Operand is increased by one. Contents of R2 are incremented by two.

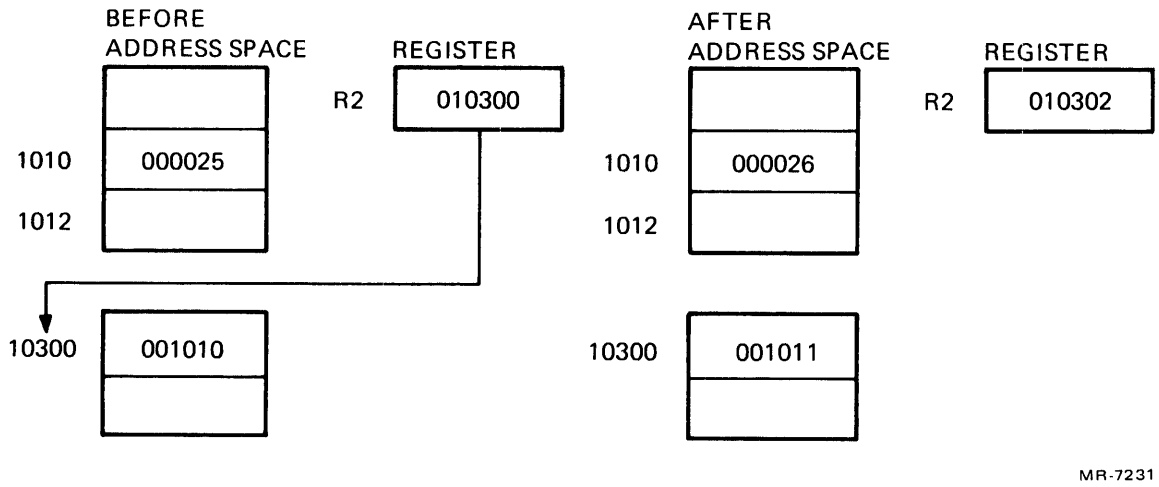


Figure 7-24 INC @ (R2) + Increment

Autodecrement 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 operand. Operand is one's complemented (i.e., logically complemented).

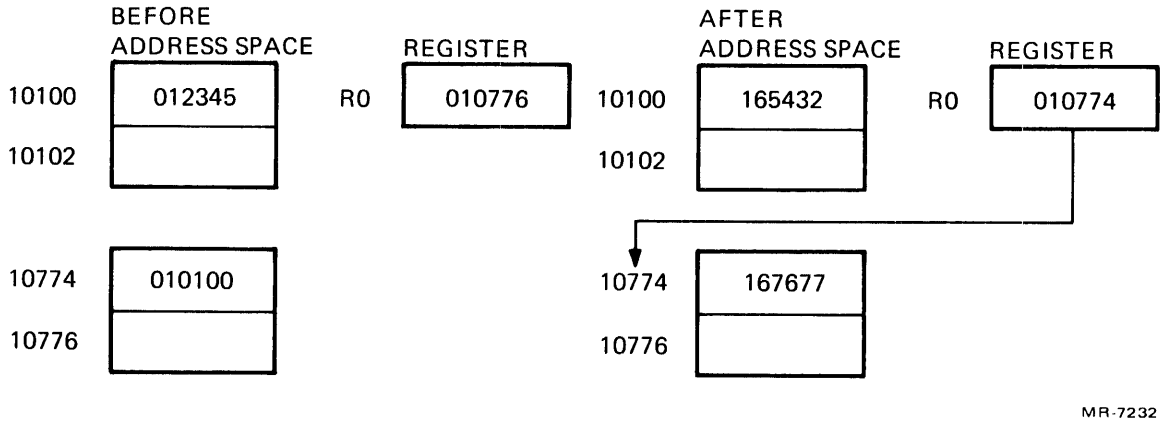


Figure 7-25 COM @-(R0) Complement

Index Deferred Mode Example (Mode 7)

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

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 stored in R1.

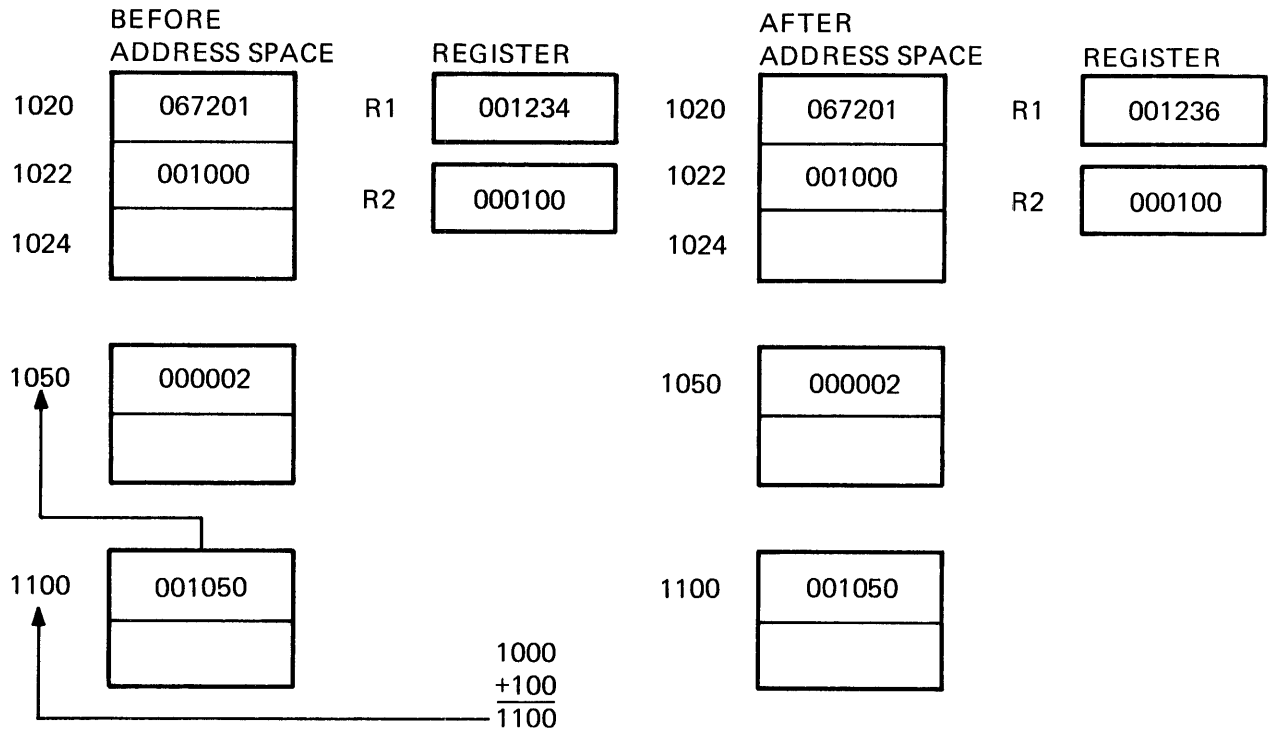


Figure 7-26 ADD @ 1000 (R2), R1 Add

7.1.5 Use of the PC as a General Register

Although register seven is a general purpose register, it doubles in function as the program counter for the microprocessor. 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 SBC-11/21 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	A	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. SBC-11/21 can accomplish the relocation of a program 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. Thus, PIC usually references locations relative to the current location.

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

7.1.5.1 Immediate Mode -- 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.

OPR #n,DD

Immediate Mode Example

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

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.

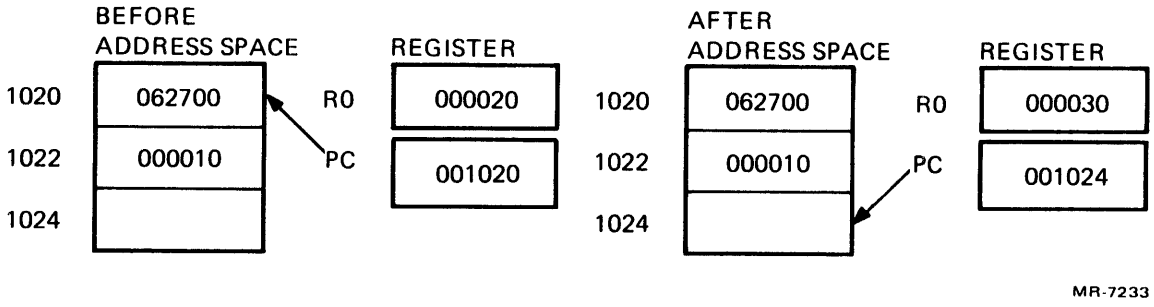


Figure 7-27 ADD # 10, R0 Add

7.1.5.2 Absolute Addressing -- 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).

OPR @ #A

Absolute Mode Examples

	Symbolic	Octal Code	Instruction Name
1.	CLR @ #1100	005037 001100	Clear

Operation: Clear the contents of location 1100.

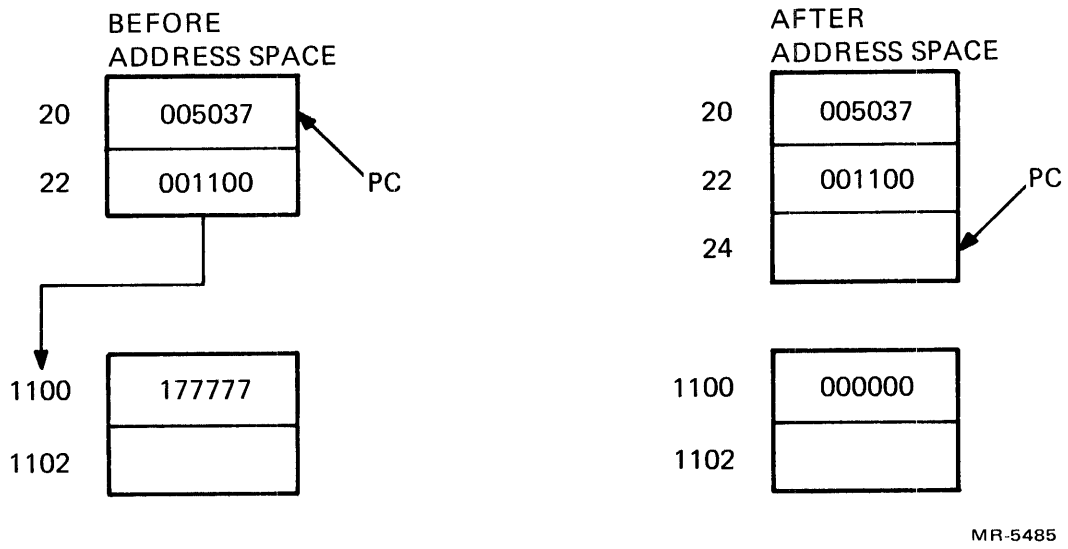


Figure 7-28 CLR @ # 1100 Clear

2. ADD @ #2000,R3 063703
002000

Operation: Add contents of location 2000 to R3.

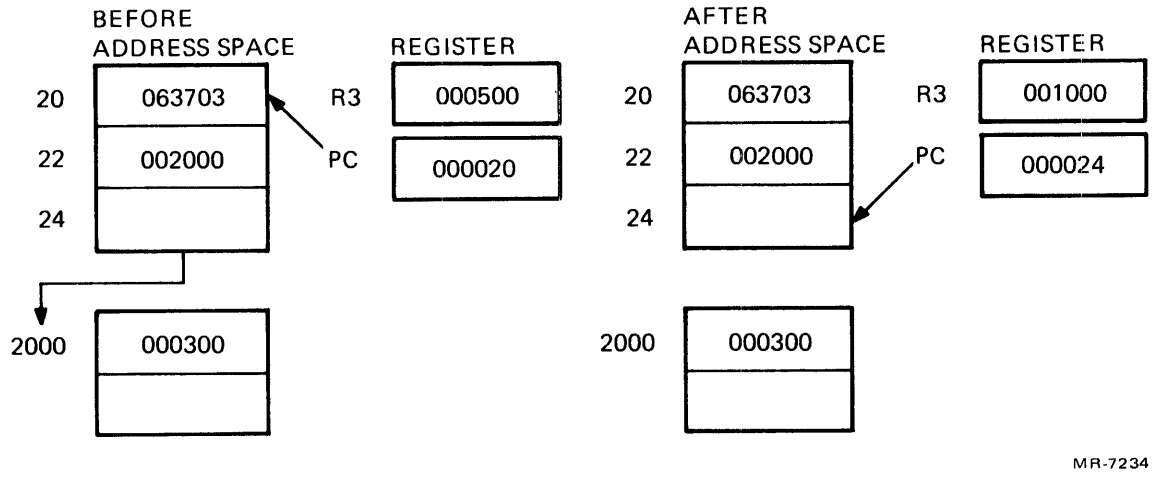


Figure 7-29 ADD @ # 2000 Add

7.1.5.3 Relative Addressing -- 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.

OPR A or OPR X (PC)
where X is the location of A relative to the instruction.

Relative Addressing Example

Symbolic	Octal Code	Instruction Name
INC A	005267 000054	Increment

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.

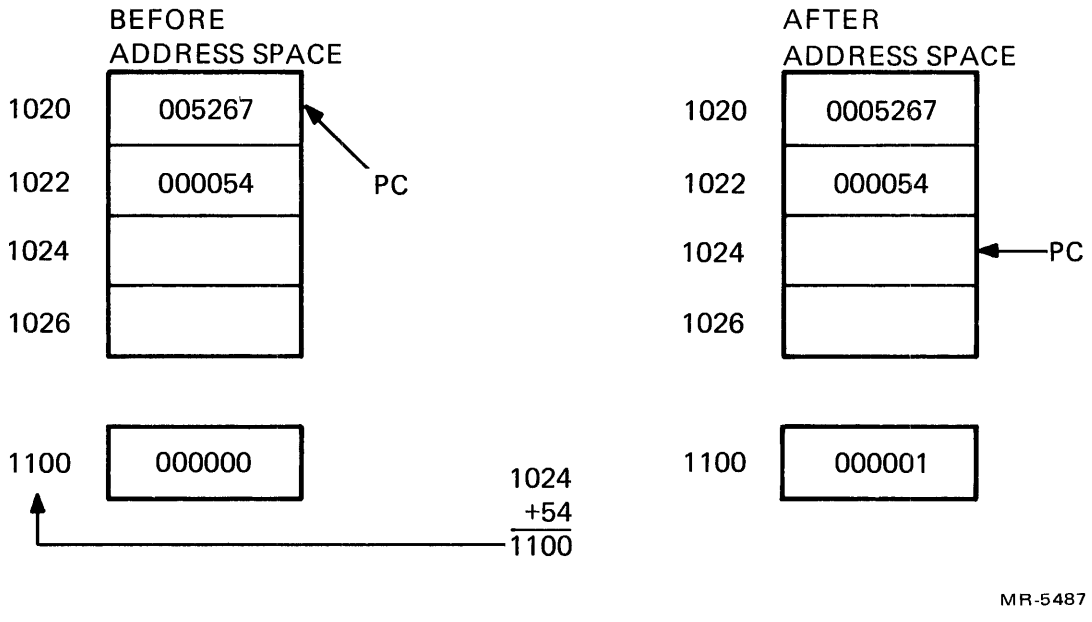


Figure 7-30 INC A Increment

7.1.5.4 Relative Deferred Addressing -- 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.

OPR @A or OPR @X(PC)
where x is location containing address of A, relative to the instruction.

Relative Deferred Mode Example

Symbolic	Octal Code	Instruction Name
CLR @A	005077 000020	Clear

Operation: Add second word of instruction to updated PC to produce address of address of operand. Clear operand.

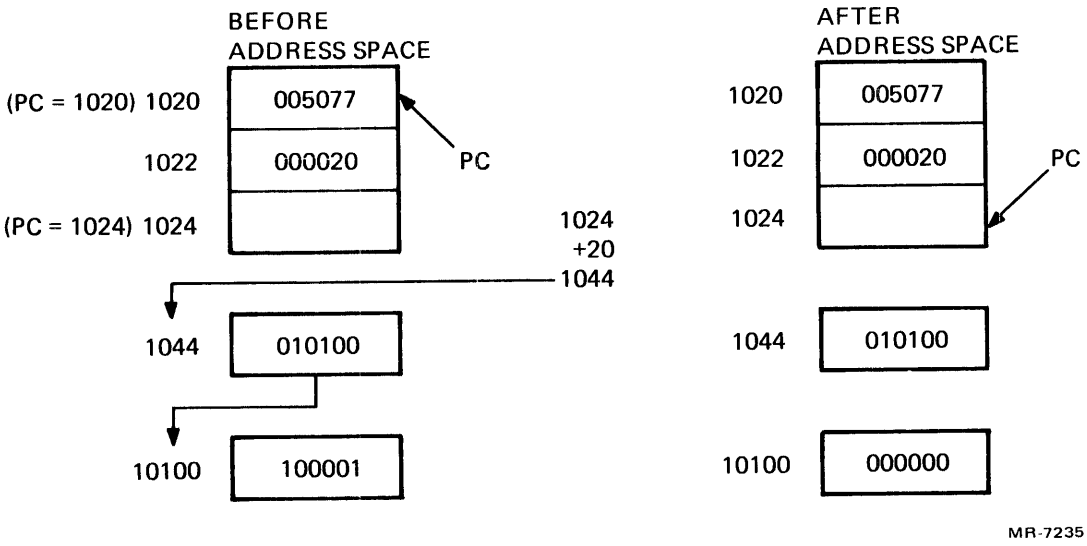


Figure 7-31 CLR @ A Clear

7.1.6 Use of Stack Pointer as General Register
The processor stack pointer (SP, register R6) is in most cases the general register used for the stack operations related to program nesting. Autodecrement with register R6 "pushes" data on to the stack and autoincrement with register R6 "pops" data off 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.

7.2 INSTRUCTION SET

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

MNEMONIC: This is indicated before each description. When the word instruction has a byte equivalent, the byte mnemonic is also shown.

INSTRUCTION FORMAT: A diagram accompanying each instruction shows the octal op code, the binary op code, and bit assignments. [Note that in byte instructions the most significant bit (bit 15) is always a one].

SYMBOLS:

() = contents of

SS or src = source address

DD or dst = destination address

loc = location

← = becomes

↑ = "is popped from stack"

↓ = "is pushed onto stack"

∧ = boolean AND

∨ = boolean OR

⊻ = exclusive OR

~ = boolean not

Reg or R = register

B = Byte

■ = 0 for word
1 for byte

, = concatenated

7.2.1 Instruction Formats

The following formats include all instructions used in the SBC-11/21. 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, TSTB, ROR, RORB, ROL, ROLB, ASR, ASRB, ASL, ASLB, JMP, SWAB, MFPS, MTPS, SXT, XOR)

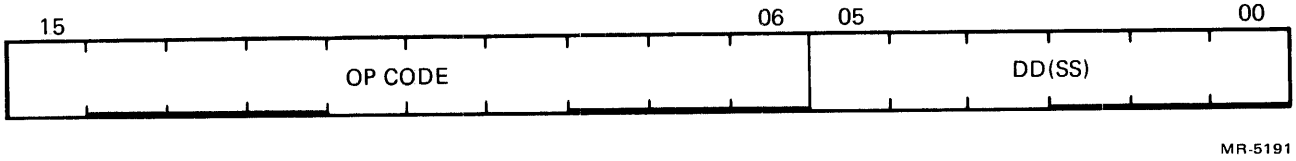


Figure 7-32 Single Operand Group

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

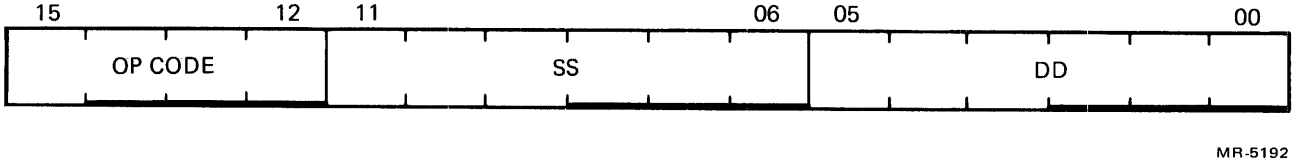


Figure 7-33 Double Operand Group

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

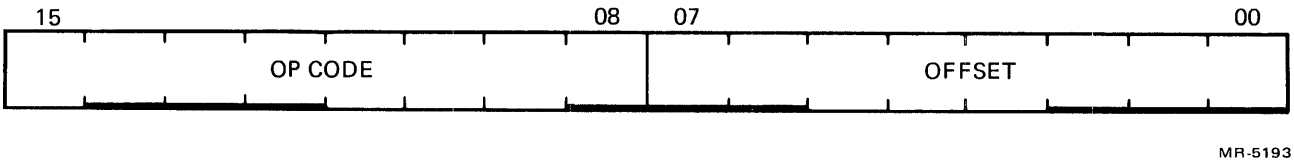


Figure 7-34 Program Control Group Branch

   b. Jump To Subroutine (JSR)

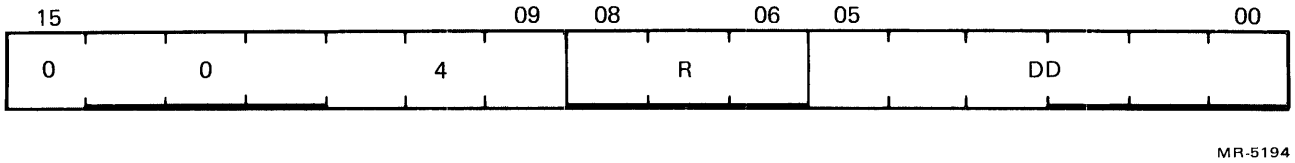


Figure 7-35 Program Control Group JSR

c. Subroutine Return (RTS)

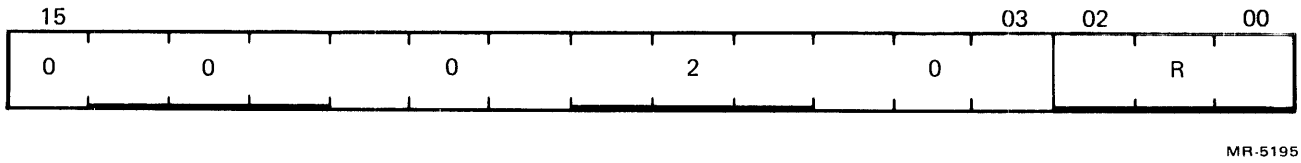


Figure 7-36 Program Control Group (RTS)

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

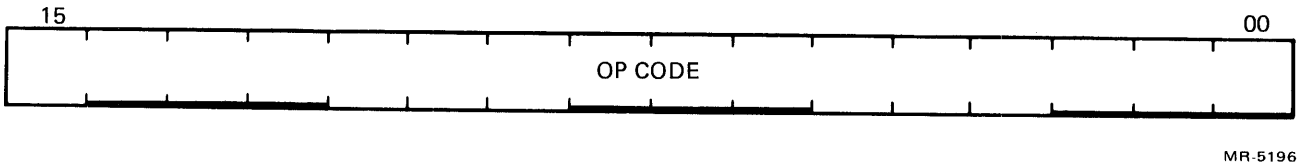


Figure 7-37 Program Control Group Traps

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

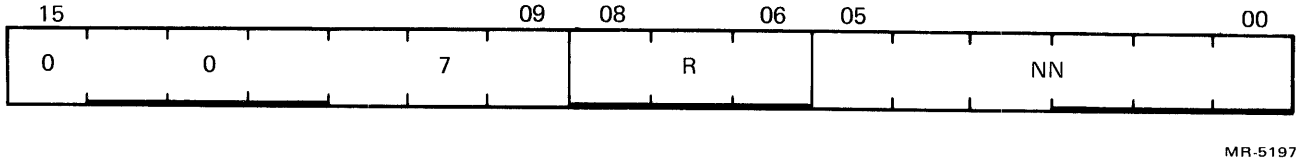


Figure 7-38 Program Control Group Subtract

4. Operate Group (HALT, WAIT, RTI, RESET, RTT, NOP, MFPT)

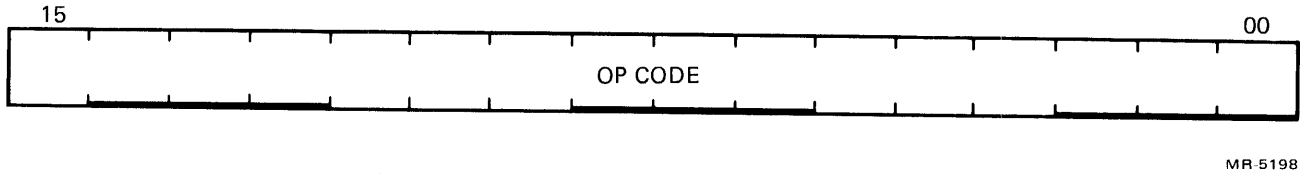


Figure 7-39 Operate Group

5. Condition Code Operators (all condition code instructions)

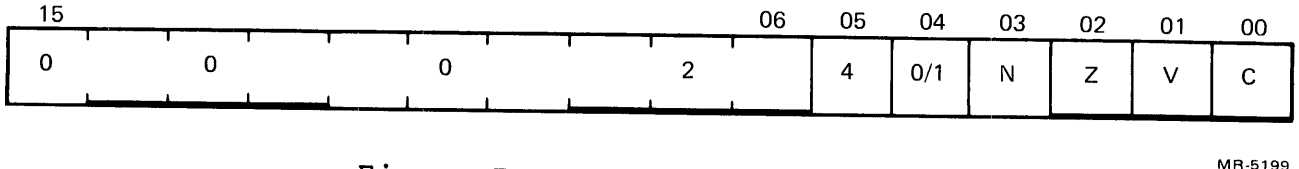


Figure 7-40 Condition Group

7.2.1.1 Byte Instructions -- The SBC-11/21 includes a full complement of instructions that manipulate byte operands. Since all microprocessor addressing is byte-oriented, byte manipulation addressing is straightforward. Byte instructions with autoincrement or autodecrement direct addressing cause the specified register to be modified by one to point to the next byte of data. Byte operations in register mode access the low-order byte of the specified register. These provisions enable the SBC-11/21 to perform as either a word or byte microprocessor. The numbering scheme for word and byte addresses in memory is:

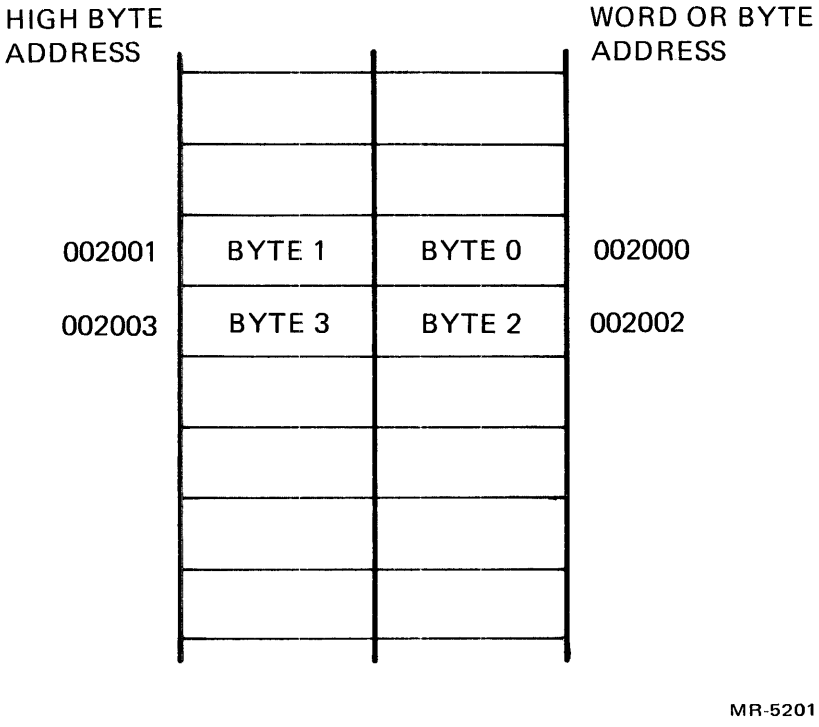


Figure 7-41 Byte Instructions

The most significant bit (Bit 15) of the instruction word is set to indicate a byte instruction.

Example:

Symbolic	Octal	
CLR	0050DD	Clear Word
CLRB	1050DD	Clear Byte

7.2.2 List of Instructions

The SBC-11/21 instruction set is shown in the following sequence.

SINGLE OPERAND

Mnemonic	Instruction	Op Code
General		
CLR(B)	clear dst	050DD
COM(B)	complement dst	051DD
INC(B)	increment dst	052DD
DEC(B)	decrement dst	053DD
NEG(B)	negate dst	054DD
TST(B)	test dst	057DD
Shift & Rotate		
ASR(B)	arithmetic shift right	062DD
ASL(B)	arithmetic shift left	063DD
ROR(B)	rotate right	060DD
ROL(B)	rotate left	061DD
SWAB	swap bytes	0003DD
Multiple Precision		
ADC(B)	add carry	055DD
SBC(B)	subtract carry	056DD
SXT	sign extend	0067DD
PS Word Operators		
MFPS	move byte from PS	1067DD
MTPS	move byte to PS	1064SS

DOUBLE OPERAND

General		
MOV(B)	move source to destination	1SSDD
CMP(B)	compare src to dst	2SSDD
ADD	add src to dst	06SSDD
SUB	subtract src from dst	16SSDD
Logical		
BIT(B)	bit test	3SSDD
BIC(B)	bit clear	4SSDD
BIS(B)	bit set	5SSDD
XOR	exclusive or	074RDD

PROGRAM CONTROL

Mnemonic	Instruction	Op Code or Base Code
Branch		
BR	branch (unconditional)	000400
BNE	branch if not equal (to zero)	001000
BEQ	branch if equal (to zero)	001400
BPL	branch if plus	100000
BMI	branch if minus	100400
BVC	branch if overflow is clear	102000
BVS	branch if overflow is set	102400
BCC	branch if carry is clear	103000
BCS	branch if carry is set	103400
Signed Conditional Branch		
BGE	branch is greater than or equal (to zero)	002000
BLT	branch if less than (zero)	002400
BGT	branch if greater than (zero)	003000
BLE	branch if less than or equal (to zero)	003400
Unsigned Conditional Branch		
BHI	branch if higher	101000
BLOS	branch if lower or same	101400
BHIS	branch if higher or same	103000
BLO	branch if lower	103400
Jump & Subroutine		
JMP	jump	0001DD
JSR	jump to subroutine	004RDD
RTS	return from subroutine	00020R
SOB	subtract one and branch (if ≠ 0)	077R00
Trap & Interrupt		
EMT	emulator trap	104000--104377
TRAP	trap	104400--104777
BPT	breakpoint trap	000003
IOT	input/output trap	000004
RTI	return from interrupt	000002
RTT	return from interrupt	000006
MISCELLANEOUS		
HALT	halt	000000
WAIT	wait for interrupt	000001
RESET	reset external bus	000005
MFPT	move processor type	000007

RESERVED INSTRUCTIONS

		00021R
		00022

CONDITION CODE OPERATORS

CLC	clear C	000241
CLV	clear V	000242
CLZ	clear Z	000244
CLN	clear N	000250
CCC	clear all CC bits	000257
SEC	set C	000261
SEV	set V	000262
SEZ	set Z	000264
SEN	set N	000270
SCC	set all CC bits	000277
NOP	no operation	00240

7.2.3 Single Operand Instructions

NOTE

In all SBC-11/21 instructions a write operation to a memory location or register is always preceded by a read operation from the same location. The exception is when writing PC and PSW to the stack in two cases.

1. The execution of the microcode preceding an interrupt or trap service routine.

2. Interrupt and trap instructions:

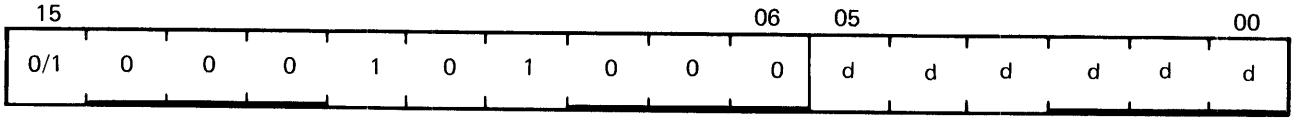
   HLT
   TRAP
   BPT
   IOT

7.2.3.1 General --

CLR
CLRB

CLEAR DESTINATION

■050DD



MR-5202

Figure 7-42 CLR

Operation: (dst)←0

Condition Codes:
N: cleared
Z: set
V: cleared
C: cleared

Description: Word: Contents of specified destination are replaced with zeros.
Byte: Same

Example:

```
                    CLR R1

          Before                        After
(R1) = 177777                 (R1) = 000000

          NZVC                          NZVC
          1111                          0100
```

COM

COMB

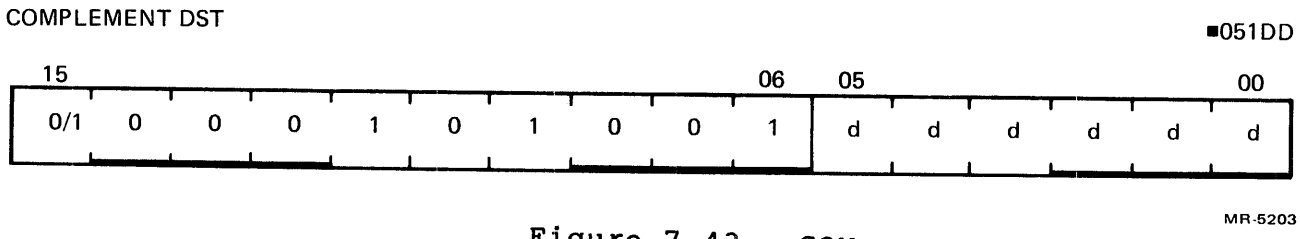


Figure 7-43 COM

Operation: (dst)←~(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

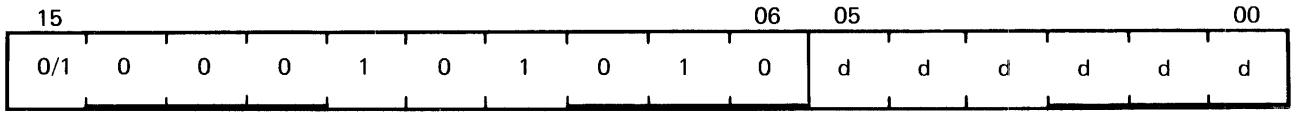
Description: Replaces the contents of the destination address by their logical complement (each bit equal to 0 is set and each bit equal to one is cleared)
Byte: Same

Example:

```
                    COM R0

          Before                        After
(R0) = 013333                 (R0) = 164444

          NZVC                          NZVC
          0110                          1001
```

INC
INCB

INCREMENT DST ■052DD



MR-5204

Figure 7-44 INC

Operation: (dst)←(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
Byte: Same

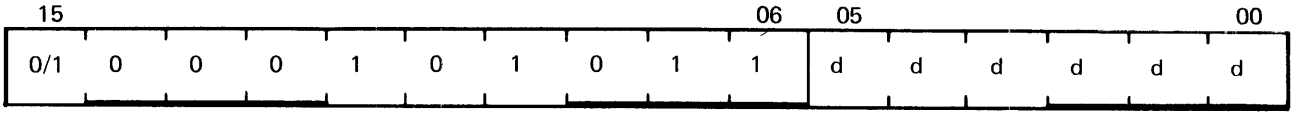
Example:

INC R2

Before	After
(R2) = 000333	(R2) = 000334
NZVC	NZVC
0000	0000

DEC
DECB

DECREMENT DST ■053DD



MR-5205

Figure 7-45 DEC

Operation: (dst)← (dst) -1

Condition Codes:
N: set if result is <0, cleared otherwise
Z: set is result is 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: not affected

Description:
Word: Subtract one from the contents of the destination
Byte: Same

Example:

```
DEC R5

        Before                  After
(R5) = 000001           (R5) = 000000

        NZVC                    NZVC
        1000                    0100
```

NEG
NEGB

NEGATE DST

■054DD

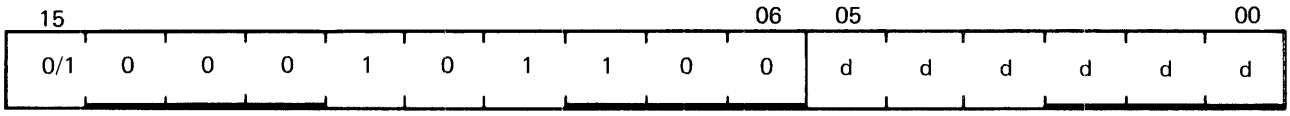


Figure 7-46 NEG

Operation: (dst)← -(dst)

Condition Codes:
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                  After
(R0) = 000010           (R0) = 177770

        NZVC                    NZVC
        0000                    1001
```

TST
TSTB

TEST DST ■057DD

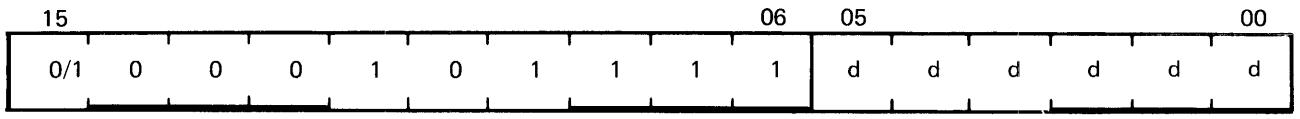


Figure 7-47 TST

Operation: (dst)←(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) = 012340
NZVC	NZVC
0011	0000

7.2.3.2 Shifts & Rotates -- Scaling data by factors of two is accomplished by the shift instructions:

ASR -- Arithmetic shift right

ASL -- Arithmetic shift left

The sign bit (bit 15) of the operand is reproduced in shifts to the right. The low order bit is filled with zero in shifts to the left. Bits shifted out of the C bit, as shown in the following examples, are lost.

The rotate instructions operate on the destination word and the C bit as though they formed a 17-bit "circular buffer." These instructions facilitate sequential bit testing and detailed bit manipulation.

ASR
ASRB

ARITHMETIC SHIFT RIGHT

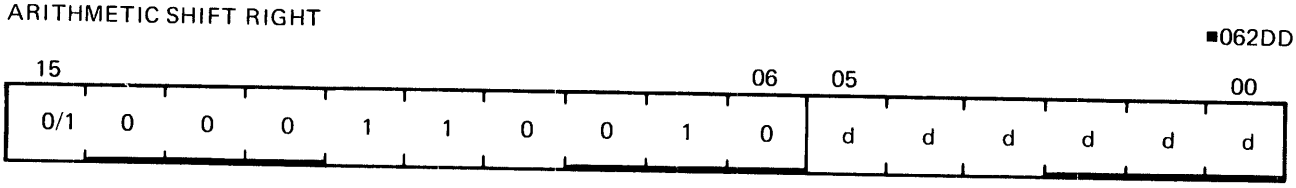


Figure 7-48 ASR

Operation: (dst)←(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 the 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 zero of the destination. ASR performs signed division of the destination by two.
Word:

Example:

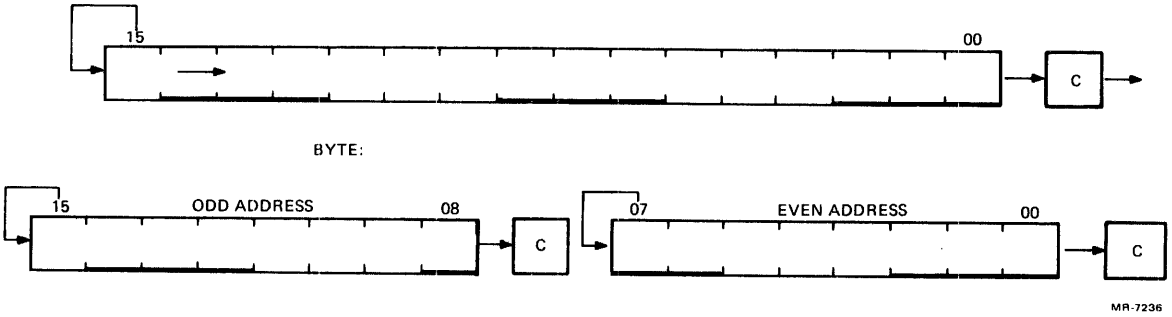


Figure 7-49 ASR Description

ASL
ASLB

ARITHMETIC SHIFT LEFT

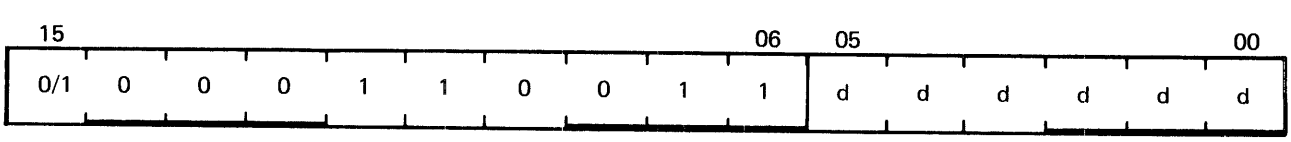


Figure 7-50 ASL

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 zero is loaded with a zero. 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 two with overflow indication.
Word:

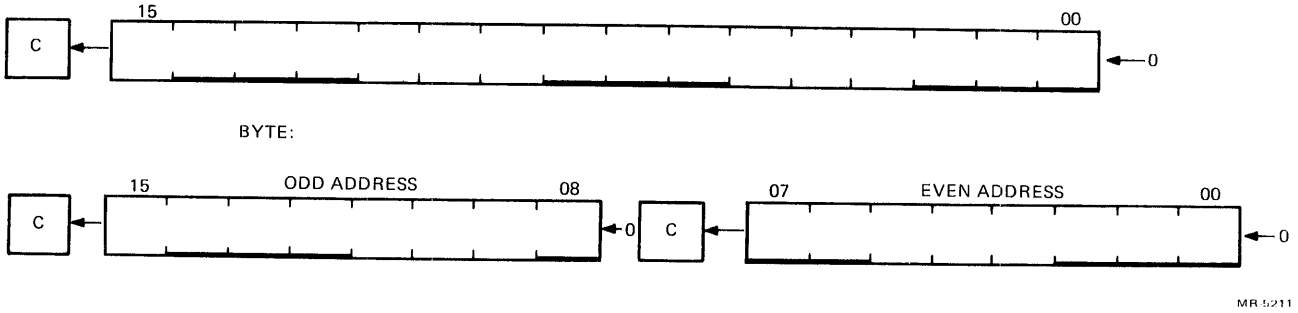


Figure 7-51 ASL Description

ROR

RORB

ROTATE RIGHT

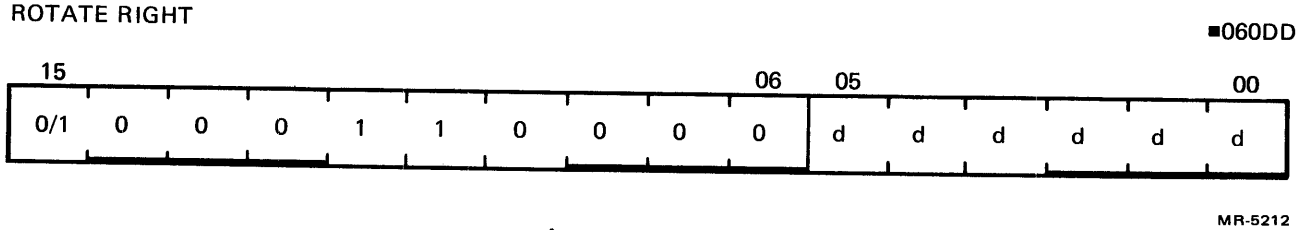


Figure 7-52 ROR

Operation: (dst)←(dst) rotate right one place

Condition Codes:
N: set if the high-order bit of the result is set (result <0); cleared othewise
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 into bit 15 of the destination.

Example:

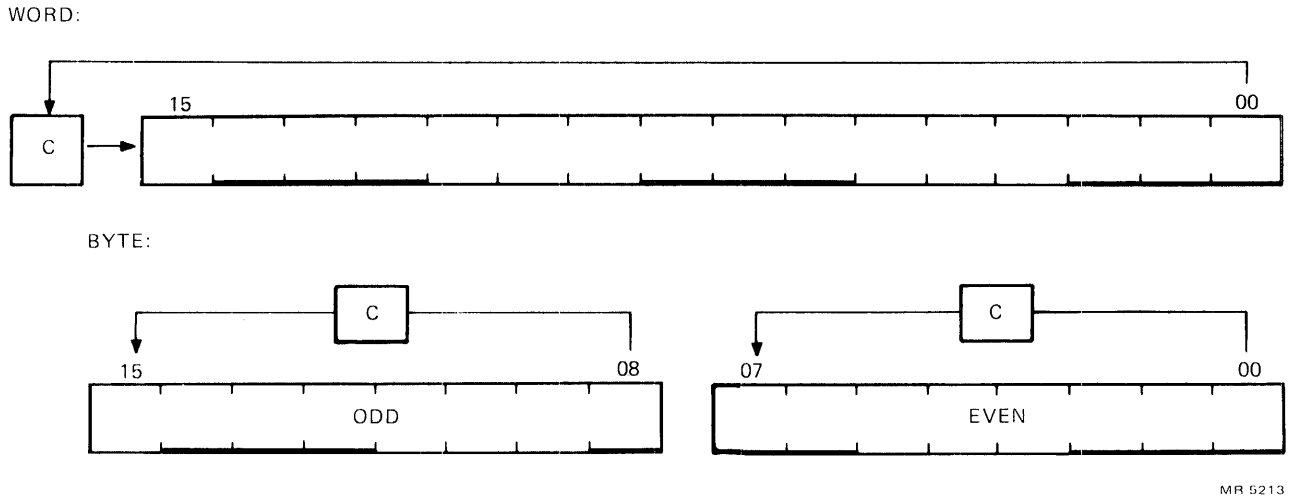


Figure 7-53 ROR Description

ROL
ROLB

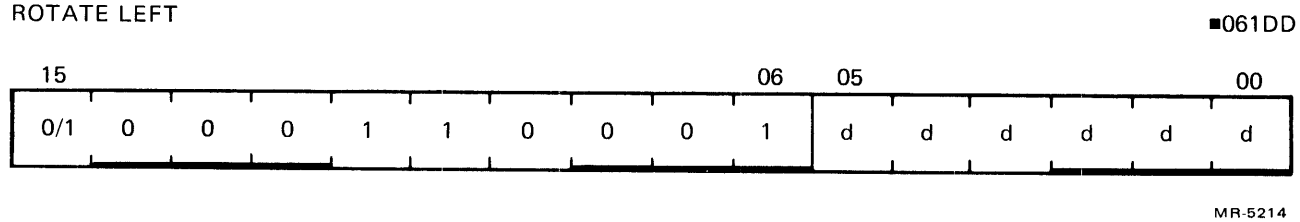


Figure 7-54 ROL

Operation: (dst) ←(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.

Example:

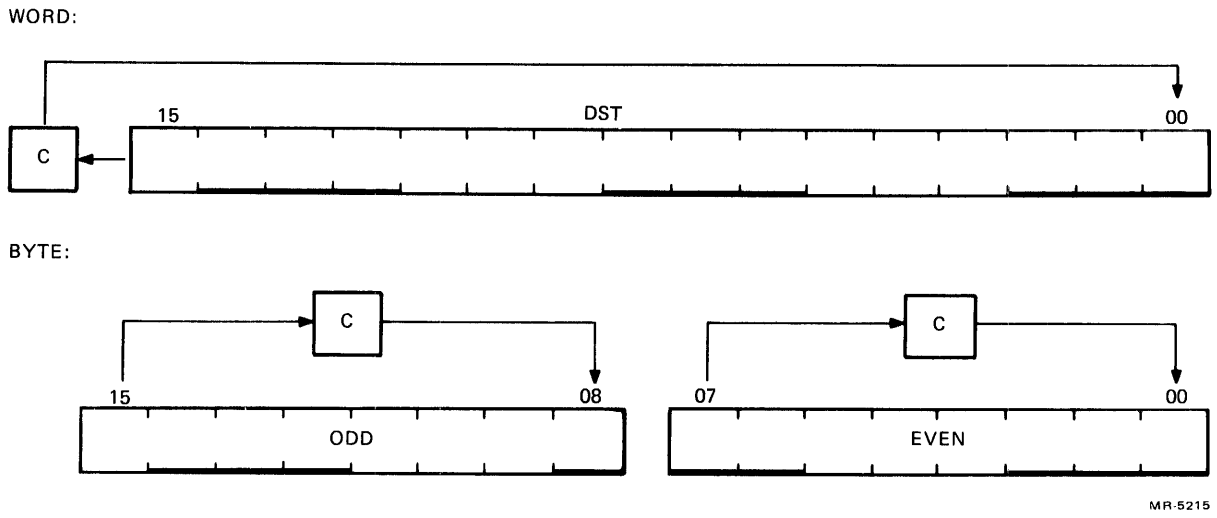


Figure 7-55 ROL Description

SWAB

SWAP BYTES

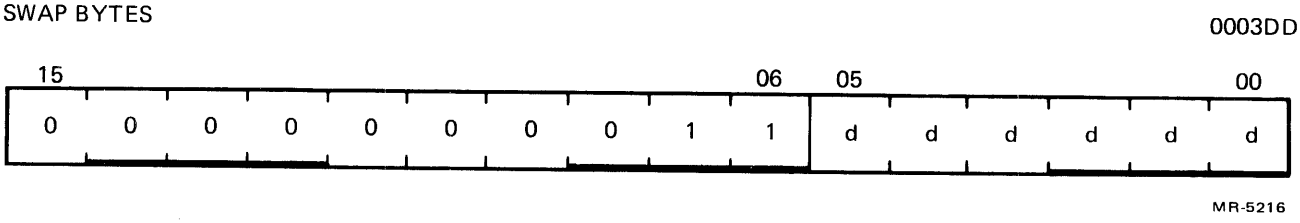


Figure 7-56 SWAB

Operation: Byte 1/Byte 0 ←Byte 0/Byte 1

Condition Codes: N: set if high-order bit of low-order 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 destination word (destination must be a word address).

Example:

SWAB R1

Before		After	
(R1) =	077777	(R1) =	177577
	NZVC		NZVC
	1111		0000

7.2.3.3 Multiple Precision -- It is sometimes necessary to do arithmetic on operands considered as multiple words or bytes. The SBC-11/21 makes special provision for such operations with the instructions ADC (Add Carry) and SBC (Subtract Carry) and their byte equivalents.

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

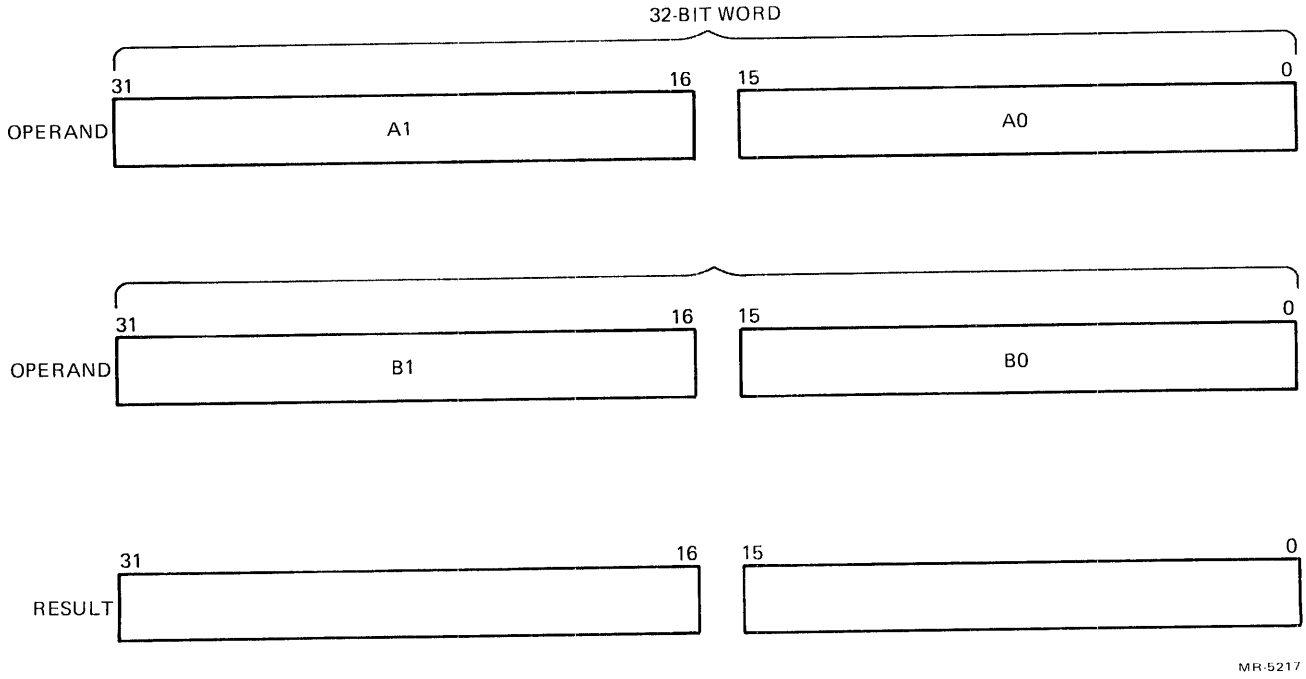


Figure 7-57 Multiple Precision

Example:

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

-1 = 37777777777

(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 37777777776 or -2

ADC
ADCB

ADD CARRY

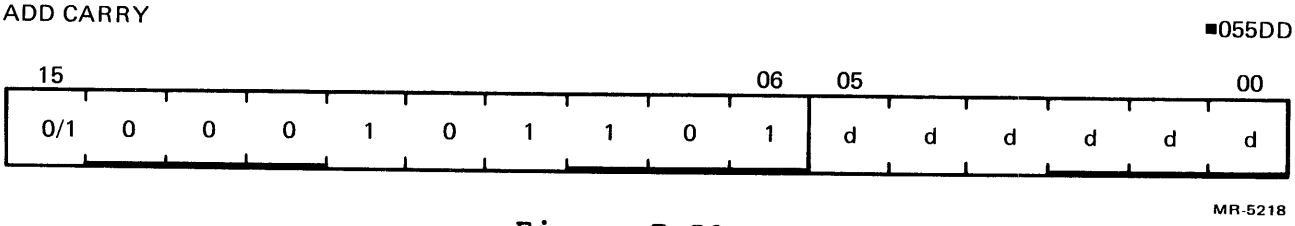


Figure 7-58 ADC

Operation: (dst)←(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
```

SBC
SBCB

SUBTRACT CARRY

■056DD

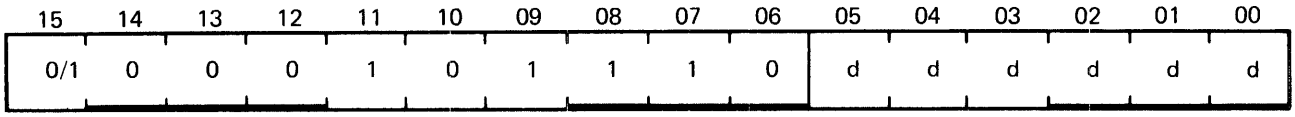


Figure 7-59 SBC

Operation: (dst)←(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
```

SXT

SIGN EXTEND

0067DD

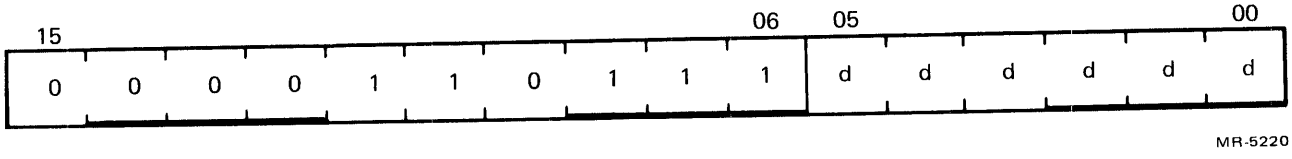


Figure 7-60 SXT

Operation:
(dst)←0 if N-bit is clear
(dst)←1 N-bit is set

Condition Codes:
N: unaffected
Z: set if N-bit 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 zero 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) = 177777

          NZVC                          NZVC
          1000                          1000
```

7.2.3.4 PS Word Operators --

MFPS

MOVE BYTE FROM PROCESSOR STATUS WORD

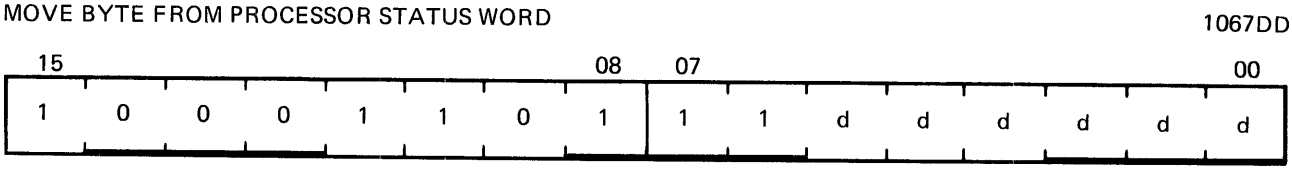


Figure 7-61 MFPS

Operation: (dst)←PS
dst lower 8 bits

Condition Code Bits:
N = set if PS bit 7 = 1; cleared otherwise
Z = set if PS <0:7> = 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]                        R0 [000014]
PS [000014]                   PS [000000]
```

MTPS

MOVE BYTE TO PROCESSOR STATUS WORD

1064SS

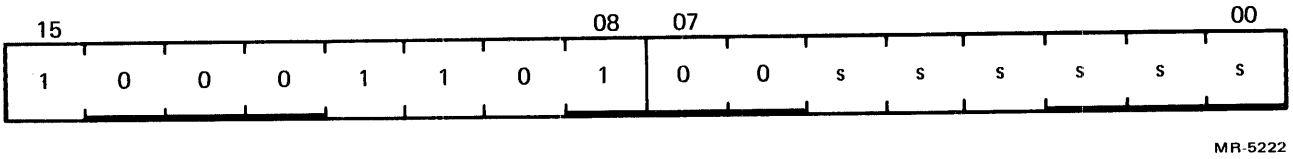


Figure 7-62 MTPS

Operation: PS←(SRC)

Condition Codes: Set according to effective SRC operand bits 0--3

Description: The eight bits of the effective operand replaces the current contents of the PS. The source operand address is treated as a byte address. Note that the T bit (PS bit four) cannot be set with this instruction. The SRC operand remains unchanged. This instruction can be used to change the priority bits (PS bits 7--5) in the PS.

7.2.4 Double Operand Instructions

Double operand instructions provide an instruction (and time) saving facility since they eliminate the need for "load" and "save" sequences such as those used in accumulator-oriented machines.

7.2.4.1 General --

MOV
MOVB

MOVE SOURCE TO DESTINATION

■1SSDD

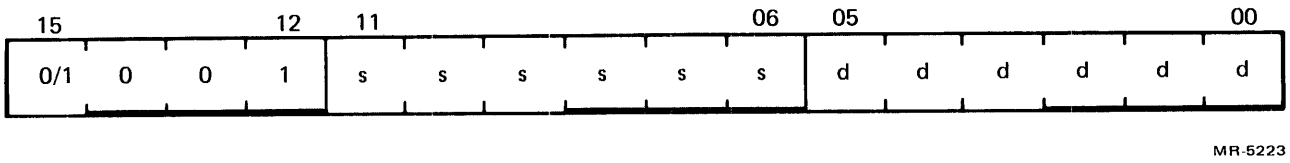


Figure 7-63 MOV

Operation: (dst)←(src)

Condition Codes:
N: set if (src) <0; cleared otherwise
Z: set if (src) = 0; cleared otherwise
V: cleared
C: not affected

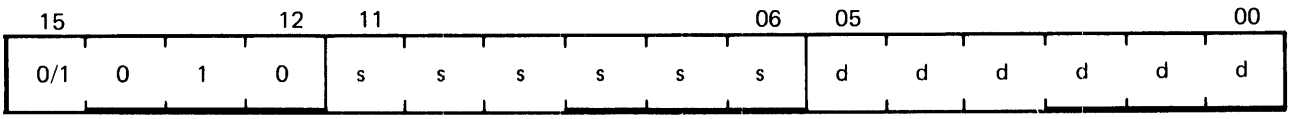
Description: Word: Moves the source operand to the destination location. The previous contents of the destination are lost. The contents of the source address are not affected.
Byte: Same as MOV. The MOVB to a register (unique among byte instructions) extends the most significant bit of the low order byte (sign extension). Otherwise MOVB operates on bytes exactly as MOV operates on words.

Example:

```
MOV  XXX,R1          ;loads Register one 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.
```

CMP
CMPB

COMPARE SRC TO DST ■2SSDD



MR-5224

Figure 7-64 CMP

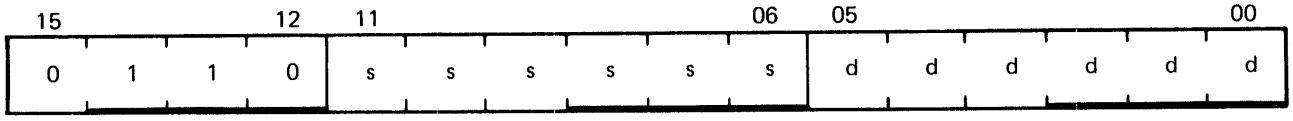
Operation: (src)-(dst)

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 otherwise.
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).

ADD

ADD SRC TO DST 06SSDD



MR-5225

Figure 7-65 ADD

Operation: (dst)←(src)+(dst)

Condition Codes:
N: set if result <0; cleared otherwise.
Z: set if result = 0; cleared othrwise.
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 othrewise.

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 affected. Two's complement addition is performed.
Note: There is no equivalent byte mode.

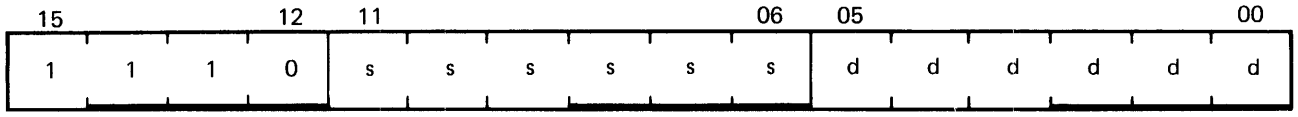
Examples:

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.

SUB

SUBTRACT SRC FROM DST 16SSDD



MR-5226

Figure 7-66 SUB

Operation: (dst)←(dst)-(src)

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 if operands were of opposite signs and the sign of the source was the same as the sign of ther esult; 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".
Note: There is no equivalent byte mode.

Example:

SUB R1,R2

Before	After
(R1) = 011111	(R1) = 011111
(R2) = 012345	(R2) = 001234
NZVC	NZVC
1111	0000

7.2.4.2 **Logical** -- These instructions have the same format as the double operand arithmetic group. They permit operations on data at the bit level.

BIT
BITB

BIT TEST

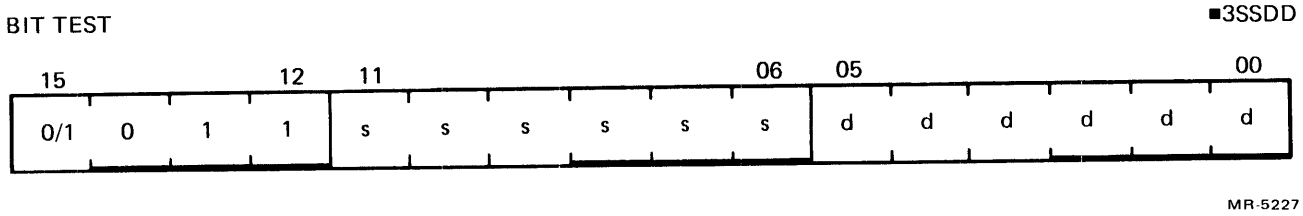


Figure 7-67 BIT

Operation: (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: Performs logical "and" comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination is 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 three and four of R3 to see if both are off

R3 = 0 000 000 000 011 000

Before	After
NZVC	NZVC
1111	0001

BIC
BICB

BIT CLEAR ■4SSDD

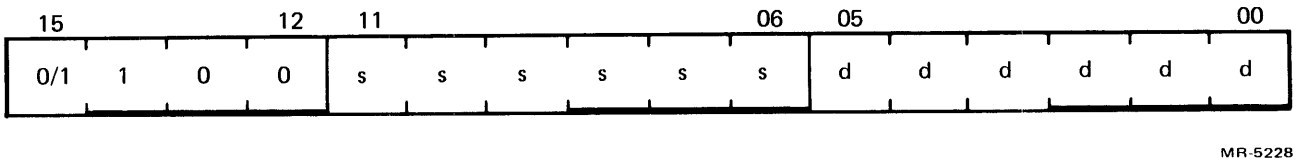


Figure 7-68 BIC

Operation: (dst)←(dst) ∧ ~(src)

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) = 001111	(R4) = 000101
NZVC	NZVC
1111	0001

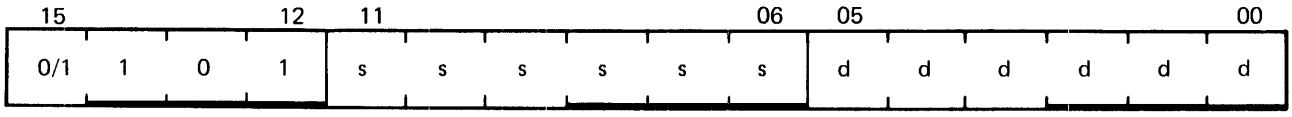
Before: (R3) = 0 000 001 010 011 100
(R4) = 0 000 001 001 001 001

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

BIS
BISB

BIT SET

■5SSDD



MR-5229

Figure 7-69 BIS

Operation: (dst)←(dst) $\vee$ (src)

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.

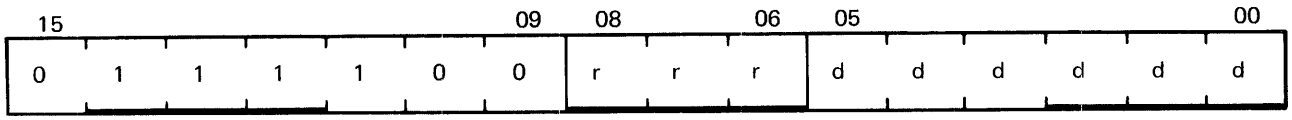
Example:

```
BIS R0,R1

        Before                          After
(R0) = 001234                   (R0) = 001234
(R1) = 001111                   (R1) = 001335

       NZVC                             NZVC
       0000                             0000

Before:      (R0) = 0 000 001 010 011 100
             (R1) = 0 000 001 001 001 001

After:       (R1) = 0 000 001 011 011 101
```

XOR

EXCLUSIVE OR

074RDD



MR-5230

Figure 7-70 XOR

Operation: (dst)←(dst)$\overline{\vee}$(Reg)

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 R0,R2

Before	After
(R0) = 001234	(R0) = 001234
(R2) = 001111	(R2) = 000325
NZVC	NZVC
1111	0001

Before: (R0) = 0 000 001 010 011 100
(R2) = 0 000 001 001 001 001

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

7.2.5 Program Control Instructions

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

1. the branch instruction is unconditional.
2. 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 seven is the sign of the offset. If it is set, the offset is negative and the branch is done in the backward direction. Similarly if it is not set, the offset is positive and the branch is done in the forward direction.

The 8-bit offset allows branching in the backward direction by 200_8 words (400 bytes) from the current PC, and in the forward direction by 177_8 words (376 bytes) from the current PC.

The microprocessor 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 OPs.

BR

BRANCH (UNCONDITIONAL) 000400 PLUS OFFSET

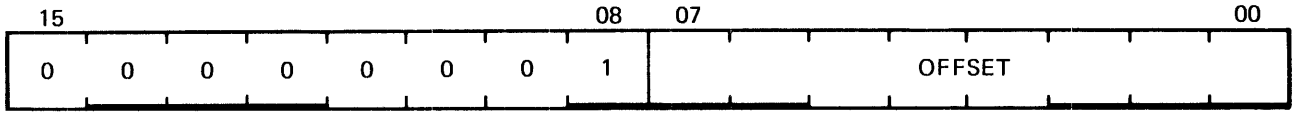


Figure 7-71 BR

Operation: PC ← PC + (2 X offset)

Condition Codes: Unaffected

Description: Provides a way of transferring program control within a range of -128_{10} to $+127_{10}$ words with a one word instruction.

New PC address = updated PC + (2 X offset)

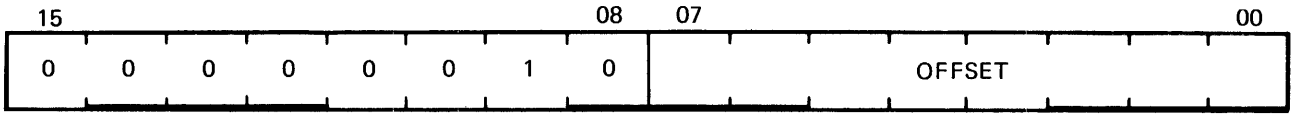
Updated PC = address of branch instruction +2

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

BNE

BRANCH IF NOT EQUAL (TO ZERO) 001000 PLUS OFFSET



MR-5232

Figure 7-72 BNE

Operation: PC ← PC + (2 X offset) 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 ≠ B

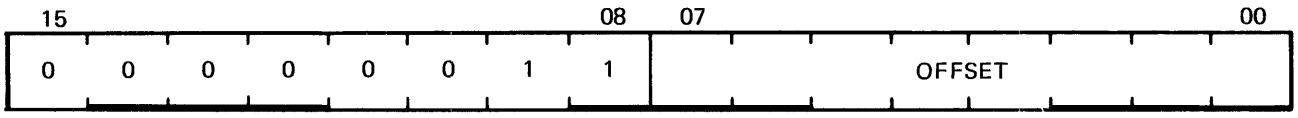
and the sequence

```
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 ≠ 0

BEQ

BRANCH IF EQUAL (TO ZERO) 001400 PLUS OFFSET



MR-5233

Figure 7-73 BEQ

Operation: PC ← PC + (2 X 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          ;compare A and B
BEQ  C            ;branch if they are equal
```

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.

BPL

BRANCH IF PLUS — 100000 PLUS OFFSET

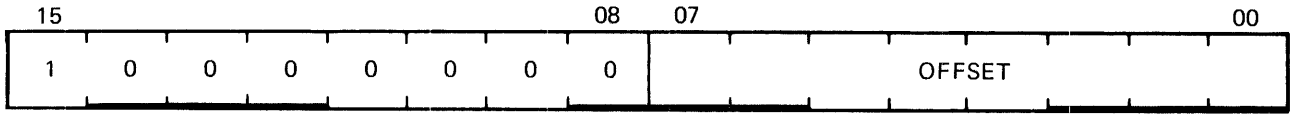


Figure 7-74 BPL

Operation: PC ← PC + (2 X 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.

BMI

BRANCH IF MINUS — 100400 PLUS OFFSET

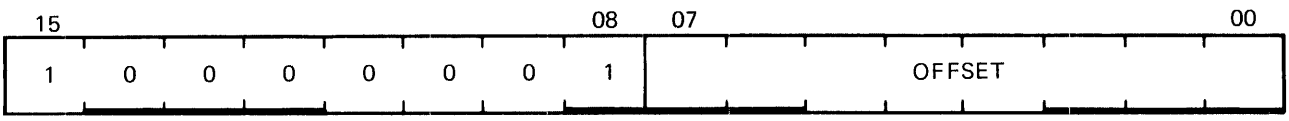


Figure 7-75 BMI

Operation: PC ← PC + (2 X offset) 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.

BVC

BRANCH IF OVERFLOW IS CLEAR

102000 PLUS OFFSET

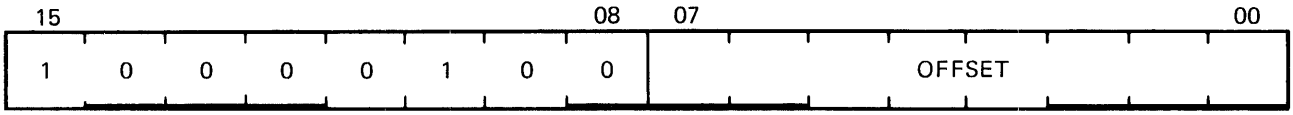


Figure 7-76 BVC

Operation: PC ← PC + (2 X 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

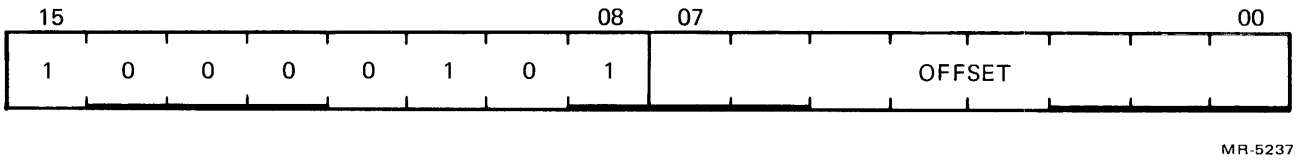


Figure 7-77 BVS

Operation: PC ← PC + (2 X offset) 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.

BCC

BRANCH IF CARRY IS CLEAR

103000 PLUS OFFSET

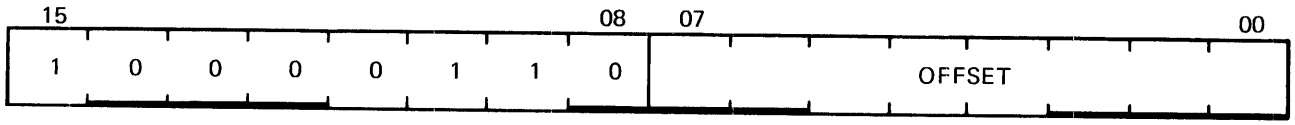


Figure 7-78 BCC

Operation: PC ← PC + (2 X 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

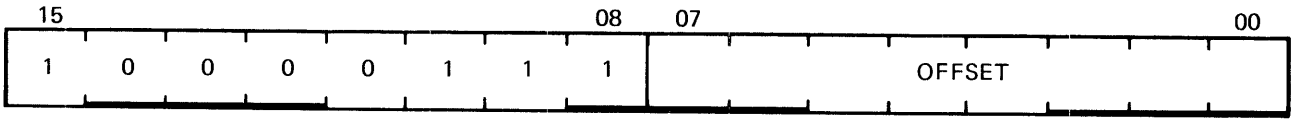


Figure 7-79 BCS

Operation: PC ← PC + (2 X 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.

7.2.5.2 Signed Conditional Branches -- Particular combinations of the condition code bits are tested with the signed conditional branches. These instructions are used to test the results of instructions in which the operands were considered as signed (two's complement) values.

Note that the sense of signed comparisons differs from that of unsigned comparisons in that in signed 16-bit, two's complement arithmetic the sequence of values is as follows.

largest 077777
077776

positive	.
	.
	.
	000001
zero	000000
	177777
	177776
	.
negative	.
	.
	100001
smallest	100000

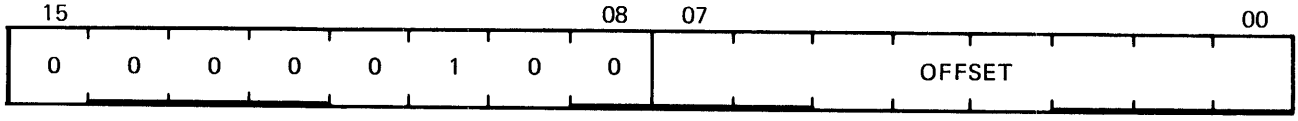
whereas in unsigned 16-bit arithmetic the sequence is considered to be

highest	177777
	.
	.
	.
	.
	.
	.
	000002
	000001
lowest	000000

BGE

BRANCH IF GREATER THAN OR EQUAL (TO ZERO)

002000 PLUS OFFSET



MR-5240

Figure 7-80 BGE

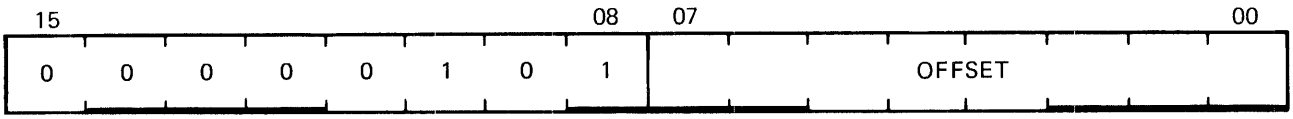
Operation: PC ← PC + (2 X offset) if N $\veebar$ V = 0

Condition Codes: Unaffected

Description: Causes a branch if N and V are either both clear or both set. BGE 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.

BLT

BRANCH IF LESS THAN (ZERO) 002400 PLUS OFFSET



MR-5241

Figure 7-81 BLT

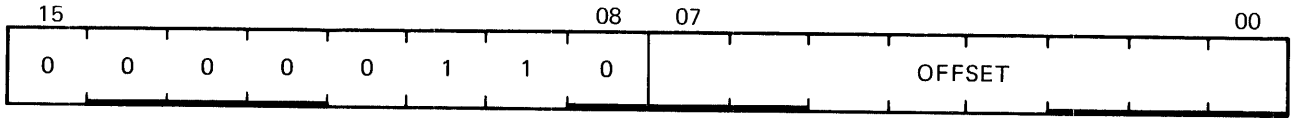
Operation: PC ← PC + (2 X offset) if N $\veebar$ V = 1

Condition Codes: Unaffected

Description: Causes a branch if the "Exclusive Or" of the N and V bits are one. 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).

BGT

BRANCH IF GREATER THAN (ZERO) 003000 PLUS OFFSET



MR-5242

Figure 7-82 BGT

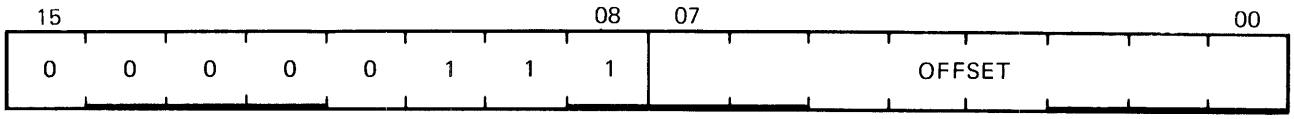
Operation: PC ← PC + (2 X offset) if Z $\vee$ (N $\veebar$ V) = 0

Condition Codes: Unaffected

Description: Operation of BGT is similar to BGE, except BGT will not cause a branch on a zero result.

BLE

BRANCH IF LESS THAN OR EQUAL (TO ZERO) 003400 PLUS OFFSET



MR-5243

Figure 7-83 BLE

Operation: PC ← PC + (2 X offset) if Z $\vee$ (N $\veebar$ V) = 1

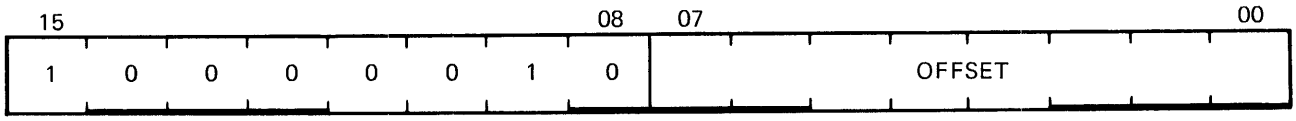
Condition Codes: Unaffected

Description: Operation is similar to BLT but in addition will cause a branch if the result of the previous operation was zero.

7.2.5.3 Unsigned Conditional Branches -- The unsigned conditional Branches provide a means for testing the result of comparison operations in which the operands are considered as unsigned values.

BHI

BRANCH IF HIGHER 101000 PLUS OFFSET



MR-5244

Figure 7-84 BHI

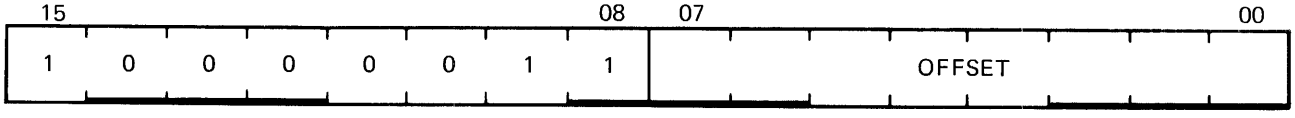
Operation: PC ← PC + (2 X offset) if C = 0 and Z = 0

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 destination.

BLOS

BRANCH IF LOWER OR SAME 101400 PLUS OFFSET



MR-5245

Figure 7-85 BLOS

Operation: PC ← PC + (2 X offset) if C ⋁ 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.

BHIS

BRANCH IF HIGHER OR SAME — 103000 PLUS OFFSET

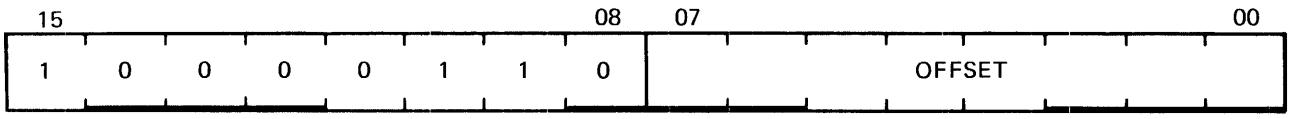


Figure 7-86 BHIS

Operation: PC ← PC + (2 X offset) if C = 0

Condition Codes: Unaffected

Description: BHIS is the same instruction as BCC. This mnemonic included only for convenience.

BLO

BRANCH IF LOWER — 103400 PLUS OFFSET

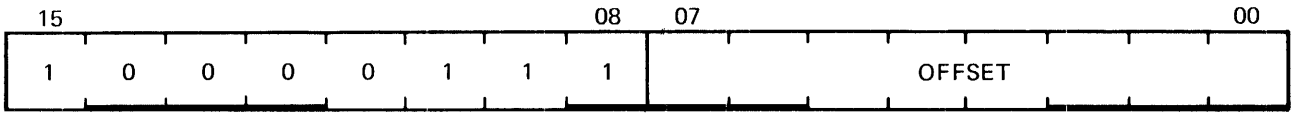


Figure 7-87 BLO

Operation: PC ← PC + (2 X offset) if C = 1

Condition Codes: Unaffected

Description: BLO is same instruction as BCS. This mnemonic is included only for convenience.

7.2.5.4 Jump & Subroutine Instructions -- The subroutine call in the microprocessor provides for automatic nesting 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.

JMP

JUMP 0001DD

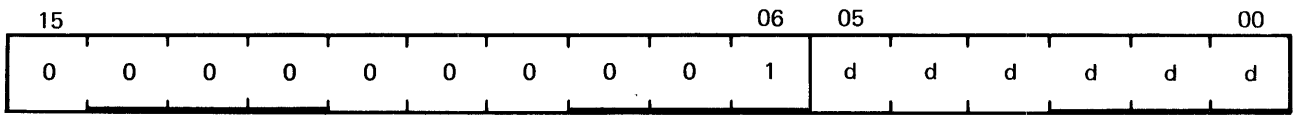


Figure 7-88 JMP

Operation: PC ← (dst)

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 four. (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.
```

JSR

JUMP TO SUBROUTINE

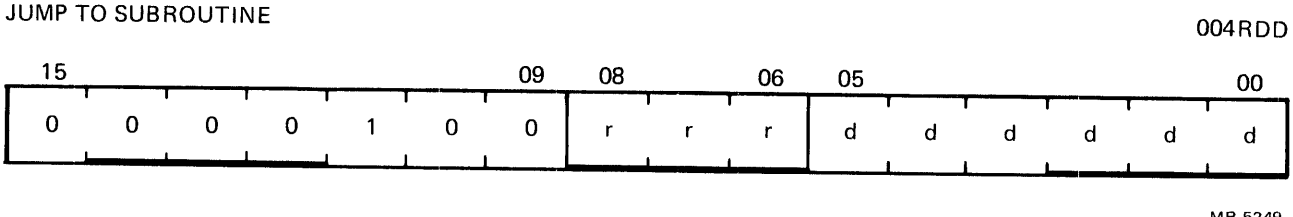


Figure 7-89 JSR

Operation:

(tmp) ← (dst) (tmp is an internal processor register)

↓(SP)←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, and 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.

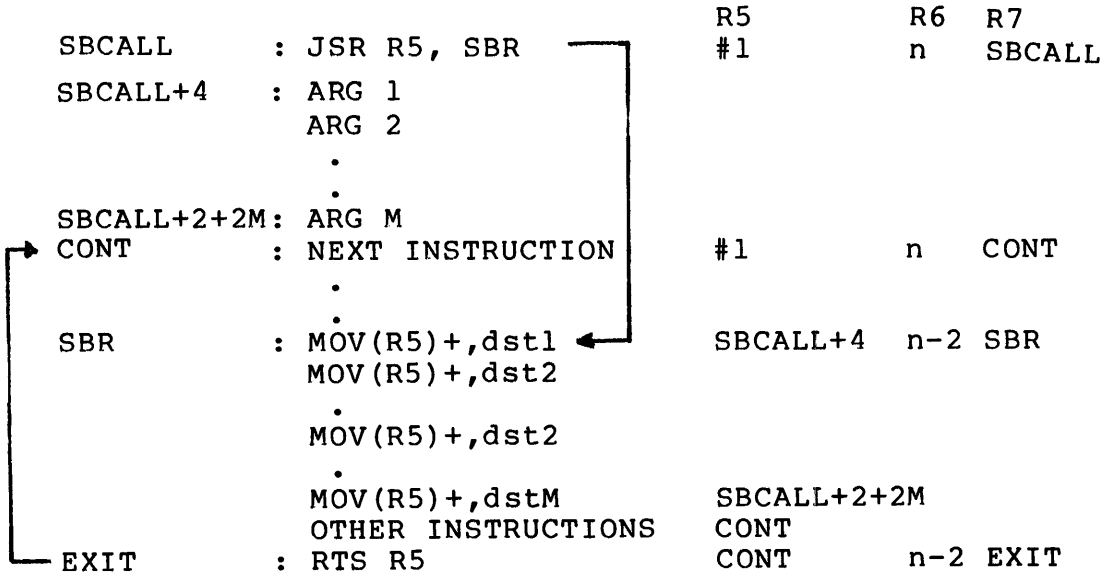
A subroutine called with a JSR reg,dst instruction can access the arguments following the call with either autoincrement addressing, (reg) +, (if arguments are accessed sequentially) or by indexed addressing, X(reg), (if accessed in random order). These addressing modes may also be deferred, @(reg)+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

JSR PC, dst is a special case of the microprocessor 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:



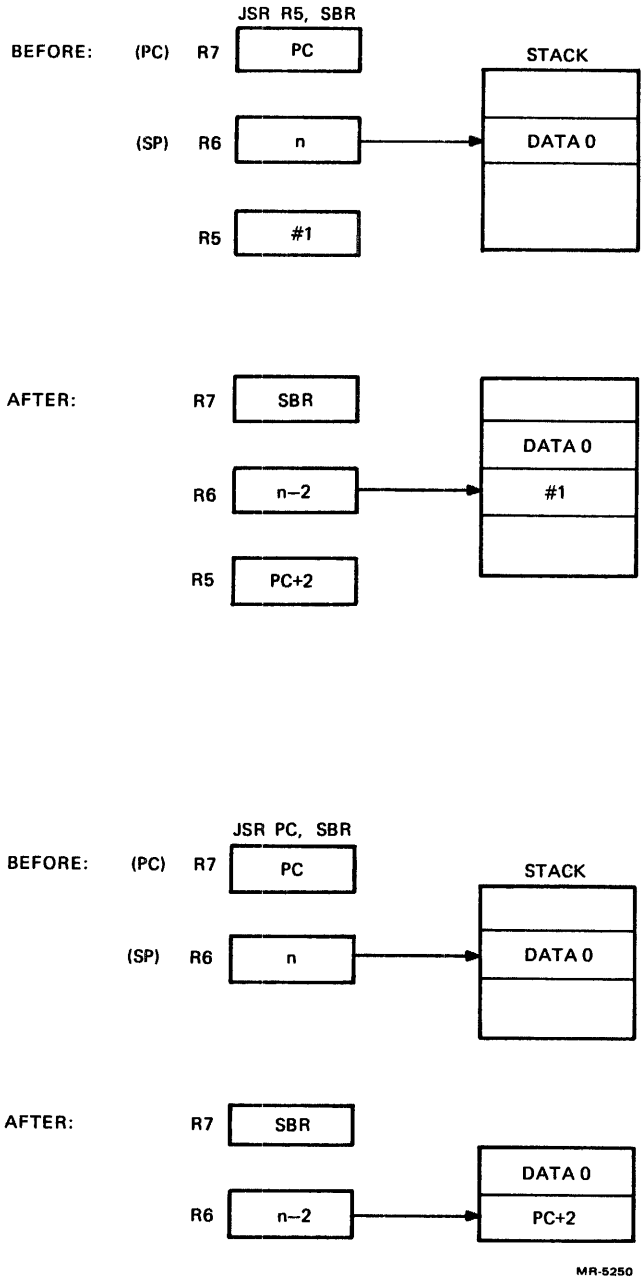


Figure 7-90 JSR Example

RTS

RETURN FROM SUBROUTINE

00020R

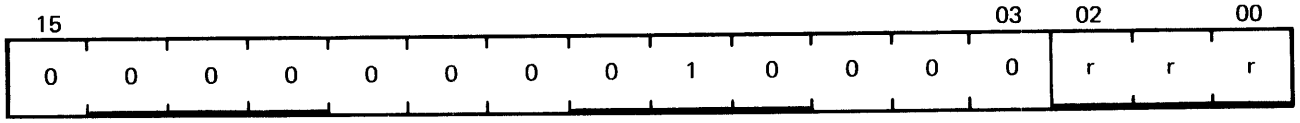


Figure 7-91 RTS

Operation: PC ← (reg)
(reg) ← (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 exits with an 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 exits, with an RTS R5.

Example:

RTS R5

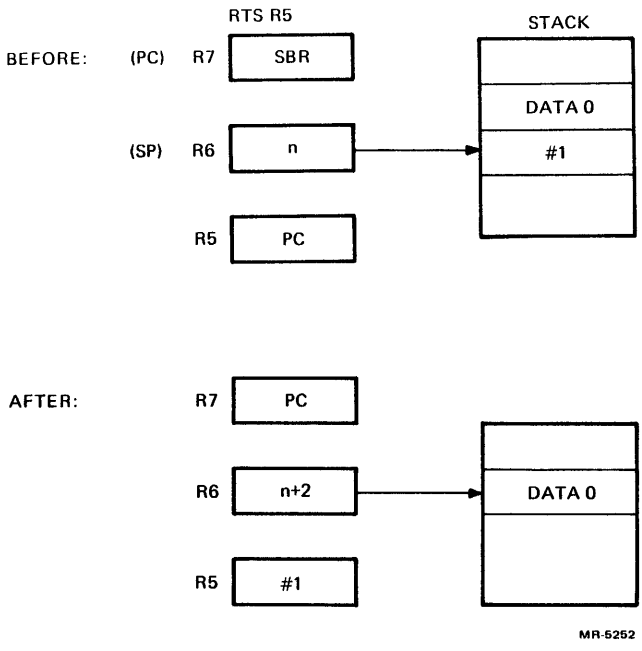


Figure 7-92 RTS Example

SOB

SUBTRACT ONE AND BRANCH (IF ≠ 0) 077RNN

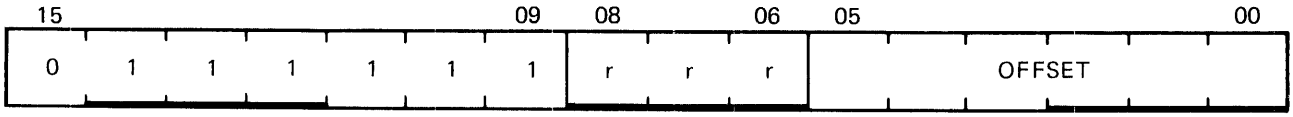


Figure 7-93 SOB

Operation: (R) ← (R) -1; if this result ≠ 0 then PC ← PC -(2 X offset), if (R) = 0 then PC ← PC

Condition Codes: Unaffected

Description: The register is decremented. If it is not equal to zero, 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 cannot be used to transfer control in the forward direction.

7.2.5.5 Traps -- 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 Program Counter (PC) and processor Status Word (PS) are pushed onto the processor stack and replaced by the contents of a two-word trap vector containing a new PC and new PS. 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.

EMT

EMULATOR TRAP 104000–104377

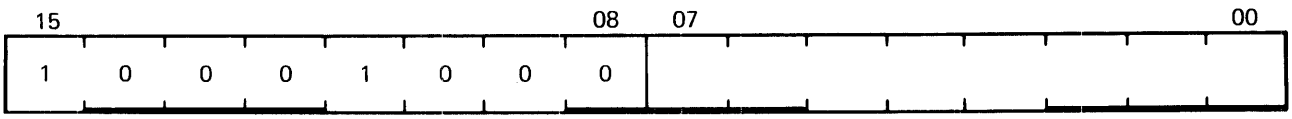


Figure 7-94 EMT

Operation:

↓(SP)←PS
↓(SP)←PC
PC←(30)
PS←(32)

Condition Codes:

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

Caution: EMT is used frequently by DEC system software and is therefore not recommended for general use.

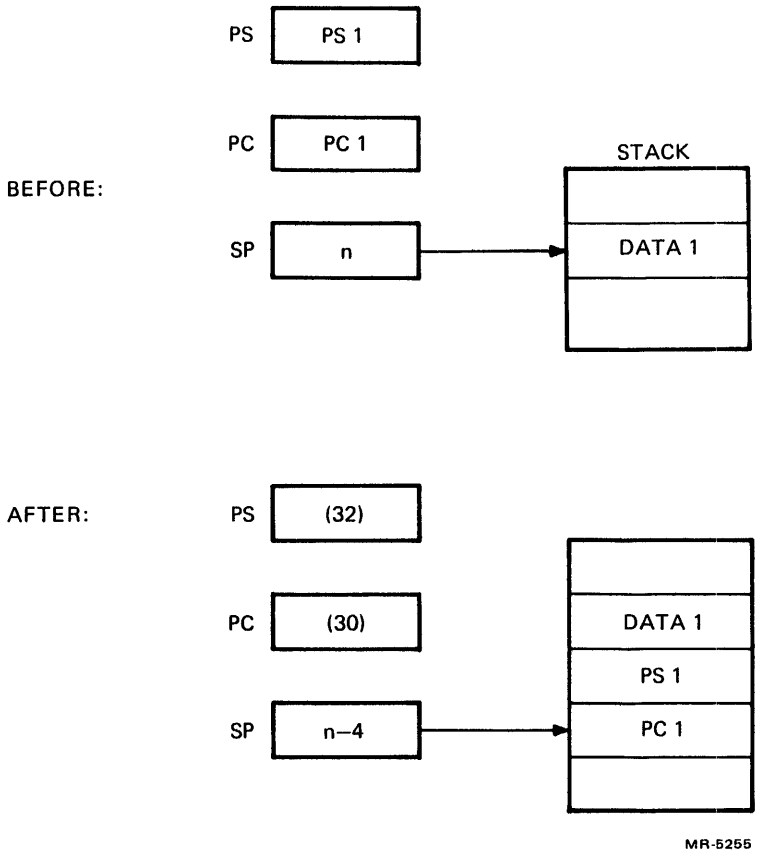


Figure 7-95 EMT Example

TRAP

TRAP

104400–104777

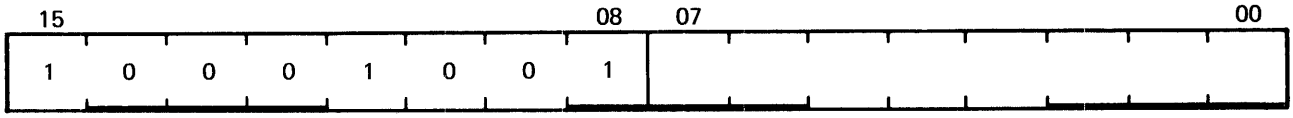


Figure 7-96 TRAP

Operation:

↓(SP)←PS
↓(SP)←PC
PC←(34)
PS←(36)

Condition Codes:

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

BPT

BREAKPOINT TRAP

000003

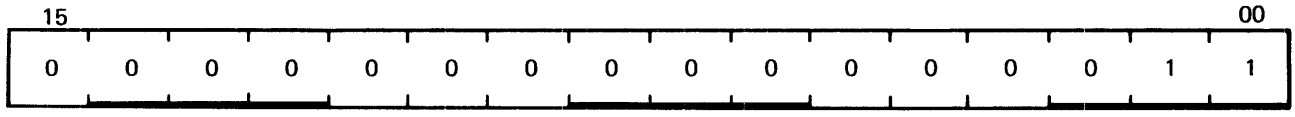


Figure 7-97 BPT

Operation:

↓(SP)←PS
↓(SP)←PC
PC←(14)
PS←(16)

Condition Codes:

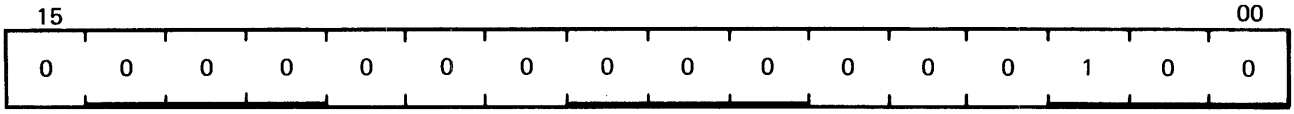
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

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.)

IOT

INPUT/OUTPUT TRAP 000004



MR-5258

Figure 7-98 IOT

Operation:
$\downarrow$(SP)$\leftarrow$PS
$\downarrow$(SP)$\leftarrow$PC
PC$\leftarrow$(20)
PS$\leftarrow$(22)

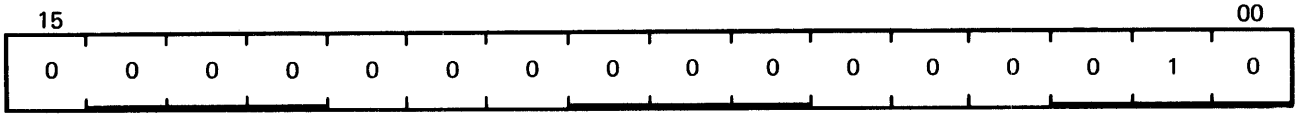
Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description: Performs a trap sequence with a trap vector address of 20.

(No information is transmitted in the low byte.)

RTI

RETURN FROM INTERRUPT 000002



MR-5259

Figure 7-99 RTI

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.

RTT

RETURN FROM INTERRUPT 000006

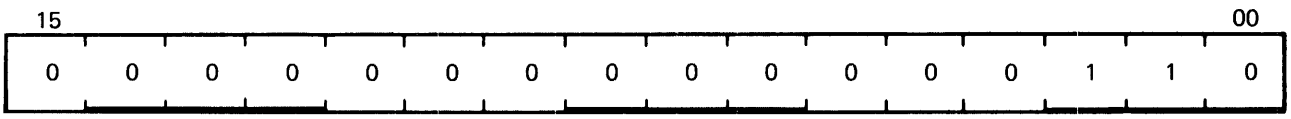


Figure 7-100 RTT

Operation: PC←(SP)↑
PS←(SP)↑

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.

7.2.5.6 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 four. Reserved instructions trap to vector address 10.

7.2.5.7 Halt Interrupt -- This is caused by the -HALT line. The -HALT interrupt saves the PC and PS and goes to the restart address with PS = 340.

7.2.5.8 Trace Trap -- Trace Trap is enabled by bit four 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. Note that the trace trap is a system debugging aid and is transparent to the general programmer.

7.2.5.9 Power Failure Interrupt -- Occurs when -PF line is asserted. Vector for power failure is location 24 and 26. Trap will occur if an RTI instruction is executed in power fail service routine.

7.2.5.10 Interrupts -- Refer to Table 5-3.

NOTE
Bit four of the PS can only be set indirectly by executing an RTI or RTT instruction with the desired PS on the stack.

7.2.5.11 Special Cases T-bit -- The following are special cases of the T-bit.

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

1. 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.

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

3. 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 exits 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.

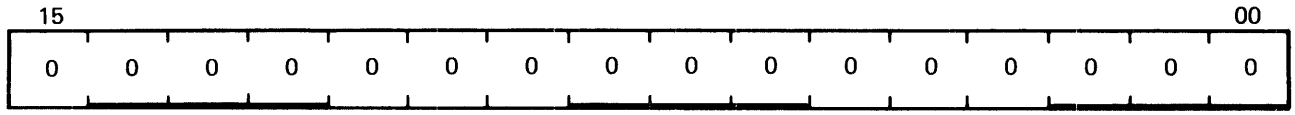
4. Interrupt Trap Priorities -- In case of multiple trap and interrupt conditions, occurring simultaneously, the following order of priorities is observed (from high to low):

Halt Line
Power Fail Trap
Trace Trap
Internal Interrupt Request
External Interrupt Request
Instruction Traps

7.2.6 Miscellaneous Instructions

HALT

HALT 000000



MR-5261

Figure 7-101 HALT

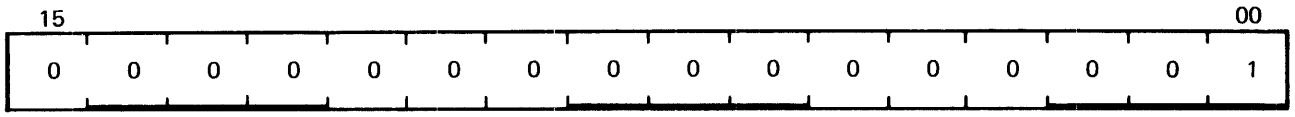
Operation:
$\Downarrow$(SP)$\leftarrow$PS
$\Downarrow$(SP)$\leftarrow$PC
PC$\leftarrow$restart address
PS$\leftarrow$340

Condition Codes: Unaffected

Description: The processor goes to the restart address after placing the current PC and PS on the stack. PS is initialized to 340.

WAIT

WAIT FOR INTERRUPT 000001



MR-5262

Figure 7-102 WAIT

Condition Codes: Unaffected

Description: 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.

RESET

RESET EXTERNAL BUS

000005

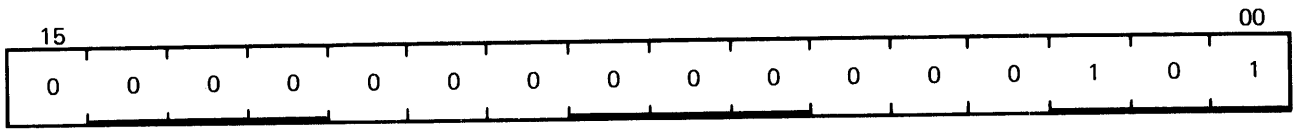


Figure 7-103 RESET

Condition Codes: Unaffected

Description: The -BCLR line is asserted and the mode register is loaded. -BCLR is negated and an ASPI transaction takes place. PC, PS, and R0--R5 are not affected.

MFPT

MOVE FROM PROCESSOR TYPE WORD

000007

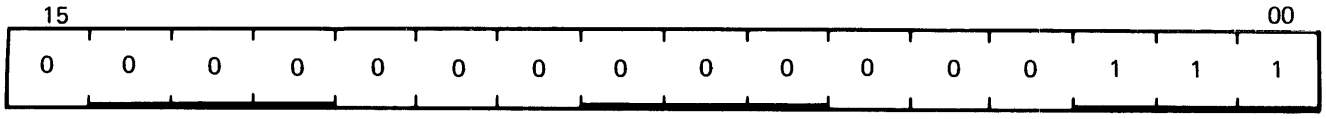


Figure 7-104 MFPT

Operation: R0←4

Condition Codes: Unaffected

Description: The number four is placed in R0 indicating to the system software that the processor type is SBC-11/21.

7.2.7 Condition Code Operators

CLN SEN
CLZ SEZ
CLV SEV
CLC SEC
CCC SCC

CONDITION CODE OPERATORS

0002XX

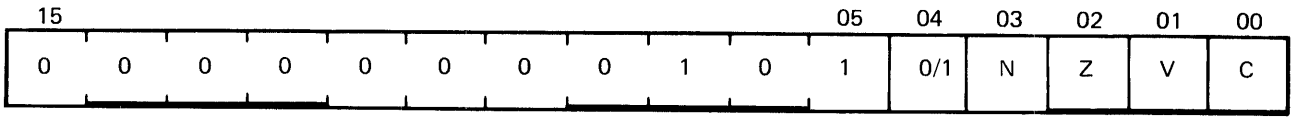


Figure 7-105 Condition Code Operators

Description: Set and clear conditon 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 four, the set/clear bit of the operator; i.e., set the bit specified by bit zero, one, two, or three, if bit four is a one. Clear corresponding bits if bit 4 = 0.

Mnemonic	Operation	OP Code
CLC	Clear C	000241
CLV	Clear V	000242
CLZ	Clear Z	000244
CLN	Clear N	000250
SEC	Set C	000261
SEV	Set V	000262
SEZ	Set Z	000264
SEN	Set N	000270
SCC	Set all CCs	000277
CCC	Clear all CCs	000257
	Clear V and C*	000243
NOP	No Operation	000240

*Combinations of the above set or clear operations may be ORed together to form combined instructions. Clear V and C represents CLC (241) ORed with CLV (code 242).

CHAPTER 8
THEORY OF OPERATION

8.0 GENERAL

This chapter provides a detailed explanation of the SBC-11/21 hardware operation from the point of view of the logic designer. It will be useful for troubleshooting the device to the chip level.

> **NOTE**
>
> The negated or inverse signal is designated by the "-" character. For example, RAS is normally low and asserted high when activated and the -RAS would be normally high and asserted low when activated. This convention is used throughout this chapter. The LSI-11 bus signals will remain consistent with the standard bus conventions.

The SBC-11/21 functional block diagram (Figure 8-1) provides an overview of the module functions and how they are related. The main components of the single board computer are shown on sheet 1. The single board computer is comprised of the microprocessor interconnected to the Serial Line Units, RAM memory, ROM memory and the Parallel I/O Interface via the on-board TDAL bus. The TDAL bus can access the LSI-11 bus (BDAL bus) by the Bus Control function shown by broken lines and is for reference only. Also on sheet 1 is the Address bus, the Memory Address Decode function and the Interrupt Control function.

The microprocessor support functions and the LSI-11 interfacing functions are described on sheet 2. The microprocessor is shown by broken lines for reference only. The Power Up, Clock, Clock Control, Ready, DMA and Halt functions are used by the microprocessor. The IAK Data In, Sync, Read/Write, Reply Timeout and Bus Control functions are used to interface the LSI-11 bus to the microprocessor.

The functional descriptions used in this chapter will first define the microprocessor and the input/output signals associated with its operation. The support functions, the LSI-11 bus interface functions and the remaining single board computer devices are also described in detail.

8.1 MICROPROCESSOR

The microprocessor is contained within a 40-pin LSI chip shown in Figure 8-2. There are eight 16-bit general purpose registers (R0--R7) of which R6 operates as the Stack Pointer (SP) and R7 operates as the microprocessor Program Counter (PC). A special purpose status register contains the current Processor Status Word

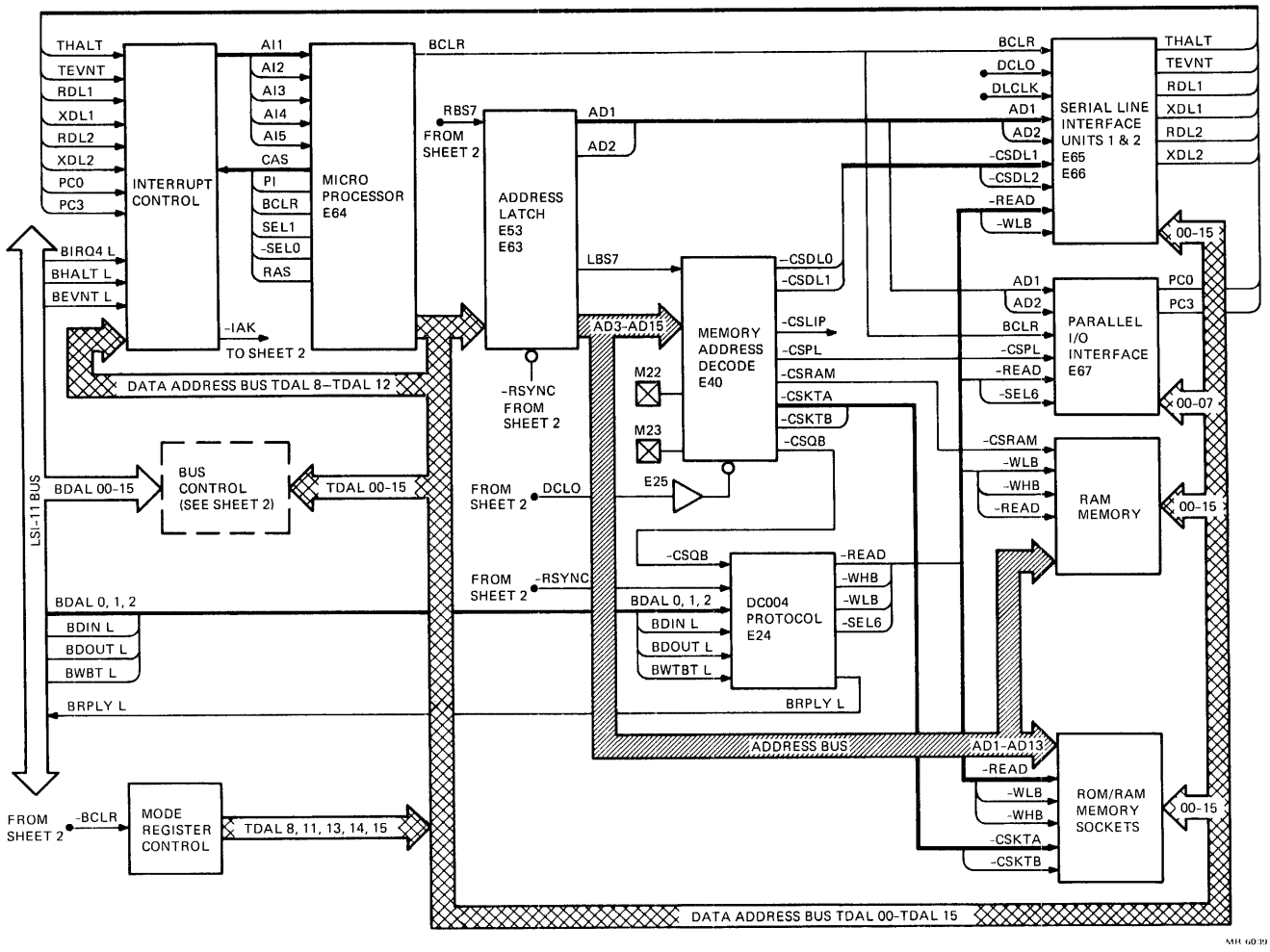


Figure 8-1 SBC-11/21 Functional Block Diagram
(Sheet 1 of 2)

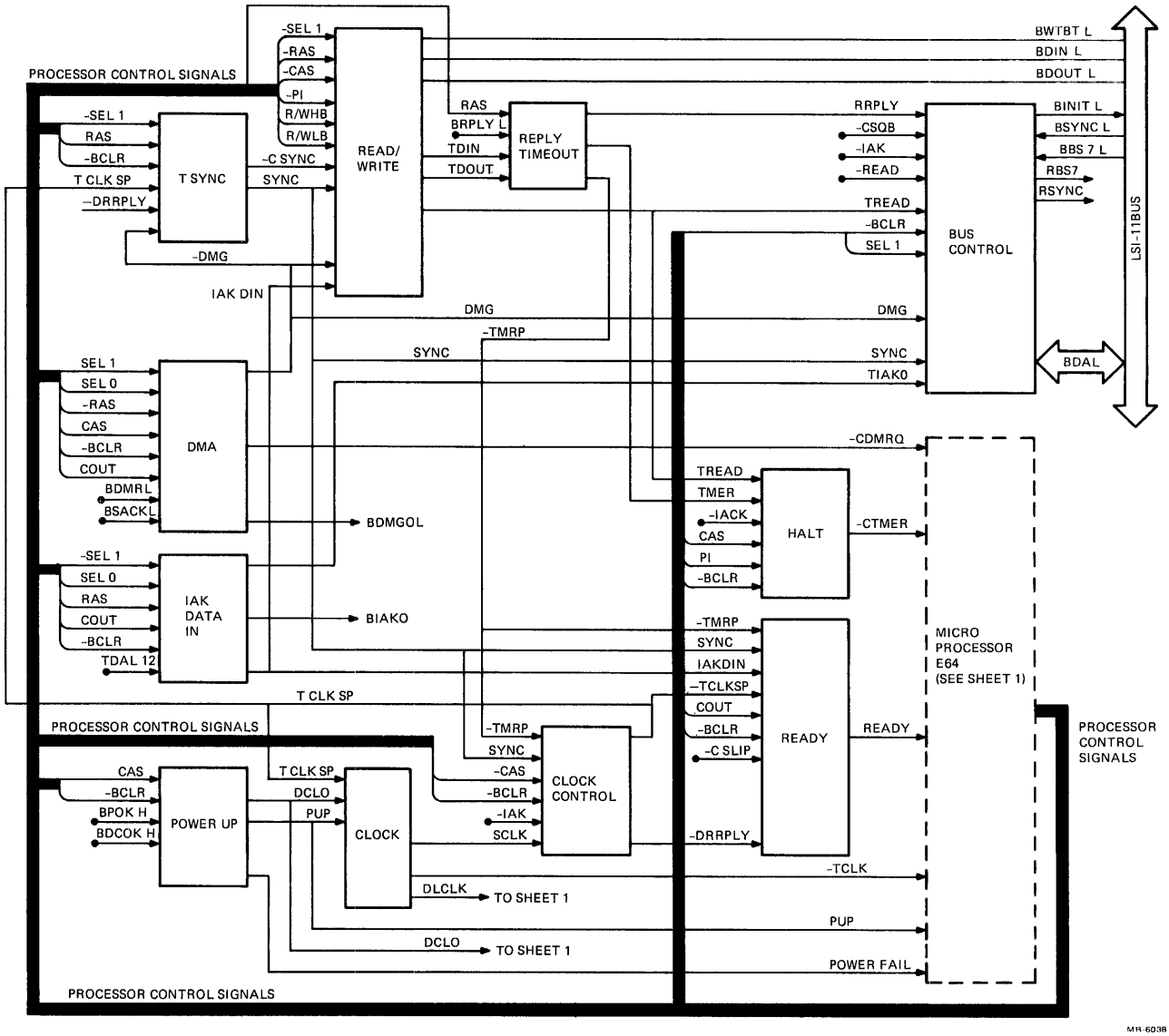


Figure 8-1 SBC-11/21 Functional Block Diagram
(Sheet 2 of 2)

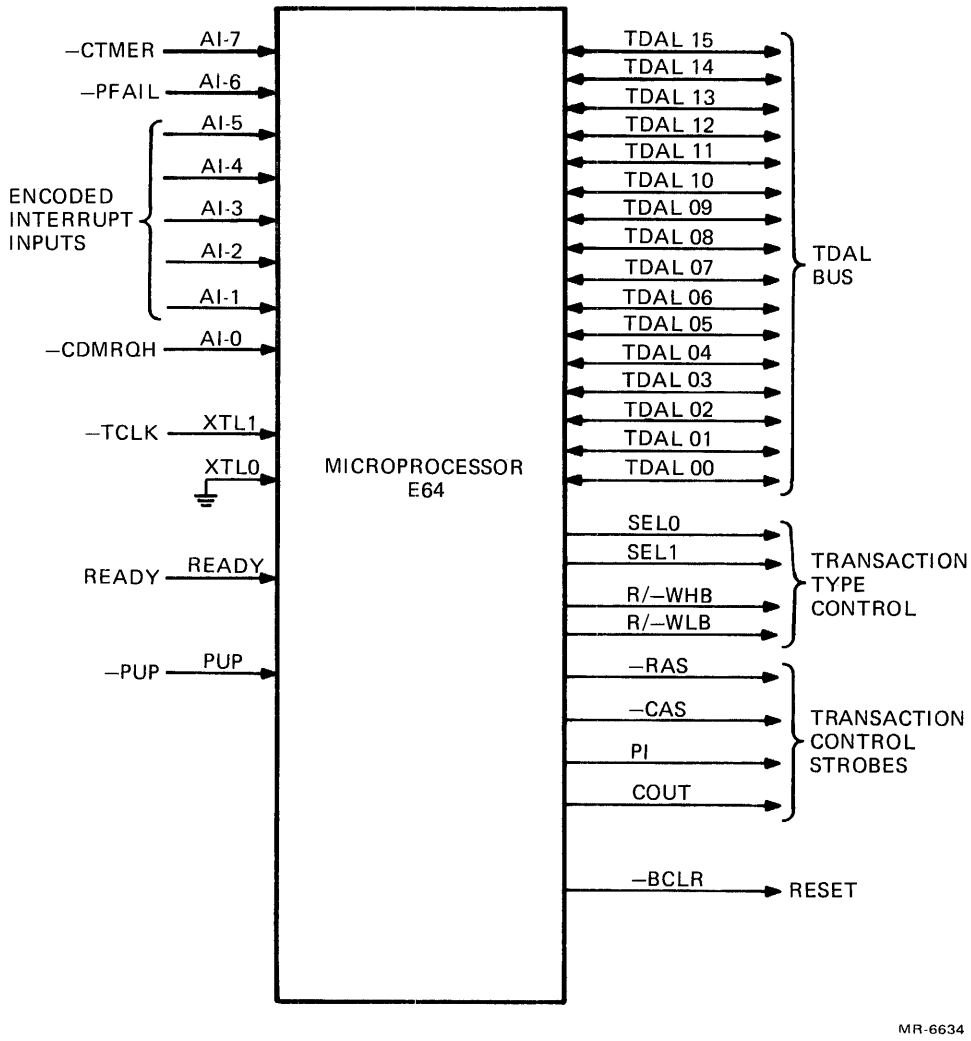


Figure 8-2 SBC-11/21 Microprocessor

(PSW). The operating characteristics of the microprocessor are affected by the mode register. It is discussed in detail in section 8.2.

8.1.1 Microprocessor Initialization

The microprocessor will initialize the SBC-11/21 module during the power up sequence or whenever the RESET instruction is executed.

8.1.1.1 RESET -- The RESET instruction asserts the -BCLR output and this clears or resets the control logic of the module to an initial state. The microprocessor loads the Mode Register from the TDAL bus with the Mode Register Control data. The LSI-11 bus transceivers are disabled when -BCLR is asserted. The RCVIE bit of the RCSRs and the XMITIE, MAINT and XMITBRK bits of the XCSRs are reset in the Serial Line Units (SLU). The port C buffer output

lines of the Parallel I/O are set high. Also, if port A and port B buffers are output to the connectors, they are set high. The LED is turned off during reset. The -BCLR output is then negated and an Assert Priority In (ASPI) transaction is performed to service any Interrupts or DMA requests. The RESET instruction does not disturb the PSW or any internal registers.

8.1.1.2 **Power Up** -- The Power Up (PUP) input goes from low to high when the 5.0 V power is first applied and this initiates the power up sequence. The -BCLR output is asserted. The module is cleared and reset in the same manner described for the RESET instruction except the Serial Line Units (SLU) registers are completely reset. After a delay, the BDCOK and BPOK are asserted, PUP is negated, and the -BCLR output goes high. The microprocessor then performs ten bus NOP transactions. The processor loads the starting address into the Program Counter (R7), location 376 into the Stack Pointer (R6) and the Processor Status Word is set to 340. An Assert Priority In (ASPI) transaction is performed to service any interrupts or DMA requests before the first instruction is fetched.

The PUP input normally remains low for all operations. If PUP is asserted high, the present transaction is terminated and the internal registers go to an undetermined state. The TDAL bus, the interrupt inputs and the microprocessor control signals will all go to an initial reset state.

8.1.2 Clock Input

The -TCLK input is a 4.9152 MHz clock that comes from the 19.6608 MHz crystal oscillator. This clock input is used for the internal time base of the microprocessor and the source of the clock output (COUT). COUT is pulsed once for every microcycle and a microcycle can represent either three or four -TCLK input pulses depending on the type of transaction. The microprocessor will halt or stop whenever the -TCLK input is disabled.

8.1.3 Ready Input

The READY input is normally high and will not interfere with the microprocessor transactions in any way. However, when this input is held low, a single microcycle slip occurs during every transaction. When READY is clocked with COUT, while RAS is asserted, then the microprocessor will slip a microcycle every time the input is pulsed. This allows the microprocessor to be placed in an idle or wait state until a peripheral device has either received or asserted data on the bus.

8.1.4 Microprocessor Control Signals

The microprocessor controls the functions of the SBC-11/21 through the use of eight microprocessor control signals. These signals are listed below with a description of the functions they perform. The RAS, CAS, PI, COUT, and BCLR are transaction control strobes used for logic transitions. The R/-WLB, R/-WHB, SEL0 and SEL1 are steady state logic signals used as transaction type control signals.

8.1.4.1 Row Address Strobe (RAS) -- The leading edge of the signal is used to acknowledge that the address is stable on the TDAL bus during Read/Write and Fetch transactions. During interrupt transactions the leading edge strobes the interrupt acknowledge data onto the TDAL 12--08 bus lines.

8.1.4.2 Column Address Strobe (CAS) -- The trailing edge of the signal is used to acknowledge that data on the TDAL bus lines during Read and Fetch transactions was read by the microprocessor. For Write transactions it is used to acknowledge that the microprocessor data will be removed after a specified time.

The leading edge is used to request that READ data be placed on the TDAL bus, and to strobe interrupt requests into latches which are read during the assertion of PI.

8.1.4.3 Priority In (PI) -- The leading edge of the signal is used to acknowledge that data on the TDAL bus lines during Write transactions is stable. The leading edge is also used to enable the microprocessor to read the interrupt inputs AI-0 to AI-7 and to initiate IAK, RESTART, POWER FAIL, or DMA transactions.

8.1.4.4 Read/Write (R/-WHB and R/-WLB) -- These two signals control the Read/Write and Fetch transactions by enabling the TDIN, TDOUT and TWTBT control signals. For Read and Fetch transactions, both are asserted high and enable the TDIN control circuits. During Write transactions the TDOUT and TWTBT control circuits are enabled when either or both signals are asserted low. If only one is asserted low the TWTBT control circuits are enabled by the leading edge of CAS and a Write byte transaction occurs for either high byte or low byte.

8.1.4.5 Select Output Flags (SEL0 and SEL1) -- These two signals indicate the transaction being performed. When both are low, a Read, Write, ASPI or NOP transaction is selected and when both are high, a DMA transaction is selected. When SEL1 is low and SEL0 high the Fetch transaction is selected and when SEL1 is high and SEL0 is low an IAK transaction is being performed.

8.1.4.6 Bus Clear (BCLR) -- This signal is used to reset the control logic and generate BINIT. It is only asserted during the Power Up sequence and during the execution of a RESET instruction.

8.1.4.7 Clock Out (COUT) -- This signal is asserted once for every microcycle and is used to time the microprocessor transactions.

8.1.5 Microprocessor Transactions

The microprocessor performs six types of transactions to support the instruction set, direct memory access, and the interrupt structure. The types of transactions are Fetch Read, Write, DMA, IAK, ASPI, and Bus NOP. A normal Fetch/Read or IAK transaction

requires either one or two microcycles and extended transactions can take as many as required before a timeout occurs. The COUT signal is asserted once for every microcycle. The transactions are used to transfer information and data via the TDAL bus which interconnects all local devices and connects them to the LSI-11 bus interface. The operation of the transactions is described below.

8.1.5.1 Fetch/Read -- This transaction is used to either fetch an instruction or read data for the microprocessor. The data may originate from the on-board memory, I/O device, or from the LSI-11 bus. The microprocessor control signals for the transaction are described by Figure 8-3. The R/-WLB and R/-WHB control signals are asserted. The SEL0 output is high and the SEL1 output is low for the Fetch transaction and both of these outputs are low for the Read transaction.

The following sequence of events then takes place.

- The microprocessor places the address onto the TDAL bus when the transaction is initiated and it is latched into the Memory Address circuits by the assertion of SYNC.
- The data is received on the TDAL bus after RRPLY is received. The microprocessor accepts the data and negates TDIN.
- Interrupt and DMA requests are latched by CAS, set up while PI is asserted, and latched into the microprocessor when PI is negated.

NOTE

A write transaction is always preceded by a read transaction except when the microprocessor pushes onto the stack. Therefore, each write consists of at least four microcycles, assert address, read data, assert address, and write data.

8.1.5.2 Write -- This transaction is used to write data from the microprocessor to memory, a local I/O device or an LSI-11 bus peripheral device. The microprocessor control signals for the transaction are described by Figure 8-4. The R/-WLB and R/-WHB control signals are asserted low when writing a word and for writing a byte, either the high or byte signal is asserted. Both SEL0 and SEL1 control signals are negated.

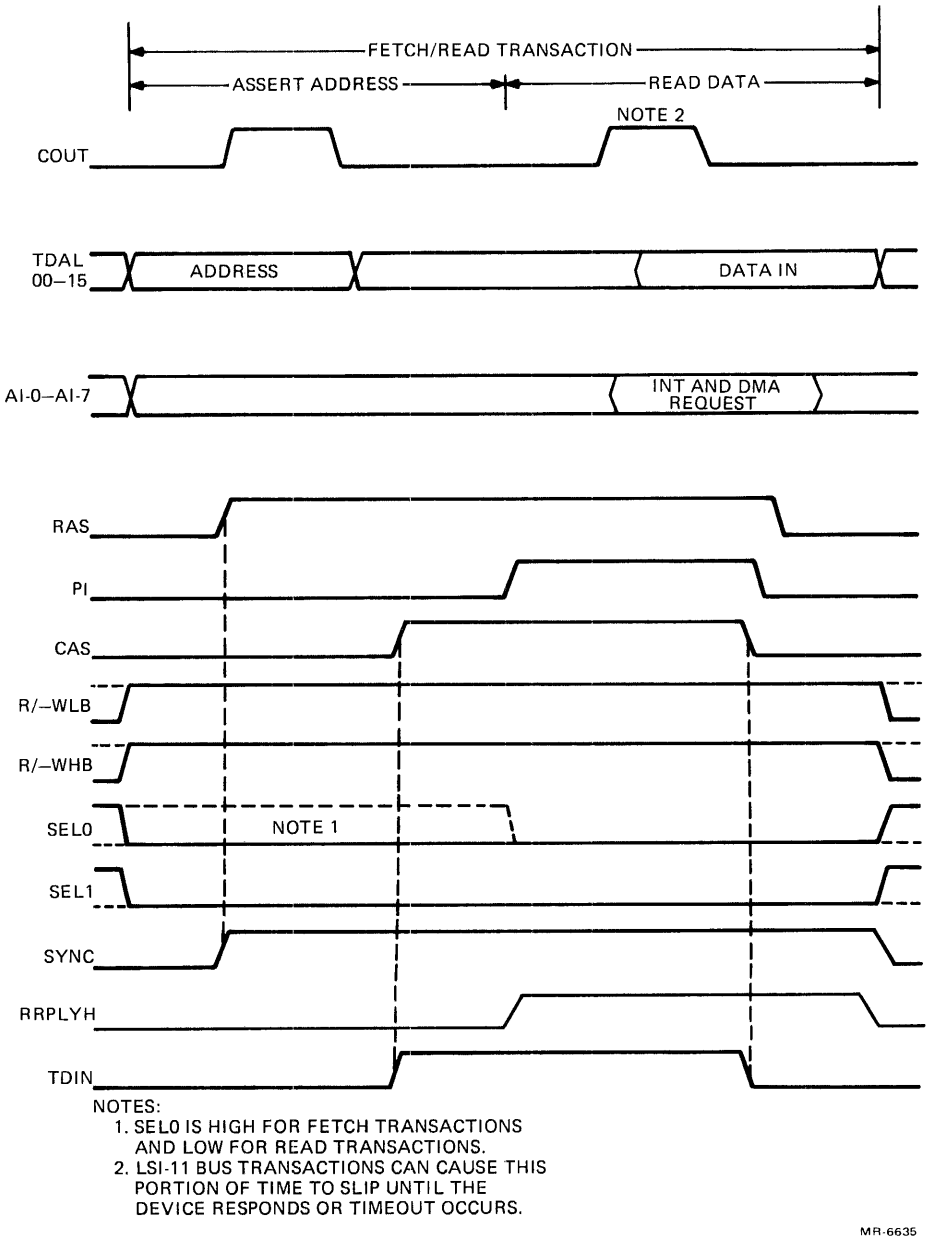


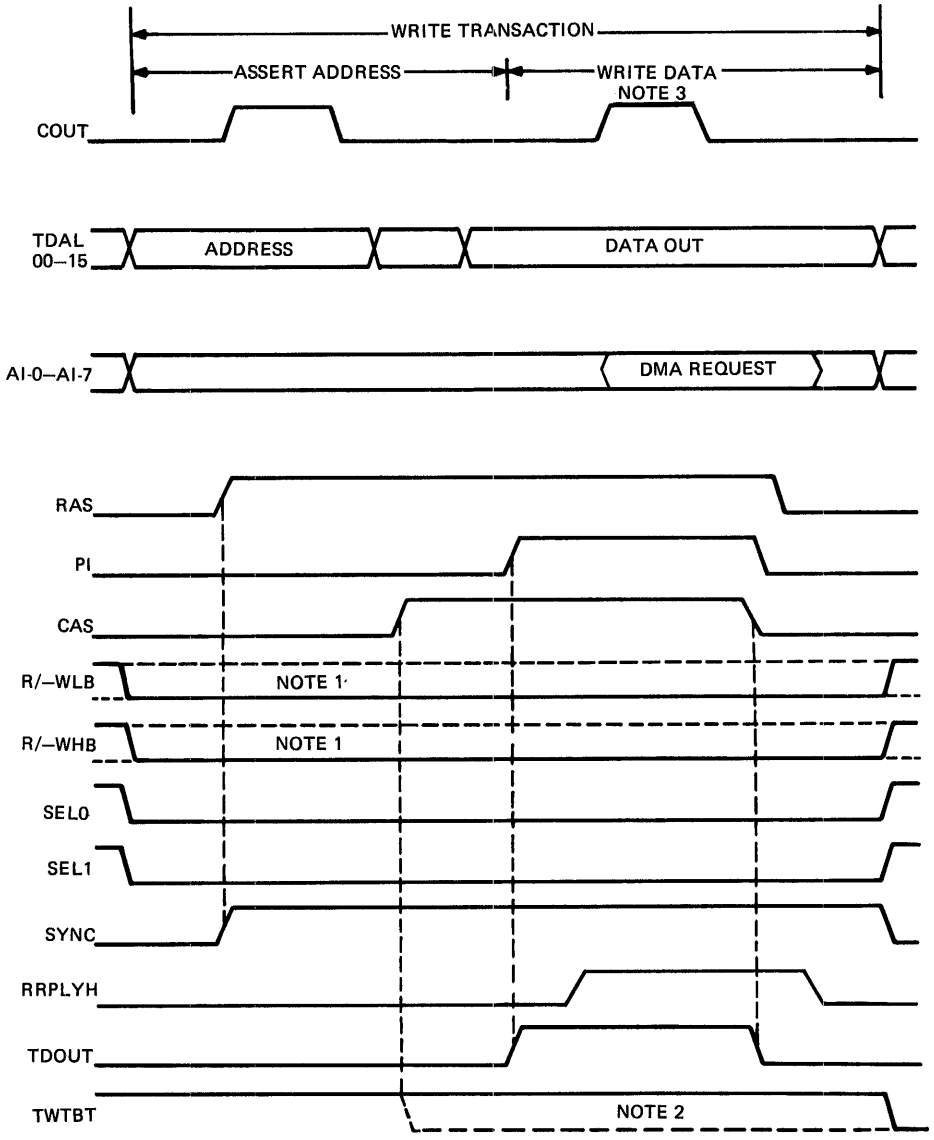
Figure 8-3 Fetch/Read Transaction

The following sequence of events take place.

- The microprocessor places the address on the TDAL bus and the state of the R/W lines causes TWTBT to be asserted. The address is latched into the Memory Address circuits by the assertion of SYNC.

- When CAS is asserted, TWTBT is negated for word transactions and left asserted for byte transactions.

- The data is placed on the TDAL bus before TDOUT is asserted and is written into the addressed location when TDOUT is negated.
- When the addressed device negates BRPLY, the SYNC and TWTBT signals are cleared.
- The DMA requests are detected while PI is asserted and latched into the microprocessor when PI is negated. No other interrupts are read by the microprocessor during write transactions.



NOTES:
1. R/–WHB OR R/–WLB CAN BE HIGH WHEN PERFORMING A WRITE BYTE TRANSACTION.
2. TWTBT IS LOW FOR WORD TRANSACTIONS.
3. LSI-11 BUS TRANSACTIONS CAN CAUSE THIS PORTION OF TIME TO SLIP UNTIL THE DEVICE RESPONDS OR TIMEOUT OCCURS.

MR-6636

Figure 8-4 Write Transaction

8.1.5.3 IAK -- The interrupt request was detected during a previous read transaction and the microprocessor initiates an IAK transaction as described by Figure 8-5. The R/-WHB and R/-WLB control signals are asserted high and CAS, PI, and SEL0 are asserted low for the transaction. The TDAL bits 12--08 represent the acknowledged input and are used to reset the interrupt request. For local interrupts TDAL bits 07--00 are ignored because the vector address is in the microprocessor. For LSI-11 bus interrupts, the vector address is read from the bus using TDAL bits 07--02. TDAL bus bit 12 is set low for this type of IAK and directs the control logic to initiate an LSI-11 bus IAK transaction. The TDIN signal is asserted for the transaction and the TIAK0 output acknowledges the interrupt. The requesting device then places the vector address on the low byte of the bus and asserts BRPLY. The microprocessor stops slipping microcycles, negates TDIN, and accepts the vector. It then negates TIAK0 on the trailing edge of RAS and continues to the next transaction.

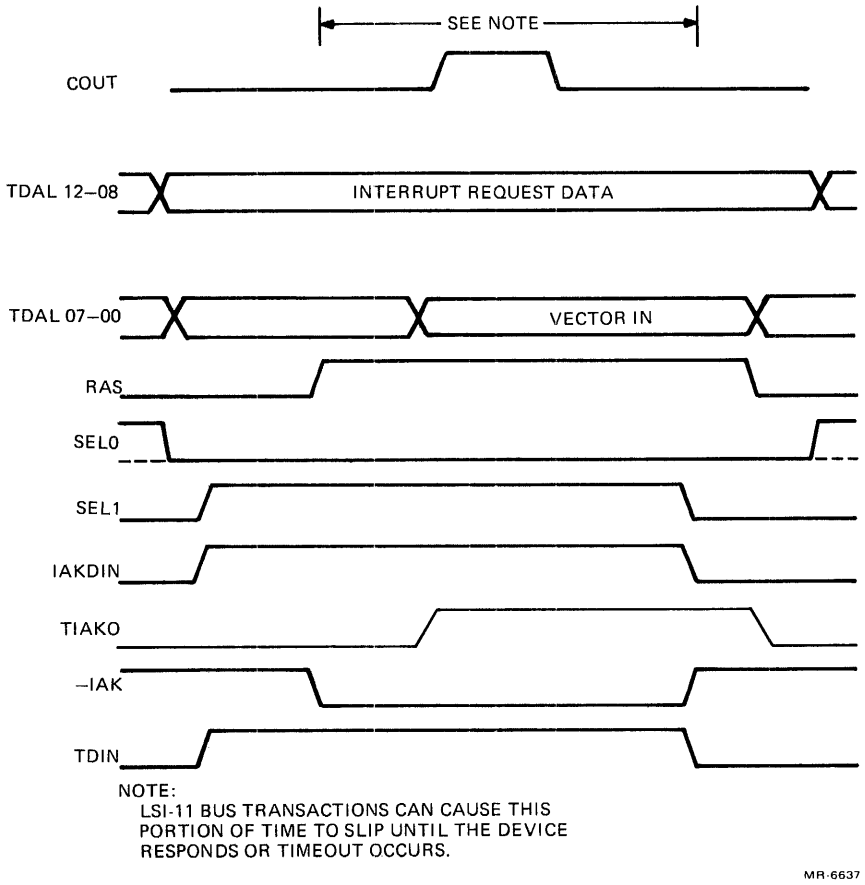


Figure 8-5 IACK Transaction

8.1.5.4 DMA -- The DMA request was read during a previous transaction. The microprocessor will acknowledge the request by tri-stating the TDAL bus as shown in Figure 8-6. The SEL0 and SEL1 outputs are asserted to indicate that the bus mastership has been relinquished. The transaction will continue with no interruptions until the DMA transfer is completed. The microprocessor will then negate SEL1 control output to indicate that it is resuming bus mastership, followed by the negation of SEL0 if the next transaction is not a fetch.

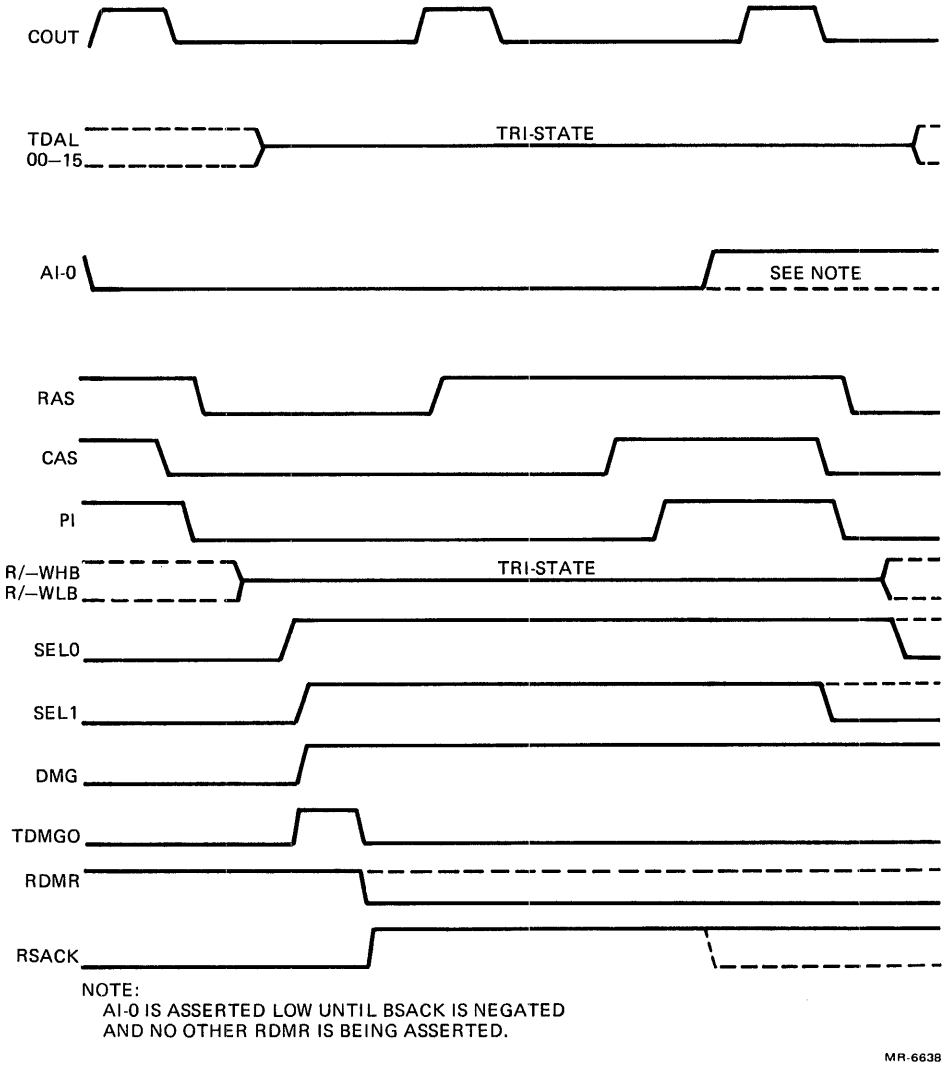


Figure 8-6 DMA Transaction

8.1.5.5 ASPI -- The Assert Priority In (ASPI) transaction is used by the RESET and WAIT instructions or the power up sequence as shown described in Figure 8-7. The CAS and PI outputs are asserted to allow the microprocessor to recognize and latch any pending interrupts or DMA requests. The R/-WHB and R/-WLB outputs are asserted and the SEL0, SEL1, and RAS outputs are negated for the transaction.

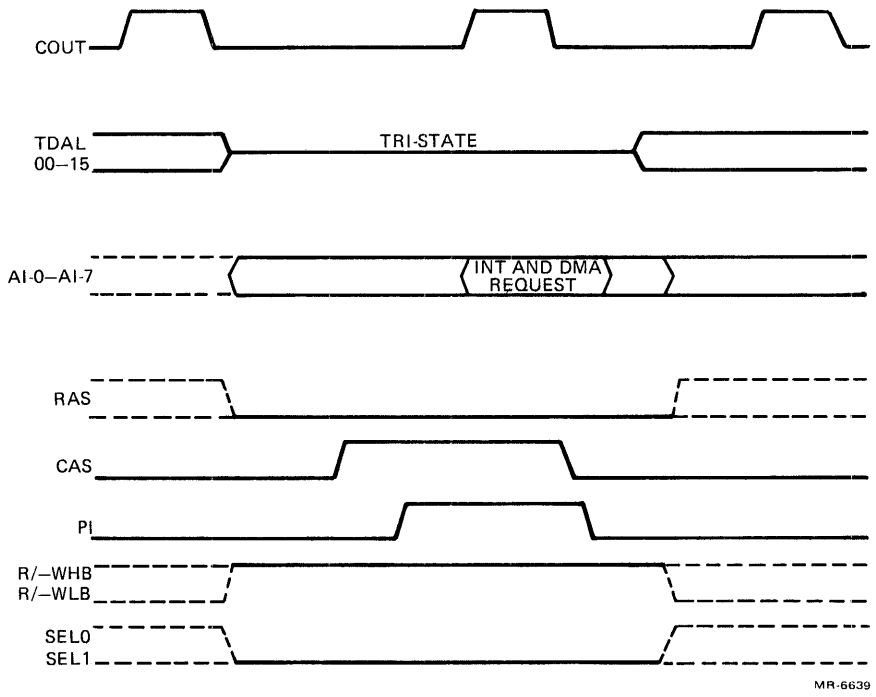


Figure 8-7 ASPI Transaction

8.1.5.6 NOP -- The Bus NOP transaction performs no operation and is used during the power-up sequence or if the programmer intentionally introduces a delay into the program. The AI-0 through AI-7 inputs are tri-stated to prevent interrupts. The R/-WHB and R/-WLB outputs are asserted and the SEL0 and SEL1 outputs are taken low. The RAS, CAS, and PI control strobes are inhibited during the transaction as shown in Figure 8-8.

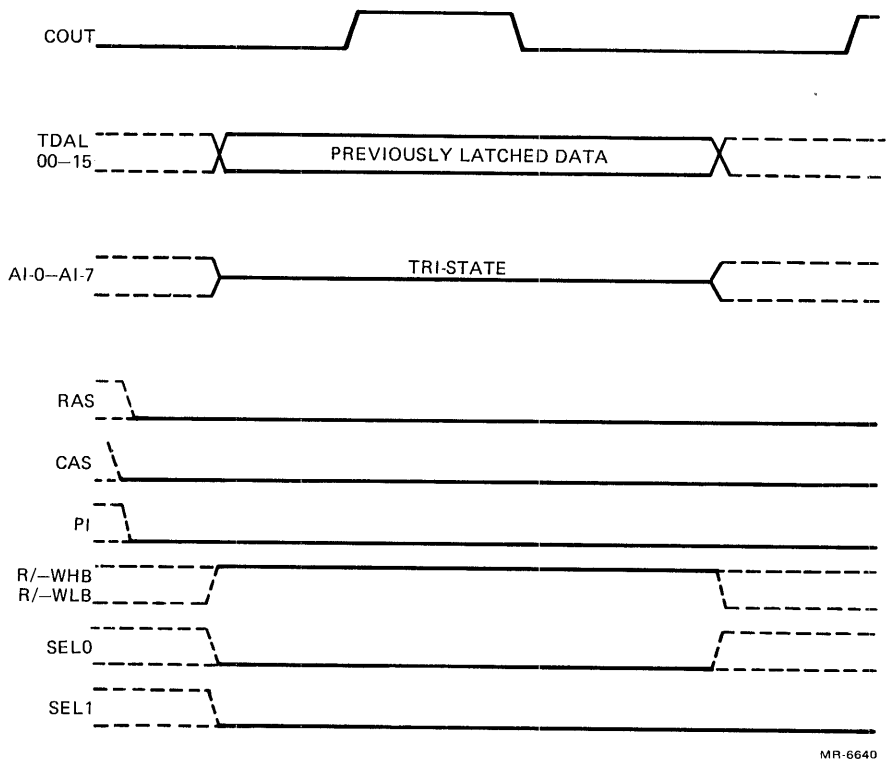


Figure 8-8 BUS NOP Transaction

8.2 MODE REGISTER CONTROL

The mode register is an internal microprocessor register used to define the operating mode of the microprocessor. The 16-bit mode register is written into from the TDAL 00--15 data lines during a power up sequence or when a reset instruction is executed. During this time the -BCLR output is low and the mode register is loaded. The mode register logic (Figure 8-9) has five tri-state drivers that are enabled when the -BCLR input goes low. TDAL bits 11 and 8 are factory set to force the microprocessor to operate in the following mode.

- The microprocessor clock mode is selected. The microprocessor pulses the COUT output once for every four XTL1 input pulses during DMA and Interrupt transactions. For all other transactions, the microprocessor pulses the COUT output once for every three XTL1 input pulses.

- The standard microcycle mode is selected. The standard microcycle uses four XTL1 input periods for DMA and Interrupts and three XTL1 input periods for all other transactions.

- The normal Read/Write mode is selected. The normal Read/Write mode establishes the read/write control lines (R/-WLB and R/-WHB) prior to the assertion of -RAS and remain valid after the negation of -CAS.

- The static memory mode is selected and therefore no dynamic memory chips may be installed on the module. The refresh function is disabled.

- The memory addressing is limited to 64K bytes.

- The bus consists of 16 bits.

- The user mode is selected. This mode performs transactions with no automatic test of the Processor Status Word.

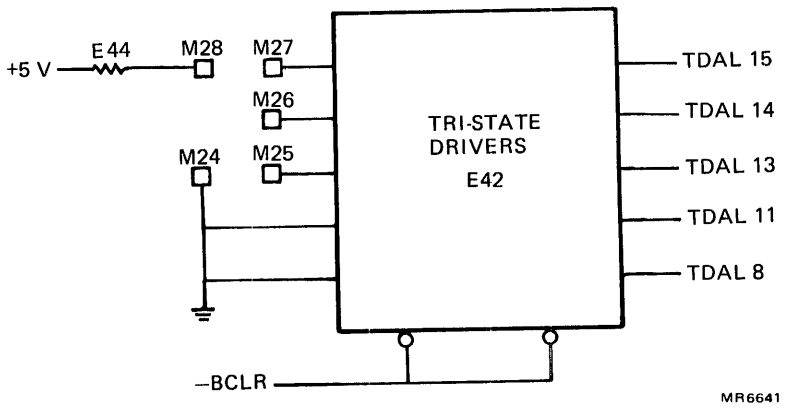


Figure 8-9 Mode Register Control

The status of the TDAL bits 13--15 are selected by the user and determine the start and restart addresses for the microprocessor. The start address is the location of the first fetch after power up and the restart address is the location of a fetch after a HALT instruction is executed or the assertion of the HALT interrupt. The wire wrap pins M27, M26, and M25 control the status of the TDAL 13--15 bits during the power up sequence. Wire wrap pin M28 is pulled up to +5 Vdc and represents a one, while wire wrap pin M24 is connected to ground and represents a zero. Pins M27, M26, and M25 are jumpered to either M28 or M24 according to the listing in **Table 8-1** to select the start and restart address for the microprocessor.

Table 8-1 Start Address Configurations

Wirewrap Pins			Start Address	Restart Address
Bit 15 M27	Bit 14 M26	Bit 13 M25		
1	1	1	172000	172004
1	1	0	173000	173004
1	0	1	000000	000004
1	0	0	010000	010004
0	1	1	020000	020004
0	1	0	040000	040004
0	0	1	100000	100004
0	0	0	140000	140004

Connection to M28 = 1
Connection to M24 = 0

8.3 INTERRUPT CONTROL

The interrupt logic, as a block diagram, is illustrated in Figure 8-10. Studying this diagram will make the detailed explanation presented later in this section easier to follow.

The elements of this scheme include:

- Five D flip-flops which latch five of the interrupt lines.

REVNT	Wire OR-ed TEVNT or BEVNT.
PARQST	Parallel I/O Port A interrupt request.
PBRQST	Parallel I/O Port B interrupt request.
BKRQ	Level 7, maskable interrupt, configurable.
HLTRQ	Produces CTMER, nonmaskable interrupt, configurable.

- Twelve interrupt synchronizing latches, which latch the following signals.

  Outputs of five latches described above

  3 interrupt signals:

IRQ4	Level 4 LSI-11 bus interrupt
DMRQ	DMA request
PFAIL	Power fail nonmaskable interrupt

  4 signals from the Interrupt Acknowledge Decoder, wire OR-ed with Interrupt Requests from SLUs.

RDL1	SLU1 Receiver Interrupt Request
XDL1	SLU1 Transmitter Interrupt Request
RDL2	SLU2 Receiver Interrupt Request
XDL2	SLU2 Transmitter Interrupt Request

The operation centers around eight microprocessor input lines, AI-0 to AI-7, driven by interrupt signals, either directly or indirectly, through the Interrupt Encoding PROM.

AI-0 serves as a DMA request line connected directly to DMRQ.

AI-1 to AI-4 are driven by the output of the Interrupt Encoder to request maskable interrupts.

AI-5 is driven by the VEC gate which detects the presence of the LSI-11 bus interrupt on the outputs of the Interrupt Encoder. It calls for a vector read transaction from the bus.

AI-6 is driven directly by the Power Fail input line to force a power fail trap.

AI-7 is driven directly by the HALT interrupt line to force a restart trap.

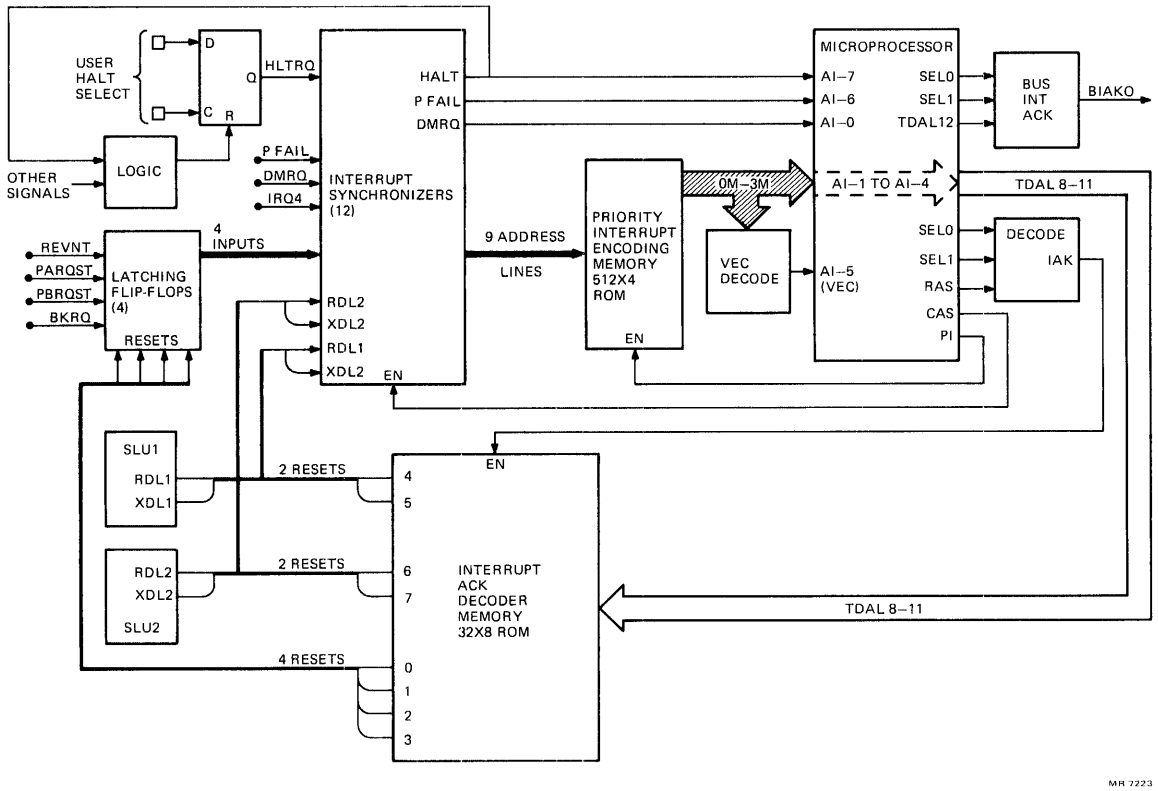


Figure 8-10 SBC-11/21 Interrupt Logic

The microprocessor reads the AI-0 to AI-7 input lines, and arbitrates the interrupt priority according to Table 8-2. In addition, the state of AI-1 to AI-5 is reproduced on TDAL 12-8 lines during the Acknowledge cycle. TDAL 11-8 lines are used as an address in the Interrupt Acknowledge Decoder, which is a 32-byte PROM. Output bits 7-4 of that PROM are the previously mentioned SLU receive and transmitter interrupt requests (RDL1, RDL2, XDL1, XDL2) which are wire-ORed to reset the latched requests in the SLUs. TDAL12 reflects the state of the -VEC signal and is used in the LSI-11 bus protocol.

Bits 0-3 are used as reset signals for the four interrupt latches mentioned earlier.

With this general understanding of the interrupt scheme in mind, the reader can proceed into the more detailed explanation that follows.

8.3.1 Details of Interrupt Control Logic

The Interrupt Logic (Figure 8-11) receives the interrupt requests from the interface devices and applies them to the microprocessor. The microprocessor will acknowledge the highest priority interrupt, provided its priority is higher than the current microprocessor status word priority. There are nine interrupts

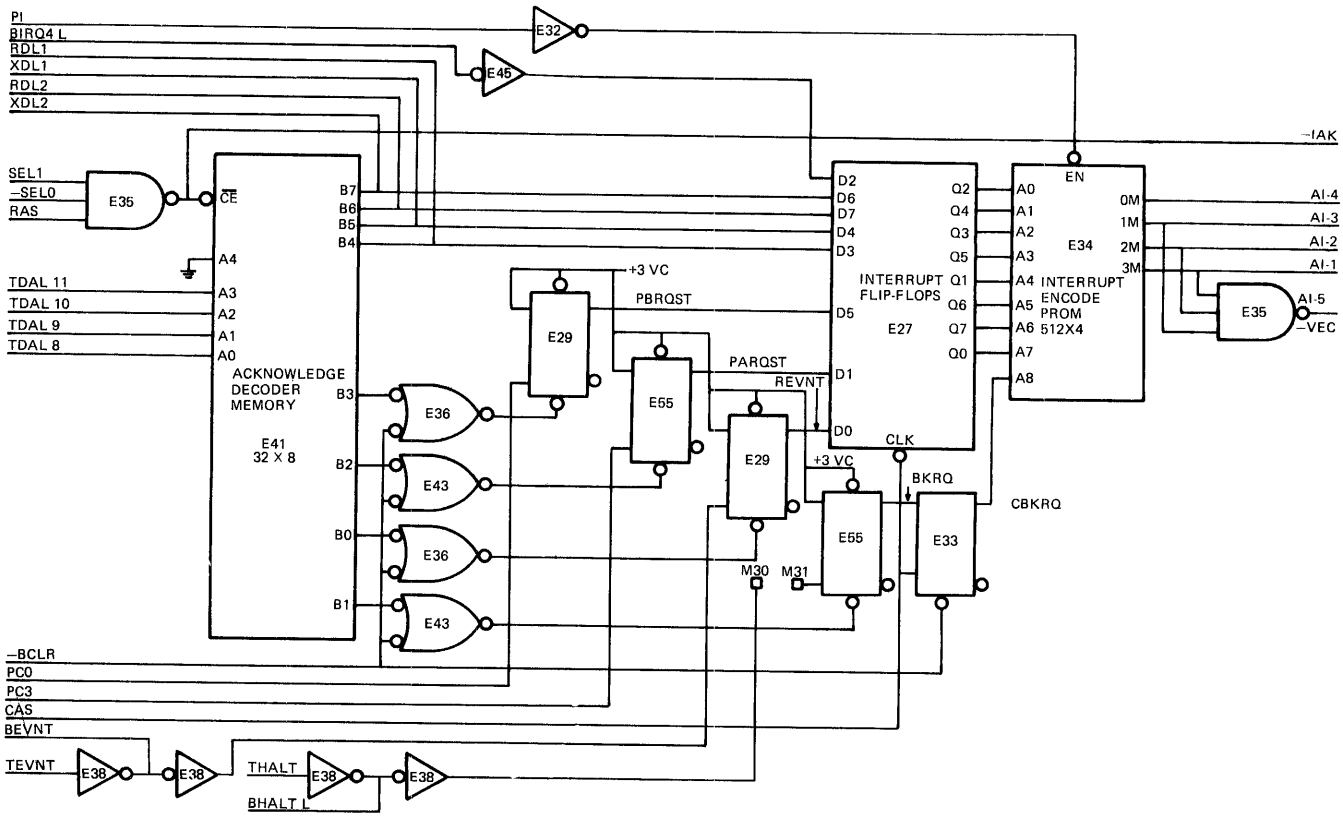


Figure 8-11 Interrupt Control

available and either one or all can be inputs to the Interrupt synchronizers E27 and E33. Any interrupt is active when the signal goes high. Five of these inputs are latched and remain high until reset. Four interrupts are clocked through flip-flops E29, and E55 to maintain a high output. The enabled interrupts are clocked through the Interrupt flip-flops by CAS asserting during the present transaction. These outputs address the locations of the Interrupt Encode Memory which is enabled by the -PI input of the present transaction. The Interrupt Encode Memory outputs an interrupt code equivalent to the highest input priority.

The interrupt codes and their priority levels are listed by descending rank in Table 8-2. When the PI output is enabled, the

Table 8-2 Designated Interrupts

Interrupt Source	Input Signal	Priority Level	Coded Input AI-1	AI-2	AI-3	AI-4	AI-5	Vector Address
HALT	HLTRQ	Non-maskable	X	X	X	X	X	Restart Address
Power Fail	PFAIL	Non-maskable	X	X	X	X	X	24
LSI-11 Bus Signal BHALT	BKRQ	7	0	0	0	0	1	140
LSI-11 Bus Signal BEVNT	REVNT	6	0	1	0	0	1	100
SLU2 REC	RDL2	5	1	0	0	0	1	120
SLU2 XMIT	XDL2	5	1	0	0	1	1	124
PARALLEL I/O B	PBRQST	5	1	0	1	0	1	130
PARALLEL I/O A	PARQST	5	1	0	1	1	1	134
SLU1 REC	RDL1	4	1	1	0	0	1	60
SLU1 XMIT	XDL1	4	1	1	0	1	1	64
LSI-11 Bus Signal BIRQ4	IRQ4	4	1	1	1	0	0	Read from LSI-11 Bus

NOTE: HALT and Power Fail interrupts are not generated by the coded inputs AI-1 to AI-5. All signals are listed in the order of descending priority.

microprocessor looks at the interrupt inputs and will following the completion of a Read transaction initiate an IAK transaction for an interrupt with the proper priority. The coded input to the microprocessor is placed on the TDAL bus using bits 08 through 12. Bit 08 represents the AI-1 input and bit 11 represents the AI-4 input. These four TDAL bus bits are inputs to the Acknowledge Decoder Memory which is enabled when the microprocessor starts the IAK transaction and the -IAK input goes low. These inputs are decoded to determine which interrupt was granted and will output a low to negate that interrupt. The interrupt flip-flop is reset by the clear line for that interrupt switching the output of the selected AND gate low. The E66 and E65 transmitter and receiver interrupt lines are latched outputs and are reset by wire OR-ing and asserting low the output of the Acknowledge Decoder PROM. The LSI-11 bus interrupt is an exception to this process. This interrupt code enables the inputs of NAND gate E35 and the low output enables the -VEC (AI-5) input to the microprocessor. This input instructs the microprocessor to receive the vector address from the TDAL bus. TDAL 12 reflects the state of the -VEC input when the microprocessor acknowledges the interrupt and is used to determine that the LSI-11 bus Interrupt Acknowledge handshake protocol must be invoked. The LSI-11 bus interrupt is not reset by the Acknowledge Decoder PROM, but it should be reset when the TDIN and TIAK0 signals are received by the bus device during the Interrupt Acknowledge Sequence.

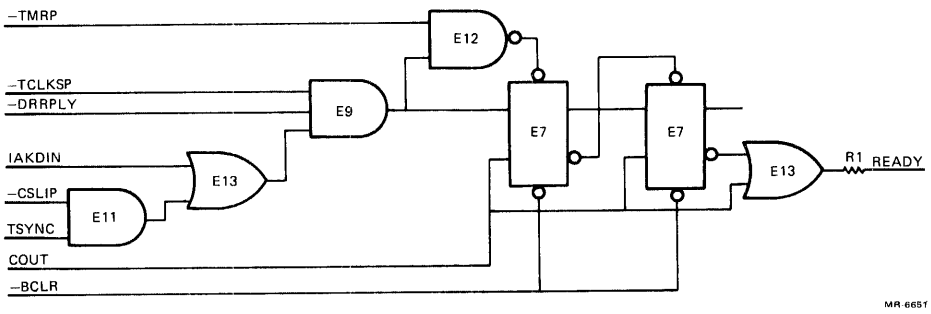
Before further discussion of the Interrupt system, the READY logic must be described.

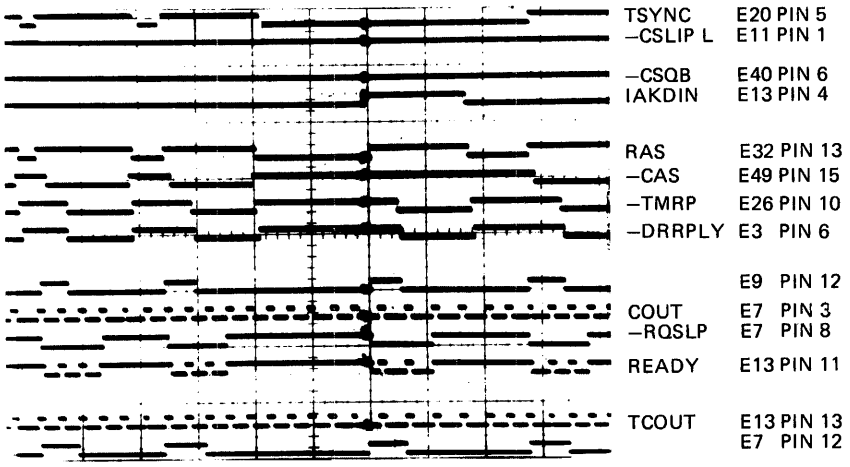
NOTE

The waveform diagrams shown in Figure 8-12 and subsequent figures are referenced to the circuit schematics in Appendix E. They were obtained by photographing the displays produced by a logic analyzer and are subject to quantization errors inherent in all logic analyzers. They are intended only as a help in understanding the logic and not as a precise representation of timing relationships.

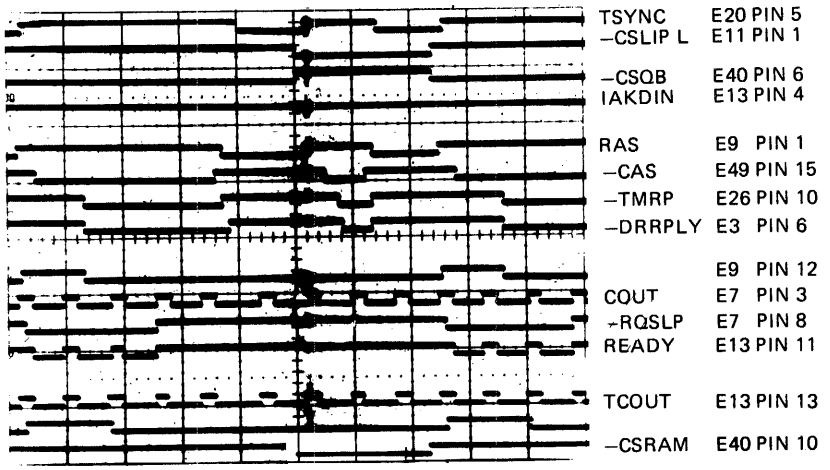
8.3.2 READY

The Ready logic (Figure 8-12) provides the READY input to the microprocessor and is used to control the cycle slip function. The microprocessor will cycle slip when the READY input is being pulsed while RAS is asserted and the cycle slip function is inhibited when the READY input is set high. The output of the Ready flip-flop and the COUT input go to the E13 OR gate and generate the READY input. When the -CSLIP input to E11 is high and the TSYNC input is high, the output of the E11 AND gate goes high. -DRRPLY is not yet asserted and the -TCLKSP and the output of E13 are high, so the output of E9 is high. This enables E12





A. Signals during LSI-11 Bus transaction



B. Signals during local transaction

Figure 8-12 Ready

and the preset input to the E7 flip-flops to go low. The flip-flop output is low at OR gate E13 and it enables the READY input with every COUT. When the IAKDIN input goes high and the -CSLIP and TSYNC inputs are negated, the output of the E9 AND gate goes high. It allows the E12 NAND gate output to go low and forces the preset terminal of the E4 flip-flops low. The output of the flip-flop to the OR gate is now low and this allows the COUT input to clock the READY output. The microprocessor will continue to cycle slip while this input is being pulsed. The -TMRP input to the NAND gate will go low when either the BRPLY or TMER input from the bus is received. This will remove the low from the preset input of the first flip-flop. Shortly after the -TMRP input goes low the -DRRPLY input also goes low and forces a high to the input of the flip-flop. The high is clocked through by the COUT clock and the flip-flop output to E13 will go high. This disables the READY input to the microprocessor and allows the transaction to be completed.

The second E7 flip-flop is required to ensure that data is stable at the microprocessor or at the peripheral prior to transaction completion. The Ready circuit is inactive during local address references.

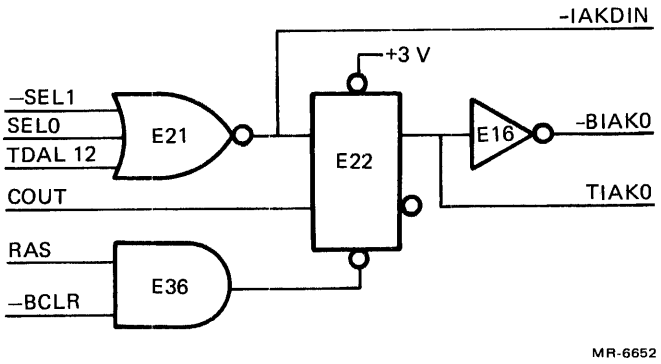
8.3.3 IAK DATA IN (IAKDIN)

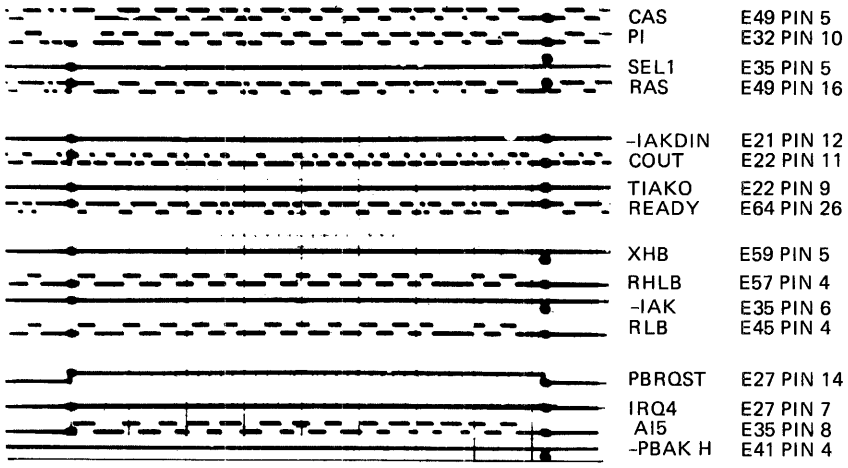
The IAKDIN output is enabled by the output of the NOR gate E21 as shown in Figure 8-13. The microprocessor acknowledges an external interrupt request, asserts -SEL1 and negates SEL0. When the microprocessor has to read the interrupt vector from the bus, the TDAL12 input is low as a result of AI-5 being low during the interrupt request read. This allows the IAKDIN output to go high and assert TDIN to the bus. The RAS input is high and this enables the TIAK0 flip-flop E22. IAKDIN is clocked by the COUT input causing TIAK0 to go high and the inverter E16 sets the BIAK0 output low. The BIAK0 output goes to the bus as an interrupt acknowledge. The TIAK0 output goes to the Bus transceiver logic and enables the low byte transceivers to receive the vector. The IAKDIN output goes to the Ready logic and allows the microprocessor to cycle slip until the interrupting device asserts the -BRPLY input or a timeout occurs. When either response is received, the SEL0 input goes high to disable the IAKDIN output and signals that the microprocessor has read the vector. The RAS goes low to clear the TIAK0 flip-flop.

The microprocessor cannot abort the reading of a vector if a timeout occurs and will read a vector of zero in all cases if -BRPLY is not asserted and the timeout counter triggers.

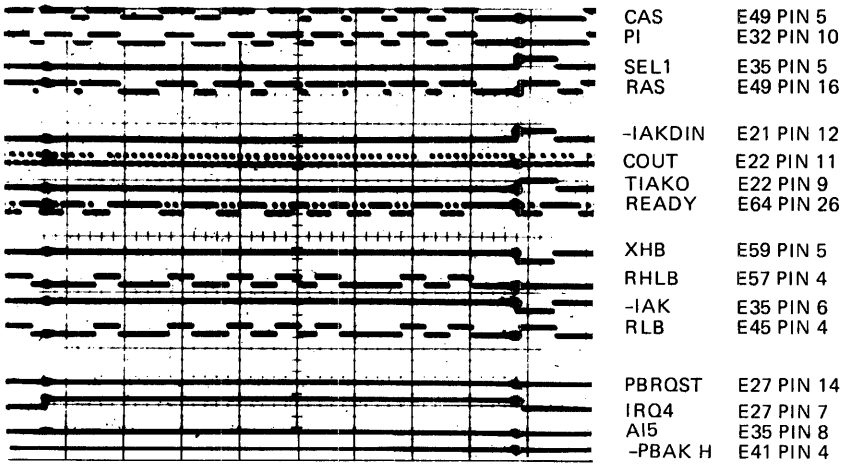
8.3.4 HALT INTERRUPT

The HALT interrupt (Figure 8-14) is designated as -CTMER and goes to the microprocessor AI-7 input. The user determines the configuration of the control signals, such as TMER, SLU BREAK request or BHALT that can trigger this interrupt. The E4



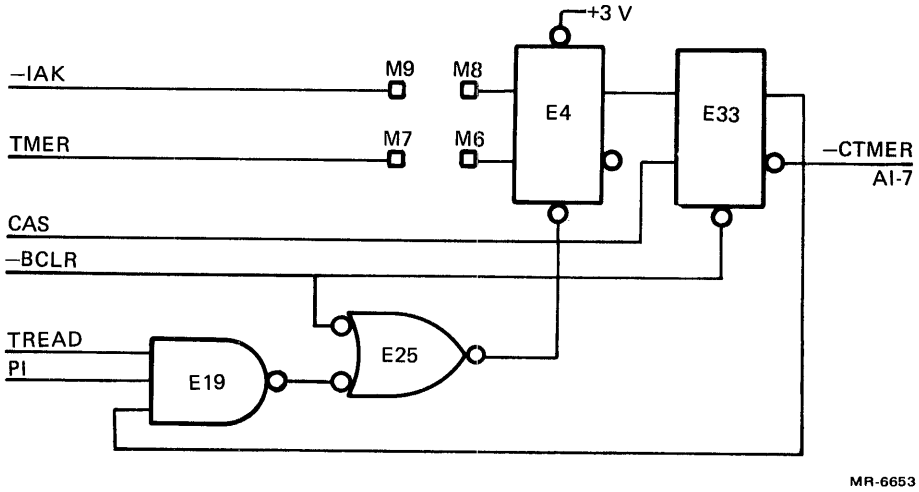


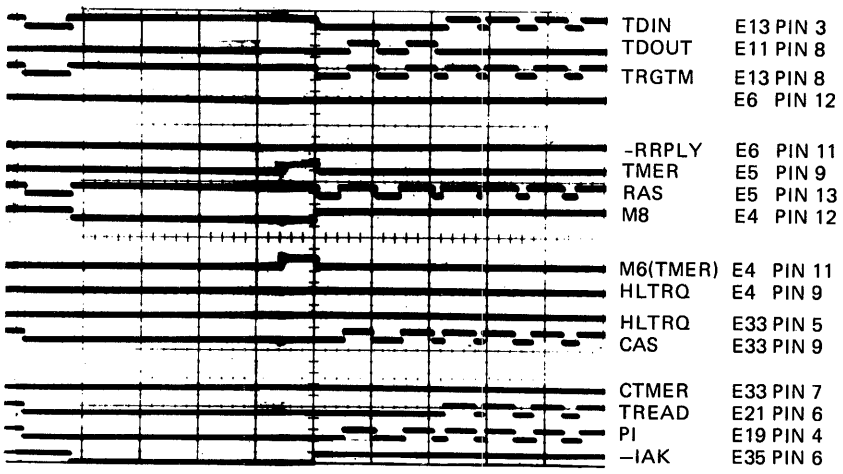
A. Signals during local interrupt



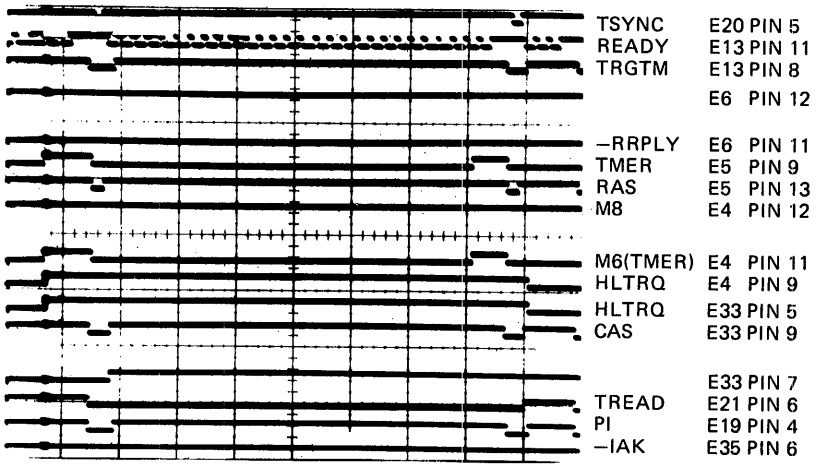
B. Signals during LSI-11 Bus interrupt

Figure 8-13 IAKDIN





A. Signals during LSI-11 Bus time out
(Interrupt Acknowledge with M8 and M9 jumpered)



B. Signals during nonexistent LSI-11 bus address
(with M8 and M9 jumpered)

Figure 8-14 HALT Interrupt

flip-flop is clocked by the input to M6 and this asserts the E33 flip-flop input. The assertion of CAS clocks the E33 flip-flop and enables the -CTMER output. The CTMER output is set high and goes to the NAND gate E19. The assertion of PI during a microprocessor Read or Fetch transaction latches -CTMER into the microprocessor and at the same time switches the E19 NAND gate output low. This sets the output of the E25 AND gate low to reset the E4 flip-flop for the next HALT interrupt. The E33 flip-flop is cleared by the next CAS strobe. The microprocessor AI-7 input is pseudo edge-sensitive, that is, it must be negated for one PI time before another trap to the restart address can be invoked.

As explained in Chapter 2, connecting M8 to M9 prevents-CTMER assertion during LSI-11 bus interrupt acknowledge transactions. This will prevent the restart trap resulting from this timeout.

8.3.5 Power Fail

The -PFAIL output is connected to the AI-6 input of the microprocessor and is recognized as the power fail interrupt which is nonmaskable. This is the second highest priority interrupt and it does not initiate an IAK transaction. When acknowledged, the microprocessor traps through octal addresses 24 and 26 to access the PC and PSW for the User's Power Fail Routine. This routine should include a RESET instruction and any other instructions required to initialize the bus and the module, as well as an MTPS instruction that will load 340 into the PSW and a wait instruction to inhibit the assertion of any LSI-11 bus control signal when battery backup is being utilized.

As an alternative to the MTPS instruction, 340 may simply be stored at location 26. Then, when the microprocessor vectors through 24, 340 will automatically be loaded into the PSW.

NOTE

BDCOK can be used as a microprocessor reset signal, unrelated to power failure. To guarantee correct restart, the BDCOK pulse must be at least 100 microseconds wide. BPOK should remain inactive during this reset operation.

8.3.6 Local

The on-board local interrupts are listed in Table 8-2 and use a coded input on the AI-1 through AI-5 inputs to the microprocessor. Some of these interrupt functions are determined by the user when configuring the module. There are eight local interrupts and they are all maskable. The multiple interrupts are arbitrated and the interrupt with the highest priority is serviced by the microprocessor. All local interrupts initiate an IAK transaction and their vector addresses are internal to the microprocessor. During IAK, the serviced interrupt is driven on TDAL lines 11--08 to address the interrupt acknowledge PROM. The outputs of the PROM reset the interrupts. TDAL bits 07--00 are ignored. The

microprocessor pushes the present PSW and PC onto the stack and receives a new PC and PSW from the vector address location and the next location.

8.3.7 External

A level 4 LSI-11 bus interrupt also uses a coded input on the AI-1 through AI-4 inputs to the microprocessor and the interrupt is maskable. For the bus interrupt the AI-5 input to the microprocessor is taken low to indicate that the vector address must be read from LSI-11 bus bits 07--02. The microprocessor does an IAK transaction and places the BDIN and BIAK0 signals on the bus to the requesting peripheral device. This device responds with -BRPLY and the vector address is read from the LSI-11 bus. The microprocessor pushes the current PC and PSW onto the stack and reads a new PSW and PC from the vector address location and the next location.

If the interrupting peripheral device fails to assert the BRPLY bus signal within 10 usec after BDIN is asserted, the module timeout signal TMER is enabled. The microprocessor completes the IAK transaction and receives a vector address of zero, since there is nothing driving the bus. The new PSW and PC are then read from locations zero and two. However, the user has the option to connect the timeout signal TMER to the HALT interrupt. The HALT interrupt can then be processed and pushes the current PSW and PC, which were read from locations 0 and 2, onto the stack. It then loads the PC with the restart address and the PSW with 340. If the HALT is ignored for the vector timeout, then only a vector through locations zero and two will occur.

8.3.8 DMA Interrupt

The DMA request is connected to the AI-0 input to the microprocessor. The DMA request is received by the microprocessor during any Read, Write, Fetch or ASPI transaction. The request is not acknowledged by an IAK transaction but is acknowledged by the microprocessor asserting the SEL0 and SEL1 outputs to initiate a DMA transaction as described in Paragraph 8.14.

8.4 DC004 PROTOCOL

The DC004 protocol logic chip (see Figure 8-1, Sheet 1) interfaces the LSI-11 read/write signals with the module read/write signals. The -CSQB input goes high and is strobed by RSYNC to enable the logic. The BDIN L input goes low to request read data and switches the -READ output low. The BDOUT L input goes low to strobe write data and switches the -WHB and -WLB outputs low if the BWTBT L input is high. When the BWTBT L input is low, the BDAL0 L input will select either the -WHB or the -WLB. A low on the BDAL0 L input switches the -WLB output low. The BRPLY L output is controlled by the -CSQB input. When -CSQB input is high, this indicates the LSI-11 bus was not selected. The BRPLY L output is enabled and is switched low, after an RC delay, whenever BSYNC L and either the BDIN L or BDOUT L outputs are switched low. If the -CSQB input is low, the LSI-11 bus is selected and the BRPLY L output is disabled The BDAL0, 1, and 2 inputs control the -SEL6 output. The output goes low when BDAL1 L and BDAL2 L inputs are low and the BDAL0 L is high.

8.5 ADDRESS LATCH

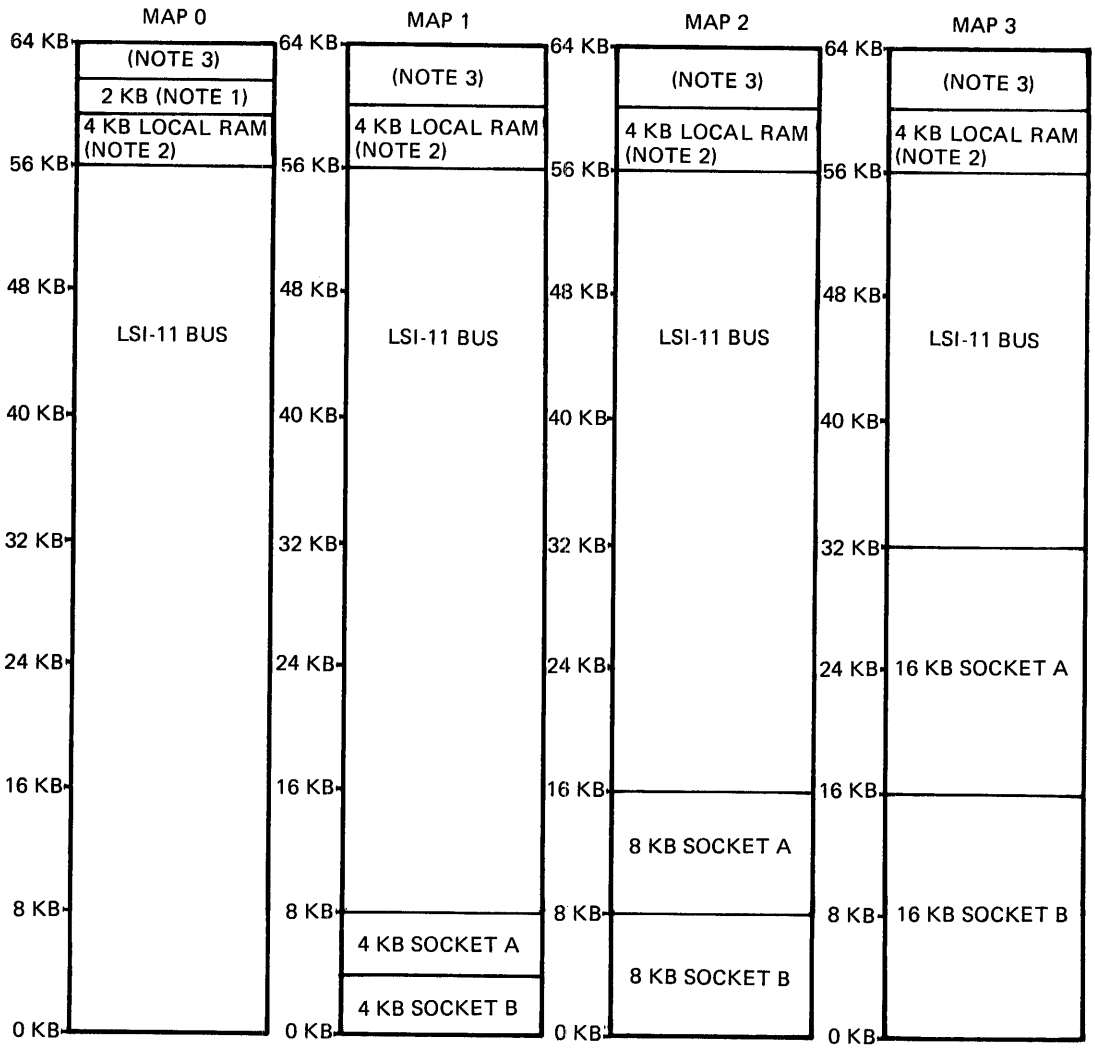
The Address Latching logic (see Figure 8-1, Sheet 1) consists of 16 transparent latches designated E53 and E63. The latches are always enabled by grounding the Output Control input. The TDAL bus bits 01--15 and the I/O page select signal RBS7 are monitored. The status of these inputs is latched to the Address Bus as bits AD1 through AD15 by the RSYNC input going high. The Address Bus and the latched LBS7 signal go to the Memory Address Decode logic (FPLA). The address bus is common to the module memories and the I/O circuits and remains stable while RSYNC is asserted.

8.6 MEMORY ADDRESS DECODE

The Memory Decode logic (see Figure 8-1, Sheet 1) consists of a Field Programmable Logic Array (FPLA) that decodes the applied address bits and the latched LBS7 signal. The FPLA selects a predetermined output according to the selected memory map. The module address range includes the on-board memory, the I/O interface registers and LSI-11 bus addresses. There are four different memory maps available to the user and these are described in Figure 8-15. The M22 and M23 wire-wrap pins allow the user to select one of these maps and these are described in Chapter 2. The FPLA is enabled provided the DCLO input is low. An address location in the RAM memory enables the -CSRAM output and an address location of either socket set A or B of PROM enables either the -CSKTA or -CSKTB outputs. A register address for either SLU 1 or SLU 2 will enable the -CSDL0 or -CSDL1 outputs. The -CSPL output is enabled when a register of the parallel I/O logic is addressed. The -CSLIP output is low for all the above address conditions. The -CSLIP output goes high only when the address is located on the LSI-11 bus and the -CSQB output is enabled low. The -CSLIP output allows the processor to cycle slip during the LSI-11 bus read/write and IAK transactions.

8.7 RAM MEMORY

The Static RAM memory is a 2K X 16-bit memory that consists of a 2K X 8-bit high byte chip and a 2K X 8-bit low byte chip as described by Figure 8-16. The memory is selected by the -CS RAM input going low to the CS pin. The memory is addressed by address bits AD1--AD11 and 16-bit data is read from or written to via TDAL bits 00--15. The memory is read by the -READ input going low to produce a low output from the AND gate E25 to the OE pin of the memories. The -WLB selects the low byte and the -WHB selects the high byte. The -WHB and -WLB inputs to the WE pin enables the write function and the output of AND gate E25 also goes low to the OE pin of the memories, to accommodate the dual CS function of some vendor's static RAMs.



NOTES:
1. SOCKET SET A IS MAPPED OVER SOCKET SET B AND IS THEREFORE LIMITED TO USING EITHER SOCKET A OR SOCKET B, BUT NOT BOTH TOGETHER.
2. ADDRESSES 160000 THROUGH 160007 ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
3. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUs AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.

MR-6643

Figure 8-15 Memory Maps

8.8 ROM/RAM MEMORY SOCKETS

The ROM/RAM memory provides the user with four 28-pin sockets, as described by Figure 8-17, to accept either 24-pin or 28-pin industry standard +5 V chips. The sockets can accommodate up to 32KB of UV PROMs, PROMs or ROMs and up to 8KB of static RAM. The socket sets are designated A and B, and each designation has a high byte socket and a low byte socket. The sockets use the -CSKTA

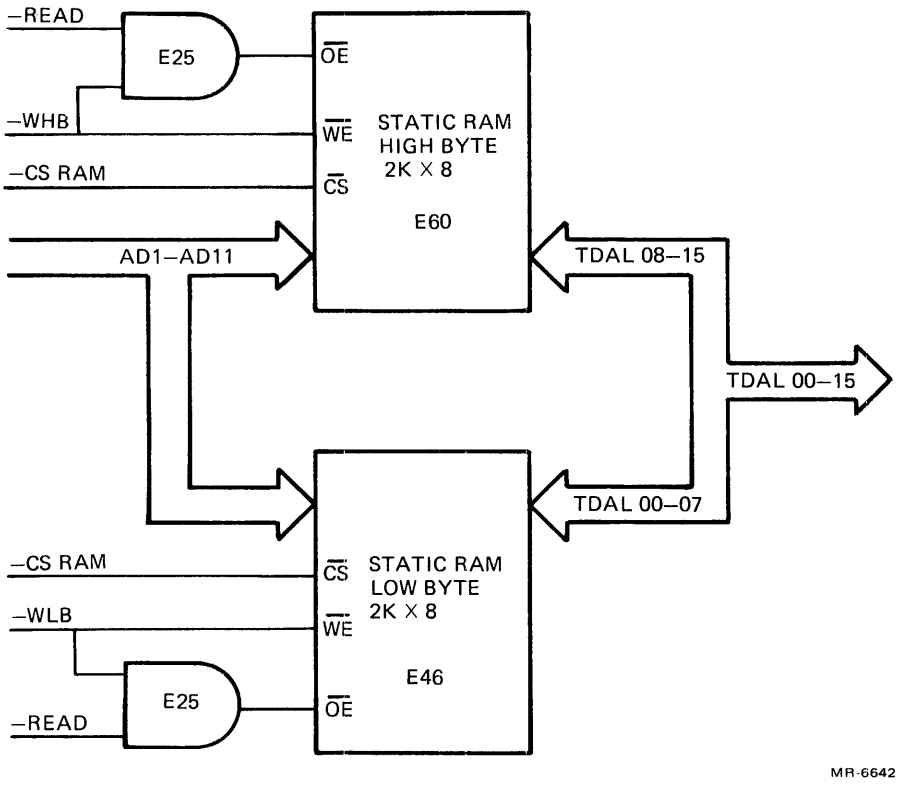


Figure 8-16 RAM Memory

and -CSKTB outputs from the Memory Address Decode and the user should refer to Figure 8-15 for the memory maps associated with these signals. The -READ, -WHB and -WLB signals from the DC004 Protocol are used to provide a High Byte Chip Enable (HBCE) and a Low Byte Chip Enable (LBCE). There are 30 wire-wrap jumper posts available for the memory configuration and detailed information is provided in Chapter 2.

NOTE

When a memory chip is placed into a socket wired for a larger capacity part, for example a 1K X 8 chip in a 2K X 8 socket, the addresses above the 1K boundary will wrap around into the start of the memory. This should be kept in mind when choosing the memory map configuration.

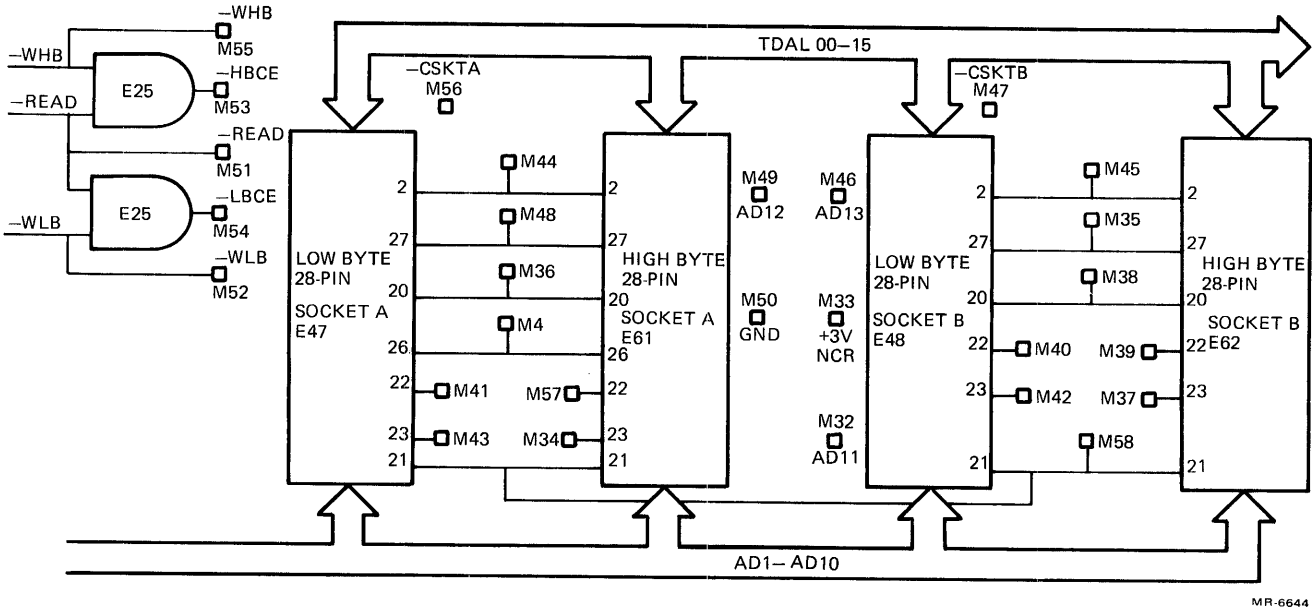


Figure 8-17 ROM/RAM Memory Sockets

8.9 SERIAL LINE INTERFACE UNITS

There are two Asynchronous Serial Line Units designated SLU1 and SLU2 to provide serial I/O interface through J1 and J2 as shown in Figure 8-18. Operational aspects are discussed in Chapter 2.

The SLUs transmit or receive 8-bit, byte-oriented data, with no parity, one start bit, and one stop bit. SLU1 provides the XDL1 and RDL1 interrupts for transmit and receive, and the BREAK output which is wired to pin M14. The user can jumper the BREAK output to the HALT interrupt (pin M13) and use SLU1 as a system console. SLU2 provides the XDL2 and RDL2 interrupts for transmit and receive, and three real time clock interrupts at 50 Hz, 60 Hz, and 800 Hz. These interrupts are wired to pins M18, M19, and M20 for use with the TEVNT interrupt (pin M17).

When the serial line units are addressed, the -CSDL0 input selects SLU1 and the -CSDL1 input selects SLU2 by enabling the chip select (CS) inputs. Address bits AD2 and AD1 are used to select individual registers within the SLUs. These registers are listed in Table 8-3 with their address and the logic states for AD2 and AD1 to access them. The -READ input will read the 16 bit register selected by -CSDL0 or -CSDL1, AD2 and AD1 by placing the contents onto the TDAL bus provided the -WLB input is not asserted low. When asserted low, the -WLB input will write the low byte of the TDAL bus into the register selected by -CSDL0 or -CSDL1, AD2 and AD1. However, only the register bits designated as read/write will be written into. The DLCLK input is a crystal controlled

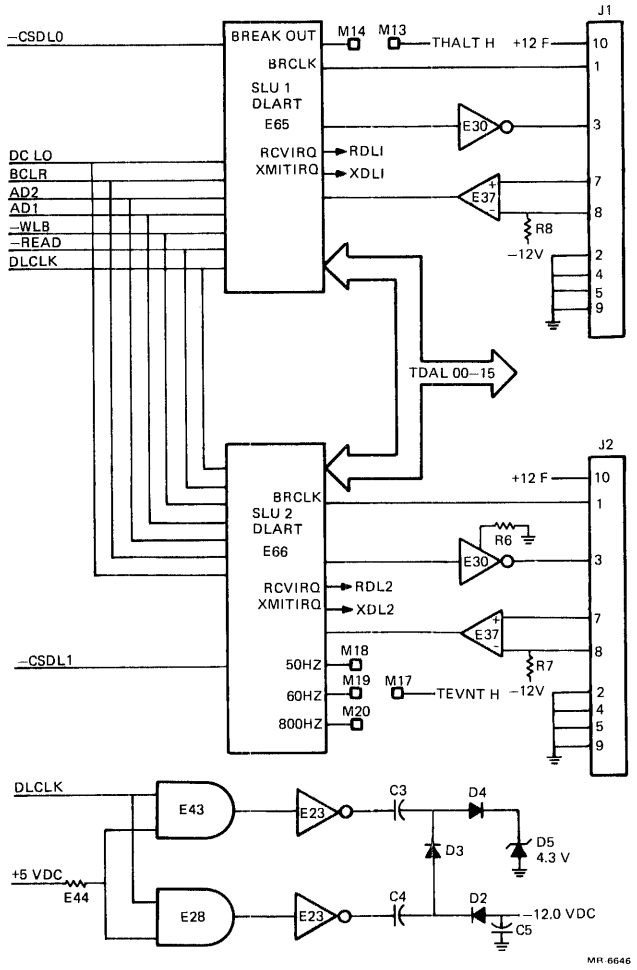


Figure 8-18 Serial Line Interface Units

clock reference used by the SLU to generate baud rates and real time clocks. The BCLR input is asserted during a RESET instruction, the RCVIE bit of the RCSR register and the XMITIE, MAINT and XMIT BRK bits of the XCSR register are reset. When the DCLO input is asserted during power up, it disables all SLU outputs and resets all internal logic and registers. The baud rate will be set at 300 baud after the SLU is initialized by DCLO.

The RS232 and RS423 signals for the interface connector are provided by 9636 (E30) and 9637 (E37) dual line drivers and dual line receivers. The slew rate for both channels is controlled by resistor R6. The factory configuration uses a 22K ohm resistor to provide a 2 usec slew rate for operating at a 38.4 K baud rate. Refer to Chapter 2 for the configuration requirements at other baud rates.

Table 8-3 Serial Line Unit Registers

Register	Description	Address	AD2	AD1
SLU1				
RCSR	Receiver Control/Status	177560	0	0
RBR	Receiver Buffer	177562	0	1
TCSR	Transmitter Control/Status	177564	1	0
TBR	Transmitter Buffer	177566	1	1
SLU2				
RCSR	Receiver Control/Status	176540	0	0
RBR	Receiver Buffer	176542	0	1
TCSR	Transmitter Control/Status	176544	1	0
TBR	Transmitter Buffer	176546	1	1

8.10 PARALLEL I/O INTERFACE

The programmable parallel I/O provides a 30 pin connector for transferring parallel data into or out of the SBC-11/21 module. The parallel I/O uses an 8255A-5 programmable interface chip, two 8-bit transceiver chips and an 8-bit buffer chip as described by Figure 8-19. The 8255A-5 has three input/output ports defined as port A, port B and port C. Port A and Port B outputs are connected to 8-bit bidirectional transceivers that are controlled by wirewrap pins M59 and M66. When a logical one is applied to these pins the data lines act as inputs to the module and when a logical zero is applied to these pins the data lines act as outputs from the module. The user can configure these as inputs or outputs using wirewrap pins M60 and M65 or as programmable inputs/outputs by programming the PC4 and PC6 lines (M64, M63) of Port C as described in the configuration description in Chapter 2. The Port C outputs are connected to directional buffers and used for interrupts and the handshake control for ports A and B. PC0 and PC3 are wired as outputs, PC3 enables the Parallel Interrupt Request for port A and PC0 enables the Parallel Interrupt Request for port B. PC4 and PC6 can be used as acknowledge or strobe inputs or can be configured to dynamically control the direction of ports A and B from either the 8255A-5 interface or the external peripheral device. PC1, PC5 and PC7 are wired as outputs and PC7 is wired to a LED that can be program controlled. PC2 is wired as an input and has a current limiting resistor for protection when PC2 might be programmed as an output from the 8255A-5 interface. Detailed configuration requirements are provided in Chapter 2 and the programming information is provided in Chapter 6.

The 8255A-5 Programmable Peripheral Interface (PPI) is enabled by the -CSPL input from the Memory Address Decode chip whenever the 176200-176207 addresses are selected. The AD1 and AD2 address lines are decoded to select one of four registers within the PPI and listed in Table 8-4. The Port A, Port B and Port C registers

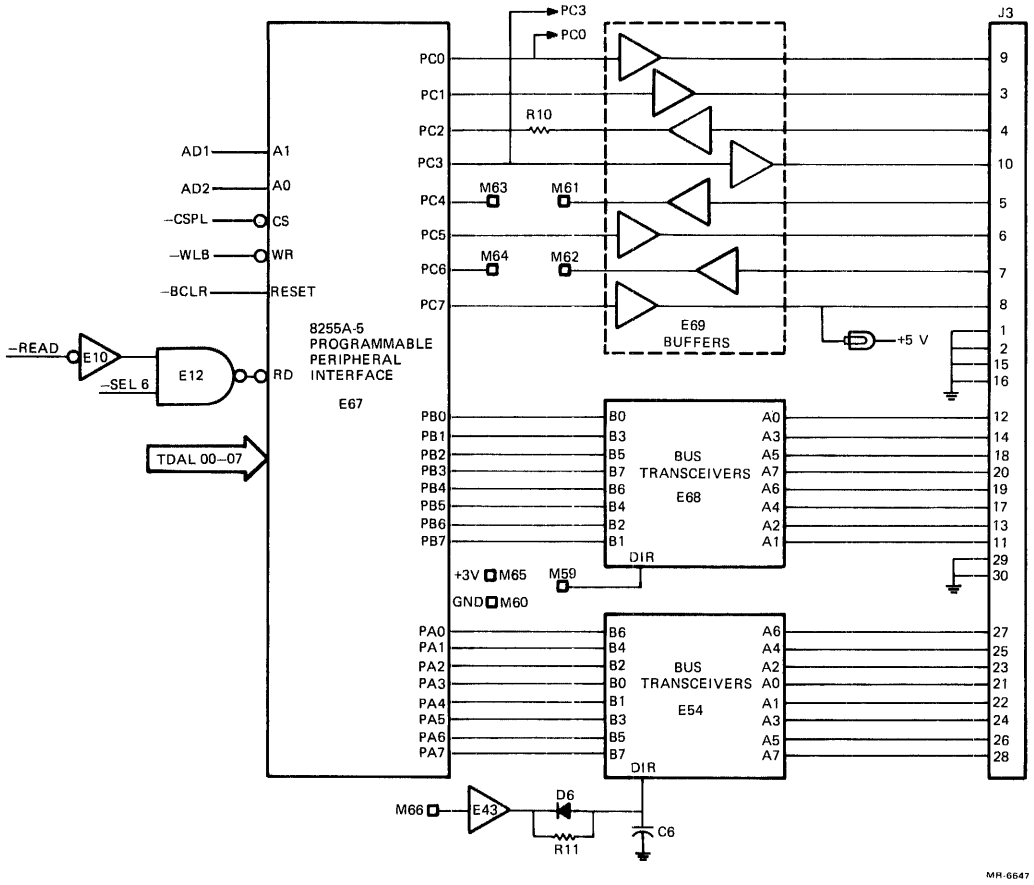
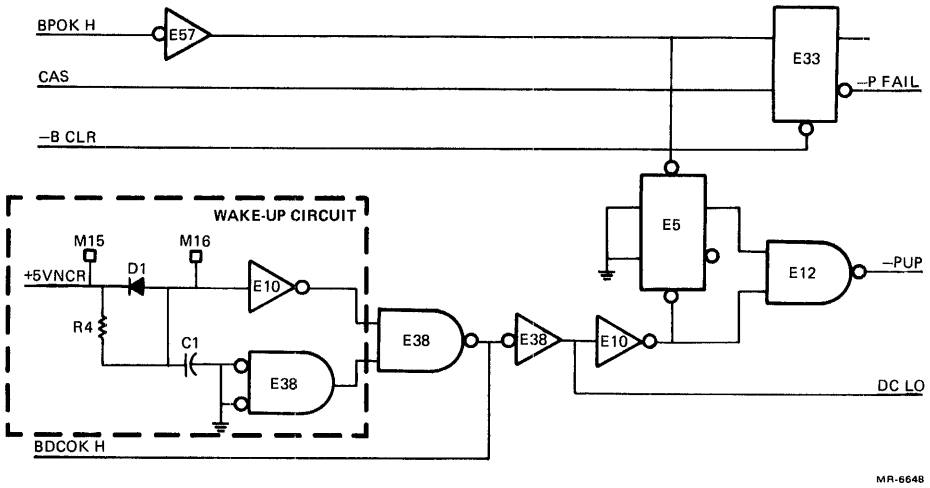
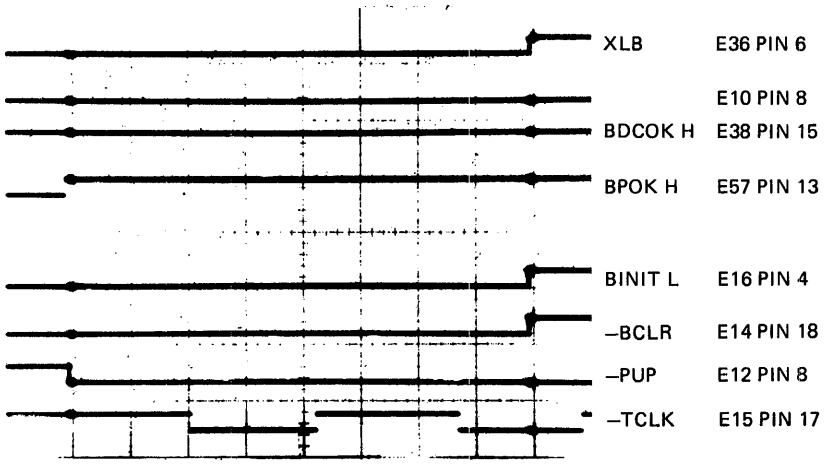


Figure 8-19 Parallel I/O Interface

Table 8-4 PPI Addressable Registers

Register	Address	Status
Port A	176200	Read/Write
Port B	176202	Read/Write
Port C	176204	Read/Write
Control Word	176206	Write Only





Signals during Power-Up

Figure 8-20 Power Up

are read/write and the Control Word register is a write only register. The addressed register is written into with the data on TDAL 07-00 bus when the -WLB input is asserted. The contents of the addressed register is placed on the TDAL 07-00 bus when the -READ input is asserted. The -SEL6 L input to NAND gate E12 inhibits the read strobe from the Control Word register and therefore any read of the control word register produces erronous data to the microprocessor. Only the low byte of the TDAL bus is used with the PPI and any data on the high byte is always treated as erroneous. The -BCLR input is used to reset the PPI when it is asserted and all 8255A-5 24 I/O lines are then defined as inputs. The buffer outputs to the connector will be driven high.

8.11 POWER UP

The power up circuits (Figure 8-20) sense the application of +5VNCR power source to the module and initiate a power-up sequence. When the +5VNCR input is first applied, the input at the

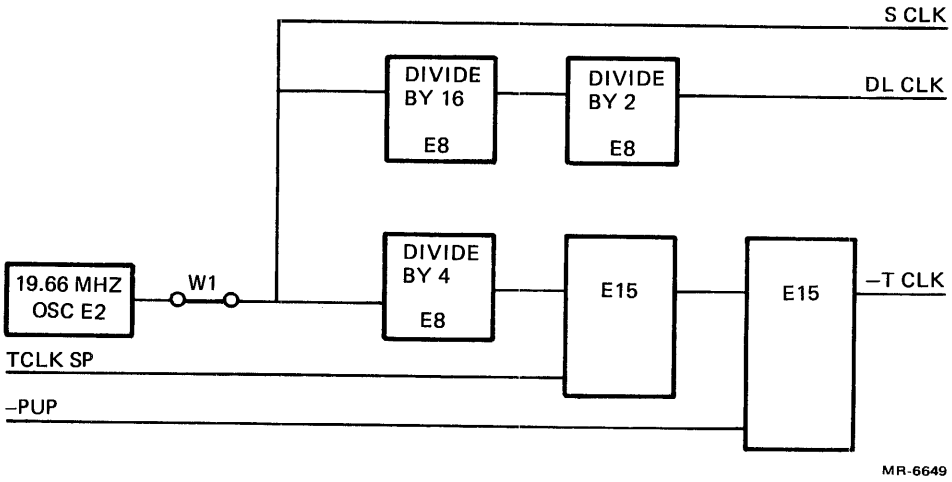
inverter E10 is low and causes the clear input of the PUP flip-flop E5 to be low, thus keeping its output low.Since an input to the nand gate E12 is low, the -PUP output is high, the microprocessor is held reset and asserts the -BCLR output. The +5VNCR input charges C1 through R4 until the threshold level of inverter E10 is reached. This occurs at approximately 2.6 Vdc and 70 ms after +5VNCR was applied. This causes the reset input to the PUP flip-flop to go high and set input to go low, setting the flip-flop. The -PUP output of the nand gate E12 goes low. This initiates the power up sequence of the processor.

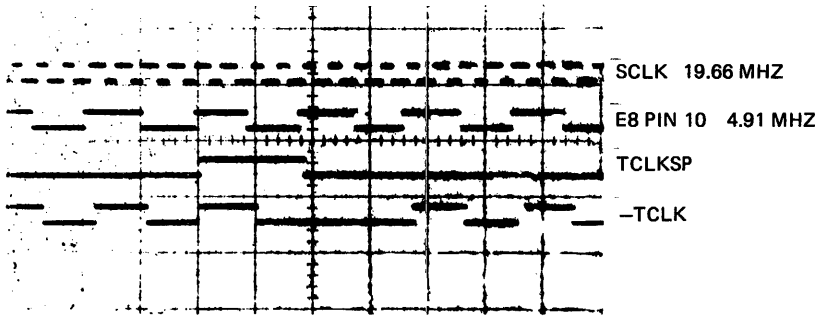
The power up delay circuit can be by-passed by inserting a jumper between M15 and M16. This allows the BDCOK H and BPOK H bus signals to control the PUP output. The +5VCNR input goes directly to the inverter E10 driving input to the inverter E38 low. E38 output is then controlled by BDCOK. The BDCOK H signal is low until power supply stabilizes, causing the reset input to the PUP flip-flop to be low. The BPOK H signal is also low and this causes the preset input to the flip-flop to be high. The low input to nand gate E12 drives the PUP output high. The microprocessor then asserts the -BCLR output resetting the PFAIL flip-flop. After a minimum of 3 ms the BDCOK bus input goes high and allows the PUP flip-flop E5 reset to go high. After a minimum of 70 ms the BPOK H bus input goes high causing the PUP preset input to go low. This allows the output to go high and when both inputs to the nand gate E12 are high the -PUP output is low. This initiates the power up sequence of the microprocessor.

The BPOK H bus input also goes to the PWR FAIL flip-flop E33. This flip-flop is not enabled until the microprocessor power up sequence is completed and therefore the BPOK H input is already high. Following the power up sequence, the first CAS pulse sets the PFAIL flip-flop. The flip-flop remains set until the BPOK input goes low indicating a power fail. The next CAS input clocks the PFAIL flip-flop and resets it. This causes the power fail interrupt and trap to location 24. This interrupt must be negated for at least one microprocessor read before another assertion will be recognized by the microprocessor.

8.12 CLOCK

The module uses a 19.6608 Mhz crystal oscillator as the basic time base reference. The oscillator output goes to the Clock Control logic (see Figure 8-21) and to the E8 binary counters. The counters are always enabled. The 19.6608 MHz output is divided by 32 and the DLCLK output, at 614.4 Khz, goes to the Serial Line units and to the charge pump. The oscillator is also divided by 4 and the 4.91 MHz output goes to the pulse sync circuit E15. When the TCLKSP input is low the circuits are enabled and the output goes to the next pulse sync circuit. When the TCLKSP input is high the circuits are inhibited and there is no output. The second pulse sync circuit is controlled by the PUP input. When the PUP input is low, TCLK to the XTL1 input is enabled and when the PUP input is high, the XTL1 input is inhibited.



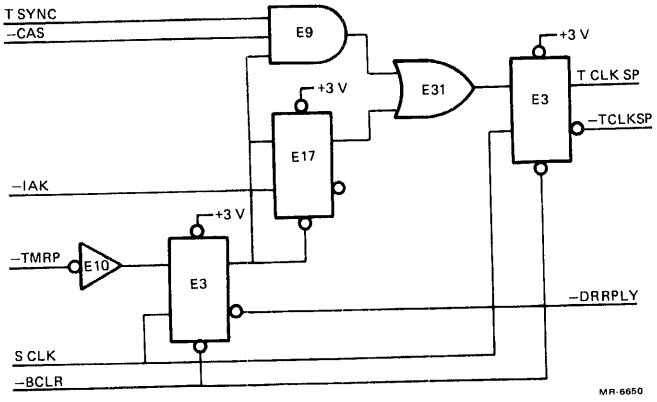


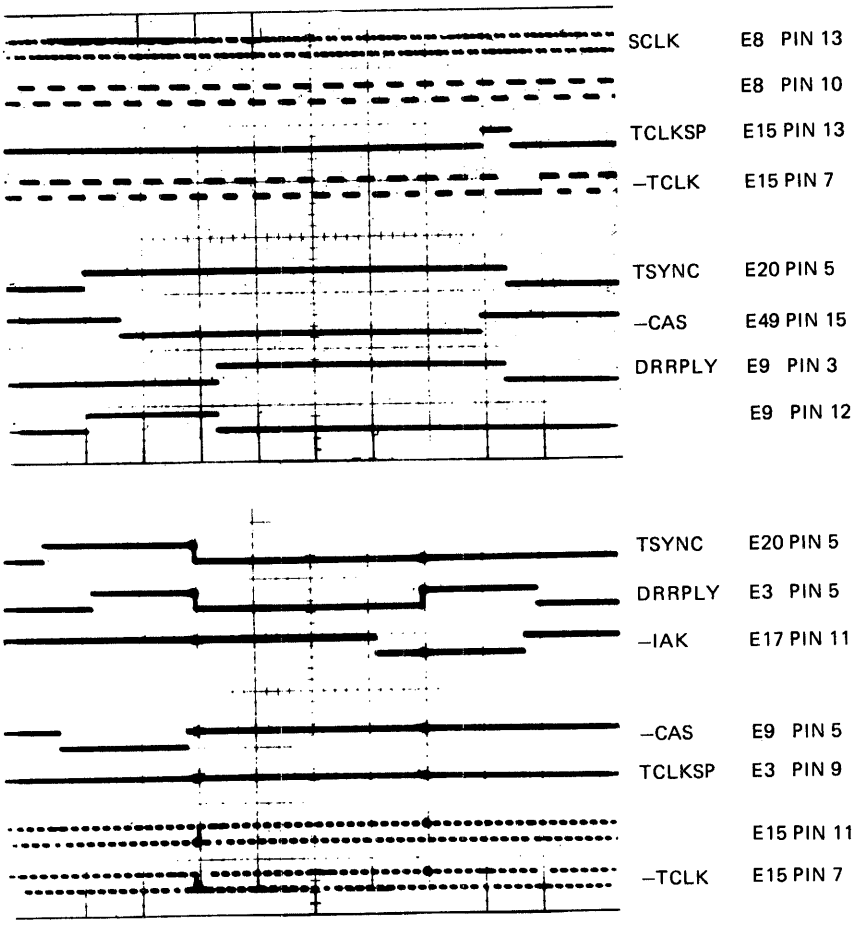
Signals for clock logic

Figure 8-21 Clock

8.13 CLOCK CONTROL

The clock control logic (Figure 8-22) is used to stop the XTL1 input to the microprocessor and forces the microprocessor to stop or wait until the XTL1 input is enabled again. The TCLKSP output is normally low to enable XTL1 and this is controlled by the TMRP input being high. This forces a low for both inputs to the OR gate E31 and the low output is clocked through the TCLKSP flip-flop by the 19.6 MHZ input. When TMRP goes low this removes the low inputs to the AND gate E9 and the IAK flip-flop E17. The TSYNC input is high for Read/Write and Fetch transactions and when the -CAS input goes high the AND gate E9 output also goes high. This is clocked through the TCLKSP flip-flop and the output goes high to stop the 4.91 MHZ clock output of E15. The TSYNC input is low for DMA and IAK transactions so that input to the AND gate E9 holds the output low. However the IAK flip-flop E17 is set when the -IAK clock input goes high at the end of an external interrupt transaction and the E31 output goes high. This is clocked through the TCLKSP flip-flop and the output goes high to stop the 4.91 MHz clock output of E15. The microprocessor XTL1 input will remain stopped until the TMRP input goes high again signaling that either BRPLY or TMER have been negated. This forces the IAK flip-flop E17 output to go low. This negates the TCLKSP output and enables the XTL1 input to the microprocessor.



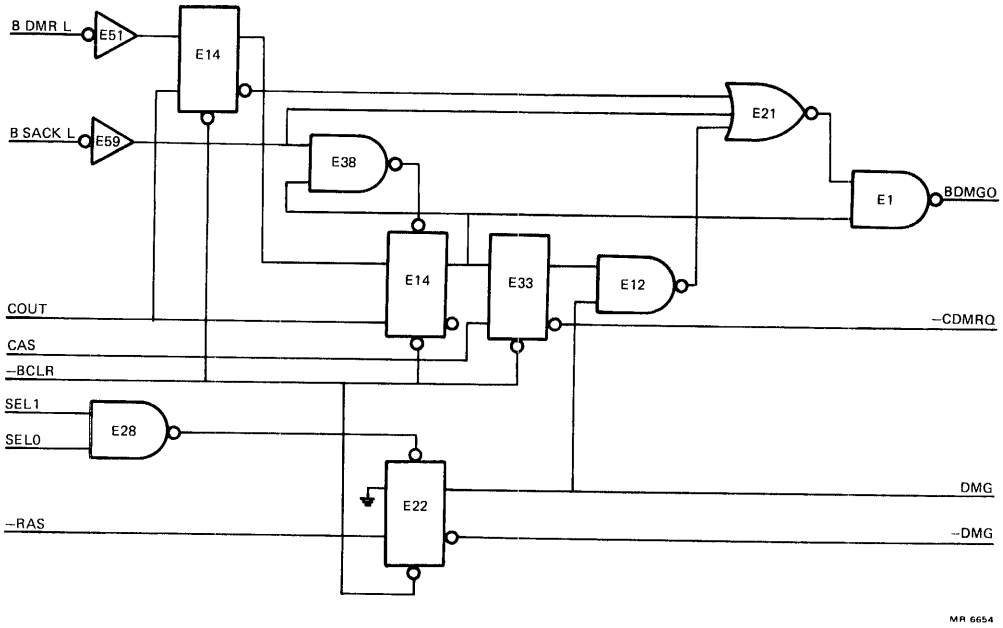


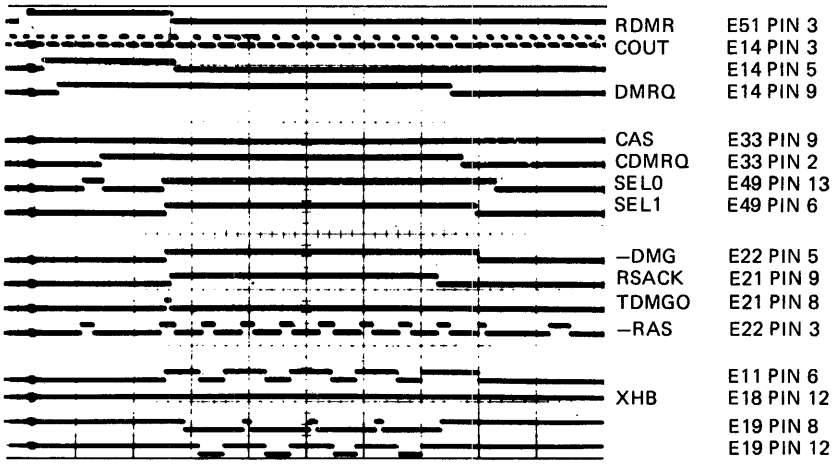
Signals for clock control

Figure 8-22 Clock Control

8.14 DMA

The DMA logic (Figure 8-23) controls the bus and microprocessor for DMA transactions. The BDMR L input goes low to initiate a DMA request. The output of the inverter goes high and is clocked through flip-flop E14 by COUT. The low output goes to the E21 NOR gate and the high output goes to flip-flop E14. The high output is clocked through by COUT and the high output enables the two NAND gates E28 and E1. The high output is also clocked through flip-flop E33 by the CAS input. The high output enables the NAND gate E12 and the -CDMRQ output (AIO Input) is switched low. The -CDMRQ output is the DMA interrupt to the microprocessor and initiates a DMA transaction. The microprocessor acknowledges the





Signals for DMA logic

Figure 8-23 DMA

request by outputing SEL1 and SEL0 high to NAND gate E28. The preset of flip-flop E22 goes low to set the DMG output high and the -DMG output low. The DMG high input to NAND gate E12 switches the output low and it goes to NOR gate E21. The BSACK L input is normally high and when inverted by E59 is a low input to the NOR gate E21. All three inputs to the NOR gate E21 are now low causing the output to switch high. Two high inputs to the NAND gate E1 switches BDMG0 low on the bus to the originator of the DMA request. The requesting device then sets the bus signal BSACK L low and the BDMR L input high. BSACK L is inverted by E59 and removes the low from the NOR gate E21 and the high input to the NAND gate E1 causing the BDMG0 output to go high. It also provides a high input to the NAND gate E28 causing the output to switch low. This low goes to the preset input of the flip-flop E14 and clamps the output high which holds the microprocessor in the DMG mode. The requesting device maintains the BSACK L input low for the duration of the DMA transfer and then sets it high. This removes the low from the preset input of flip-flop E14 and enables the flip-flop. Previously the BDMR L input went high and was inverted as a low to flip-flop E14. This low was clocked through by COUT and provides a low input to the enabled flip-flop E14. The low is now clocked through causing the -CDMRQ output to go high which removes the request from the microprocessor. The microprocessor completes the DMA interrupt transaction and negates the SEL1 and SEL0 outputs. The preset input of flip-flop E22 is no longer low and the low data input is clocked through when RAS goes high. The DMG output goes low and the -DMG output goes high to complete the DMA transaction.

8.15 TSYNC

The TSYNC output (Figure 8-24) is normally high for the microprocessor controlled Fetch/Read and Write transactions and low for IAK and DMA transactions. These conditions follow the -SEL1 input which is high and low for the same transactions. The exclusive OR gate E32 acts as a non inverting buffer and when RAS goes high the -SEL1 input of the TSYNC flip-flop E20 is clock through as the output. Whenever the -CSYNC clear input goes low, it forces the output of the TSYNC flip-flop E20 to go low. The CSYNC flip-flop E20 normally has the clear input pulled low by TCLKSP and the output to the AND gate E11 is high. When the TCLKSP input goes high the input of the CSYNC flip-flop is enabled. At this time the -DRRPLY clock input is low and goes high to clock the flip-flop shortly before the TCLKSP input gets reset. If a DMA transaction is in progress the -DMG input is high and the CSYNC flip-flop output remains low when clocked by -DRRPLY going high. For any transaction other than the DMA, the -DMG input is low and the CSYNC flip-flop output goes high when clocked by -RPTM going high. This allows the CSYNC output to go high and clear the TSYNC flip-flop E20 the write byte flip-flop E17 and the disable WT flip-flop E4 in Figure 8-25.

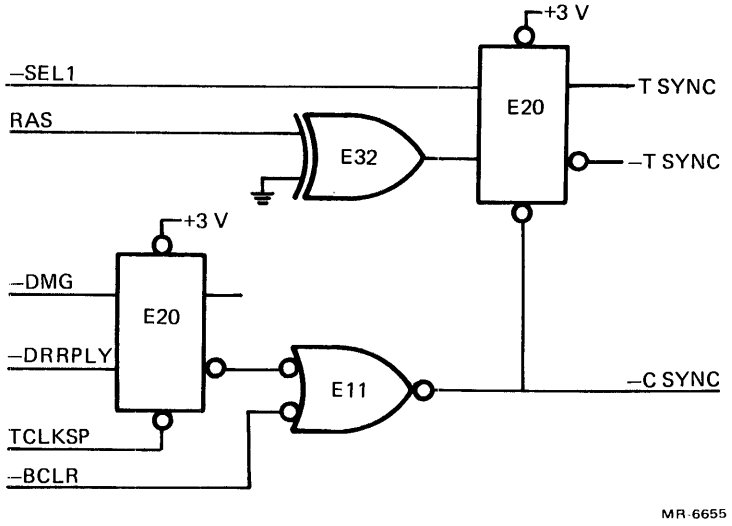
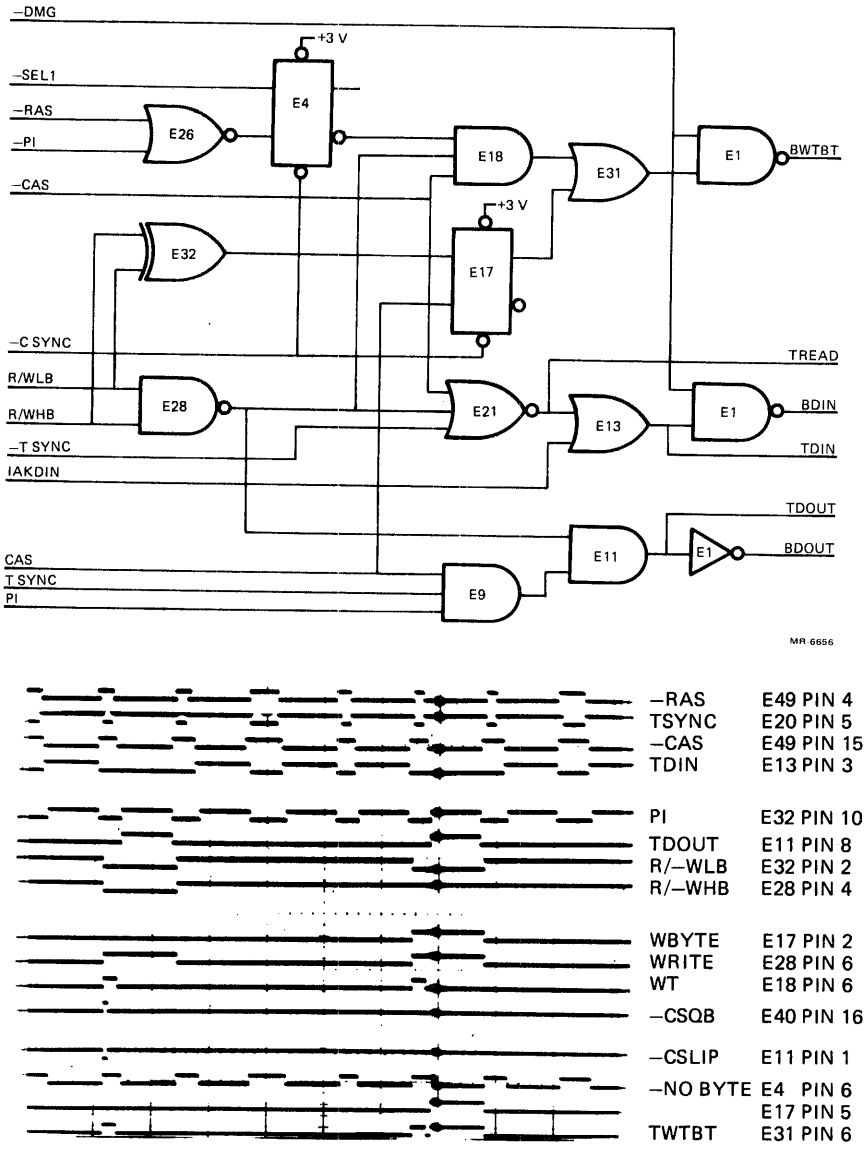


Figure 8-24 TSYNC

8.16 READ/WRITE

The Read/Write logic (Figure 8-25) controls the read, write, and fetch transactions for the microprocessor and supports the IAK and DMA transactions. The microprocessor controls the R/-WLB and R/-WHB inputs to select either BDIN, BDOUT or BWTBT bus signals. To select the BDIN output the microprocessor sets both R/-WLB and R/-WLB inputs high to the NAND gate E28. The output goes low to enable the NOR gate E21 and disables the AND gates E18 and E11. The -TSYNC input to E21 is low for read/write transactions. When the -CAS input goes low, the TREAD output goes high. The TDIN output of OR gate E13 goes high and the BDIN output of NAND gate E21 goes low. The -DMG input to the NAND gate is always high except for DMA transactions. During interrupt transactions, the IAKDIN input to E13 is enabled high, and also causes TDIN to go high and BDIN to go low.

The microprocessor determines any write condition by setting either R/-WLB or R/-WHB input low or both of these inputs low. The output of NAND gate E28 goes high and enables the AND gates E11 and E18. The output of flip-flop E4 is high and the -CAS input to AND gate E18 is also high. The output of AND gate E18 goes high and the output of OR gate E31 goes high. The DMA input to NAND gate E1 is high and allows the BWTBT output to go low. At this time the write destination address is written onto the bus. The logic now determines if the data being written is a word or a byte. The exclusive OR gate E32 monitors the R/-WLB and R/-WHB inputs and the output goes high when the inputs are different. A high output indicates the data is a byte and a low output indicates the data is a word. The output goes to flip-flop E17.



Signals for Read/Write logic

Figure 8-25 Read/Write

The microprocessor asserts CAS, the CAS input to E17 and E9 goes high and -CAS input to E18 goes low. The -CAS input to AND gate E18 switches the output low, to remove BWTBT but the CAS input clocks flip-flop E17 and enables the WBYTE signal to E31. The output of the flip-flop E17 is high for byte transactions and low for word transactions. The BWTBT L signal will either remain asserted low for a byte transaction or be negated high for a word transaction. The TSYNC and CAS inputs to AND gate E9 are set high and when the PI input goes high the gate output goes high. The AND gate E11 is enabled and the output of E9 switches the TDOUT output high. This is inverted and the BDOUT output is enabled by going low and writes the data word.

At the same time the -RAS and -PI inputs to NOR gate E26 are both low switching the output high. The high clocks flip-flop E4 and the output goes low. This inhibits the AND gate E18 when the -CAS input goes high again. The flip-flops are reset by CSYNC at the end of the transaction.

8.17 REPLY TIMEOUT

The Reply Timeout logic (Figure 8-26) monitors the bus BRPLY L input to indicate that an LSI-11 bus device responds to an address. The TMER flip-flop E5 output is normally set low by the RAS input to clear the flip-flop. The BRPLY L input is high and inverted so the RRPLY output is low. The -TMRP NOR gate inputs are both low and the -TMRP output is high. The bus transaction is initiated by either TDIN or TDOUT inputs going high. This enables the 10 usec timeout (50 cycle slips) monostable multivibrator to start. The microprocessor starts to cycle slip while waiting for the BRPLY L input to go low indicating the bus transaction can complete. When BRPLY L switches low, the RRPLY output goes high and the -TMRP output goes low. The TMER output remains low. If the BRPLY L does not go low and the 50 usec timeout circuit allows the 50 cycle slips and the TMER flip-flop is clocked and the TMER output goes high. This also forces the -TMRP output to go low. The assertion of the TMER output goes to the HALT logic and the microprocessor action is dependent upon the configuration of the module. The -TMRP output goes to the Clock Enable, SYNC, Ready logic, and disables cycle slips and the start/stop of the XTL1 clock. The RRPLY output goes to the Bus Control logic, and enables bus data to be received during LSI-11 bus device reads.

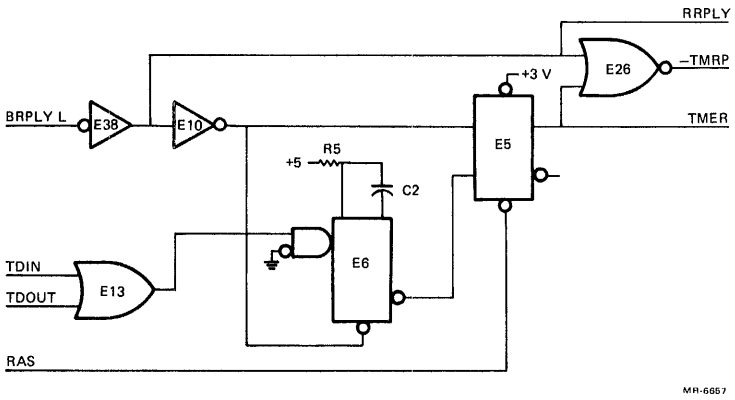


Figure 8-26 Reply Timeout

8.18 BUS CONTROL

The Bus Control logic (Figure 8-27) controls the transmit and receive functions of the Bus Transceivers. The transceivers are in transmit mode for microprocessor controlled Read/WRite and Fetch transactions to local memory, Local I/O and during LSI-11 bus writes. The transceivers go to the receive mode during an LSI-11 bus read. During DMA, the transceivers go to the receive mode to accept the local device address and will stay in this mode until the device is addressed. When a read transaction occurs, the transceivers go into the transmit mode. When the -BCLR input is high the transceivers are able to transmit data and when -BCLR is asserted low, the transceivers are disabled. During an IAK transaction, the -IAK input to AND gate E18 goes low to disable the transceiver high byte and the low byte goes to the receive mode to accept the vector.

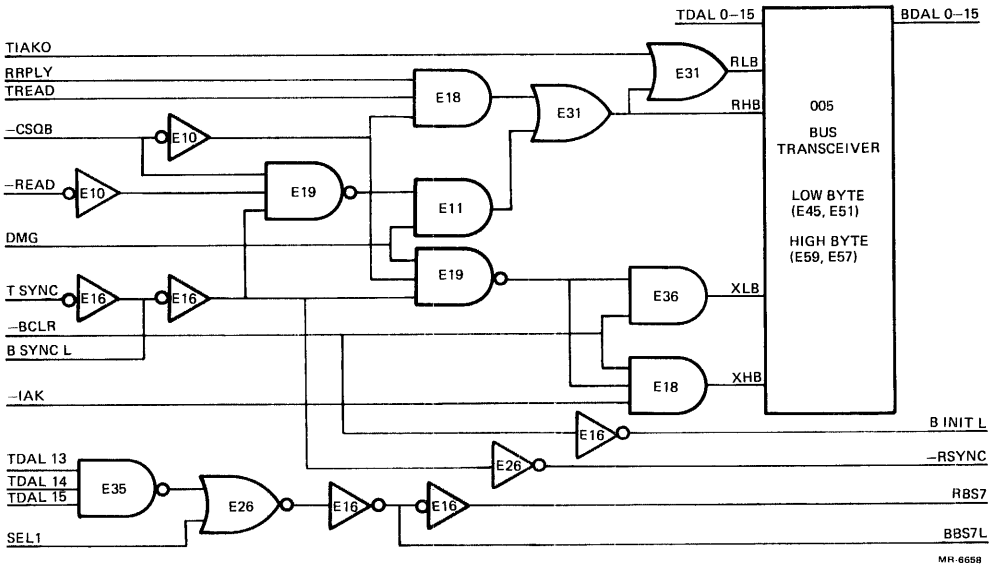


Figure 8-27 Bus Control

The receive function of the bus transceivers will override the transmit function any time the receive inputs are enabled high. When data is to be read from an LSI-11 bus device, the -CSQB input is low and inverter E10 makes it a high input to AND gate E18. The TREAD input to AND gate E18 is set high for the receive function. When the data is on the bus the DRRPLY input to AND gate E18 goes high and the output of the gate goes high. The two OR gates E31 allow the high output to enable the Receive Low Byte and Receive High Byte inputs to the transceivers. The data is now read onto the TDAL bus. During an interrupt transaction, the TIAK0 input goes high and enables only the Receive Low Byte input of the transceivers.

The DMG transaction grants bus control to the external device that requested the direct memory grant. The DMG input goes high for the duration of the DMG transaction. This input enables the AND gate E11 and the NAND gate E19. The BSYNC L input is high and inverted low to the two NAND gates E19. This switches the NAND gate outputs high and the receive and transmit functions are both enabled. However, the receive function overrides the transmit function and the TDAL bus receives data from the BDAL bus. This condition exists until the bus master asserts the BDIN L input low. It is inverted high and enables the NAND gate E19. The -CSQB input is dependent upon the address received from the BDAL bus. This input is low if the address is a bus location and high if the address is for the local memory or I/O device. A low input sets the output of NAND gate E19 high and enables the receive function of the transceivers. At the same time, the -CSQB low input is inverted high and the output of NAND gate E19 is switched low to disable the transmit function. When the -CSQB input is high indicating the local memory is being addressed, the NAND gate E19 is enabled. The -CSQB high input is also inverted low to NAND gate E19 and enables the receive function. The bus master now asserts either BDIN L or BDOUT L bus signals. The -READ input goes low for the BDIN L signal and goes high for the BDOUT L signal. If -READ goes high, it is inverted low and switches the output of NAND gate E19 high to enable the receive function. If -READ goes low, it is inverted high and switches the output of NAND gate E19 low to inhibit the receive function and the transmit function remains enabled. Therefore, when the bus master asserts the BDIN L bus signal, the data is transmitted from the module and when it asserts the BDOUT L bus signal the data is received by the module even if it was not addressed.

The BBS7 L bus signal is enabled low whenever the bus addresses the I/O page during the address portion of a transaction. This consists of the upper 8K bytes from 56KB to 64KB. This page is normally reserved for I/O devices on the LSI-11 bus, but the 4K bytes of local RAM memory resides within this page. It is also possible to have an additional 2K bytes of memory located within this page.

To address this page, the TDAL bus bits 13, 14, and 15 are set high and are inputs to NAND gate E35. The output is switched low and goes to the NOR gate E26. The SEL1 input to NOR gate E26 is low for read, write and fetch transactions. When both inputs to NOR gate E26 are low, the output is switched high. This is inverted to a low for BBS7 L output. It is inverted again to set RBS7 high.

9.0 GENERAL

The LSI-11 bus provides interconnections for LSI-11 type modules such as processors, memories, and interfaces to communicate with each other. Not all of the bus functions are supported by the SBC-11/21. Only the supported functions are described in this chapter. For a more complete treatment of the LSI-11 bus the reader is referred to the PDP-11 Bus Handbook.

The LSI-11 bus has 40 signal lines, 18 are devoted to data and 22 are for control. The SBC-11/21 supports only 16 data lines and 18 control lines.

There are four groups of control lines.

a. Six data transfer control lines:

   BBS7
   BDIN
   BDOUT
   BRPLY
   BSYNC
   BWTBT

b. Four direct memory access control lines:

   BDMGI
   BDMGO
   BDMR
   BSACK

c. Six interrupt control lines:

   BIAKI
   BIAKO
   BIRQ4
   BIRQ5 Not used by SBC-11/21
   BIRQ6 Not used by SBC-11/21
   BIRQ7 Not used by SBC-11/21

d. Six system control lines:

   BDCOK
   BPOK
   BHALT
   BINIT
   BREF Not used by SBC-11/21
   BEVNT

(Refer to Table 9-4 for functional description of these signals.)

Most LSI-11 Bus signals are bidirectional and use terminations for a negated (high) signal level. Modules connect to these lines via high-impedance bus receivers and open collector drivers. The asserted state is produced when a bus driver asserts the line low. Although bidirectional lines are electrically bidirectional (any point along the line can be driven or received), certain lines are functionally unidirectional. These lines communicate to or from a bus master or signal source, but not both. Interrupt acknowledge (BIAK) and direct memory access grant (BDMG) signals are physically unidirectional in a daisy-chain fashion. These signals originate at the processor output signal pins. Each is received on device input pins (BIAKI or BDMGI) and conditionally retransmitted via device output pins (BIAKO or BDMGO). These signals are received from higher priority devices and are retransmitted to lower priority devices along the bus.

9.1 SBC-11/21 SINGLE BOARD COMPUTER

The SBC-11/21 module functions on the LSI-11 bus and can act as a bus-master, a bus slave, or a bus arbitrator and allows a DMA master to access the on board functions. The module supports only 16 data/address lines and terminates the excess lines. It also contains its own on-board memory and accesses the bus for external memory or devices. It should be noted, however, that while accessing its on-board devices the SBC-11/21 asserts bus control signals in the same manner as when communicating with the LSI-11 bus. The memory maps defining on-board and external addressing are described in Chapter 2. The module's microprocessor supports an on board multilevel interrupt structure and the BIRQ4 bus interrupt control line is an active bus interrupt with a level four priority. Therefore the BIRQ5, BIRQ6, and BIRQ7 bus control interrupt lines are not recognized or accepted by the SBC-11/21 module. The DMA request is recognized by the module at the lowest interrupt level, but once the DMA master has accessed the bus, there are no other interrupts until the transfer is complete or the DMA master relenquishes the bus. The module does not use or support the BREF control line for refreshing dynamic memory.

9.2 MASTER/SLAVE RELATIONSHIP

Communication between devices on the bus is asynchronous. A master/slave relationship exists throughout each bus transaction. At any time, there is 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. The bus master (typically the processor or a DMA device) initiates a bus transaction. The slave device responds by acknowledging the transaction in progress and by receiving data from, or transmitting data to, the bus master. LSI-11 Bus control signals transmitted or received by the bus master or bus slave device must complete the sequence according to bus protocol.

The processor controls bus arbitration, i.e., who becomes bus master at any given time. 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 disk, as master, transferring data to memory as slave. Communication on the LSI-11 Bus is interlocked so that for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the LSI-11 Bus asynchronous. The asynchronous operation precludes the need for synchronizing with, and waiting for, clock pulses.

Since bus cycle completion by the bus master requires response from the slave device, each bus master must include a time-out error circuit that will abort the bus cycle if the slave device does not respond to the bus transaction within 10 microseconds.

The actual time before a time-out error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus.

The signal assignments are shown in Table 9-1.

Table 9-1 Signal Assignments

Number of Pins	Functional Category	Signal Names
16	Data/Address	BDAL0, BDAL1, BDAL2...BDAL15,
6	Data Control	BDOUT, BRPLY, BDIN, BSYNC, BWTBT, BBS7
3	Interrupt Control	BIRQ4, BIAKO, BIAKI
4	DMA Control	BDMR, BDGO, BDMGI, BSACK
5	System Control	BHALT, BDCOK, BPOK, BEVNT, BINIT
3	+5 Vdc	
2	+12 Vdc	
2	-12 Vdc	
1	+5 B (battery)	
8	GND	
8	SSPARES	
4	MSPARES	
2	PSPARES	

9.3 DATA TRANSFER BUS CYCLES

Data transfer bus cycles are listed and defined in Table 9-2.

NOTE

The SBC-11/21 microcomputer performs a read transaction before every write transaction. It does not perform DATIO or DATIOB bus transactions as one address. It executes read-modify-write instructions by addressing the source as one transaction and addressing the destination as another transaction.

Table 9-2 Data Transfer Operations

Bus Cycle Mnemonic	Description	Function (with Respect to the Bus Master)
DATI	Data word input	Read
DATO	Data word output	Write
DATOB	Data byte output	Write byte

These bus cycles, executed by bus master devices, transfer 16-bit words or 8-bit bytes to or from slave devices. The bus signals listed in Table 9-3 are used in the data transfer operations described in Table 9-2.

Table 9-3 Bus Signals Used in Data Transfer Operations

Mnemonic	Description	Function
BDAL <15:00> L	16 Data/address lines	BDAL<15:00> L are used for word and byte transfers.
BSYNC L	Bus Cycle Control	Strobe signal.
BDIN L	Data input indicator	Strobe signal.
BDOUT L	Data output indicator	Strobe signal.
BRPLY L	Slave's acknowledge of bus cycle	Strobe signal.
BWTBT L	Write/byte control	Control signal.
BBS7	I/O device select Indicates address is in the I/O page	Control signal.

Data transfer bus cycles can be reduced to two basic types: DATI, and DATO(B). These transactions occur between the bus master and one slave device selected during the addressing portion of the bus cycle.

9.3.1 Bus Cycle Protocol

Before initiating a bus cycle, the previous bus transaction must have been completed (BSYNC L negated) and the device must become bus master. The bus cycle can be divided into two parts, an addressing portion, and a data transfer portion. During the addressing portion, the bus master outputs the address for the desired slave device, memory location or device register. The selected slave device responds by latching the address bits and holding this condition for the duration of the bus cycle until BSYNC L becomes negated. During the data transfer portion, the actual data transfer occurs.

9.3.1.1 Device Addressing -- The device addressing portion of a data transfer bus cycle comprises an address setup and deskew time and an address hold and deskew time. During the address setup and deskew time, the bus master does the following:

- Asserts BDAL <15:00> L with the desired slave device address bits
- Asserts BBS7 L if a device in the I/O page (56--64 Kb for SBC-11/21) is being addressed. Devices in the I/O page ignore BDAL <15:13>, and decode BBS7 L along with BDAL <12:00>.
- Asserts BWTBT L if the cycle is a DATO(B) bus cycle. Inactive BWTBT L indicates a DATI or DATIO (B) operation.
- Asserts BSYNC at least 150 ns after BDAL <15:00> L, BBS7 L and BWTBT L are valid.

The address, BBS7 L, and BWTBT L signals must be asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. The address hold and deskew time begins after BSYNC L is asserted.

The slave device uses the active BSYNC L bus receiver output to clock BDAL address bits, BBS7 L, and BWTBT L into its internal logic. BDAL <15:00> L, BBS7 L, and BWTBT L will remain active for 25 ns (minimum) after BSYNC L bus receiver goes active. BSYNC L remains active for the duration of the bus cycle.

Memory devices generally do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read-only memory bootstraps or diagnostics, etc.

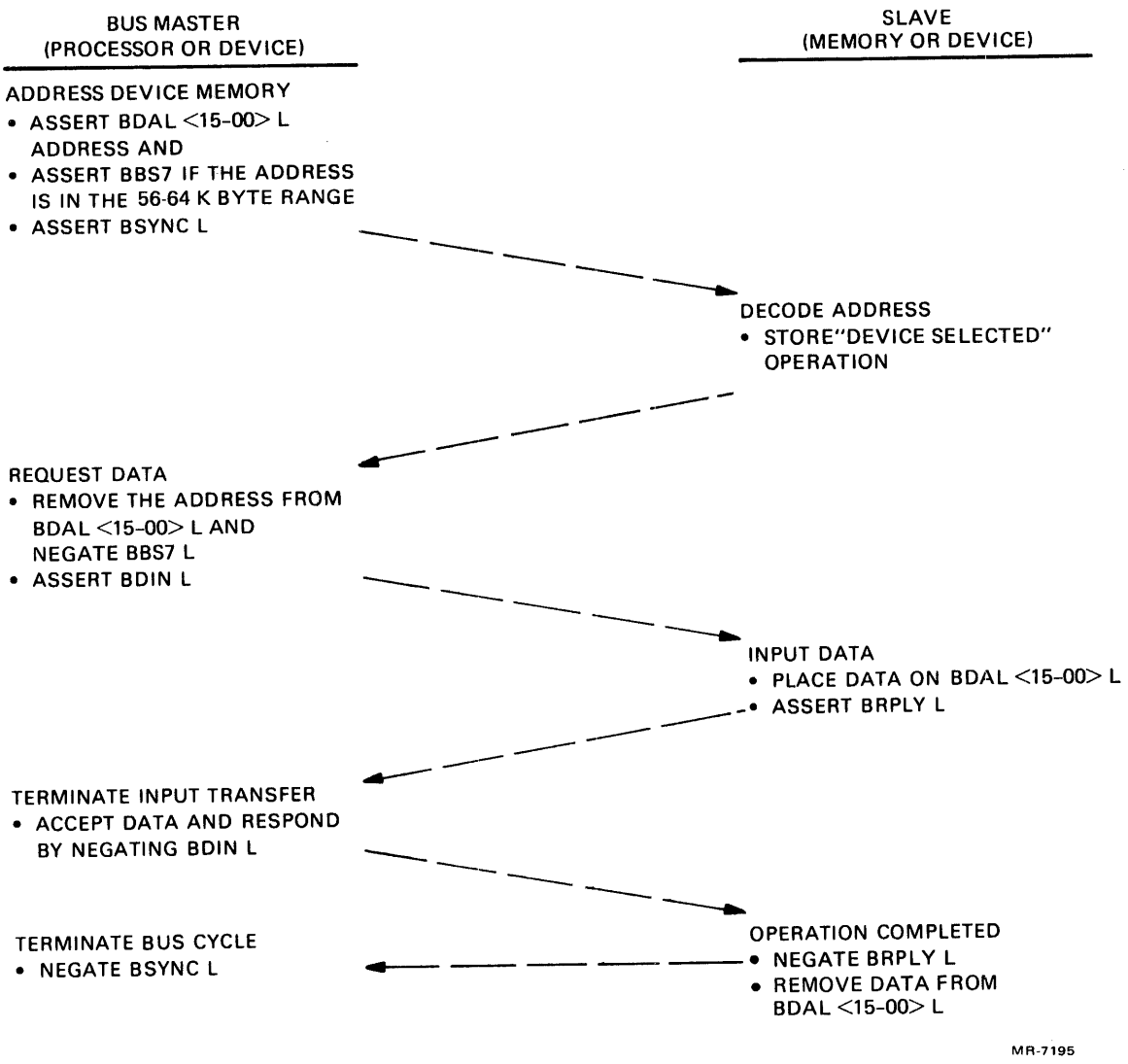


Figure 9-1 DATI Bus Cycle

DATI -- The DATI bus cycle, illustrated in Figure 9-1, is a read operation. During DATI, data are input to the bus master. Data consist of 16-bit word transfers over the bus. During the data transfer portion of the DATI bus cycle, the bus master asserts BDIN L 100 ns minimum after BSYNC L is asserted. The slave device responds to BDIN L active as follows:

- Asserts BRPLY L after receiving BDIN L and 125 ns (maximum) before BDAL bus driver data bits are valid
- Asserts BDAL <15:00> L with the addressed data

When the bus master receives BRPLY L, it does the following:

- Waits at least 200 ns deskew time and then accepts input data at BDAL <15:00> L bus receivers.
- Negates BDIN L 150 ns (minimum) to 2 microseconds (maximum) after BRPLY L goes active.

NOTE

Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted, which will cause the master to keep BSYNC L asserted.

The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drives. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

Conditions for the next BSYNC L assertion are as follows:

- BSYNC L must remain negated for 200 ns (minimum)
- BSYNC L must not become asserted within 300 ns of previous BRPLY L negation

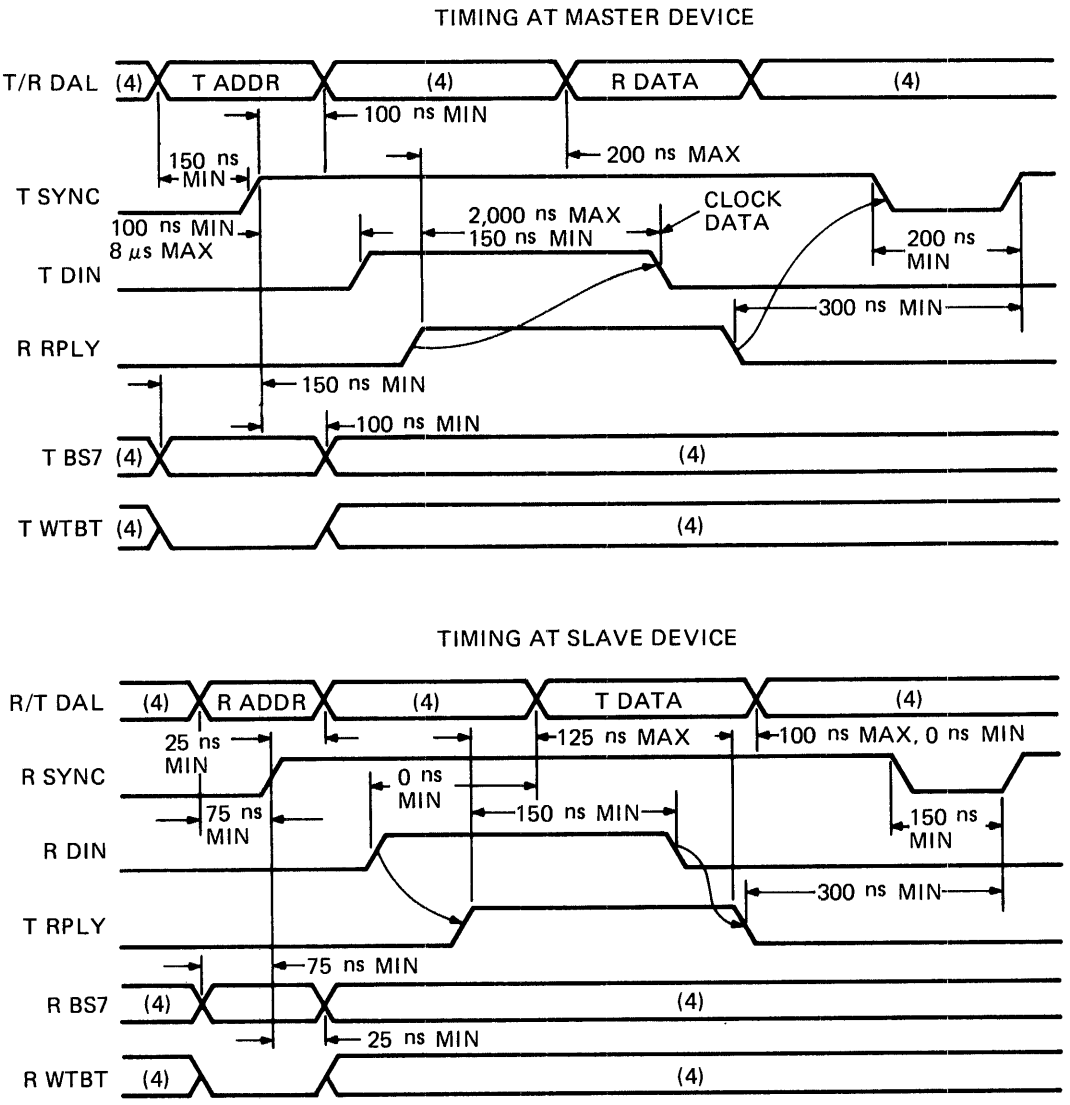
Figure 9-2 illustrates DATI bus cycle timing.

DATO(B) -- DATO(B), illustrated in Figure 9-3, is a write operation. Data is transferred in 16-bit words (DATO) or 8-bit bytes (DATOB) from the bus master to the slave device. The data transfer output can occur after the addressing portion of a bus cycle when BWTBT L has been asserted by the bus master.

The data transfer portion of a DATO(B) bus cycle comprises a data setup and deskew time and a data hold and deskew time.

During the data setup and deskew time, the bus master outputs the data on BDAL <15:00> L at least 100 ns after BSYNC L is asserted. If it is a word transfer, the bus master negates BWTBT L at least 100 ns after BSYNC L assertion. BWTBT L remains negated for the length of the bus cycle. If the transfer is a byte transfer, BWTBT L remains asserted. During a byte transfer, BDAL 00 L selects the high or low byte. This occurs while in the addressing portion of the cycle. If asserted, the high byte (BDAL <15:08> L) is selected; otherwise, the low byte (BDAL <07:00> L) is selected. The bus master asserts BDOUT L at least 100 ns after BDAL and BWTBT L bus drives are stable. The slave device responds by asserting BRPLY L within 10 microseconds to avoid bus time-out. This completes the data setup and deskew time.

During the data hold and deskew time the bus master receives BRPLY L and negates BDOUT L. BDOUT L must remain asserted for at least 150 ns from the receipt of BRPLY L before being negated by the bus master. BDAL <15:00> L bus drivers remain asserted for at least 100 ns after BDOUT L negation. The bus master then negates BDAL inputs.



NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS

2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT

3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX

4. DON'T CARE CONDITION

MR-7180

Figure 9-2 DATI Bus Cycle Timing

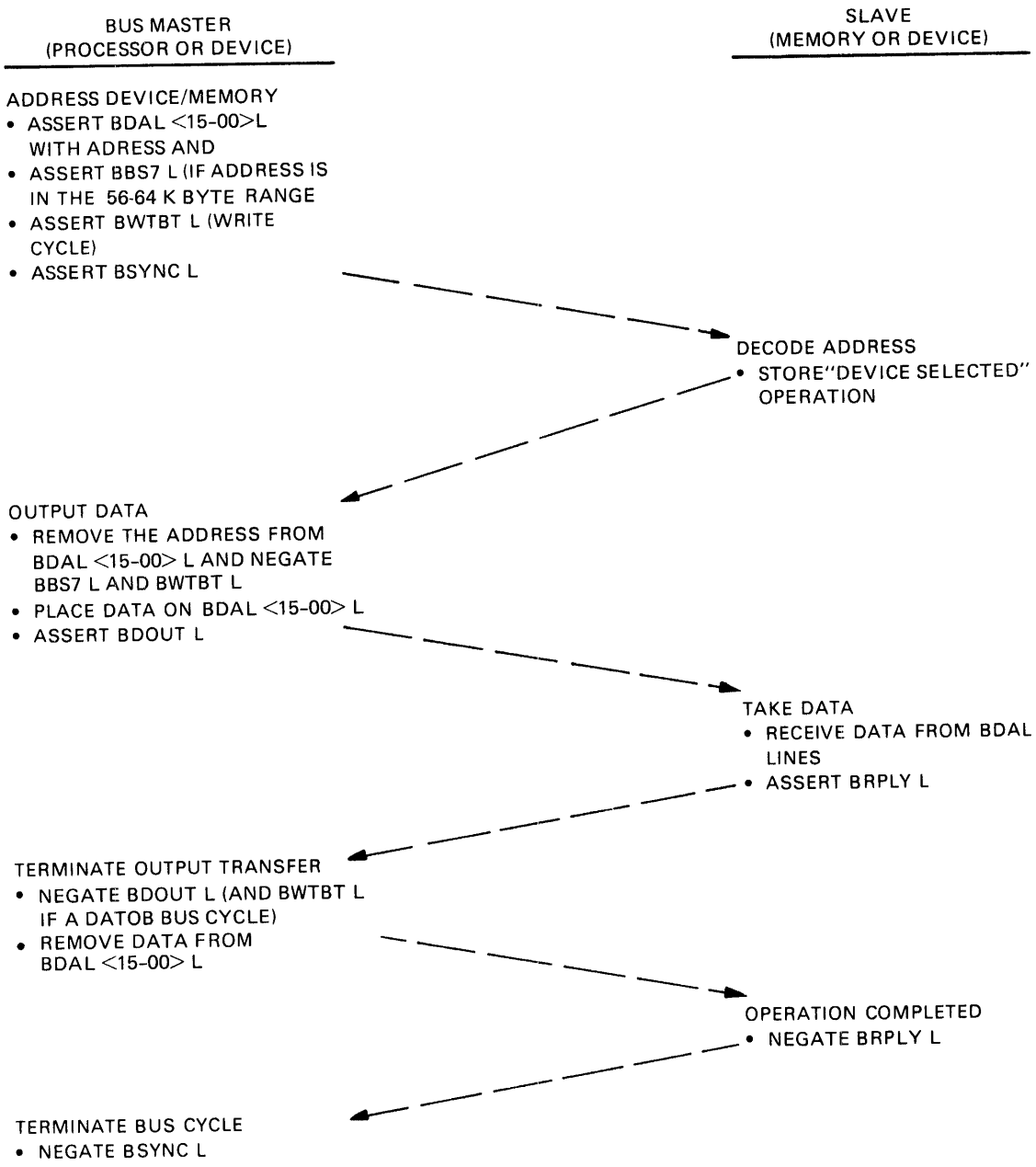
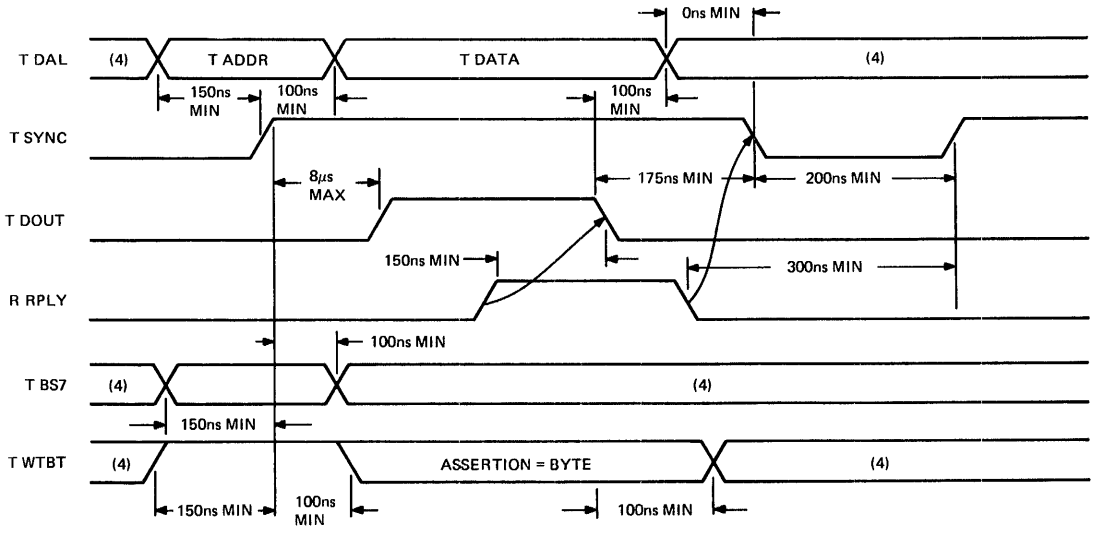
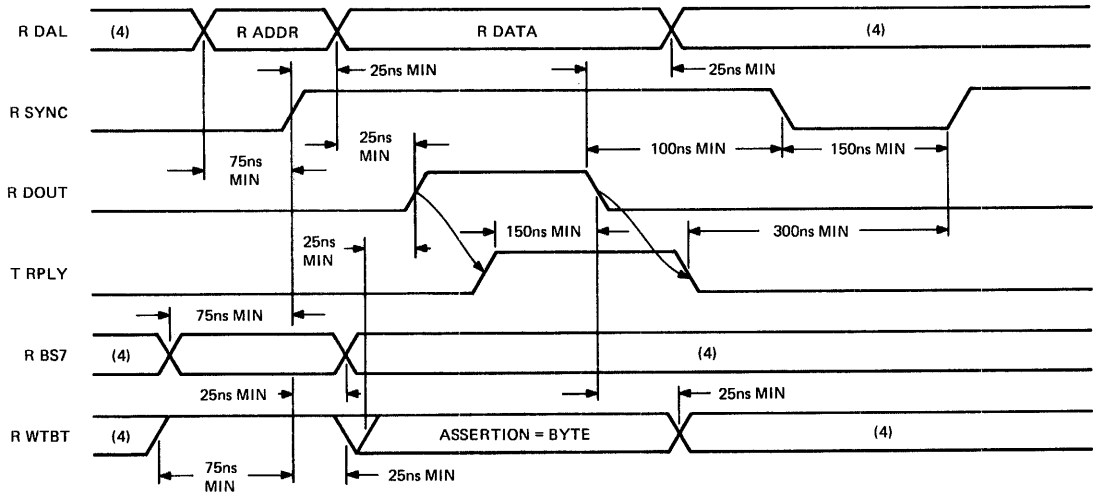


Figure 9-3 DATO or DATOB Bus Cycle MR-7196

During this time, the slave device senses BDOUT L negation. The data are accepted and the slave device negates BRPLY L. The bus master responds by negating BSYNC L. However, the processor will not negate BSYNC L for at least 175 ns after negating BDOUT L. This completes the DATO(B) bus cycle. Before the next cycle BSYNC L must remian unasserted for at least 200 ns. Figure 9-4 illustrates DATO(B) bus cycle timing.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR-1179

Figure 9-4 DATO or DATOB Bus Cycle Timing

9.3.2 Direct Memory Access

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device located electrically closest to it. A DMA device remains bus master indefinitely until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA Grant Input
BDMGO L	DMA Grant Output
BDMR L	DMA Request Line
BSACK L	Bus Grant Acknowledge

A DMA transaction can be divided into three phases:

- Bus mastership acquisition phase
- Data transfer phase
- Bus mastership relinquish phase

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L. The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This this is processor-dependent. BDMGO L/BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus will be hung.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it receives BDMGI L and its BSYNC L and BRPLY L become negated.

During the bus mastership relinquish phase, the DMA device relinquishes the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure 9-5 illustrates the DMA protocol and

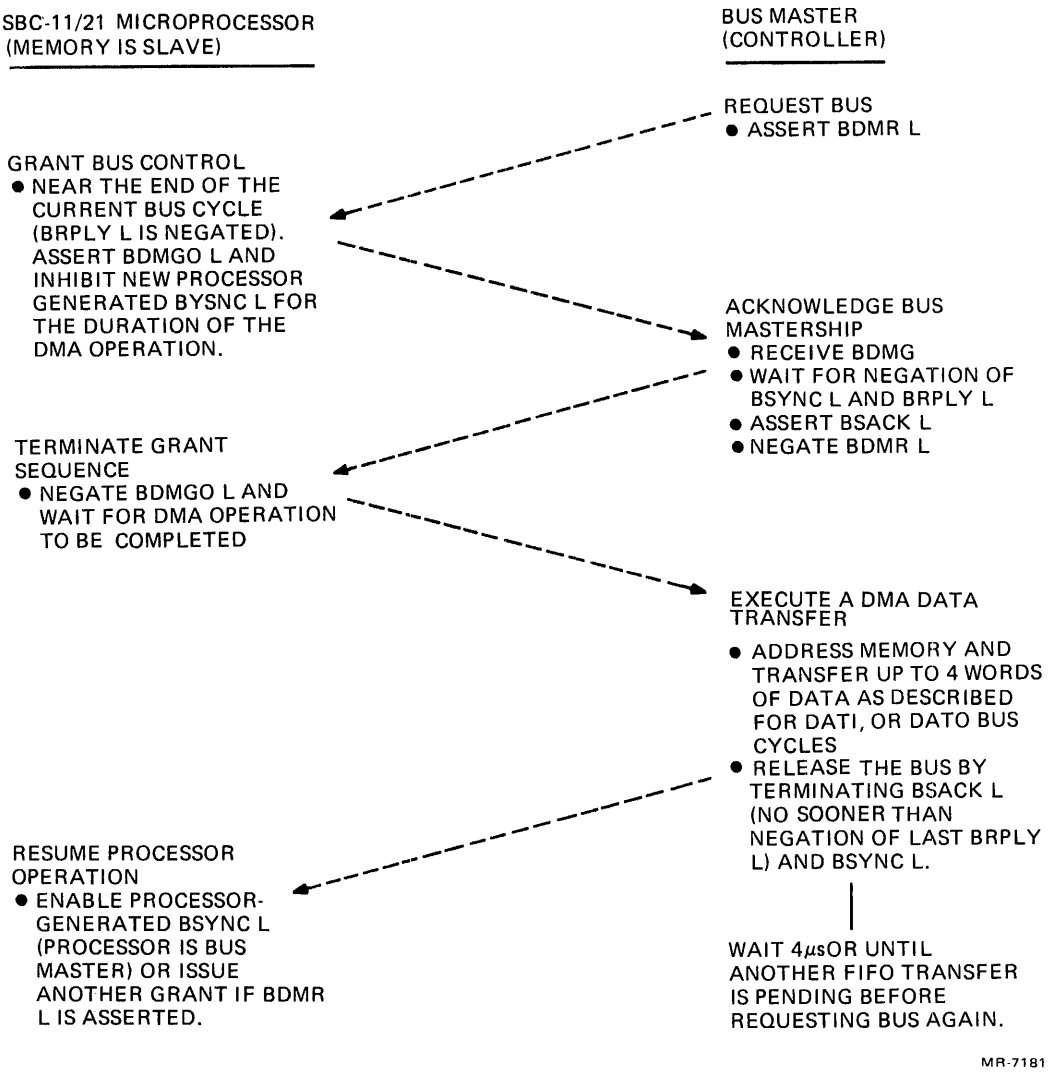


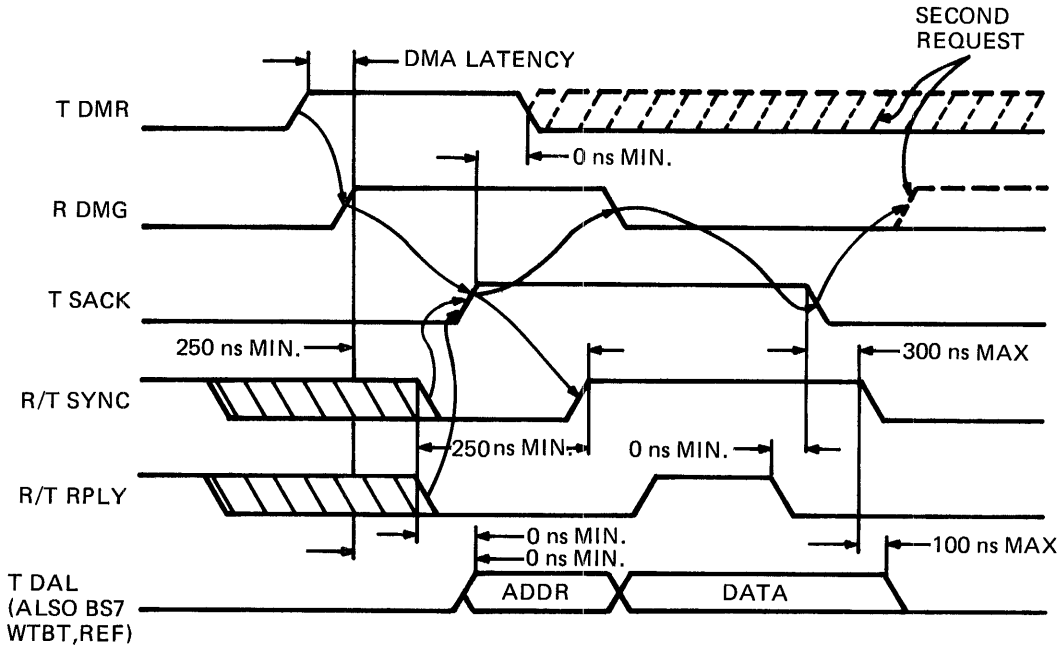
Figure 9-5 DMA Protocol

Figure 9-6 illustrates DMA request/grant timing.

9.4 INTERRUPTS

The LSI-11 Bus signals used in interrupt transactions are:

BIRQ4 L	Interrupt request priority level 4
BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output
BDAL <15:00> L	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply



NOTES:

1 TIMING SHOWN AT REQUESTING DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.

2 SIGNAL NAME PREFIXES ARE DEFINED BELOW
T = BUS DRIVER INPUT
R = BUS RECEIVER OUTPUT

3 BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX

MR-7178

Figure 9-6 DMA Request/Grant Timing

9.4.1 Device Priority

The SBC-11/21 supports only one method of device priority arbitration: position-defined arbitration -- priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

9.4.2 Interrupt Protocol

Interrupt protocol on the SBC-11/21 has three phases: interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase. Figure 9-7 illustrates the interrupt request/acknowledge sequence.

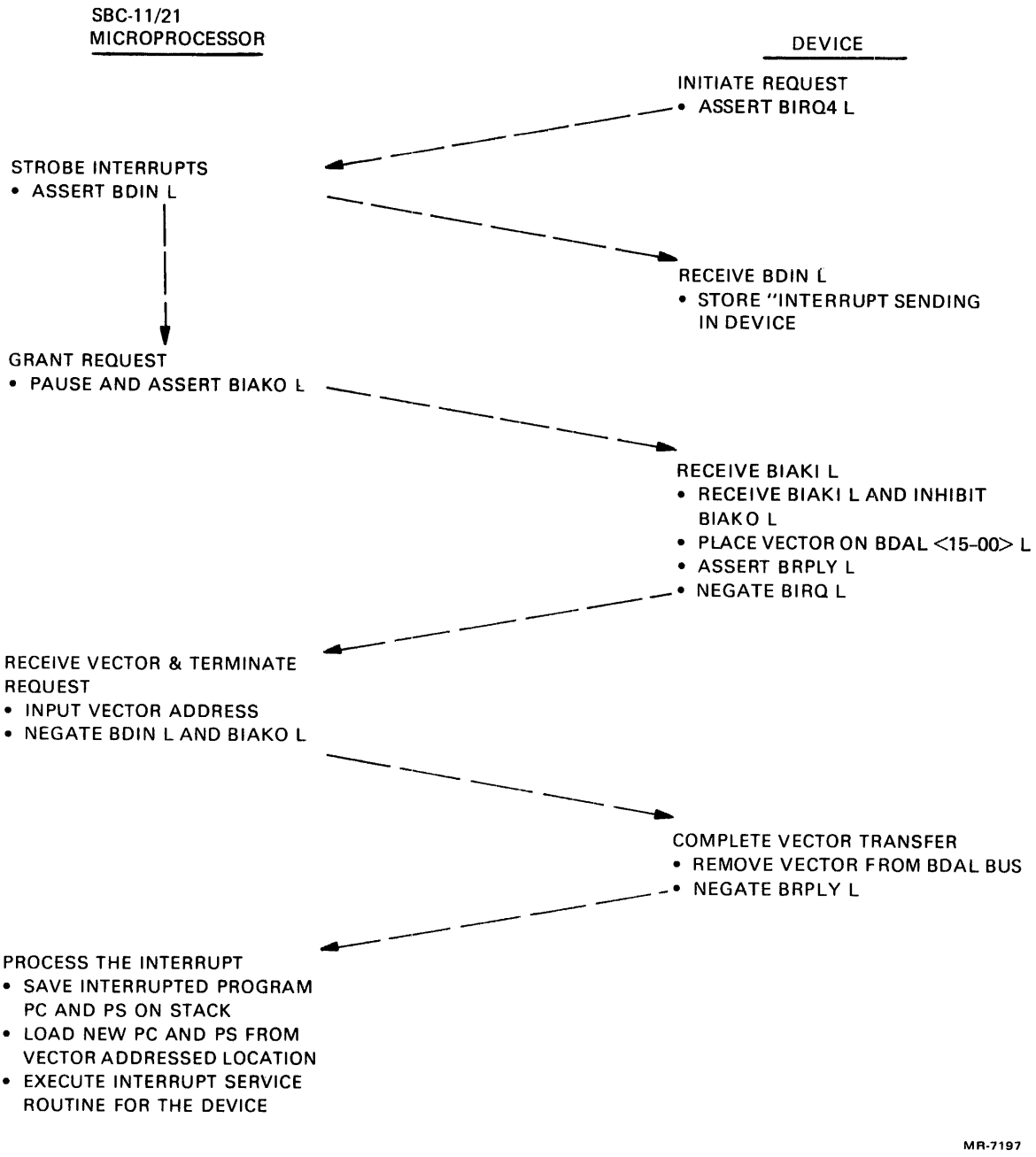


Figure 9-7 Interrupt Request/Acknowledge Sequence

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line. BIRQ4 L is the only hardware priority level on SBC-11/21 and is asserted for all interrupt requests.

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase the SBC-11/21 processor will acknowledge interrupts under the following conditions:

1. The device interrupt priority is higher than the current PS<7:5>.

2. The processor has completed instruction execution and no additional bus cycles are pending.

The processor acknowledges the interrupt request by asserting BDIN L, and 225 ns (minimum) later asserting BIAKO L. The device electrically closest to the processor receives the acknowledge on its BIAKI L bus receiver.

When the device receives the acknowledge, it reacts as follows:

- If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.

- If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L and arbitration is won. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL <15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and 100 ns (maximum) later removes the vector address bits. The processor then enters the device's service routine.

NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per LSI-11 Bus slot.

The device must assert BRPLY L within 10 microseconds (maximum) after the processor asserts BIAKI L.

9.5 CONTROL FUNCTIONS

The following LSI-11 Bus signals provide control functions.

BHALT L	Processor halt
BINIT L	Initialize
BPOK H	Power OK
BDCOK H	DC power OK
BEVNT L	External Event

9.5.1 Halt

Refer to Chapter 2 for explanation of BHALT L response.

9.5.2 Initialization

Devices along the bus are initialized when BINIT L is asserted. The microprocessor can assert BINIT L as a result of executing a RESET instruction or as part of a power-up sequence. BINIT L is asserted for approximately 17 microseconds when RESET is executed.

9.5.3 Power Status

Power status protocol is controlled by two signals, BPOK H and BDCOK H. These signals are driven by some external device (usually the power supply).

BDCOK H -- When asserted, this indicates that dc power has been stable for at least 3 ms. Once asserted, this line remains asserted until the power fails. Its negation indicates that only 5 microseconds of dc power reserve remains.

BPOK H -- When asserted this indicates that there is at least an 8 ms reserve of dc power and that BDCOK H has been asserted for at least 70 ms. Once BPOK H has been asserted, it must remain asserted for at least 3 ms. The negation of this line, the first event in the power-fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remains.

9.5.4 Power-Up/Down Protocol

Power-up protocol (Figure 9-8) begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply or other external device asserts BDCOK H. The processor responds by clearing the PS. BINIT L is asserted for 17 microseconds and then negated for 110 microseconds. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPOK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3.0 ms before a power-down sequence can begin.

A power-down sequence begins when the power supply negates BPOK H. When the current instruction is completed, the microprocessor traps to a power-down routine at location 24. The routine must provide for loading 340 into the PSW, execute a RESET instruction and must terminate in a WAIT instruction or branch on itself.

There should be no DMA requests issued after the RESET is executed. This procedure prevents any possible memory corruption in the battery supported system as the dc voltages decay.

NOTE

SBC-11/21 does not generate BINIT L during the power-down sequence. The power-down routine must therefore include a RESET instruction to set bus devices into a known state.

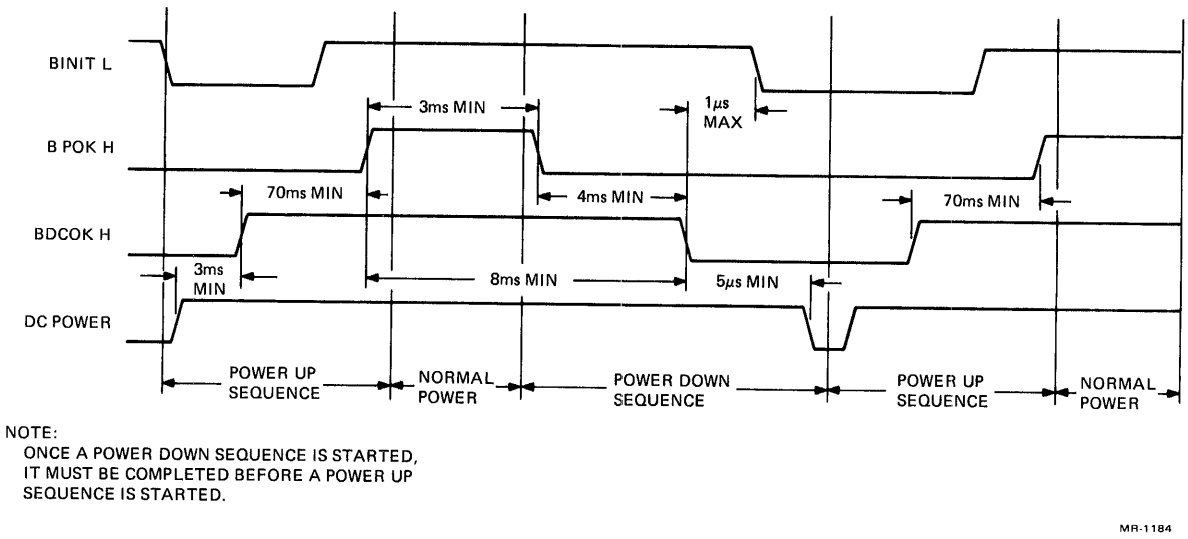


Figure 9-8 Power-Up/Power-Down Timing

9.6 LSI-11 BUS ELECTRICAL CHARACTERISTICS

Configuring LSI-11 bus systems requires an appreciation of its transmission line characteristics which can be best obtained from the PDP-11 Bus Handbook.

9.7 MODULE CONTACT FINGER IDENTIFICATION

DIGITAL plug-in modules, including the SBC-11/21, all use the same contact finger (pin) identification system. The LSI-11 bus is based on the use of double-height modules that plug into a 2-slot bus connector. Each slot contains 36 lines (18 each on component and solder sides of circuit board).

Slots, shown as row A and row B in Figure 9-9, include a numeric identifier for the side of the module. The component side is designated side 1 and the solder side is designated side 2. Letters ranging from A through V (excluding G, I, O, and Q) identify a particular pin on a side of a slot. Table 9-4 lists and identifies the bus pins of the double-height module. For a quick summary refer to Table 1-1. The bus pin identifier ending with a 1 is found on the component side of the board, while a bus pin identifier ending with a 2 is found on the solder side of the board. A typical pin is designated as follows.

AE2: Row A, pin E, Side 2

The positioning notch between the two rows of pins mates with a protrusion on the connector block for correct module positioning.

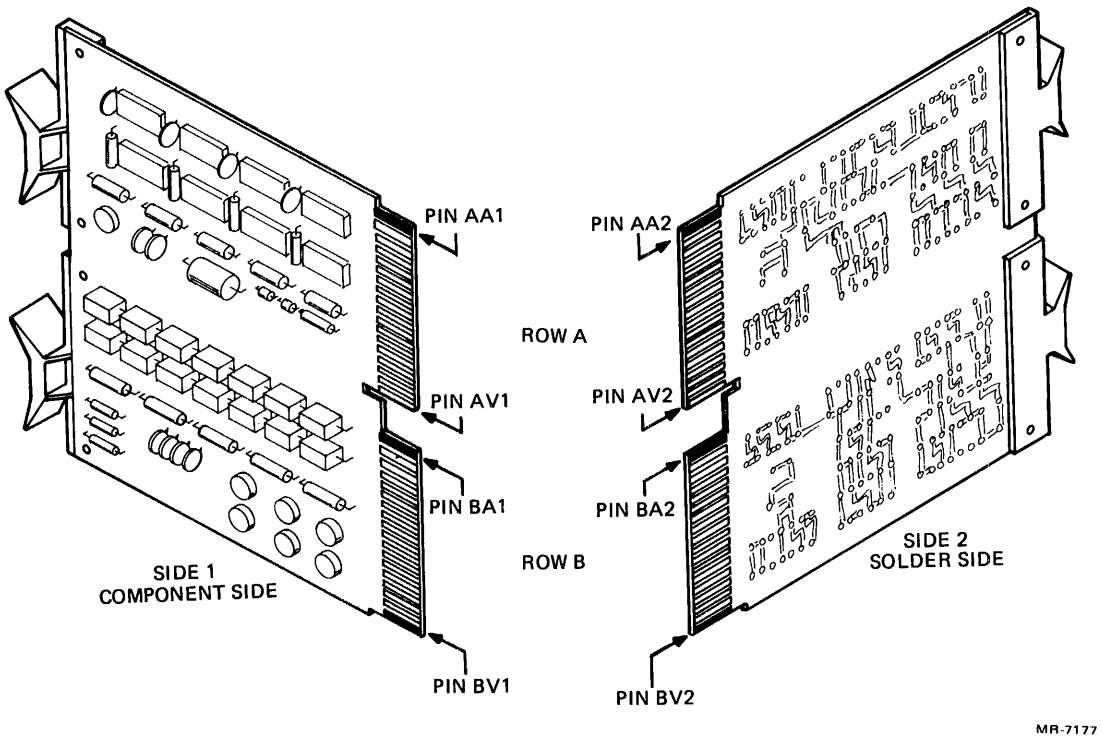


Figure 9-9 Double-Height Module Contact Finger Identification

Table 9-4 Bus Pin Identifiers

Bus Pin	Mnemonics	Description
AE1	SSPARE1 (Alternate +5B)	Special Spare -- not assigned or bused in DIGITAL cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5 V battery (+5B) backup power to keep critical circuits alive during power failures. A jumper is required on LSI-11 bus options to open (disconnect) the +5B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special Spare -- not assigned or bused in DIGITAL cable or backplane assemblies; available for user interconnection.
AJ1	GND	Ground -- System signal ground and dc return.
AK1	MSPAREA	
AL1	MSPAREA	Maintenance Spare -- Normally connected together on the backplane at each option location (not bused connection).
AM1	GND	Ground -- System signal ground and dc return.
AN1	BDMR L	Direct Memory Access (DMA) Request -- A device asserts this signal to request bus mastership. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor Halt -- Refer to Chapter 2.
AT1	GND	Ground -- System signal ground and dc return.
AU1	PSPARE 1	Spare (Not assigned. Customer usage not recommended.) Prevents damage when modules are inserted upside down.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonics	Description
AV1	+5B	+5 V Battery Power -- Secondary +5 V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC Power OK -- Power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation.
BB1	BPOK H	Power OK -- Asserted by the power supply 70 ms after BDCOK. Negated when ac power drop below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power fail trap sequence is initiated.
BH1	SSPARE8	Special Spare -- Not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground -- System signal ground and dc return.
BK1	MSPAREB	
BL1	MSPAREB	Maintenance Spare -- Normally connected together on the backplane at each option location (not a bused connection).
BM1	GND	Ground -- System signal ground and dc return.
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BR1	BEVNT L	External Event Interrupt Request -- When asserted, the processor responds (if PS bit 7 is 0) by entering a service routine via vector address 100. A typical use of this signal is a line time clock interrupt.
BS1	PSPARE 4	Power Spare 4 (Not assigned a function, not recommended for use).

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonics	Description
BT1	GND	Ground -- System signal ground and dc return.
BU1	PSPARE2	Power Spare 2 (not assigned a function, not recommended for use). If a module is using -12 V (on pin AB2) and if the module is accidentally inserted upside down in the backplane, -12 Vdc appears on pin BU1.
BV1	+5	+5 V Power -- Normal +5 Vdc system power.
AA2	+5	+5 V Power -- Normal +5 Vdc system power.
AB2	-12	-12 V Power -- -12 Vdc (optional) power for devices requiring this voltage. NOTE LSI-11 modules which require negative voltages contain an inverter circuit (on each module) which generates the required voltage(s). Hence, -12 V power is not required with DIGITAL-supplied options.
AC2	GND	Ground -- System signal ground and dc return.
AD2	+12	+12 V Power -- 12 Vdc system power.
AE2	BDOUT L	Data Output -- BDOUT, when asserted, implies that valid data is available on BDAL <0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOUT L is deskewed with respect to data on the bus. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.
AF2	BRPLY L	Reply -- BRPLY L is asserted in response to BDIN L or BDOUT L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonics	Description
AH2	BDIN L	Data Input -- BDIN L is used for two types of bus operation: When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master, and requires a response (BRPLY L). BDIN L is asserted when the master device is ready to accept data from a slave device. When asserted without BSYNC L, it indicates that an interrrupt operation is occurring. The master device must deskew input data from BRPLY L.
AJ2	BSYNC L	Synchronize -- BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL <0:15> L. The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	Write/Byte -- BWTBT L is used in two ways to control a bus cycle: It is asserted at the leading edge of BSYNC L to indicate that an output sequence is to follow (DATO or DATOB), rather than an input sequence. It is asserted during BDOUT L, in a DATOB bus cycle, for byte addressing.
AL2	BIRQ4 L	Interrupt Request Priority Level 4 -- A level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PS word bit 7 is 0, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonics	Description
AM2 AN2	BIAKI L BIAKO L	Interrupt Acknowledge -- In accordance with interrupt protocol, the processor asserts BIAKO L to acknowledge receipt of an interrupt. The bus transmits this to BIAKI L of the device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions: 1. The device requested the bus by asserting BIRQ4L, and 2. the device has the highest priority interrupt request on the bus at that time. If these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy-chain fashion until the device with the highest interrupt priority receives the interrupt acknowledge signal.
AP2	BBS7 L	Bank 7 Select -- The bus master asserts this signal to reference the I/O page (including that portion of the I/O page reserved for nonexistent memory). The address in BDAL <0:12> L when BBS7 L is asserted is the address within the I/O page.
AR2 AS2	BDMGI L BDMBO L	Direct Memory Access Grant -- The Bus arbitrator asserts this signal to grant bus mastership to a requesting device, according to bus mastership protocol. The signal is passed in a daisy-chain from the arbitrator (as BDMGO L) through the bus to BDMGI L of the next priority device (electrically closest device on the bus). This device accepts the grant only if it requested to be bus master (by a BDMR L). If not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonics	Description
AT2	BINIT L	Initialize -- This signal is used for system reset. All devices on the bus are to return to a known, initial state; i.e., registers are reset to zero, and logic is reset to state 0. Exceptions should be completely documented in programming and engineering specifications for the device.
AU2	BDAL0 L	Data/Address Lines -- These two lines are part of the 16-line data/address bus over which address and data information are communicated. Address information is first placed on the bus by the bus master device. The same device then either receives input data from, or outputs data to the addressed slave device or memory over the same bus lines.
AV2	BDAL1 L	
BA2	+5	+5 V Power -- Normal +5 Vdc system power.
BB2	-12	-12 V Power -- -12 Vdc (optional) power for devices requiring this voltage.
BC2	GND	Ground -- System signal ground and dc return.
BD2	+12	+12 V Power -- +12 V system power.
BE2	BDAL2 L	Data/Address Lines -- These 14 lines are part of the 16-line data/address bus previously described.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

INSTRUCTION TIMING

The following fetch and execute times assume that the SBC-11/21 is transacting with local devices that do not require cycle slips when accessed.

Single Operand Instructions	Destination Mode	Fetch and Execute Time (usec)	Number of Bus Transactions	Number of Microcycles
CLR(B), COM(B),	0	2.44	1	4
INC(B), DEC(B),	1	4.27	3	7
NEG(B), ROR(B),	2	4.27	3	7
ROL(B), ASR(B),	3	5.49	4	9
ASL(B), SWAP,	4	4.88	3	8
ADC(B), SBC(B),	5	6.10	4	10
SXT, MFPS,	6	6.10	4	10
XOR	7	7.32	5	12
	0	2.44	1	4
	1	3.66	2	6
	2	3.66	2	6
TST(B)	3	5.49	3	8
	4	4.27	2	7
	5	5.49	3	9
	6	5.49	3	9
	7	6.71	4	11
	0	4.88	1	8
	1	6.10	2	10
	2	6.10	2	10
MTPS	3	7.32	3	12
	4	6.71	2	11
	5	7.93	3	13
	6	7.93	3	13
	7	9.16	4	15

Double Operand Instructions	Source Mode	Source Mode Time (usec) Includes Fetch		
MOV(B), CMP(B),	0	1.83	1	3
ADD, SUB,	1	3.05	2	5
BIT(B), BIC(B),	2	3.05	2	5
BIS(B)	3	4.27	3	7
	4	3.66	2	6
	5	4.88	3	8
	6	4.88	3	8
	7	6.10	4	10

Double Operand Instructions	Destination Mode	Destination Mode Time (usec)		
MOV(B), CMP(B),	0	0.61	0	1
ADD, SUB	1	2.44	2	4
BIT(B), BIC(B),	2	2.44	2	4
BIS(B)	3	3.66	3	6
	4	3.05	2	5
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9
	0	0.61	0	1
	1	1.83	1	3
	2	1.83	1	3
CMP(B), BIT(B)	3	3.05	2	5
	4	2.44	1	4
	5	3.66	2	6
	6	3.66	2	6
	7	4.88	3	8

Jump and Subroutine Instructions	Destination Mode	Fetch and Execute Time (usec)		
	1	3.05	2	5
	2	3.66	2	6
JMP	3	3.66	3	6
	4	3.66	2	6
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9
	1	5.49	4	9
	2	6.10	4	10
JSR	3	6.10	5	10
	4	6.10	4	10
	5	6.71	5	11
	6	6.71	5	11
	7	7.90	6	13
RTS	NA	4.27	2	7
SOB	NA	3.66	1	6

Branch, Trap and Interrupt Instructions	Destination Mode	Fetch and Execute Time (usec)		
BR, BNE, BEQ, BPL, BMI, BVC,	NA	2.44	1	4

BVS, BCC, BCS, BGE, BLT, BGT, BLE, BHI, BLOS, BHIS, BLO				
EMT, TRAP BPT, IOT	NA	9.77	7	16
RTI	NA	4.88	3	8
RTT	NA	6.71	3	11

Miscellaneous and Condition Code Instructions	Destination Mode	Fetch and Execute Time (usec)		
HALT	NA	8.54	5	14
WAIT	NA	2.44	1	4 then loop
RESET	NA	22.28	1	39
NOP	NA	3.66	1	6
CLC, CLV, CLZ, CLN, CCC, SEC, SEV, SEZ, SEN, SCC	NA	3.66	1	6
MFPT	NA	3.05	1	5

The measure of LSI-11 BUS interrupt latency is the time from assertion of BIRQ until BIAKI is accepted by the interrupting device electrically closest to the processor on the LSI-11 BUS.

The measure of local interrupt latency is the time from assertion of the request until the time the microprocessor is ready to fetch the first instruction in the interrupt service routine. This time is primarily comprised of the time to perform two pushes and a PC and PSW restore.

DMA latency is known as the period of time between a device asserting its BDMR and receiving BDMGI when it resides on the LSI-11 BUS as the electrically closest DMA device to the processor.

Interrupt Latency:	LOCAL	23.2 usec
	LSI-11 BUS	9.3 usec

NOTE

Assume that the stack and vector memory reside on the SBC-11/21 and that the LSI-11 BUS device can assert BRPLY and vector within 600 nsec after receiving IAKI. The service latency (time from BIRQ until the time the microprocessor is ready to fetch the first instruction in the interrupt service routine) depends on the response time of the interrupting device i.e., RDIN to TRPLY and negation of TRPLY.

DMA latency: 1.3 usec minimum 11.0 usec maximum

WAIT instruction latencies:

internal vector	11.8 usec
external vector	12.4 usec
DMA	5.06 usec

APPENDIX B
PROGRAMMING DIFFERENCE LIST

DIFFERENCE LIST

The following table presents a consise comparison between the SBC-11/21, LSI-11/2, and LSI-11/23.

Activity	SBC-11/21	LSI-11/2	LSI-11/23
OPR %R, (R)+ or OPR %R,-(R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	X		X
OPR %R, @(R)+ or OPR %R,@-(R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	X		X
In the above two cases, initial contents of R are used as the source operand.		X	
OPR PC, X(R); OPR PC,@X(R); OPR PC,@A; OPR PC,A: Location A will contain the PC of OPR + 4.	X		X
In the above case, location A will contain the PC of OPR + 2.		X	
JMP (R)+ or JSR reg,(R)+: Initial contents of R are used as the new PC.	X	X	X
JMP %R or JSR reg,%R traps to 4 (illegal instruction).	X	X	X
Only one LSI-11 bus interrupt level (BR4) exists.	X	X	
Four local interrupt levels exist.	X		
Four LSI-11 interrupt levels exist.			X
Stack overflow not implemented.	X	X	
A stack overflow trap exists.			X

The first instruction in an interrupt routine will not be executed if another interrupt occurs at a higher priority level than assumed by the first interrupt.	X	X	X
Eight general purpose registers.	X	X	X
PSW address, 177776, not implemented. Must use MTPS and MFPS instructions.	X	X	
Only implicit references (RTI, RTT, traps and interrupts) can load T bit. Console cannot load T bit.	X	X	X
If an interrupt occurs during an instruction that has the T bit set, the T bit trap is acknowledged before the interrupt.	X	X	X
If RTI sets the T bit, T bit trap is acknowledged immediately following RTI.	X	X	X
T bit trap will sequence out of WAIT instruction.	X		X
If RTT sets the T bit, the T bit trap occurs after the instruction following RTT.	X	X	X
RESET instruction consists of 10 usec of INIT followed by a 90 usec pause. Power fail not recognized until the instruction is complete.		X	X
RESET instruction consists of 17 usec of INIT followed by a minimum 3.2 μsec pause. Power fail not recognized until the instruction is complete.	X		
Odd address references using the SP do not trap.	X		
Non-existent address references using the SP trap to the restart address.	X		

MOVB instruction does a READ (DATIP) and a WRITE (DATO) bus sequence for last memory cycle.		X
MOV instruction does a WRITE (DATO) bus sequence for the last memory cycle.		X X
MOV instruction does a READ (DATI) and a WRITE (DATO) bus sequence for last memory cycle.	X	

From	Through	Response	11/21	11/2	11/23	
210	217	Trap to 10	X	NOTE 1	X	Reserved Instr.
210	227	Trap to 10	X	X	X	Reserved Instr.
70000	73777	Trap to 10	X	NOTE 2	NOTE 2	Extended Instr. Set
75000	75037	Trap to 10	X	X	NOTE 2	Floating point
75040	75777	Trap to 10	X	NOTE 2	X	Reserved Instr.
170000	177777	Trap to 10	X	NOTE 2	NOTE 2	Reserved Instr.

NOTE 1: Maintenance instructions
NOTE 2: Response depends on processor options

CLR(B) and SXT do a READ (DATI) and a WRITE (DATO) sequence for the last bus cycle.	X		
CLR(B) and SXT do a READ (DATI) and a WRITE (DATO) bus sequence for the last bus cycle.		X	
CLR(B) and SXT do a WRITE (DATO) bus sequence for the last bus cycle.			X
MARK instruction.		X	X
SOB, RTT, SXT, XOR instructions.	X	X	X
SWAB clears V.	X	X	X
ASH, ASHC, DIV, MUL instructions.		X	X

Register addresses (177700-177717) are handled as regular memory addresses. No internal registers are addressable from either the bus or the console.	X		
Register addresses (177000-177717) time-out when used as a program address by the CPU.		X	X
If PC contains a non-existent memory address and a bus error occurs, PC will have been incremented.	X	X	X
If register contains a non-existent memory address in mode 2 and a bus error occurs, register will be incremented.	X	X	X
If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X	X	X
HALT in user mode traps to 10.			X
HALT instruction pushes PC and PSW on the stack and loads the PS with 340 and the PC with the restart address.	X		
Only power-up mode 2 implemented.	X		
Resident ODT microcode.		X	X
Instruction execution runs to completion regardless of bus error.	X		
BEVNT line interrupt on level 6.	X		X
Bus error traps to restart address. Instruction runs to completion before trap.	X		
Bus error during IAK vectors through 0 and traps to restart address. The first instruction of service routine is guaranteed to execute.	X		
Only sixteen-bit addressing supported.	X	X	

The no-BSACK 18 usec time-out implemented. If time-out occurs BDMGO aborted.		X
Bus halt line is a jumper-configured nonmaskable interrupt. Acknowledgement causes PC and PSW to be stacked and the processor vectors thru level 7 internal vector 140.	X	
Vector address accepted only on BDAL<7-2>. This limits vector address space to 374.	X	
Certain vector addresses are reserved for "local" devices other than BEVNT.	X	

SBC-11/21 Priorities

Priority of DMA, system traps, external interrupts, internal interrupts, HALT trap, and WAIT:

DMA	(highest priority)
HALT trap (timeout request)	
Power Fail Trap	
Traps (illegal instruction, T bit, EMT)	
Internal Interrupt Request	
External Interrupt Request	
WAIT instruction	(lowest priority)

APPENDIX C
SOFTWARE DEVELOPMENT

C.0 GENERAL

This section of the manual presents programming hints which may be helpful to application programmers in gaining familiarity with the SBC-11/21. The following three topics are discussed:

- Running stand-alone programs
- The software development process
- An application example

A method of creating, loading and running stand-alone programs is briefly explained. This is followed by a discussion of the software development process, as it applies to a ROM based single board computer. The last section of this appendix presents a practical example of a real-time program written to run on the SBC-11/21. The output chosen for the program is deliberately simple, but the methodology is applicable to programs with a much greater degree of complexity. The program has been thoroughly tested and studying it should prove quite rewarding to the first time users of the SBC-11/21.

C.1 RUNNING STAND-ALONE PROGRAMS

The user can develop stand-alone programs, that is programs not requiring an operating system, on a separate RT-11 based system. The .SAV image can then be loaded into the SBC-11/21 and run. Macro-ODT option is required to load the program and to run it.

To be used with Macro-ODT, the stand-alone program must have the address of Macro-ODT BREAK service routine in location 140 and a PSW value of 300 in location 142. This will enable the program to transfer control to Macro-ODT when the BREAK key is depressed.

In order to be able to load the stand-alone program from the mass storage device into the SBC-11/21, the device's boot block must be modified. This modification extends to locations 0, 2, 4 and 6. Location 0 normally contains 240 and must be changed to 260. When the device is booted, this tells the Macro-ODT that the mass storage device contains a stand-alone program. Macro-ODT will then interpret the contents of locations 2, 4 and 6 as a RADIX-50 encoded six character file name, and search the directory of the volume for that file. The volume must have the RT-11 file structure. When the file is found, the entire file is loaded into contiguous memory starting at location zero. Then Macro-ODT loads register R0 with the number of the unit or drive, and register R1 with the CSR address of the booted device.

This information may be of interest to the stand-alone program if it uses overlays. Then the Stack Pointer (SP) is loaded with the contents of location 42, the Program Counter (PC) is loaded with the contents of location 40, and the program begins execution. Please note, that a stand-alone program developed on an RT-11

based system will already have the correct values for PC and SP in locations 40 and 42.

The detailed procedure for performing these modifications in the boot block and the stand-alone program will now be outlined. This will be done on an RT-11 based system using the SIPP utility.

In the examples below, the program that is to be loaded and run from the stand-alone volume is named FOOBAR.SAV and resides on DK. The characters entered by the operator are underlined. "<CR>" is a carriage return and not the four characters "<", "C", "R" and ">". The "^C" and "^Y" symbols are obtained by holding down the "CTRL" key and typing "C" or "Y" respectively before releasing "CTRL". The "XXXXXX" is a string of octal digits whose value can be anything but is not relevant to the process.

First modify the stand-alone program:

```
. R SIPP <CR>                                  ;Run the SIPP utility

* DK:FOOBAR.SAV <CR>                           ;Name of file to be
                                                patched

Base? <CR>                                     ;Defaults to zero

Offset? 140 <CR>

Base    Offset  Old     New?

000000  000140  xxxxxx  170000 <CR>  ;load address of Break
                                      ;routine at Break vector

000000  000142  xxxxxx  300 <CR>      ;PSW during Break routine

000000  000144  xxxxxx  ^Y <CR>       ;Exit patching

*^C                                            ;Exit SIPP
```

NOTE

If you are using your own BREAK intercepting routine, put its address at location 140 in place of the value 170000.

Now modify the bootblock:

```
.R SIPP <CR>

*DK:/A <CR>

Base? <CR>

Offset? <CR>
```

```
Base    Offset  Old     New?

000000  000000  xxxxxx  000260

000000  000002  xxxxxx  ;RFOO <CR>

000000  000004  xxxxxx  ;RBAR <CR>

000000  000006  xxxxxx  ;RSAV <CR>

000000  000010  xxxxxx  ^Y <CR>

*^C
```

C.2 THE SOFTWARE DEVELOPMENT PROCESS

Software development for the SBC-11/21 can be thought of as consisting of four distinct steps as shown in Figure C-1.

1. Design the software and code the source tasks.

2. Enter, edit and assemble the tasks that make up the application.

3. Build the application into a runable memory image.

4. Load the program into the SBC-11/21 and execute the application program. This step will involve the debugging of the application.

C.2.1 Design of the Software

An important consideration in the design of application software is the run-time memory configuration. Since the SBC-11/21 is a ROM/RAM system, the location of the ROM/RAM boundaries must be defined. All instructions and constants must be grouped separately for location in the ROM portion of memory. Variable information must be grouped together for location in the RAM portion of memory. Throughout the development process, the segregation of ROM and RAM information must be maintained. MACRO-11 Language Reference Manual describes methods of data and code separation.

C.2.2 Editing and Assembly

The second step in the development cycle is the entry, editing, and assembly of the application software. To enter and create the application software involves the use of an editor on the development system. Once the application software is entered and the designer is satisfied with the contents, it can be saved on a mass storage device. The assembler must then be used to convert the source code instructions into executable code. The result of the assembly process is an object file.

The assembler detects common assembly language coding errors and issues appropriate warnings. If errors are detected, corrections should be made by re-editing the source and assembling again.

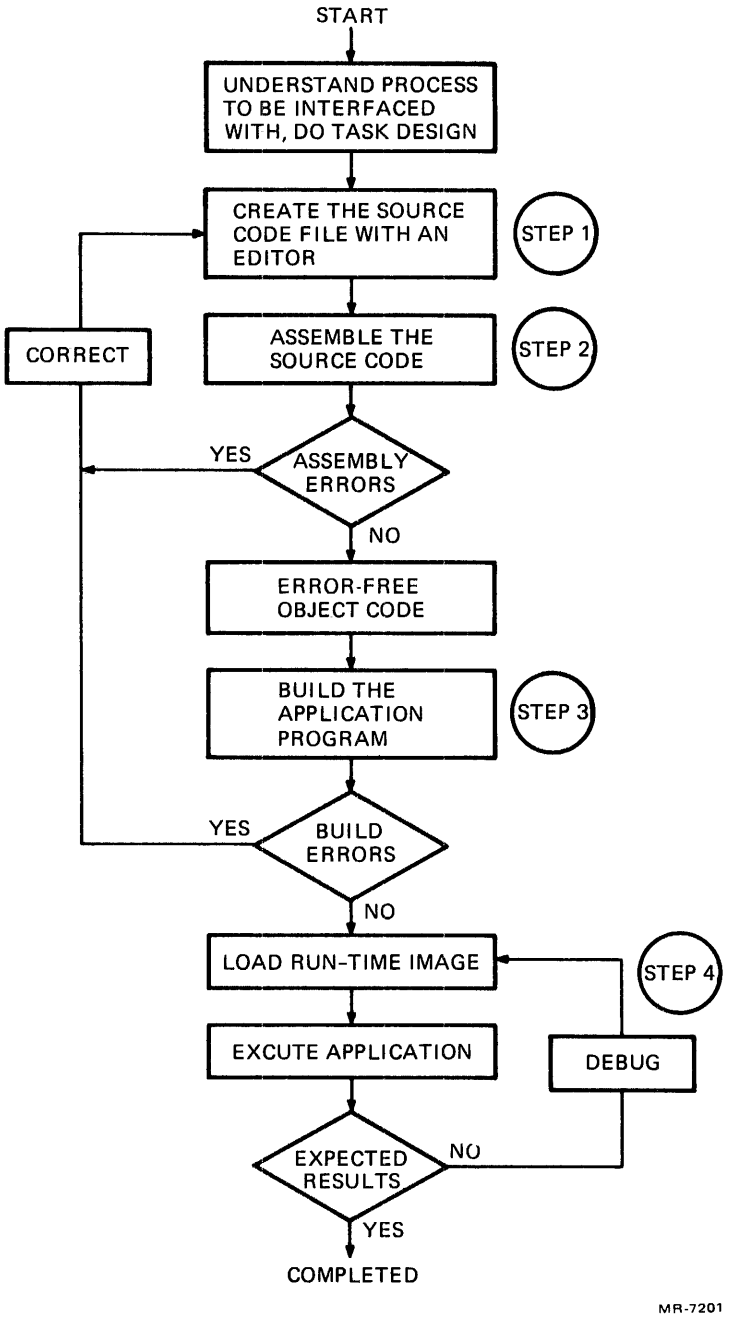


Figure C-1 Overview of Software Development

Once the application software has been translated error-free into object form, it is ready for the next step.

C.2.3 Building Process

The third step in the development cycle is the building process to create a runable memory image. The build process involves the linking of the tasks that make up the application software into a single memory image. The building process takes an object module or modules and assigns absolute memory references to the information contained in the object code. The user assigns these locations by the sectioning of the code that took place during design. The result of the build phase is an executable run-time memory image that can now be loaded and tested.

C.2.4 Running and Debugging the Program

The fourth step in the development cycle is the loading of the runable memory image into the SBC-11/21. Once loaded, the program can be run and debugged. There are three techniques that can be employed to transfer the software to the target. The methods are:

1. ROM transfer. This method involves the programming of ROMS via a PROM blasting utility, such as PB-11, and placing the PROMS into the target configuration. This simple loading technique most closely resembles the final target configuration since actual ROM storage is used.

2. Media transfer. When this method is used, the application program is loaded, in stand-alone form, into the target from a mass storage system. The directions on creating a stand-alone bootable program are provided in the first part of this appendix. The target configuration uses LSI-11 bus RAM memory in place of the SBC-11/21 on-board ROM during initial start-up and debug. The SBC-11/21 configuration must contain the Macro-ODT ROMs described in Chapter 4. The ODT ROMs provide the means of loading the application program and are used during program debug. Media transfer does not reflect the final configuration, but execution from RAM makes debugging and testing easier. The speed of the program in this mode is approximately half that of the ROM based system.

3. Down-line loading. This method of loading allows transfer of the controller software from the development system to the target system via a serial communication link. The down-line loader must be a development system utility. The target configuration is similar to the media transfer configuration. In addition to the LSI-11 bus, RAM, and the Macro-ODT ROMs one of the serial I/O lines on the SBC-11/21 must be dedicated to the communication with the development system.

When the desired loading technique is implemented, the final phase of development is to debug and run. The loading technique employed

dictates the approach that will be taken during debug.

If the application is being loaded via the ROM transfer method, initial testing and debugging is difficult. When ROM transfer is used there must be embedded code in the application that will report the state of the control system periodically. Another way to check the system is to observe changes that occur in the external devices. If errors are found, correction requires a complete reprogramming of the PROMs. This type of testing and debugging can be quite cumbersome.

When the application is loaded via media transfer, the inital testing and debugging becomes easier than the ROM method. Once the application program is loaded into LSI-11 bus RAM or into on-board RAM, it can be run using the features of Macro-ODT. The designer can also embed reporting tasks and periodic halts in the application to examine the current state of the system. It should be noted that executing out of LSI-11 bus RAM during debug will be approximately twice as slow as running out of the SBC-11/21 on-board memory. If errors are found minor modifications can be made in the application code since testing is being done in RAM. This eliminates the loop of making new run-time memory images for every change. Once the target system is running successfully, with all of the tasks integrated, the run-time configuration can be set-up. The last step is to commit the application program to ROM and run in the SBC-11/21.

C.3 AN APPLICATION EXAMPLE

This is a sample application (Figure C-2) to show the development of a controller program using MACRO-11. The sample program will simply light the LED used by Port C of the SBC-11/21. The LED will light for ten seconds whenever an input is detected on the console port (SLU1).

The controller program for this simple system is best served by using an interrupt driven environment. An interrupt service routine is used to monitor the console port. When an input is received, a routine is entered that will set the timer for ten seconds and light the LED. A second interrupt service routine is used to count up to ten seconds and then turn off the LED. This routine is serviced by the BEVNT interrupts. In addition to the application tasks, there are tasks to initialize the input/output devices and data structures. Also there are diagnostic programs for the SBC-11/21 and programs used to handle any exceptions. The controller program is developed as individual tasks and then integrated into a complete final program.

The monitor program is described by Figure C-3 and consists of Power Up programs, Diagnostic programs, Task programs and Exception programs. The Power Up programs consist of POWRUP and RECOVR which is initiated by POWRUP. The Diagnostic programs

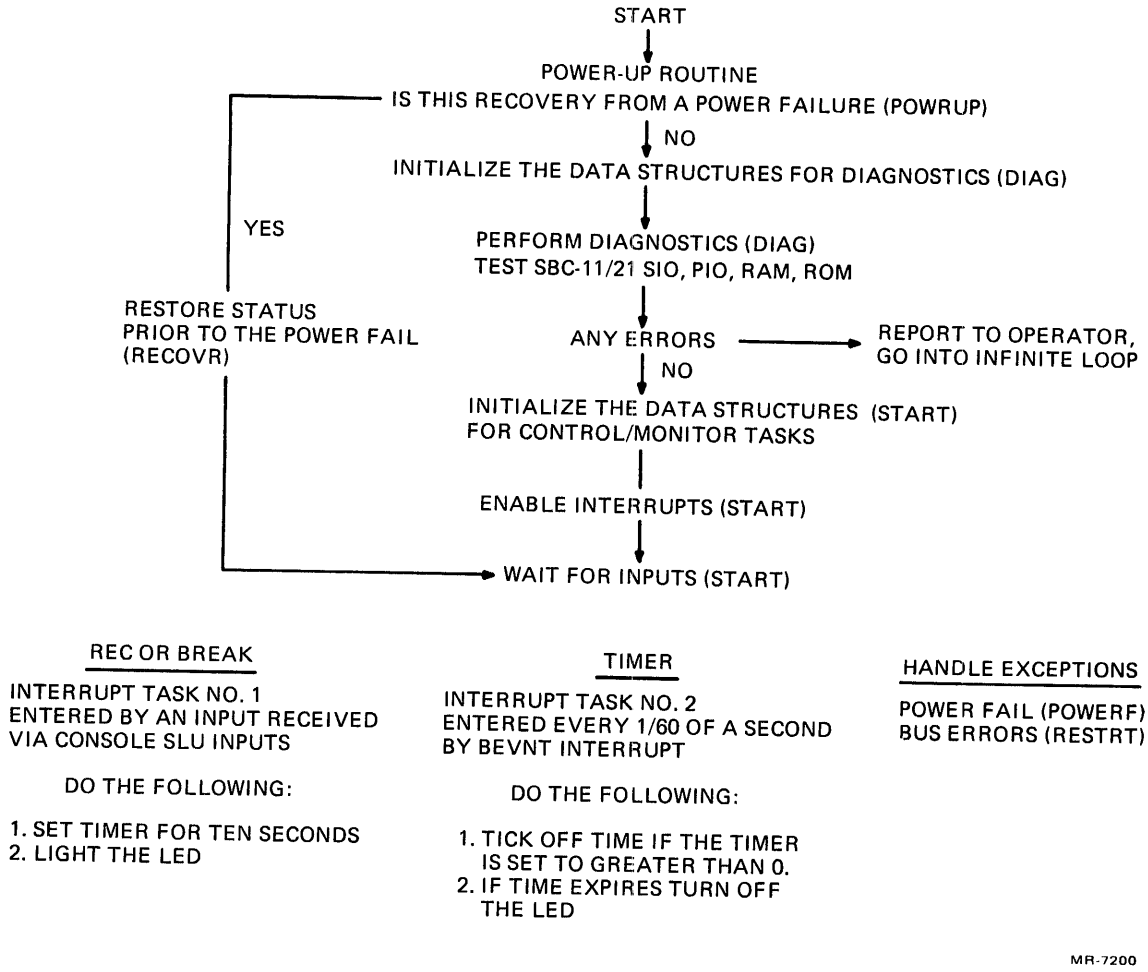


Figure C-2 Application Overview

consist of SLUTST, PIOTST, RAMTST and ROMTST. The Task programs consist of TIMER, REC and BREAK. The Exception programs consist of POWERF, RESTRT and PRINT. All these programs with the constants and instruction data are located in the ROM memory. The variable data for the application and the stack are located in the RAM memory. Memory Map 1 is assumed (Table 2-8), with the program in ROM socket set B, and data in battery-backed RAM starting at 160000. The Load Map in Figure C-4 shows actual memory locations assigned to code and data.

C.3.1 Power Up Programs

The controller program begins when the system power is applied. The microprocessor accesses location zero which is the jumper-configured start address. This location contains a jump to the power up routine POWRUP (see Figure C-5). This routine determines if this is a normal power up or a recovery from a power failure. This is determined by checking the POWER FAIL flags in the RAM memory. If the flag is set to indicate that the system is recovering from a power fail condition the program jumps to the RECOVR program (see Figure C-6).This program restores the system to the conditions that existed prior to the power fail and resumes program execution. If the flag is not set, an initial power up program is executed and then the program branches to the Diagnostic programs.

```
                                        .TITLE FALCON DEVELOPMENT EXAMPLE
                                        .ENABL  LC
                                        .GLOBL  POWERF,BREAK,REC,TIMER,RECOVR,SLUTST,PIOTST,RAMTST,ROMTST
                                        .GLOBL  POWER1,POWER2,ERROR,STACK,RESTRT

                                ;+
                                ; This is an example of a simple controller application for the KXT11-AA.
                                ;-


                                        .SBTTL  Program section definitions
                                ;+
                                ;  Define the three program sections that will be used
                                ;-
000000                                  .ASECT                          ; Assign absolute memory locations
000000                                  .PSECT  ROM                     ; For instructions and constant data
                                                                        ; that will be stored in rom memory
000000                                  .PSECT  RAM,D                   ; To define all RAM locations


                                        .SBTTL  Equates
                                ;+
                                ; Constant definitions
                                ;-
                176540                  RCSR1   == 176540               ; Auxiliary SLU addresses

                177560                  RCSRC   == 177560               ; Console  SLU addresses
                000052                  CONBR   == 52                   ; Programmable baud rate mask (9600)

                176200                  PPA     == 176200               ; Parallel port A
                176206                  PCW     == 176206               ; Parallel control word
                000261                  LEDON   == 261                  ; Parallel CSW to light the LED
                000017                  LEDOFF  == 17                   ; Parallel CSW to turn off the LED

                160010                  RAMBGN  == 160010               ; Bottom of the user RAM
                167776                  RAMTOP  == 167776               ; Top of the user RAM

                106016                  CSUM    == 106016               ; Checksum value for the system tasks


                                        .SBTTL  Macro definitions
                                ;+
                                ; Define macros that will be used by the application
                                ;-
                                        .MACRO  PUSH ARG        ; stack push operation
                                          MOV   ARG,-(SP)               ; move the argument onto the stack
                                        .ENDM

                                        .MACRO  POP ARG         ; stack pop operation
                                          MOV   (SP)+,ARG               ; move the argument from the stack
                                        .ENDM


                                        .SBTTL  Entry points
                                ;+
                                ;  Define entry point, interrupt, and trap service routine addresses
                                ;-
000000                                  .ASECT
                000000                  .=0
000000  000167  000000'                 JMP     POWRUP                  ; Jump to the power-up routine
000004  000167  000000G                 JMP     RESTRT                  ; Jump to the restart routine
                000024                  .=24
000024  000000G 000340                  .WORD   POWERF,340              ; Power fail service routine

                000060                  .=60
000060  000000G 000300                  .WORD   REC,300                 ; Console receiver service routine

                000100                  .=100
000100  000000G 000300                  .WORD   TIMER,300               ; Timer service routine

                000140                  .=140
000140  000000G 000300                  .WORD   BREAK,300               ; Console break service routine


                                        .SBTTL  Power up routine
000000                                  .PSECT  ROM

000000                                  POWRUP::
                                ;+
                                ; Come here first under all circumstances and decide if this is a normal
                                ; power up or recovery from a power fail
                                ;-
000000  026727  000000G 123456          CMP     POWER1,#123456          ; Is this recovery from  power failure
000006  001006                          BNE     DIAG                    ; or is it a normal power-up ?
000010  026727  000000G 135724          CMP     POWER2,#135724          ; Increase the chance to distinquish
000016  001002                          BNE     DIAG                    ; by checking against a 32 bit pattern

000020  000167  000000G                 JMP     RECOVR                  ; This a recovery from power fail
```

Figure C-3 Monitor Program

```
                                        .SBTTL Diagnostics
000024                          DIAG::
                                ;+
                                ; Do the system diagnostics
                                ;-
000024  012706  000000G                 MOV     #STACK,SP               ; Initialize the stack
000030  005067  000000G                 CLR     ERROR                   ; Initialize the error flag
000034  052737  000052  177564          BIS     #CONBK,@#RCSRC+4        ; Initialize the console SLU

000042  004767  000074                  CALL    PRINT                   ; Tell the operator that the power-
000046  000174'                         .WORD   DIAGM                   ; up diagnostics are running
000050  004767  000000G                 CALL    RAMTST                  ; Perform the KXT11 RAM memory test
000054  004767  000000G                 CALL    ROMTST                  ; Perform the KXT11 ROM memory test
000060  004767  000000G                 CALL    SLUTST                  ; Perform the KXT11 serial line test
000064  004767  000000G                 CALL    PIOTST                  ; Perform the KXT11 parallel I/O test

000070  005767  000000G                 TST     ERROR                   ; Is the error flag zero
000074  001404                          BEQ     1$                      ; Yes, no errors proceed to init
000076  004767  000040                  CALL    PRINT                   ; No, diagnostic failure
000102  000332'                         .WORD   EMESS
000104  000777                          BR      .                       ; Wait until there is operator action
000106  004767  000030          1$:     CALL    PRINT                   ; Indicate that things are OK
000112  000253'                         .WORD   HMESS                   ; and move on


                                        .SBTTL  Initialization completion, allow application tasks to run

000114                          START::
                                ;+
                                ; This is the start of the main body of the application
                                ;-
000114  105737  177562                  TSTB    @#RCSRC+2               ; Flush the receiver buffer
000120  052737  000100  177560          BIS     #100,@#RCSRC            ; Enable interrupt on the receiver
000126  106427  000000                  MTPS    #0                      ; Allow interrupts to happen
000132  004767  000004                  CALL    PRINT                   ; Tell the operator that the
000136  001015'                         .WORD   GO                      ; application is up and running

000140  000777                          BR      .                       ; Sit and wait for interrupts


000142                          PRINT::
                                ;+
                                ; This subroutine prints the actual messages
                                ;-
                                        .ENABL  LSB
000142  017604  000000                  MOV     @(SP),R4        ; Point to the beginning of the message
000146  005216                          INC     (SP)            ; Increment beyond message address in the
000150  005216                          INC     (SP)            ; calling routine
000152  112405                  1$:     MOVB    (R4)+,R5        ; Move the next character to be printed
000154  001406                          BEQ     3$              ; Is this the end of message marker ?
000156  105737  177564          2$:     TSTB    @#RCSRC+4       ; No, output another character
000162  100375                          BPL     2$              ; Transmitter ready
000164  110537  177566                  MOVB    R5,@#RCSRC+6    ; Output the character
000170  000770                          BR      1$              ; Get another character
000172  000207                  3$:     RETURN                  ; Go back
                                        .DSABL  LSB


                                        .SBTTL  Messages sent to the operator
                                        .NLIST  BEX

000174      015     012     040 DIAGM:: .ASCIZ  <15><12>/ The power-up diagnostics are running ... /<15><12>
000253      015     012     040 HMESS:: .ASCIZ  <15><12>/ System checked out, there were no faults /<15><12>
000332      015     012     040 EMESS:: .ASCIZ  <15><12>/ System did not pass initial power up test /<15><12>
000412      015     012     007 FMESS:: .ASCIZ  <15><12><7><7><7><7>/ RUN-TIME FAILURE /<15><12>
000445      015     012     040 SLUE::  .ASCIZ  <15><12>/ Serial line unit diagnostic failure /<15><12>
000517      015     012     040 SLGOOD::.ASCIZ  <15><12>/ Serial line unit passed diagnostics /<15><12>
000571      015     012     040 MESRA1::.ASCIZ  <15><12>/ RAM failure /<15><12>
000613      015     012     040 RAGOOD::.ASCIZ  <15><12>/ RAM passed diagnostics /<15><12>
000650      015     012     040 MESRO1::.ASCIZ  <15><12>/ ROM checksum error /<15><12>
000701      015     012     040 ROGOOD::.ASCIZ  <15><12>/ ROM passed diagnostics /<15><12>
000736      015     012     040 PGOOD:: .ASCIZ  <15><12>? Parallel input/output passed diagnostics ?<15><12>
001015      015     012     040 GO::    .ASCII  <15><12>/ The application is running ... / <15><12>
001061      040     040     040         .ASCIZ  /       Type any key to light the KXT11-AA LED for 10 secs./
                                        .EVEN


        000001                          .END
```

Figure C-3 Monitor Program (Cont)

```
RT-11 LINK  V06.01C     Load Map        Tue 06-Oct-81 09:02:01
C     .SAV      Title:  FALCON  Ident:          /B:000400

Section  Addr   Size    Global  Value   Global  Value   Global  Value

 . ABS.  000000 000400   (RW,I,GBL,ABS,OVR)
                         LEDOFF  000017  CONBR   000052  LEDON   000261
                         CSUM    106016  RAMBGN  160010  RAMTOP  167776
                         PPA     176200  PCW     176206  RCSR1   176540
                         RCSRC   177560
 ROM     000400 157400   (RW,I,LCL,REL,CON)
                         POWRUP  000400  DIAG    000424  START   000514
                         PRINT   000542  DIAGM   000574  HMESS   000653
                         EMESS   000732  FMESS   001012  SLUE    001045
                         SLGOOD  001117  MESRA1  001171  RAGOOD  001213
                         MESRO1  001250  ROGOOD  001301  PGOOD   001336
                         GO      001415  RECOVR  001556  REC     001634
                         TIMER   001656  BREAK   001702  LAST    001716
                         POWERF  001720  RESTRT  001764  SLUTST  001774
                         RAMTST  002132  ROMTST  002212  PIUTS1  002262
 RAM     160000 000332   (RW,D,LCL,REL,CON)
                         POWER1  160010  POWER2  160012  SAVER6  160014
                         ERROR   160016  TIME    160020  STACK   160332

Transfer address = 000001, High limit = 160330 = 28780. words
```

Figure C-4 Load Map

```
 1                                      .ENABL  LC
 2 000000                               .PSECT  RAM,D
 3                                      ;
 4                                      ; The variable data is assigned to the user RAM space on the KXT11-AA
 5                                      ;
 6
 7 000000                               .BLKW   4               ; Non existant KXT11-AA memory
 8 000010  000000                       POWER1::.WORD   0       ; Power failure 32-bit comparision
 9 000012  000000                       POWER2::.WORD   0       ; flag
10 000014  000000                       SAVER6::.WORD   0       ; Stack pointer area for power failure
11 000016  000000                       ERROR:: .WORD   0       ; Diagnostic error flag
12
13 000020  000000                       TIME::  .WORD   0       ; Time flag
14 000022                               .BLKW   100.            ; This is the stack
15 000332                               STACK::
16
17         000001                       .END
```

Figure C-5 Power Up Task

```
 1                                      .ENABL  LC
 2                                      .GLOBL  SAVER6,TIME,RCSRC,CONBR,LEDON,PCW
 3                                      .MCALL  POP
 4 000000                               .PSECT  ROM
 5
 6 000000                               RECOVR::
 7                                      ;+
 8                                      ;This routine is entered if a recovery from a power failure is taking place
 9                                      ;- Entered from POWRUP
10 000000  016706  000000G              MOV     SAVER6,SP       ; Restore the stack pointer
11 000004                               POP     TIME            ; Restore     any variable information
12 000010                               POP     R5              ; Restore the general purpose
13 000012                               POP     R4              ; registers
14 000014                               POP     R3
15 000016                               POP     R2
16 000020                               POP     R1
17 000022                               POP     R0
18 000024  052737  000100  000000G      BIS     #100,@#RCSRC    ; Re-initialize console SLU, enable
19 000032  052737  000000G 000004G      BIS     #CONBR,@#RCSRC+4; interrupts and set-up baud rate
20 000040  005767  000000G              TST     TIME            ; Is the LED timer set
21 000044  001403                       BEQ     3$              ; No, continue
22 000046  012737  000000G 000000G      MOV     #LEDON,@#PCW    ; Yes turn the LED on for the rest
23                                                              ; of the time prior to power-fail
24 000054  000002                       3$:     RTI             ; Return from point of power-fail
25                                                              ; interrupt
26
27         000001                       .END
```

Figure C-6 Power Fail Recovery

C.3.2 Diagnostic Programs

The diagnostic programs are entered via a diagnostic initialization routine. The SLUTST (see Figure C-7) program is the first diagnostic and it tests the auxiliary Serial Line Unit on the SBC-11/21. The diagnostic enables the SLU maintenance mode and transmits various test patterns. After a certain amount of time the program checks to see that the test patterns were correctly received. It should be noted that the SLU maintenance mode allows data to be transmitted to the EIA port as well as through the internal loopback. Therefore if a device is connected to the port it will respond to this data.

The second diagnostic is RAMTST (see Figure C-8) which tests the RAM memory. This test is performed by writing known data into a RAM location and then checking that the correct information is in that location.

```
 1                                          .ENABL  LC,LSB
 2                                          .GLOBL  RCSR1,ERROR,PRINT,SLUE,SLGOOD
 3 000000                                   .PSECT  ROM
 4
 5 000000                           SLUTST::
 6                                  ;+
 7                                  ;  This routine checks the auxiliary SLU port on the KXT11-AA
 8                                  ;-
 9
10 000000  012701  000000G                 MOV     #RCSR1,R1               ; Point to the address
11 000004  105761  000002                  TSTB.   2(R1)                   ; Flush the contents of RBUF
12 000010  012761  000006  000004          MOV     #6,4(R1)                ; Set the SLU for maintenance and
13                                                                         ; programmable baud rates
14 000016  012702  000010                  MOV     #8.,R2                  ; Initialize the baud rate counter
15 000022  012703  000132'         1$:     MOV     #PATERN,R3              ; Point to the test patterns
16 000026  005005                  2$:     CLR     R5                      ; Initialize time out counter
17 000030  105761  000004          3$:     TSTB    4(R1)                   ; Loop the pattern around
18 000034  100402                          BMI     4$                      ; Branch if ready to send
19 000036  077504                          SOB     R5,3$                   ; If not ready, bump time out counter
20 000040  000422                          BR      100$                    ; IF timed out then - ERROR -
21 000042  111361  000006          4$:     MOVB    (R3),6(R1)              ; Send the information out
22 000046  005005                          CLR     R5                      ; Initialize the time out counter
23 000050  105711                  5$:     TSTB    (R1)                    ; Is the receiver ready ?
24 000052  100402                          BMI     6$                      ; Yes it is and branch
25 000054  077503                          SOB     R5,5$                   ; If not ready, bump time out counter
26 000056  000413                          BR      100$                    ; If timed out then -ERROR-
27 000060  126113  000002          6$:     CMPB    2(R1),(R3)              ; Was the information sent OK ?
28 000064  001010                          BNE     100$                    ; No it was not -ERROR-
29 000066  105723                          TSTB    (R3)+                   ; All of the test paterns done ?
30 000070  001356                          BNE     2$                      ; No, go do another pattern
31 000072  005302                          DEC     R2                      ; All of the baud rates tested ?
32 000074  001412                          BEQ     200$                    ; Yes, get out of this routine
33 000076  062761  000010  000004          ADD     #10,4(R1)               ; No, set-up the next baud rate
34 000104  000746                          BR      1$                      ; Do another loop, reinit patterns
35
36 000106  005267  000000G         100$:   INC     ERROR                   ; Bump the error counter
37 000112  004767  000000G                 CALL    PRINT                   ; Print the error message
38 000116  000000G                         .WORD   SLUE
39 000120  000403                          BR      150$                    ; Go back
40 000122  004767  000000G         200$:   CALL    PRINT                   ; The test was successful
41 000126  000000G                         .WORD   SLGOOD
42 000130  000207                  150$:   RETURN                          ; Bye
43
44 000132     177      040      000  PATERN: .BYTE   177,40,0                ; Test patterns for SLU
45                                         .EVEN
46
47                                         .DSABL  LSB
48         000001                          .END
```

Figure C-7 SLU Diagnostic Task

```
 1                                              .ENABL  LC,LSB
 2                                              .GLOBL  RAMBGN,PRINT,RAMTOP,MESRA1,RAGOOD,ERROR
 3 000000                                       .PSECT  ROM
 4
 5 000000                               RAMTST::
 6                                      ;+
 7                                      ;  This routine checks the user RAM on the KXT11-AA
 8                                      ;-
 9
10 000000  011602                              MOV     (SP),R2                 ; Save the return address
11 000002  016703  000000G                     MOV     ERROR,R3                ; Save the contents of the ERROR flag
12 000006  012700  000000G                     MOV     #RAMBGN,R0              ; Point to the start of the user RAM
13
14 000012  010010                      1$:     MOV     R0,(R0)                 ; Write the address
15 000014  020010                              CMP     R0,(R0)                 ; Read it back
16 000016  001405                              BEQ     2$                      ; Was the value read correctly
17 000020  004767  000000G                     CALL    PRINT                   ; No, report the failure,
18 000024  000000G                             .WORD   MESRA1                  ;
19 000026  005203                              INC     R3                      ; set the error flag,
20 000030  000407                              BR      3$                      ; and go back
21 000032  005720                      2$:     TST     (R0)+                   ; Go onto the next location
22 000034  020027  000002G                     CMP     R0,#RAMTOP+2            ; Until there is no more to test
23 000040  103764                              BLO     1$
24 000042  004767  000000G                     CALL    PRINT                   ; Indicate RAM test success
25 000046  000000G                             .WORD   RAGOOD
26
27 000050  010216                      3$:     MOV     R2,(SP)                 ; Restore the return address
28 000052  010367  000000G                     MOV     R3,ERROR                ; Restore the ERROR flag
29 000056  000207                              RETURN                          ; Test completed
30
31                                             .DSABL  LSB
32         000001                              .END
```

Figure C-8 RAM Diagnostic Task

The third diagnostic is ROMTST (see Figure C-9) which checks the ROM memory. This test calculates a checksum on the actual control and monitoring tasks. If there is a checksum mismatch then there is a potential failure at some ROM location.

THe last diagnostic is PIOTST (see Figure C-10) which checks the Parallel I/O port on the SBC-11/21. This test verifies that the Parallel I/O registers can be addressed. The send/receive capability cannot be checked unless there is a loopback connector installed on the J3 connector. When data are written into these registers and a device is connected to the port, the device can respond to the data.

```
 1                                              .ENABL  LC,LSB
 2                                              .GLOBL  REC,LAST,CSUM,PRINT,MESRO1,ROGOOD,ERROR
 3 000000                                       .PSECT ROM
 4
 5 000000                               ROMTST::
 6                                      ;+
 7                                      ;  This routine will check the ROM on the KXT11-AA, this test checks
 8                                      ;  the portion of the ROM that contains the actual control/monitor tasks
 9                                      ;-
10
11 000000  012700  000000G                     MOV     #REC,R0                 ; Point to the control task address
12 000004  005001                              CLR     R1                      ; Initialize checksum value
13 000006  062001                      1$:     ADD     (R0)+,R1                ; Update value
14 000010  022700  000002G                     CMP     #LAST+2,R0              ; Until there are no values to sum
15 000014  001374                              BNE     1$                      ; If there are still some go get them
16 000016  022701  000000G                     CMP     #CSUM,R1                ; Are the checksums equal ?
17 000022  001406                              BEQ     2$                      ; Yes, leave the test
18 000024  004767  000000G                     CALL    PRINT                   ; No, report the
19 000030  000000G                             .WORD   MESRO1                  ; failure
20 000032  005267  000000G                     INC     ERROR                   ; Set the error flag
21 000036  000403                              BR      3$                      ; Leave the test
22 000040  004767  000000G             2$:     CALL    PRINT                   ; Report the test passed
23 000044  000000G                             .WORD   ROGOOD
24 000046  000207                      3$:     RETURN
25
26                                             .DSABL  LSB
27         000001                              .END
```

Figure C-9 ROM Diagnostic Task

```
 1                                      .ENABL  LC,LSB
 2                                      .GLOBL  PPA,PRINT,PGOOD
 3 000000                               .PSECT  ROM
 4
 5 000000                       PIOTST::
 6                                      ;+
 7                                      ;  This routine checks the parallel ports on the KXT11-AA this only
 8                                      ;  tests the ability to address the port
 9                                      ;-
10
11 000000  012701  000003              MOV     #3,R1           ; Initialize loop counter
12 000004  005000                      CLR     R0              ; Initialize counting index
13 000006  005760  000000G     1$:     TST     PPA(R0)         ; Attempt to address PIO port if the
14                                                             ; attempt fails a trap through the
15                                                             ; restart will occur and report a run
16                                                             ; time error
17 000012  005720                      TST     (R0)+           ; Increment the index, this will not
18                                                             ; time out since there is memory at
19                                                             ; locations 2-4
20 000014  077104                      SOB     R1,1$           ; Do the port
21 000016  004767  000000G             CALL    PRINT           ; Indicate success
22 000022  000000G                     .WORD   PGOOD
23
24 000024  000207                      RETURN
25
26                                      .DSABL  LSB
27         000001                      .END
```

Figure C-10 Parallel I/O Diagnostic Task

When any of the above diagnostics detect a failure, the program will set an Error Flag. The diagnostic program will check the status of all Error Flags before it enters the task programs. If an error is found, the operator is informed that a diagnostic test failed and the program enters a loop to wait for the operator to intervene. Each diagnostic will print a message to the operator indicating success or failure. If there are no failures, then a success message is printed and the program enters the task programs.

C.3.3 Control Task Programs

The Control Tasks programs (see Figure C-11) complete the initialization of the system by clearing the receive buffer, enabling the interrupts, and lowering the microprocessor priority to accept interrupts. The operator is then informed that the system is running and waiting for interrupts. The TIMER receives a BEVNT input sixty times per second. The REC program is entered when an interrupt is received from the console. The program will then turn on the LED and load the ten second counter. The BREAK program is entered whenever a break is detected and performs the same task as REC. A TIMER program will decrement the ten second counter, if it is enabled, every time BEVNT is received. When the ten second counter is decremented to zero, the program will turn off the LED. If the LED is turned on and another break or interrupt occurs, the ten second counter is reset for ten seconds. The program also anticipates any exception conditions.

C.3.4 Exception Programs

The system is now running and the exception programs are entered only when a power fail occurs or a bus timeout occurs. The PRINT program is entered only to establish communication with the operator.

```
                                        .SBTTL  CONTROL AND MONITORING TASKS
                                        .ENABL  LC
                                        .GLOBL  TIME,LEDON,PCW,RCSRC,LEDOFF
000000                                  .PSECT  ROM

000000                          REC::
                                ;+
                                ; This interrupt routine accepts an input from the console. When the input is
                                ; received a ten second counter is initialized and the LED is turned on.
                                ;-

000000  012767  001130  000000G         MOV     #<10.* 60.>,TIME        ; Set timer for ten seconds
000006  012737  000000G 000000G         MOV     #LEDON,@#PCW            ; Turn the LED on
000014  105737  000002G                 TSTB    @#RCSRC+2               ; Flush the receive buffer
000020  000002                          RTI                             ; Go back

000022                          TIMER::
                                ;+
                                ; This interrupt routine when entered every clock tick will decrement the ten
                                ; second counter and turn off the LED if the time is expired, otherwise it
                                ; returns immediately.
                                ;-

000022  005767  000000G                 TST     TIME                    ; If the time is set update the
000026  001406                          BEQ     GOBACK                  ; counter otherwise go back
000030  005367  000000G                 DEC     TIME                    ; Yes, bump the counter and if it is
000034  001003                          BNE     GOBACK                  ; The last tick then shut the LED off
000036  012737  000000G 000000G         MOV     #LEDOFF,@#PCW           ; Otherwise go back
000044  000002                  GOBACK: RTI

000046                          BREAK::
                                ;+
                                ; This interrupt service routine will be entered if a break detected , this
                                ; is treated as a regular input on the KXT11-AA console port.
                                ;-

000046  012767  001130  000000G         MOV     #<10. * 60.>,TIME       ; Set the timer for ten seconds
000054  012737  000000G 000000G         MOV     #LEDON,@#PCW            ; Turn the LED on
000062  000002                  LAST::  RTI                             ; Go back

        000001                          .END
```

Figure C-11 Control Task

A timeout will occur when an address does not respond or if a device does not reply to an interrupt acknowledge. When a timeout occurs the SBC-11/21 will trap to location 4 which is the restart address. The start address is defined as location 0 and restart address is defined as location 4 by the factory configuration. The RSTRT program is entered via location 4 and informs the operator that a run-time error has occurred and waits for the operator to intervene.

An imminent power failure is detected when the system power is going down. This enables the powerfail interrupt and causes a trap to location 24. The POWERF program (see Figure C-12) is entered via location 24 and the POWER FAIL flags are set in the RAM memory. The RAM memory incorporates the Battery Backup feature of the SBC-11/21 module. Program information contained in the general purpose registers, the stack pointer and other pertinent data are stored in the non-volatile RAM memory. The program then puts the bus into a known state with RESET instruction and waits for the power loss to occur. When power is eventually restored, POWRUP routine is executed and data are recovered as the system restarts.

```
                                        .ENABL  LC
                                        .GLOBL  POWER1,POWER2,TIME,SAVER6,PRINT,FMESS
                                        .MCALL  PUSH
000000                                  .PSECT  ROM

000000                          POWERF::
                                ;+
                                ; This routine is entered when a power fail is detected and saves the
                                ; pertinent information in non-volatile RAM
                                ;-
000000  012767  123456  000000G         MOV     #123456,POWER1          ; Initialize the 32-bit power recovery
000006  012767  135724  000000G         MOV     #135724,POWER2          ; test pattern in first two words of RAM
000014                                  PUSH    R0                      ; Save the general purpose registers
000016                                  PUSH    R1                      ; and any pertinent data in a non-
000020                                  PUSH    R2                      ; volatile RAM area
000022                                  PUSH    R3
000024                                  PUSH    R4
000026                                  PUSH    R5
000030                                  PUSH    TIME
000034  010667  000000G                 MOV     SP,SAVER6               ; Save the stack pointer in the non-
                                                                        ; volatile ram area
000040  000005                          RESET                           ; Put the bus in a known state
000042  000777                          BR      .                       ; and wait for loss of power

000044                          RESTRT::
                                ;+
                                ; When a bus error occurs such as an interrupt time-out or bus time-out
                                ; a trap thru the restart takes place and comes here
                                ;-
000044  004767  000000G                 CALL    PRINT                   ; Indicate that a run-time error has
000050  000000G                         .WORD   FMESS                   ; occurred and wait for operator
000052  000777                          BR      .                       ; intervention


        000001                          .END
```

Figure C-12 Power Fail Task

APPENDIX D
MACRO-ODT ROM

This appendix provides the user with the program listing of the Macro-ODT ROM firmware.

TABLE OF CONTENTS

3- 1 COPYRIGHT NOTICE
4- 1 KXT11-A2 EDIT HISTORY
5- 1 Equates
6- 1 General DLART Equates
8- 1 General PPI Equates
9- 1 Program-specific Equates
11- 1 MACRO DEFINITIONS
13- 1 RAM Definition
14- 3 TRAPS-Trap-handling routines
14- 4 TRAPS-LTC Trap-killer
14- 5 TRAPS-BREAK handler
15- 1 RESTART-Introduction
19- 1 RESTART-Entry point
20- 1 RESTART-See if stack exists
20- 19 RESTART-Exit if in IN-ROM state
21- 1 RESTART-Cause determination
22- 1 RESTART-Exits
23- 1 POWERUP-Introduction
24- 1 POWERUP-Turn on LED
24- 22 POWERUP-Test console DLART
25- 1 POWERUP-Test and set up I/O-page RAM
26- 1 POWERUP-Turn off LED
26- 29 POWERUP-Test for "low core"
27- 1 POWERUP-Exit
27- 20 POWERUP-Subroutine to initialize vectors
28- 1 AUTOBAUD-Synchronize with Console
30- 1 macroODT-Introduction
32- 1 macroODT-Save status and print prompt
33- 1 macroODT-Get ODT command
35- 1 macroODT- Go and Proceed
36- 1 macroODT-Register and PS command
37- 1 macroODT-Examine and Deposit
39- 1 macroODT-Get and echo character
40- 1 macroODT-Type ASCII string
41- 1 macroODT-Get octal digits
42- 1 macroODT-OCTSTR--type binary in R0 as ASCII
43- 1 macroODT-Output messages
44- 1 DIAGNOSTICS-for SLU2 and PPI
45- 1 HARDWARE ENTRY POINT
46- 1 DIAGNOSTICS-Continued
47- 1 BOOTS-Description
48- 9 BOOTS-RX Controller Definitions
48- 56 BOOTS-TU58 Definitions and Protocol Equates
48- 114 BOOTS-RX11 Definitions and Equates
49- 1 BOOTS-Program entry point
49- 42 -----> HALT AT PC=172234 INDICATES "Illegal device name"
49- 51 -----> HALT AT PC=172264 INDICATES "Illegal unit number"
49- 58 -----> HALT AT PC=172304 INDICATES "No low memory, can't boot"
49- 92 -----> HALT AT PC=172376 INDICATES "Unexpected timeout during boot"
50- 1 BOOTS-RX01/RX02 Bootstrap
51- 1 BOOTS-Distinguishing type of boot block
51- 23 -----> HALT AT PC=172454 INDICATES "No boot block on volume"
52- 1 BOOTS-TU58 Bootstrap
52- 29 -----> HALT AT PC=172542 INDICATES "TU58 initialization error"
52- 37 -----> HALT AT PC=172562 INDICATES "TU58 block 0 read error"
53- 1 BOOTS-Stand-alone volume bootstrap
53- 24 -----> HALT AT PC=172614 INDICATES "Directory read error"

TABLE OF CONTENTS

53- 36 -----> HALT AT PC=172652 INDICATES "File not found"
54-   1 BOOTS-Load Stand-Alone Program File
54-   8 -----> HALT AT PC=172732 INDICATES "Stand-alone file read error"
54- 12 -----> HALT AT PC=172750 INDICATES "Illegal transfer address"
55-   1 173000G ENTRY POINT
56-   1 BOOTS-Continued
57-   1 BOOTS-RX01/RX02 Read routines
57- 36 -----> HALT AT PC=173070 INDICATES "Floppy drive not ready"
57- 114 -----> HALT AT PC=173262 INDICATES "Floppy read error"
60-   1 BOOTS-TU58 Read routines
61- 27 -----> HALT AT PC=173556 INDICATES "TU58 END packet missing"
61- 37 -----> HALT AT PC=173610 INDICATES "TU58 checksum error"
63-   1 END STATEMENT

```
                                        .TITLE KXT11-A2 1K FIRMWARE
                                        .IDENT  /V1.00/
                                        .ENABL LC

                                    ; Place identification number in last ROM location:
000000                              .ASECT
                 173776             .=173776
173776    000                       .BYTE   0
173777    001                       .BYTE   1.
```

```
                        .SBTTL COPYRIGHT NOTICE
                ;
                ; COPYRIGHT (C) 1980, 1981 BY
                ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                ;
                ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
                ; ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
                ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER
                ; COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
                ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
                ; TRANSFERRED.
                ;
                ; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE
                ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
                ; CORPORATION.
                ;
                ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
                ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                ;
                ; VERSION V1.00
                ;
                ; EDWARD P. LUWISH 29-SEP-81
```

```
                                                        .SBTTL  KXT11-A2 EDIT HISTORY

                                ;EDIT HISTORY:
                                ;
```

```
                                                        .SBTTL Equates
                                ; BIT EQUATES
000001                          BIT0    =       1
000002                          BIT1    =       2
000004                          BIT2    =       4
000010                          BIT3    =       10
000020                          BIT4    =       20
000040                          BIT5    =       40
000100                          BIT6    =       100
000200                          BIT7    =       200
000400                          BIT8    =       400
001000                          BIT9    =       1000
002000                          BIT10   =       2000
004000                          BIT11   =       4000
010000                          BIT12   =       10000
020000                          BIT13   =       20000
040000                          BIT14   =       40000
100000                          BIT15   =       100000

                                ; ASCII CHARACTER EQUATES

000012                          LF      =       12              ;Line feed
000015                          CR      =       15              ;Carriage return
000040                          SPACE   =       40              ;Space
```

```
                                        .SBTTL General DLART Equates

                        ; DLART EQUATES

177560                  RCSR$1  =       177560          ;SLU1 Receive CSR
177562                  RBUF$1  =       177562          ;SLU1 Receive buffer
177564                  XCSR$1  =       177564          ;SLU1 Xmit CSR
177566                  XBUF$1  =       177566          ;SLU1 Xmit buffer
176540                  RCSR$2  =       176540          ;SLU2 Receive CSR
176542                  RBUF$2  =       176542          ;SLU2 Receive buffer
176544                  XCSR$2  =       176544          ;SLU2 Xmit CSR
176546                  XBUF$2  =       176546          ;SLU2 Xmit buffer

                        ; DLART RECEIVE CSR BITS

004000                  RC.ACT  =       BIT11           ;Receiver active (R/O).  Set
                                                        ; while character is being
                                                        ;  received.
000200                  RC.DUN  =       BIT7            ;Receiver done (R/O).  A
                                                        ; character has been completely
                                                        ;  received and now resides
                                                        ;   in RBUF.
000100                  RC.IEN  =       BIT6            ;Receiver int. enable (R/W).
                                                        ; when set, enables "keyboard"
                                                        ;  interrupts, using vector
                                                        ;   at 60.

                        ; DLART RECEIVE BUFFER BITS (R/O)

100000                  RB.ERR  =       BIT15           ;Error.  Framing error or
                                                        ; overrun has occurred.
040000                  RB.OVR  =       BIT14           ;Overrun error.  Character was
                                                        ; received before previous one
                                                        ;  was read.
020000                  RB.FRM  =       BIT13           ;Framing error.  No valid stop
                                                        ; bit was detected.
004000                  RB.BRK  =       BIT11           ;Break detect.  Set when break
                                                        ; is detected, reset when next
                                                        ;  start bit arrives.
```

```
                        ; DLART TRANSMIT CSR BITS
                        
000200                  XC.RDY  =       BIT7            ;Transmitter ready (R/O).
                                                        ; When set, indicates that the
                                                        ;  last character was completely
                                                        ;   sent and XBUF is ready for
                                                        ;    a new one.
000100                  XC.IEN  =       BIT6            ;Transmit int. enable (R/W).
                                                        ;when set, enables "console
                                                        ; printer" interrupts, using
                                                        ;  vector at 64.
                        
                        ; Programmable baud rate bits
                        
000010                  PBR0    =       BIT3
000020                  PBR1    =       BIT4
000040                  PBR2    =       BIT5
                        
                        ; PBR0-2 set baud rates as follows:
                        
000000                  BD.003  =       0               ;Baud rate =   300
000010                  BD.006  =       PBR0            ;Baud rate =   600
000020                  BD.012  =       PBR1            ;Baud rate =  1200
000030                  BD.024  =       PBR1!PBR0       ;Baud rate =  2400
000040                  BD.048  =       PBR2            ;Baud rate =  4800
000050                  BD.096  =       PBR2!PBR0       ;Baud rate =  9600
000060                  BD.192  =       PBR2!PBR1       ;Baud rate = 19200
000070                  BD.384  =       PBR2!PBR1!PBR0  ;Baud rate = 38400
                        
000004                  XC.MNT  =       BIT2            ;Maintenance (R/W).  When set,
                                                        ; creates an internal "loop-
                                                        ;  back" between the transmitter
                                                        ;   and receiver.  Also dis-
                                                        ;    connects the external
                                                        ;     serial input.
000002                  XC.PBE  =       BIT1            ;Prog. baud rate enable.  When
                                                        ; set, the baud rate is deter-
                                                        ;  mined by bits 3-5 as
                                                        ;   tabulated above.  WHEN
                                                        ;    CLEAR, BAUD RATE IS DETER-
                                                        ;     MINED BY VOLTAGES APPLIED
                                                        ;      TO DLART IC PINS.
000001                  XC.BRK  =       BIT0            ;Transmit break (R/W).  When
                                                        ; set, serial output is a
                                                        ;  continuous BREAK.
```

```
                                        .SBTTL  General PPI Equates

                        ; PROGRAMMABLE PERIPHERAL INTERFACE (PPI) EQUATES

176206                  PP.CWR  =       176206          ;PPI Control Word Register
176200                  PP.A    =       176200          ;PPI Port A Register
176202                  PP.B    =       176202          ;PPI Port B Register
176204                  PP.C    =       176204          ;PPI Port C Register

                        ; PPI MODE-SETTING BITS

                        ; KXT11-AA board configuration does not permit all combinations of
                        ; the mode bits.  Consult the manual before using the PPI.

000200                  PP.MOD  =       BIT7            ;This MUST be or'd with other
                                                        ; bits to set mode.
000100                  PP.MD2  =       BIT6            ;Sets mode 2
000040                  PP.MDA  =       BIT5            ;If bit 6 is low, determines
                                                        ; mode of port A
                                                        ;  (hi=mode 1, lo=mode 0)
000020                  PP.DRA  =       BIT4            ;Direction of port A.
                                                        ; Hi=IN, lo=OUT.
000010                  PP.CHI  =       BIT3            ;Direction of port C upper half
                                                        ; Hi=IN, lo=OUT.
000004                  PP.MDB  =       BIT2            ;Mode of port B.
                                                        ; Hi=mode 1, lo=mode 0.
000002                  PP.DRB  =       BIT1            ;Direction of port B.
                                                        ; Hi=IN, lo=OUT.
000001                  PP.CLO  =       BIT0            ;Direction of port C lower half
                                                        ; Hi=IN, lo=OUT.

                        ; PPI BIT SET/RESET CONTROL BITS

                        ;       When bit 7 is low, writing to the PPI CSR will set or reset
                        ;       individual bits in Port C, depending on the mode and direction
                        ;       of the port's bits, and on the combination of bits you write.

000016                  PP.BI7  =       BIT3!BIT2!BIT1  ;Use ONE
000014                  PP.BI6  =       BIT3!BIT2       ;of these
000012                  PP.BI5  =       BIT3!BIT1       ;to select
000010                  PP.BI4  =       BIT3            ;which bit
000006                  PP.BI3  =       BIT2!BIT1       ;is desired
000004                  PP.BI2  =       BIT2            ;to be
000002                  PP.BI1  =       BIT1            ;SET or
000000                  PP.BI0  =       0               ;CLEARed

000001                  PP.BIS  =       BIT0            ;SET specified bit.
000000                  PP.BIC  =       0               ;CLEAR specified bit.
```

```
                                        .SBTTL  Program-specific equates

                        ; EQUATES USED TO TURN LED ON AND OFF

000221          MODE    =       PP.MOD!PP.DRA!PP.CLO    ;Port A = Mode 0 IN
                                                        ;Port B = Mode 0 OUT
                                                        ;Port C upper nibble = OUT
                                                        ;Port C lower nibble = IN

000017          LEDOFF  =       PP.BIS!PP.BIT           ;Set PC7

                        ; EQUATES USED TO SET UP DLARTS

000032          BAUDRS  =       BD.024!XC.PBE           ;Initial console baud rate to
                                                        ; be 2400, with prog. baud
                                                        ;  rates enabled.

000072          TUBAUD  =       BD.384!XC.PBE           ;TU58 Baud rate = 38,400

                        ; MEMORY CONFIGURATION EQUATES

160010          RAMBOT  =       160010                  ;Bottom address of RAM
167776          RAMTOP  =       167776                  ;Top address of RAM

                        ; SOFTWARE FLAGS AND MASKS

000300          PR16    =       300                     ;PS for priority of 6
000340          PR17    =       340                     ;PS for priority of 7

                        ; USED BY ODT MODULE

000200          RFLAG   =       BIT7                    ;Register flag bit- indicates
                                                        ; register is being examined
000020          T.BIT   =       BIT4                    ;Trace bit in PSW
```

PROGRAM-SPECIFIC EQUATES

```
                                ; RESTART TYPE WORD BITS
                        
000000                  R.HALT  =       BIT15           ;HALT or BREAK occurred
000200                  R.NXM   =       BIT7            ;Accessed non-existent memory
000001                  R.STAK  =       BIT0            ;Double-bus error
                        
                                ; BOOT CONTROL WORD BITS
                        
000000                  NO.LOW  =       BIT15           ;No memory found at 000000-
                                                        ; do not boot
000200                  DEVBIT  =       BIT7            ;1 = RX01/02 floppy
                                                        ;0 = TU58 cassette
000001                  DEVNUM  =       BIT0            ;Unit no. (0 or 1)
                        
                                ; DIAGNOSTIC MESSAGES
                        
000100                  E.EXT   =       100             ;SLU2 loopback test failed
000010                  E.INT   =       10              ;SLU2 internal loopback failed
000001                  E.PAR   =       1               ;Parallel port loopback failed
```

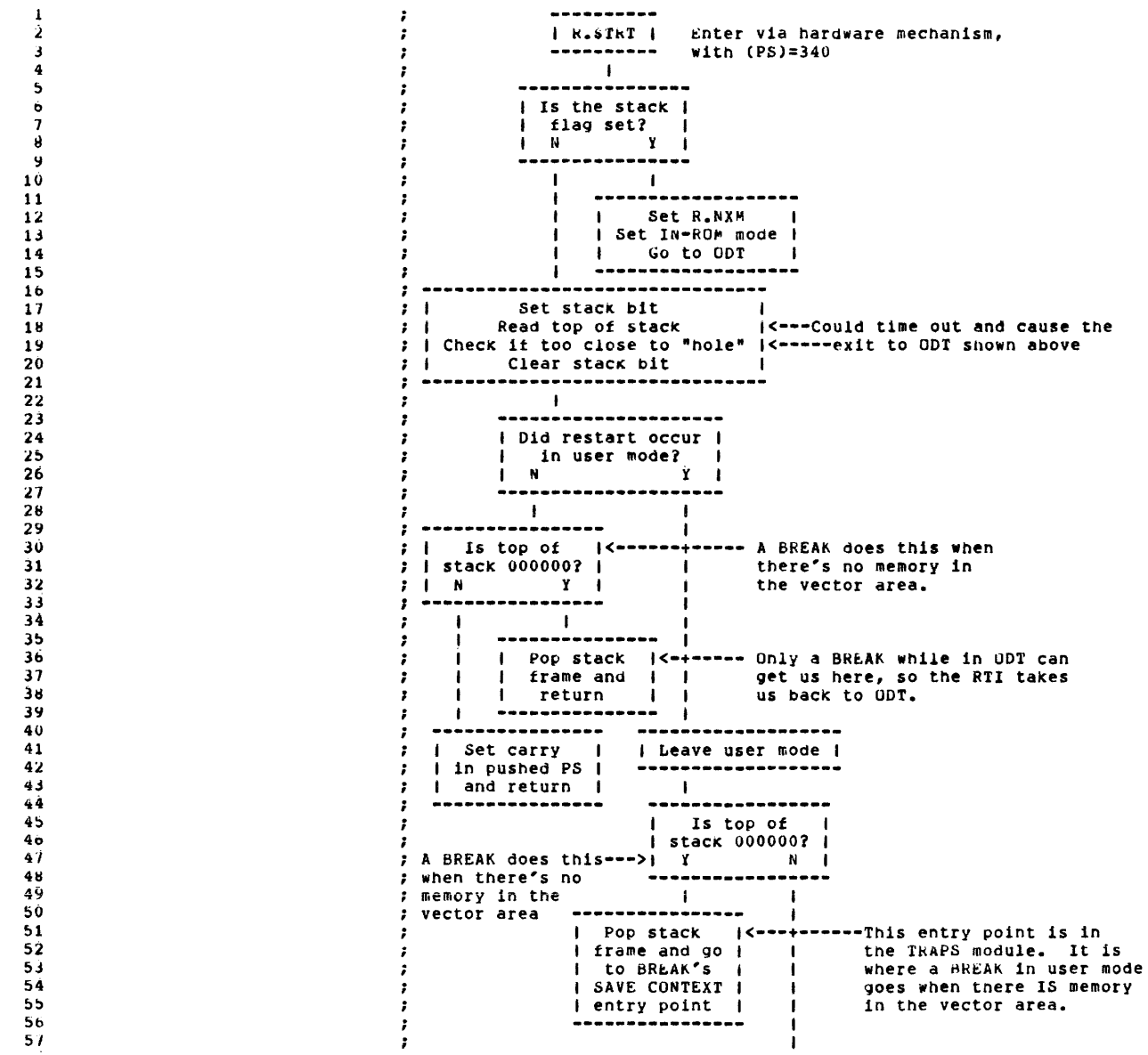
```
                                                .SBTTL MACRO DEFINITIONS

                        ;
                        ;       MACRO DEFINITIONS
                        ;

                        ;+
                        ;
                        ; This macro will insert ABORTS into the code which will halt the
                        ; program, exit to ODT with the PC printed on the console, and generate
                        ; an entry in the table of contents which describes the error condition.
                        ;
                        ;-

                        .MACRO  ABORT   TEXT
                                HALT
                        .IRP    PC$,\.
                        .SBTTL  -----> HALT AT PC='PC$ INDICATES "'TEXT"
                        .ENDR
                                BR      .-2
                        .ENDM
```

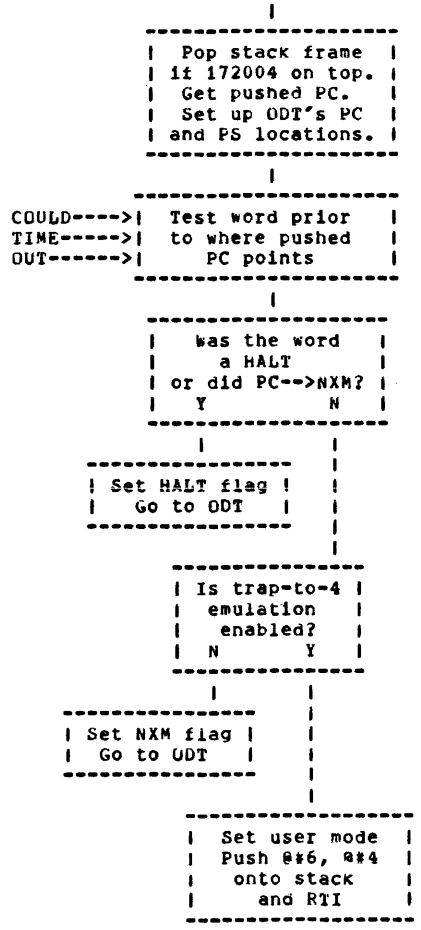
```
                        ;+
                        ; DELAY A,B,N
                        ;       where A and B are names of registers that are free (both will
                        ;       be clear when through) and N is an integer.
                        ;
                        ; This macro produces a delay whose duration (when running in KXT11-AA
                        ; ROM) is .2399N seconds.
                        ; When N<4, it is more efficient to use the following code:
                        ;
                        ;       CLR     Rn                      ;1W             2.44
                        ;       SOB     Rn,.                    ;1W        239861.76
                        ;       [SOB    Rn,.                    ;1W        239861.76]
                        ;       [SOB    Rn,.                    ;1W        239861.76]
                        ;
                        ; The macro generates code like the following:
                        ;
                        ;       MOV     #N,Ra                   ;2W             3.66
                        ;n$:    CLR     Rb                      ;1W           N*2.44
                        ;       SOB     Rb,.                    ;1W      65536N*3.66
                        ;       SOB     Ra,n$                   ;1W           N*3.66
                        ;-

                                .MACRO  DELAY   A,B,N,?L
                                MOV     #N,A
                        L:      CLR     B
                                SOB     B,.
                                SOB     A,L
                                .ENDM
```

```
                                        .SBTTL RAM Definition

                        ; SCRATCH RAM AREA

167776                  TRAP4   ==      167776          ;Enables trap-to-4 emulation
                                                        ; when non-zero
167774                  ODTWHY  ==      167774          ;User-readable copy of R.TYPE.
                                                        ; Restart cause.  See R.TYPE
                                                        ;  table in RESTART routine.
167772                  O.CNTL  ==      167772          ;ODT Control word.  Set Bit 15
                                                        ; to disable T-Bit filter, set
                                                        ;  Bit 7 to disable Priority 7
                                                        ;   filter.
167770                  B.CNTL  ==      167770          ;Boot control word.
167766                  R.PC    ==      167766          ;where restart saves top of stack
167764                  IN.USR  ==      167764          ;Enables user-caused restart
                                                        ; and BREAK when non-zero
167762                  R.TYPE  ==      167762          ;Restart cause.  See table in
                                                        ; RESTART routine.
167760                  USERSP  ==      167760          ;Used by ODT to store the user's
                                                        ; stack pointer.
167756                  RPOINT  ==      167756          ;Used by ODT to point to the image
                                                        ; of user's R0 in its stack.
167754                  SAVPS   ==      167754          ;Store halted PS here for ODT
167752                  SAVPC   ==      167752          ;Store halted PC here for ODT
167750                  ODTFLG  ==      167750          ;Used by ODT for internal flags.
167746                  ODTLOC  ==      167746          ;Used by ODT to point to location
                                                        ; currently open.
167744                  ODTSTK  ==      167744          ;Bottom of ODT's stack
167644                  $STACK  ==      ODTSTK-100      ;Bottom of default user stack
```

```
          170000                            .=170000

                                                    .SBTTL TRAPS-Trap-handling routines
                                                    .SBTTL TRAPS-LTC Trap-killer
                                                    .SBTTL TRAPS-BREAK handler

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;                                                         ;;;;
                                    ;;;;                BREAK-HANDLING ROUTINE                   ;;;;
                                    ;;;;       AND LINE TIME CLOCK INTERRUPT KILLER              ;;;;
                                    ;;;;                                                         ;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 170000                             $$$BRK::
 170000  005767  177760                     TST     IN.USR                  ;Are we in user mode?
 170004  001001                             BNE     BRKNO0                  ;YES-Go to ODT

 170006                             $$$LTC::
 170006  000002                             RTI                             ;NO-Go back to ROM program.
                                                                            ; BREAKS are ignored by ODT,
                                                                            ;  RESTART, POWERUP and the
                                                                            ;   DIAGNOSTICS.  The BOOTS
                                                                            ;    can be interrupted, though.
 170010                             BRKNO0::
 170010  012667  177736                     MOV     (SP)+,SAVPC             ;Save context
 170014  012667  177734                     MOV     (SP)+,SAVPS             ;for ODT.
 170020  012767  100000  177734             MOV     #R.HALT,R.TYPE          ;Causes PC to be printed
                                                                            ; upon entry to ODT.
 170026  005067  177732                     CLR     IN.USR                  ;Get out of user mode
 170032  000167  000544                     JMP     ODT
```

```
                .SBTTL RESTART-Introduction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                    ;;;;
;;;;                   RESTART MODULE                   ;;;;
;;;;                                                    ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;+
;
;The purpose of the RESTART routine is to restore the FALCON to a
;known state following those exceptions which cause a RESTART hardware
;action.  This action consists of stacking the current PSW and program
;counter, then setting the PSW to 340 and jumping to the hardwired
;RESTART location.  This location is at the address START+4 where
;START is jumper selectable as 000000, 010000, 020000, 040000, 100000,
;140000, 172000 or 173000 (all in octal).  This program is designed
;for a START location of 172000, thus RESTARTs jump to 172004.
;
;There are several different ways in which RESTART performs its
;function, depending on the value of IN.USR, TRAP4, the contents
;of the location the SP points to, and one bit (R.STAK) in R.TYPE.
;
;R.TYPE, the restart type word, is RESTART's output to ODT.
;
;-

;+
;
;The goal is to maximize PDP-11 software compatibility and to provide
;useful debugging information to the program developer.
;
;-
```


R.STRT
Enter via hardware mechanism,
with (PS)=340
Is the stack flag set?
N
Y
Set R.NXM
Set IN-ROM mode
Go to ODT
Set stack bit
Read top of stack
Check if too close to "hole"
Clear stack bit
Could time out and cause the
exit to ODT shown above
Did restart occur in user mode?
N
Y
Is top of stack 000000?
N
Y
A BREAK does this when
there's no memory in
the vector area.
Pop stack frame and return
Only a BREAK while in ODT can
get us here, so the RTI takes
us back to ODT.
Set carry in pushed PS and return
Leave user mode
Is top of stack 000000?
A BREAK does this
when there's no
memory in the
vector area
N
Y
Pop stack frame and go to BREAK's SAVE CONTEXT entry point
This entry point is in
the TRAPS module. It is
where a BREAK in user mode
goes when there IS memory
in the vector area.


Pop stack frame
if 172004 on top.
Get pushed PC.
Set up ODT's PC
and PS locations.
COULD----->
TIME----->
OUT------>
Test word prior
to where pushed
PC points
was the word
a HALT
or did PC-->NXM?
Y
N
Set HALT flag
Go to ODT
Is trap-to-4
emulation
enabled?
N
Y
Set NXM flag
Go to ODT
Set user mode
Push @#6, @#4
onto stack
and RTI

```
                        ;+
                        ;
                        ;Exception-type word (R.TYPE) is passed to ODT and is RESTARTs "best guess"
                        ;as to why a restart happened:
                        ;
                        ;       Note: A user-readable copy of this word is at ODTWHY.
                        ;
                        ;|------|-----|--------|-----------------------------------------------
                        ;| EXIT | BIT |  NAME  | CAUSE
                        ;|------|-----|--------|-----------------------------------------------
                        ;|------|     |        |
                        ;| ODT  | 15  | R.HALT | HALT instruction in user code-RESTART POPS STACK.
                        ;|      |     |        | Note-BREAK also sets this bit (see the TRAPS
                        ;|      |     |        | module).  ODT uses this bit for PDP-11 ODT
                        ;|      |     |        | compatibility.
                        ;|------|     |        |
                        ;| ODT  | 14  |        | Reserved
                        ;| OR   | 13  |        | Reserved
                        ;| TRAP | 12  |        | Reserved
                        ;| TO   | 11  |        | Reserved
                        ;| FOUR | 10  |        | Reserved
                        ;|      |  9  |        | Reserved
                        ;|      |  8  |        | Reserved
                        ;|      |  7  | R.NXM  | Timeout during user access of non-existant
                        ;|      |     |        |   memory
                        ;|      |  6  |        | Reserved
                        ;|      |  5  |        | Reserved
                        ;|      |  4  |        | Reserved
                        ;|      |  3  |        | Reserved
                        ;|      |  2  |        | Reserved
                        ;|      |  1  |        | Reserved
                        ;|------|-----|--------|
                        ;| ODT  |  0  | R.STAK | Indicates that a timeout was caused by RESTART
                        ;|      |     |        |   itself accessing non-existant memory.  This
                        ;|      |     |        |   occurs in conjunction with testing for
                        ;|      |     |        |   validity of the stack pointer.
                        ;|      |     |        |   In PDP-11 parlance, this is a
                        ;|      |     |        |   "double-bus error"
                        ;|------|-----|--------|-----------------------------------------------
                        ;
                        ;-
```

```
                                              .SBTTL RESTART-Entry point

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                      ;;;;
                                ;;;;                RESTART ENTRY POINT                   ;;;;
                                ;;;;                                                      ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 170036                         R.STRT::

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                      ;;;;
                                ;;;;     IF THE RESTART ROUTINE CAUSED THE RESTART        ;;;;
                                ;;;;              GO TO ODT AND PRINT "?"                 ;;;;
                                ;;;;                                                      ;;;;
                                ;;;;      THIS EXCEPTION CAN BE CAUSED BY RESTART'S       ;;;;
                                ;;;;                STACK MANIPULATIONS                   ;;;;
                                ;;;;                                                      ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; R.TYPE will have been cleared prior to entering
                                ; any ODT command.  So, if the stack bit is set, only RESTART
                                ; itself could have caused the trap.  Since the stack is always
                                ; valid in in-ROM mode, bad stack means we are in in-USER mode.

                                ;State: X=don't care, U=user, R=in-ROM----
                                ;                                        |
 170036  005767  177720                 TST     R.TYPE                  ;X|Did the stack test fail?
 170042  001406                         BEQ     1$                      ;X|NO- go to next test
 170044  052767  000200  177710         BIS     #R.NXM,R.TYPE           ;U|YES- set R.NXM
                                                                        ;U| this forces "?" from ODT
 170052  005067  177706                 CLR     IN.USR                  ;R|enter in-ROM mode
 170056  000476                         BR      8$                      ;R|go to ODT
```

```
                                                  .SBTTL  RESTART-See if stack exists

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;                STACK VALIDITY TEST                  ;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170060  052767  000001  177674  1$:     BIS     #R.STAK, R.TYPE         ;X|If we timeout, we want RESTART
                                                                        ;X| to know we were diddling SP
170066  005716                          TST     (SP)                    ;X|see if stack is valid
170070  000240                          NOP                             ;X|(in case times out)
170072  005766  000004                  TST     4(SP)                   ;X|see if too close to top of
170076  000240                          NOP                             ;X| valid memory
170100  005067  177656                  CLR     R.TYPE                  ;X|stack is OK

                                                  .SBTTL RESTART-Exit if in IN-ROM state

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;     RETURN WITH CARRY SET IF IN "IN-ROM" MODE       ;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;  OR, GO BACK TO ODT IF A BREAK WITH NO LOW MEMORY   ;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170104  005767  177654                  TST     IN.USR                  ;X|Are we in user mode?
170110  001007                          BNE     3$                      ;U|YES-go to next test
170112  005716                          TST     (SP)                    ;R|NO-see if BREAK brought
                                                                        ;R| us here
170114  001002                          BNE     2$                      ;R|NO-Just a RESTART
170116  022626                          CMP     (SP)+,(SP)+             ;R|YES-Behave like a BREAK that
170120  000002                          RTI                             ;R| happened with RAM

170122  005266  000002          2$:     INC     2(SP)                   ;R|Set carry in pushed PS
                                                                        ;R| UNLESS ALREADY SET
170126  000002                          RTI                             ;R|and return to ROM code that
                                                                        ;R| caused timeout
```

```
                                        .SBTTL RESTART-Cause determination

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                       ;;;;
                                ;;;;        DETERMINE HOW USER CAUSED A RESTART            ;;;;
                                ;;;;                                                       ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170130  005067  177630          3$:     CLR     IN.USR                  ;U|we were in user mode,
                                                                        ;R| but no longer.
170134  005716                          TST     (SP)                    ;R|See if BREAK brought
                                                                        ;R| us here without low "core".
170136  001003                          BNE     4$                      ;R|NO-Just a RESTART
170140  022626                          CMP     (SP)+,(SP)+             ;R|YES-Behave like a BREAK that
170142  000167  177642                  JMP     BRKNUO                  ;R| happened while in user prog.


                                ;
                                ;       If the CPU attempts to fetch an instruction from non-existent
                                ;       memory, two traps (the first from executing a HALT, the second
                                ;       from timing out) will occur, the result being that second
                                ;       trap pushes the restart address and 340 on the stack.
                                ;       This information is useless and gets popped here.

170146  021627  172004          4$:     CMP     (SP),#RESTAR            ;X|Get rid of double stacking
170152  001001                          BNE     5$                      ;X|caused by EXECUTION of NXM
170154  022626                          CMP     (SP)+,(SP)+             ;X|

                                ;       Note: Because the contents of the stack is assumed to remain
                                ;       unchanged following the first instruction below, it is imperative
                                ;       that interrupts be disabled during the next three instrutions.

170156  012667  177604          5$:     MOV     (SP)+,R.PC              ;R|Get pushed PC
170162  011667  177566                  MOV     (SP),SAVPS              ;R|ODT would like
170166  014667  177560                  MOV     -(SP),SAVPC             ;R|to see these

170172  162767  000002  177566          SUB     #2,R.PC                 ;R|Set pointer to last word fetched
                                                                        ;R| before restart occurred
170200  005777  177562                  TST     @R.PC                   ;R|Is contents of pushed PC - 2
                                                                        ;R| a zero (eg a HALT)?
170204  000240                          NOP                             ;R|Make sure next instruction
                                                                        ;R| won't execute if we time out
170206  001005                          BNE     6$                      ;R|NO- it was an NXM
170210  052767  100000  177544          BIS     #R.HALT,R.TYPE          ;R|YES- Flag a HALT,
170216  022626                          CMP     (SP)+,(SP)+             ;R|pop the non-PDP-11 stack frame
170220  000415                          BR      8$                      ;R|and go to ODT.
```

```
                                                   .SBTTL RESTART-Exits

                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;                                                    ;;;;
                                         ;;;;               EXIT APPROPRIATELY                   ;;;;
                                         ;;;;                                                    ;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170222  005767  177550           6$:     TST     TRAP4                   ;R|Trap-to-4 emulation enabled?
170226  001407                           BEQ     7$                      ;R|NO-go to ODT
170230  005167  177530                   COM     IN.USR                  ;U|YES-Set user mode
170234  013746  000006                   MOV     @#6,-(SP)               ;U|Emulate a
170240  013746  000004                   MOV     @#4,-(SP)               ;U|trap to
170244  000002                           RTI                             ;U|four

170246  052767  000200  177506   7$:     BIS     #R.NXM, R.TYPE          ;R|flag NXM error
170254  000167  000322           8$:     JMP     ODT                     ;R|go to ODT
```

```
                    .SBTTL POWERUP-Introduction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                    ;;;;
;;;;                  POWER-UP MODULE                   ;;;;
;;;;                                                    ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; This module contains a series of routines which perform
; tests on the on-board RAM and the console DLART.  These
; tests are preceded by the lighting of the LED on the
; KXT11-AA board, and followed by its extinguishing.  Should
; the LED fail to either light or go out, there may be a
; defect in the board or its configuration.

; Following these tests, the on-board RAM is written with the
; default values of certain control words, and, if there is
; memory in the vector region (i.e., near 000000), the BREAK
; and clock vectors are set up.  If not, a bit is set in the
; boot control word to disable the bootstraps.
```

```
                                                  .SBTTL POWERUP-Turn on LED

                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;                                                    ;;;;
                                   ;;;;                   TURN ON LED                      ;;;;
                                   ;;;;                                                    ;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


170260                             PWRSUP::
170260  012706  167644                     MOV     #$STACK,SP              ;Initialize stack pointer

                                   ; Because a mode-setting command automatically clears all the internal
                                   ; registers in the PPI, and clearing Port C Bit 7 turns on the LED, all
                                   ; we have to do is set the mode, which is port A and lo half of C as
                                   ; input, ports B and hi half of C as output.

170264  012737  000221  176206             MOV     #MODE,@#PP.CWR          ;Set proper PPI mode

                                                  .SBTTL POWERUP-Test console DLART

                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;                                                    ;;;;
                                   ;;;;           CHECK THE CONSOLE DLART                  ;;;;
                                   ;;;;                                                    ;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170272  005037  177564                     CLR     @#XCSR$1                ;Disable XMIT interrupts,
                                                                           ; BRK XMIT, maint. mode
                                                                           ;  Set baud rate to default
170276  005737  177562                     TST     @#RBUF$1                ;Take out the trash.
170302  032737  000300  177560             BIT     #<RC.IEN!RC.DON>,@#RCSR$1
                                                                           ;Should be clear.
170310  001377                             BNE     .                       ;If not, drop dead.
170312  023727  177564  000200             CMP     @#XCSR$1,#XC.RDY        ;Should be set
170320  001377                             BNE     .                       ;If not, rest in peace.
```

```
                                                 .SBTTL POWERUP-Test and set up I/O-page RAM

                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 ;;;;                                                     ;;;;
                                 ;;;;                  I/O PAGE RAM TEST                  ;;;;
                                 ;;;;                  AND INITIALIZATION                 ;;;;
                                 ;;;;                                                     ;;;;
                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                 ; Write the location's address into the location and read it back.
                                 ; Do this for all I/O page RAM locations.
                                 ; If it fails, enter tight loop.

                                 ; In the process, clear all of this RAM.  Note that the default
                                 ; value of most of the control and flag words is zero.

   170322  012700  160010                MOV     #RAMBOT,R0              ;Lowest address of RAM
   170326  010010                1$:     MOV     R0,(R0)                 ;Write the address
   170330  020010                        CMP     R0,(R0)                 ;Read it back
   170332  001377                2$:     BNE     2$                      ;Tight failure loop
   170334  005020                        CLR     (R0)+                   ;Clear and go on to next location
   170336  020027  170000                CMP     R0,#RAMTOP+2            ;Until no more to test.
   170342  103771                        BLO     1$
```

```
                                                .SBTTL POWERUP-Turn off LED

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;  DELAY SO THAT LED IS ON A VISIBLE LENGTH OF TIME  ;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170344  005000                              CLR     R0
170346  077001                      3$:     SOB     R0,3$                   ;This leaves a 0 in R0, which
170350  077001                      4$:     SOB     R0,4$                   ; is essential for testing for
170352  077001                      5$:     SOB     R0,5$                   ;  the presence of memory at
170354  077001                      6$:     SOB     R0,6$                   ;   zero below.

                                    ; Under no circumstances can R0 be altered until "low core" test below.

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;                  TURN OFF LED                      ;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170356  012737  000017  176206              MOV     #LEDOFF,@#PP.CWR

                                                .SBTTL POWERUP-Test for "low core"

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;              TEST FOR MEMORY AT 000000             ;;;;
                                    ;;;;                                                    ;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                    ; Read memory at 000000, discard result.  If this fails, exit to
                                    ; AUTOBAUD rather than continuing with normal powerup sequence.

170364  005710                              TST     (R0)
170366  000240                              NOP                             ;This will execute even if
                                                                            ; last instruction times out
170370  103403                              BCS     7$                      ;Timed out, don't set vectors
170372  004767  000042                      CALL    VECSET                  ;Didn't time out, so go ahead
170376  000403                              BR      8$                      ;Don't set the NO.LOW flag

170400  052767  100000  177362      7$:     BIS     #NO.LOW,B.CNTL          ;Did time out,
                                                                            ; so let the world know.
```

```
                                        .SBTTL POWERUP-Exit

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;            EXIT FROM POWER-UP SEQUENCE             ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170406  012767  000300  177340  8$:     MOV     #PRI6,SAVPS             ;If P is typed in reponse to
170414  012767  170424  177330          MOV     #FAKOUT,SAVPC           ; ODT prompt before loading R7,
170422  000423                          BR      AUTOBA                  ;  will force yet more ODT.

170424                          FAKOUT:
170424  005067  177334                  CLR     IN.USR                  ;   BUT IN THE RIGHT MODE!
170430  012706  167644                  MOV     #$STACK,SP              ;    And without running out of
170434  000167  000142                  JMP     ODT                     ;     stack, either.

                                        .SBTTL POWERUP-Subroutine to initialize vectors

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;                INITIALIZE VECTORS                  ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Note: This subroutine is also used by the bootstrap module, to
                                ; restore the vector area in the event that an invalid boot block
                                ; was read into low memory.

170440                          VECSET::
170440  012737  170000  000140          MOV     #$$$BRK,@#140           ;Set up the BREAK-detect
170446  012737  000340  000142          MOV     #PRI7,@#142             ; vector.
170454  012737  170006  000100          MOV     #$$$LTC,@#100           ;Set up the line time clock
170462  012737  000340  000102          MOV     #PRI7,@#102             ; vector.
170470  000207                          RETURN
```

```
                                         .SBTTL  AUTOBAUD-Synchronize with Console

                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;                                                    ;;;;
                               ;;;;                 AUTOBAUD MODULE                    ;;;;
                               ;;;;                                                    ;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                               ; Description:
                               ;
                               ;       AUTOBAUD allows the FALCON to automatically synchronize its
                               ;       console DLART to the baud rate of the console terminal.
                               ;       On power-up, the user must type a carriage return character.
                               ;       Upon synchronization, AUTOBAUD will proceed to ODT where an '@'
                               ;       character will be displayed on the console.
                               ;
                               ;       Autobaud will loop indefinitely until synchronization is successful.
                               ;
                               ;       The algorithm requires that the console terminal generates a
                               ;       zero (space) for the eighth bit in the carriage return.  This
                               ;       will happen if the terminal is capable of sending eight-bit-
                               ;       no-parity or seven-bit-odd-parity characters.
                               ;
                               ; Environment:
                               ;
                               ;       Interrupts must be disabled for the algorithm to execute correctly
                               ;       since time durations are critical and delays due to long
                               ;       service routines may cause DLART overruns, which this routine
                               ;       ignores but cannot tolerate.
                               ;


                               ; VT103/FALCON configurations leave garbage in the DLART long after the
                               ; powerup sequence has begun.  We must delay a bit before clearing garbage
                               ; out of the DLART, otherwise the garbage would arrive after the clear
                               ; (i.e., while polling for input).  The "garbage" is an X-ON (<CTRL-q>)
                               ; that the VT-100 hardware sends after its power-up diagnostics have
                               ; completed successfully.
                               ;

170472                         AUTOBA::
170472  012737  000032  177564          MOV     #BAUDR$,@#XCSR$1        ;Set 2400 baud
170500  005000                          CLR     R0                      ;Delay
170502  077001                          SOB     R0,.                    ;       .5
170504  077001                          SOB     R0,.                    ;          seconds
```

```
                                ; AUTOBAUD proper:
                                
170506  105737  177562        10$:    TSTB    @#RBUFS1                ; discard any garbage
                                
170512  105737  177560        20$:    TSTB    @#RCSR$1                ; wait for input
170516  100375                         BPL     20$
170520  113700  177562                 MOVB    @#RBUFS1, R0            ; R0 = input character
170524  012701  170550                 MOV     #INBYTE, R1             ; R1 -> scrambled char table
170530  120021                 30$:    CMPB    R0, (R1)+               ; in the table?
170532  001411                         BEQ     HVBAUD                  ; yes
170534  020127  170556                 CMP     R1, #INBYT$             ; end of table reached?
170540  001373                         BNE     30$                     ; not yet
170542  005000                         CLR     R0                      ; uh oh, wait for DLART to clear out
170544  077001                 40$:    SOB     R0, 40$                 ; wait for a while
170546  000757                         BR      10$                     ; and try for another character
                                
                                ; Table of what you would see if an octal 15 were sent at the following
                                ; baud rates.
                                
170550                          INBYTE:
170550      200                         .BYTE   200                     ; 300
170551      170                         .BYTE   170                     ; 600
170552      346                         .BYTE   346                     ; 1200
170553      015                         .BYTE   15                      ; 2400
170554      362                         .BYTE   362                     ; 4800
170555      377                         .BYTE   377                     ; 9600, 19200, 38400
170556                          INBYT$:
                                
                                ; We have a match.  Set baud rate into DLART.
                                
170556                          HVBAUD:
170556  162701  170551                 SUB     #INBYTE+1, R1           ; turn pointer into bit mask
170562  006301                         ASL     R1
170564  006301                         ASL     R1
170566  005201                         INC     R1                      ; turn on XC.PBE
170570  006301                         ASL     R1                      ; set the baud rate
170572  010137  177564                 MOV     R1, @#XCSR$1            ; into CSR
170576  005000                         CLR     R0                      ; delay .24 seconds for rest
170600  077001                         SOB     R0,.                    ; of char. at slo baud rates
                                
                                ; Fall into ODT.
```

```
                        .SBTTL  macroODT-Introduction

        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;                                                        ;;;;
        ;;;;                        macroODT                        ;;;;
        ;;;;                                                        ;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ; macroODT is the user interface to the functions contained
        ; in the KXT11-A2 firmware product.  It interprets commands
        ; entered via the console terminal keyboard (see tables below)
        ; to permit the user to load a program into memory, execute
        ; it and debug it.

        ;   COMMAND
        ;   1- Slash (/)
        ;      a-OPEN MEMORY LOCATION
        ;      b-OPEN GENERAL REGISTER
        ;      c-OPEN STATUS REGISTER
        ;   2- Carriage return (<CR>)
        ;      a-CHANGE AND CLOSE MEMORY LOCATION OR REGISTER
        ;      b-CLOSE WITHOUT CHANGE
        ;   3- Line feed (<LF>)
        ;      a- CHANGE AND CLOSE MEMORY LOCATION AND OPEN NEXT
        ;      b- CLOSE MEMORY LOCATION WITHOUT CHANGING AND OPEN NEXT
        ;   4- Go (G)
        ;   5- Proceed (P)
        ;   6- Execute I/O diagnostics (X)
        ;   7- Execute bootstraps (D)
```

```
                        ;SYNTAX OF COMMANDS LISTED ABOVE, SHOWING CONSOLE BEFORE,
                        ;DURING AND AFTER THE TYPING OF THE COMMAND.
                        ;       Key: n-an octal integer typed by the user, only
                        ;              last 6 digits significant
                        ;            x-a single octal digit
                        ;            u-the digits 0 or 1
                        ;            all other characters are literals
                        ;
                        ;   BEFORE          DURING                AFTER
                        ;
                        ;1a @               @n/                   @n/xxxxxx
                        ;1b @               @Rx/                  @Rx/xxxxxx
                        ;1c @               @RS/                  @RS/xxxxxx
                        ;2a @n/xxxxxx       @n/xxxxxx n<CR>       @
                        ;2a @Rx/xxxxxx      @Rx/xxxxxx n<CR>      @
                        ;2a @RS/xxxxxx      @Rx/xxxxxx n<CR>      @
                        ;2a xxxxxx/xxxxxx   xxxxxx/xxxxxx n<CR>   @
                        ;2b @n/xxxxxx       @n/xxxxxx <CR>        @
                        ;2b @Rx/xxxxxx      @Rx/xxxxxx <CR>       @
                        ;2b @RS/xxxxxx      @RS/xxxxxx <CR>       @
                        ;2b xxxxxx/xxxxxx   xxxxxx/xxxxxx <CR>    @
                        ;3a @n/xxxxxx       @n/xxxxxx n<LF>       xxxxxx/xxxxxx
                        ;3a xxxxxx/xxxxxx   xxxxxx/xxxxxx n<LF>   xxxxxx/xxxxxx
                        ;3b @n/xxxxxx       @n/xxxxxx <LF>        xxxxxx/xxxxxx
                        ;3b xxxxxx/xxxxxx   xxxxxx/xxxxxx <LF>    xxxxxx/xxxxxx
                        ;4  @               @nG
                        ;5  @               @P
                        ;6  @               @X                    xxxxxx
                        ;                                         @
                        ;7  @               @DDu
                        ;7  @               @DXu
                        ;7  @               @DYu
                        ;7  @               @DD<CR>
                        ;7  @               @DX<CR>
                        ;7  @               @DY<CR>
```

```
                                        .SBTTL macroODT-Save status and print prompt

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;      SAVE CONTEXT, PRINT MESSAGES AND PROMPT       ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170602                          ODT::
170602  105737  177562                  TSTB    @#RBUF$1                ;Clear out console garbage

                                ; Copy the restart type word into user area

170606  016767  177150  177160          MOV     R.TYPE,ODTWHY

                                ; Protect against stack timeouts, but save user's SP first

170614  010667  177140                  MOV     SP,USERSP               ;SAVE USERS STACK POINTER
170620  012706  167744                  MOV     #ODTSTK,SP              ;LOAD NEW SP

                                ; Save rest of user program's context

170624  016716  177130                  MOV     USERSP,(SP)             ;RESERVE LOCATION FOR R6
170630  010546                          MOV     R5,-(SP)                ;SAVE
170632  010446                          MOV     R4,-(SP)                ; ALL
170634  010346                          MOV     R3,-(SP)                ;  OF
170636  010246                          MOV     R2,-(SP)                ;   USER'S
170640  010146                          MOV     R1,-(SP)                ;    GENERAL
170642  010046                          MOV     R0,-(SP)                ;     REGISTERS
170644  010667  177106                  MOV     SP,RPOINT               ;POINTER TO R0

                                ; Determine whether "?" or PC message is appropriate, and print it

170650  005767  177106                  TST     R.TYPE                  ;Did we get a HALT or BREAK?
170654  100004                          BPL     QODT                    ;NO-next question
                                                                        ;YES-PRINT PC
170656  016700  177070                  MOV     SAVPC,R0                ;GET STOPPED PC
170662  004767  000764                  CALL    OCTSTO                  ;TYPE THE PC ON TERMINAL
170666                          QODT:
170666  105767  177070                  TSTB    R.TYPE                  ;SEE IF RESTART OCCURRED
                                                                        ;(NXM ONLY-BIT 7 SET)
170672  100003                          BPL     KBD$                    ;TYPE PROMPT

                                ; Here's where the prompt gets printed, with or without leading "?"

170674                          KBDQ:
170674  012700  171730                  MOV     #MSGQ,R0                ; GET ? ADDRESS
170700  000402                          BR      PRINT                   ;TYPE IN MESSAGE
170702                          KBD$:
170702  012700  171731                  MOV     #MSG$,R0                ;GET PROMPT MESSAGE ADDRESS

170706  005067  177050          PRINT:  CLR     R.TYPE                  ;So reentry gives no error msg.
170712  106427  000300                  MTPS    #PRI6                   ;Allow BREAKs to happen
170716  004767  000620                  CALL    PUTSTR                  ;TYPE THE PROMPT ALREADY
170722  005067  177022                  CLR     ODTFLG                  ;CLEAR FLAG FOR NEW ENTRY
```

```
                                     .SBTTL  macroODT-Get ODT command

                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;                                                     ;;;;
                             ;;;;       INTERPRET FIRST CHARACTER OF COMMAND          ;;;;
                             ;;;;                                                     ;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                             ; Note: Following CALL GETCHR, the character (7 bit ASCII)
                             ;  appears in R2.
                             ; Note: Following CALL GETNUM, if carry is clear, the octal integer
                             ;  was followed by a carriage return.
                             ; Note: On exit to LCSET or falling through to GO routine, R0 contains
                             ;  the address typed in.

170726  004767  000556               CALL    GETCHR                  ;...INPUT CHARACTERS
170732  120227  000104               CMPB    R2,#'D                  ;BOOTSTRAPS?
170736  001002                       BNE     1$                      ;NO
170740  000167  001220               JMP     BOOTS                   ;YES

170744  120227  000130       1$:     CMPB    R2,#'X                  ;DIAGNOSTICS?
170750  001002                       BNE     2$                      ;NO
170752  000167  000776               JMP     DIAGNO                  ;YES

170756  120227  000120       2$:     CMPB    R2,#'P                  ;PROCEED?
170762  001430                       BEQ     PCMD                    ;YES
170764  120227  000122               CMPB    R2,#'R                  ;REGISTER?
170770  001465                       BEQ     RCMD                    ;YES
170772  120227  000060               CMPB    R2,#'0                  ;OCTAL DIGIT?
170776  103736                       BLO     KBDQ                    ;NO,ERROR
171000  120227  000070               CMPB    R2,#'8                  ;VALID DIGIT?
171004  103333                       BHIS    KBDQ                    ;NO,ERROR
171006  005000                       CLR     R0                      ;ITS A DIGIT
171010  004767  000576               CALL    GETNUM                  ;GET REST OF THE DIGIT OR CMD
171014  103327                       BCC     KBDQ                    ;CR WAS ISSUED,ERROR

                             ; The last character at the end of the number could be a valid command-
                             ; Let's check:

171016  120227  000057               CMPB    R2,#'/                  ;EXAMINE LOCATION?
171022  001511                       BEQ     LCSET                   ;YES
171024  120227  000107               CMPB    R2,#'G                  ;GO TO?
171030  001321                       BNE     KBDQ                    ;NO,ERROR
```

```
                        ; TABLE OF PERMISSABLE STATES
                        ;
                        ;       NO.     STATE           VALID INPUTS            COMMENT
                        ;       1-      prompt @          0-7       --------> digit.
                        ;                                 P         --------> proceed.
                        ;                                 R         --------> register designator.
                        ;                                 X         --------> execute diagnostic
                        ;                                 D         --------> boot from device
                        ;       2-      @175620           0-7       --------> another digit.
                        ;               [input digit]     /         --------> examine loc.
                        ;                                 G         --------> go from loc n.
                        ;       3-      @176000/000002    0-7       --------> input new value.
                        ;                                 LF        --------> display next loc.
                        ;                                 CR        --------> close loc go to prompt.
                        ;       4-      @200/000023 12    0-7       --------> input more digits.
                        ;                                 LF        --------> save data display next.
                        ;                                 CR        --------> save data go to prompt.
                        ;       5-      @R                0-7       --------> register number.
                        ;                                 S         --------> PSW.
                        ;       6-      @R5               /         --------> examine.
                        ;       7-      @R5/000024        0-7       --------> input new value.
                        ;                                 CR        --------> close location.
                        ;       8-      @R5/000024 16     0-7       --------> more digits input
                        ;                                 CR        --------> save value go to prompt
                        ;
```

```
                                        .SBTTL  macroODT- Go and Proceed

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;        PROCESS GO AND PROCEED ODT COMMANDS         ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

171032  010067  176714                  MOV     R0,SAVPC                ;PUT SUPPLIED PC IN MEMORY LOCATION

                                ; Prepare the environment for the Go command

171036  000005                          RESET                           ;BUS INITIALIZE
171040  005067  176710                  CLR     SAVPS                   ;CLEAR PSW

                                ; Entry point for the Proceed command

                                ; First, check for valid stack:

171044                          PCMD:

171044  016600  000014                  MOV     14(SP),R0               ;User's stack pointer
171050  005740                          TST     -(R0)                   ;Where SAVPS will go (see below)
171052  000240                          NOP                             ; (in case of time out)
171054  103403                          BCS     1$                      ;No good.  Timed out.
171056  005740                          TST     -(R0)                   ;Where SAVPC will go
171060  000240                          NOP                             ;
171062  103004                          BCC     2$                      ;Sufficient stack.

                                ; EITHER Stack no good, so simulate a double bus trap without losing the
                                ;        user's context as stored in the ODT stack.

171064  012767  000201  176702  1$:     MOV     #R.STAK!R.NXM,ODTWHY    ;Sneaky! (R.TYPE untouched-
                                                                        ; only the user image of it)
171072  000700                          BR      KBDQ                    ;Error prompt.

                                ; OR     Stack is OK, so restore user's context.

171074  012600                  2$:     MOV     (SP)+,R0                ;RESTORE
171076  012601                          MOV     (SP)+,R1                ; ALL
171100  012602                          MOV     (SP)+,R2                ;  OF
171102  012603                          MOV     (SP)+,R3                ;   USER'S
171104  012604                          MOV     (SP)+,R4                ;    GENERAL
171106  012605                          MOV     (SP)+,R5                ;     REGISTERS

171110  106427  000340                  MTPS    #PRI7                   ;No BREAKS allowed until out of
                                                                        ; ODT!
171114  042716  000001                  BIC     #BIT0,(SP)              ;Odd stacks are too odd for T-11
171120  011606                          MOV     (SP),SP                 ;RESTORE USER SP
171122  005167  176636                  COM     IN.USR                  ;Set user mode
171126  016746  176622                  MOV     SAVPS,-(SP)             ;RESTORE PC AND PS TO ...
171132  016746  176614                  MOV     SAVPC,-(SP)             ;...STACK WHERE RTT WILL LOOK
171136  000006                          RTT                             ;RETURN TO USERS PROGRAM
171140  000655                  HKBDQ:  BR      KBDQ                    ;HELP IN BR
171142  000657                  HKBDS:  BR      KBDS                    ;HELP IN BR
```

```
                                        .SBTTL  macroODT-Register and PS command

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;            PROCESS ODT REGISTER COMMANDS               ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Entry point for Rx and RS commands

171144                          RCMD:
171144  052767  000200  176576          BIS     #RFLAG,ODTFLG           ;SET REGISTER FLAG
171152  004767  000420                  CALL    ONENUM                  ;GET REGISTER NUMBER
171156  103246                          BCC     KBDQ                    ;A VALID CMD DID NOT FOLLOW
171160  120227  000123                  CMPB    R2,#'S                  ;IS IT THE RS?
171164  001412                          BEQ     SWCMD                   ;YES,BRANCH
171166  120227  000057                  CMPB    R2,#'/                  ;EXAMINE?
171172  001240                          BNE     KBDQ                    ;NO,ERROR
171174  020027  000007                  CMP     R0,#7                   ;>7?
171200  101235                          BHI     KBDQ                    ;YES,ERROR
171202  001013                          BNE     RCMD1                   ;IS IT EXACTLY SEVEN
171204  012700  167752                  MOV     #SAVPC,R0               ;YES,GET PC ADDRESS
171210  000413                          BR      REGOUT                  ;DISPLAY

                                ; Status register (PS) selected:

171212                          SWCMD:
171212  004767  000272                  CALL    GETCHR                  ;WHAT YOU WANT TO DO WITH RS?
171216  120227  000057                  CMPB    R2,#'/                  ;EXAMINE?
171222  001224                          BNE     KBDQ                    ;NO,ERROR
171224  012700  167754                  MOV     #SAVPS,R0               ;GET ADDRESS WHERE PS IS
171230  000403                          BR      REGOUT                  ;GO AND DISPLAY

171232  006300                  RCMD1:  ASL     R0                      ;SHIFT FOR OFFSET IN MEMORY
171234  066700  176516                  ADD     RPOINT,R0               ;GET EXACT ADDRESS OF REG.
171240  010067  176502          REGOUT: MOV     R0,ODTLOC               ;STORE LOCATION
171244  000402                          BR      LOCDSP                  ;DISPLAY
```

```
                                        .SBTTL macroODT-Examine and Deposit

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;  PROCESS ODT MEMORY AND REGISTER EXAMINE/DEPOSIT      ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; ODTLOC points to register or memory location
                                ; Following CALL GETNUM, if carry is clear, CR followed digit.
                                ; ODTFLG: If register bit set indicates register is being examined

                                ;ENTRY FROM CMD ROUTINE AFTER LOC. VALUE IS GIVEN

171246  010067  176474          LCSET:  MOV     R0,ODTLOC               ;SAVE NEW LOCATION
171252  011000                  LOCDSP: MOV     (R0),R0                 ;GET DATA
171254  000240                          NOP                             ;So next inst. does not execute
                                                                        ;if we time out.
171256  103730                          BCS     HKBDQ                   ;Print "?" if we timed out
171260  004767  000372                  CALL    OCTSTR                  ;PRINT IT
171264  112702  000040                  MOVB    #SPACE,R2               ;Print a space after the data
171270  004767  000226                  CALL    PUTCHR
171274  004767  000210                  CALL    GETCHR                  ;GET NEXT CHARACTER
171300  120227  000015                  CMPB    R2,#CR                  ;FINISH
171304  001716                          BEQ     HKBD$                   ;YES,CLOSE LOCATION
171306  120227  000060                  CMPB    R2,#'0                  ;DEPOSIT?
171312  103450                          BLO     4$                      ;NO,CHECK LF
171314  120227  000070                  CMPB    R2,#'8                  ;MAYBE!
171320  103307                          BHIS    HKBDQ                   ;NO,FORGET IT
171322  005000                          CLR     R0                      ;YES
171324  004767  000262                  CALL    GETNUM                  ;GET REST OF NUMBER
171330  103006                          BCC     1$                      ;CR FOUND, STORE NEW VALUE

171332  120227  000012                  CMPB    R2,#LF                  ;Not CR, must be LF
171336  001300                          BNE     HKBDQ                   ;Print error message
171340  105767  176404                  TSTB    ODTFLG                  ;If LF, cannot be register
171344  100675                          BMI     HKBDQ                   ;(Error exit)

                                ;T-BIT FILTER.  The T-BIT can be set from the keyboard via ODT.
                                ;This can either be useful for debugging or disastrous.  So, you can
                                ;do it only if you first set FILT.T in O.CNTL (BIT 15).

171346  022767  167754  176372  1$:     CMP     #SAVPS,ODTLOC           ;Are we diddling the PS?
171354  001021                          BNE     3$                      ;No, we're not.
171356  042700  177400                  BIC     #^C<377>,R0             ;PS is not a word.
171362  005767  176404                  TST     O.CNTL                  ;Is BIT 15 (FILT.T) SET?
171366  100402                          BMI     2$                      ;Yes, the filter's disabled
171370  042700  000020                  BIC     #T.BIT,R0               ;KILL THE T-BIT

                                ;2$:                                    ;Fall thru to Priority 7
                                                                        ; Filter
```

```
                                        ;Priority-7 filter: unless FILT.7 (BIT 7) in O.CNTL is set, you cannot
                                        ;actually set the PS to priority 7 using ODT from the keyboard.  This
                                        ;protects the ability to break.
                                        
171374  105767  176372          2$:   TSTB    O.CNTL                  ;ODT control word
171400  100407                        BMI     3$                      ;Do nothing-filter disabled
171402  105700                        TSTB    R0                      ;Intended new PS
171404  100005                        BPL     3$                      ;Do nothing-Priority < 4
171406  032700  000100                BIT     #BIT6,R0                ;Check again
171412  001402                        BEQ     3$                      ;Do nothing-Priority < 6
171414  042700  000040                BIC     #BIT5,R0                ;LOWER THE BOOM
171420  010077  176322          3$:   MOV     R0,@ODTLOC              ;STORE NEW VALUE
171424  120227  000012                CMPB    R2,#LF                  ;Go on to next location?
171430  001407                        BEQ     5$                      ;Sure, why not.
171432  000643                        BR      HKBD$                   ;GO TO PROMPT
                                        
171434  120227  000012          4$:   CMPB    R2,#LF                  ;IS A LF ISSUED
171440  001237                        BNE     HKBDQ                   ;NO,ERROR
171442  105767  176302                TSTB    ODTFLG                  ;IS REGISTER FLAG SET
171446  100634                        BMI     HKBDQ                   ;YES, LF NOT PERMITTED
171450  112702  000015          5$:   MOVB    #CR,R2                  ;TO LINE UP CURSOR
171454  004767  000042                CALL    PUTCHR                  ;SEND IT
171460  062767  000002  176260        ADD     #2,ODTLOC               ;GET ADDRESS OF NEXT LOC.
171466  016700  176254                MOV     ODTLOC,R0               ;GET NEXT ADDRESS VALUE...
171472  004767  000160                CALL    OCTSTR                  ;...AND PRINT IT
171476  112702  000057                MOVB    #'/,R2                  ;SEND A SLASH BEFORE...
171502  004767  000014                CALL    PUTCHR                  ;...SHOWING THE CONTENTS...
171506  000661                        BR      LOCDSP                  ;...OF THE LOCATION
```

```
                                        .SBTTL  macroODT-Get and echo character

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;        CHARACTER INPUT AND ECHO SUBROUTINE         ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Get a character from the console keyboard and echo it back
                                ; exactly as received including parity bits if any.  Return with
                                ; character in R2, eighth bit (and high byte) zero.

 171510                         GETCHR:
 171510  105737  177560                 TSTB    @#RCSR$1                ;CHARACTER READY?
 171514  100375                         BPL     GETCHR                  ;BRANCH IF NOT AND KEEP TRYING
 171516  113702  177562                 MOVB    @#RBUF$1,R2             ;TRANSFER CHARACTER
 171522                         PUTCHR:
 171522  105737  177564                 TSTB    @#XCSR$1                ;PRINTER READY
 171526  100375                         BPL     PUTCHR                  ;NO, TRY AGAIN
 171530  110237  177566                 MOVB    R2,@#XBUF$1             ;YES, XMIT CHARACTER
 171534  042702  177600                 BIC     #^C<177>,R2             ;CLEAR PARITY
 171540  000207                         RETURN                          ;CONTINUE
```

```
                                        .SBTTL  macroODT-Type ASCII string

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;             MESSAGE PRINT SUBROUTINE               ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Print message starting with character pointed to by R0 and
                                ; ending with first character with eighth bit set (this character
                                ; is not printed).

171542                          PUTSTR:
171542  112002                          MOVB    (R0)+,R2                ;GET ASCII CHAR
171544  100413                          BMI     DONE                    ;IS IT THE END MARK?
171546  004767  177750                  CALL    PUTCHR                  ;NO, PRINT IT
171552  000773                          BR      PUTSTR                  ;MORE

                                ;ENTRY FOR CARRIAGE RETURN

171554                          PUTCLF:
171554  112702  000015                  MOVB    #CR,R2                  ;PRINT CR
171560  004767  177736                  CALL    PUTCHR                  ;FALL THRU AND PRINT LF

                                ;ENTRY FOR LF

171564  112702  000012          PUTLF:  MOVB    #LF,R2                  ;PRINT LF
171570  004767  177726                  CALL    PUTCHR
171574  000207                  DONE:   RETURN
```

```
                                      .SBTTL  macroODT-Get octal digits

                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;                                                     ;;;;
                              ;;;;                NUMERIC INPUT ROUTINE                ;;;;
                              ;;;;                                                     ;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                              ; On exit, R0 contains the binary representation of the number entered
                              ; If the carry bit is clear, a <CR> followed number
                              ; If the carry bit is set, some other character followed the number,
                              ; possibly a command.

171576                        ONENUM:
171576  005000                        CLR     R0                      ;CLEAR ACCUMULATOR
171600                        NEXNUM:
171600  004767  177704                CALL    GETCHR                  ;GET DIGIT OR TERMINATOR
171604  120227  000015                CMPB    R2,#CR                  ;CLEAR CARRY AND RETURN
171610  001412                        BEQ     SRET                    ;IF <CR> WAS TYPED
171612                        GETNUM:
171612  162702  000070                SUB     #'8,R2                  ;CONVERT TO BINARY...
171616  062702  000010                ADD     #'8-'0,R2               ;...AND TEST IF OCTAL OR NOT
171622  103007                        BCC     NOCT                    ;NOT VALID DIGIT.
171624  006300                        ASL     R0                      ;MAKE ROOM FOR NEW DIGIT
171626  006300                        ASL     R0                      ;DITTO
171630  006300                        ASL     R0                      ;DITTO
171632  050200                        BIS     R2,R0                   ;PUT IT IN PLACE
171634  000761                        BR      NEXNUM                  ;GET NEXT

171636  000241                SRET:   CLC                             ;CLEAR CARRY
171640  000207                        RETURN                          ;CONTINUE

171642  062702  000060        NOCT:   ADD     #'0,R2                  ;RESTORE ASCII BECAUSE...
171646  000261                        SEC                             ;...POSSIBLE COMMAND
171650  000207                        RETURN                          ;CONTINUE
```

```
                                    .SBTTL  macroODT-OCTSTR--type binary in R0 as ASCII

                            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                            ;;;;                                                    ;;;;
                            ;;;;              NUMERIC OUTPUT ROUTINE                ;;;;
                            ;;;;                                                    ;;;;
                            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                            ; Prints, as a 6-digit octal integer, the value of the binary
                            ; number in R0.

171652  004767  177676      OCTSTO: CALL    PUTCLF                  ;NEED CRLF AT ODT ENTRY
171656                      OCTSTR:
171656  010046                      MOV     R0,-(SP)                ;SAVE VALUE
171660  012746  000006              MOV     #6,-(SP)                ;NO. OF CHARACTERS
171664  005002                      CLR     R2                      ;OUTPUT HOLD
171666  006100              5$:     ROL     R0                      ;SHIFT MSB INTO LSB
171670  006102                      ROL     R2                      ;" " " " " " " " " "
171672  062702  000060              ADD     #'0,R2                  ;MAKE A DIGIT
171676  004767  177620              CALL    PUTCHR                  ;OUTPUT A CHARACTER
171702  005316                      DEC     (SP)                    ;COUNT
171704  001406                      BEQ     10$                     ;DONE
171706  005002                      CLR     R2                      ;NEXT
171710  006100                      ROL     R0                      ;GET NEXT DIGIT INTO
171712  006102                      ROL     R2                      ;R2
171714  006100                      ROL     R0                      ;FIRST TWO BITS
171716  006102                      ROL     R2                      ;" "  " "" " "
171720  000762                      BR      5$                      ;CONTINUE
171722  005726              10$:    TST     (SP)+                   ;CLEAR COUNT
171724  012600                      MOV     (SP)+,R0                ;ORIGINAL VALUE
171726  000207                      RETURN                          ;
```

```
                                                .SBTTL  macroODT-Output messages

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                            ;;;;
                                        ;;;;                        MESSAGES                            ;;;;
                                        ;;;;                                                            ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                                .NLIST  BEX
 171730     077                         MSGQ:   .ASCII  '?'                     ; ERROR MESSAGE
 171731     015     012     100         MSGS:   .ASCII  <CR><LF>'@'<200>        ; PROMPT
                                                .EVEN
                                                .LIST   BEX
```

```
                                                  .SBTTL DIAGNOSTICS-for SLU2 and PPI

                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;                                                     ;;;;
                                          ;;;;                 DIAGNOSTIC MODULE                   ;;;;
                                          ;;;;                                                     ;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                          ; Diagnose PPI in mode 0 with loopback connectors installed.
                                          ; Diagnose SLU2 internal circuitry (maintenance mode) and
                                          ; SLU2 drivers/receivers (with external loopback connector).

                                          ; List of error bit definitions to return to user.

171736  000100                                    .WORD   E.EXT
171740  000010                                    .WORD   E.INT
171742                                    ERRBIT:

                                          ; List of masks to put in XCSR82.  Perform internal loopback
                                          ; test first, then external loopback test.

171742  000000                                    .WORD   0
171744  000002                                    .WORD   XC.PBE                  ; 300 baud
171746  000006                                    .WORD   XC.PBE ! XC.MNT         ; 300 baud and maintenance
171750                                    INITS:

                                          ; List of pattern bytes to loop around.
                                          ; All bits on, alternating bits, all bits off.
                                          ; Note: last byte must be 0.

171750     377      252      000          PATERN: .BYTE   377, 252, 0
                                                  .EVEN

                                                  .ENABL  LSB
171754                                    DIAGNO:
171754  012737  000221  176206                    MOV     #MODE,@#PP.CWR          ; set proper PPI mode- LED
171762  012737  000017  176206                    MOV     #LEDOFF,@#PP.CWR        ; must immediately be turned
                                                                                  ; off as a consequence.

171770  005000                                    CLR     R0                      ; assume success

                                          ; Perform parallel port diagnostic

171772  005001                                    CLR     R1                      ; R1 = loop pattern
171774  000405                                    BR      AROUN2                  ; SKIP OVER THE ENTRY POINT
```

```
                                                        .SBTTL HARDWARE ENTRY POINT
                
                        172000                          .=172000
172000                                  START::
172000  000167  176254                                  JMP     PWR$UP
172004                                  RESTAR::
172004  000167  176026                                  JMP     R.STRT
```

```
                                        .SBTTL DIAGNOSTICS-Continued
172010                          AROUN2:
172010  110137  176202          1$:     MOVB    R1, @#PP.B              ; send it out port B
172014  123701  176200                  CMPB    @#PP.A, R1              ; check input in port A
172020  001402                          BEQ     2$                      ; branch if same
172022  052700  000001                  BIS     #E.PAR, R0              ; else set error flag
172026  077110                  2$:     SOB     R1, 1$                  ; loop for all values

                                ; Perform SLU 2 diagnostic

172030  012702  171742                  MOV     #ERRBI1, R2             ; R2->error flags
172034  012701  176540                  MOV     #RCSR$2, R1             ; R1 -> SLU2
172040  016146  000002                  MOV     2(R1), -(SP)            ; ignore garbage, make temp
172044  012704  171750                  MOV     #INITS, R4              ; R4->initial XCSR value
172050  014461  000004          3$:     MOV     -(R4), 4(R1)            ; init XCSR
172054  001436                          BEQ     11$                     ; branch if done
172056  005742                          TST     -(R2)                   ; R2->next error flag
172060  012716  000010                  MOV     #8., (SP)               ; (SP)=baud rate counter
172064  012703  171750          4$:     MOV     #PATERN, R3             ; R3->patterns
172070  005005                  5$:     CLR     R5                      ; init timeout counter
172072  105761  000004          6$:     TSTB    4(R1)                   ; loop pattern around
172076  100402                          BMI     7$                      ; branch if ready
172100  077504                          SUB     R5, 6$                  ; else bump timeout counter
172102  000422                          BR      10$                     ; branch if timeout

172104  111361  000006          7$:     MOVB    (R3), 6(R1)
172110  005005                          CLR     R5                      ; initialize timeout counter
172112  105711                  8$:     TSTB    (R1)
172114  100402                          BMI     9$                      ; branch if ready
172116  077503                          SOB     R5, 8$                  ; else bump timeout counter
172120  000413                          BR      10$                     ; branch if timeout

172122  126113  000002          9$:     CMPB    2(R1), (R3)             ; come back OK?
172126  001010                          BNE     10$                     ; no, set error bit & exit
172130  105723                          TSTB    (R3)+                   ; done all bit patterns?
172132  001356                          BNE     5$                      ; no
172134  005316                          DEC     (SP)                    ; yes, done all bauds?
172136  001744                          BEQ     3$                      ; yes
172140  062761  000010  000004          ADD     #10, 4(R1)              ; no, to next baud rate
172146  000746                          BR      4$

172150  051200                  10$:    BIS     (R2),R0                 ; set error bit
172152  005726                  11$:    TST     (SP)+                   ; rid of temp
172154  004767  177472                  CALL    OCTST0                  ; print error flags
172160  000167  176516                  JMP     KBD$                    ; and just get out.
                                        .DSABL  LSB
```

```
                                         .SBTTL BOOTS-Description

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                     ;;;;
                        ;;;;                  BOOTSTRAP MODULE                   ;;;;
                        ;;;;                                                     ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        000000                  .REPT   0

                        This is a short bootstrap program designed to handle floppy disks or TU58
                        tape cassettes in either our standard bootable format or in the stand-alone
                        volume format (RT-11 ".SAV"-structured files).

                        The bootstrap sequence is as follows:

                        1.  Since entry is effected by typing D in response to ODT prompt, get next
                            character (D, X or Y).  Get optional device number next (default is 0).

                        2.  If floppy boot is selected:

                            a.  Attempt to read 512 bytes from specified unit of the floppy
                                disk, starting from logical block zero, into memory locations
                                starting at 0 at the density of the medium present in the
                                drive at the time.

                            b.  If the drive is not ready or does not contain a bootable
                                medium, go back to ODT.

                        3.  If TU58 boot is selected, read the first block from the selected
                            drive into locations starting at 0.

                        4.  If the first byte read into RAM is 240 octal, jump to it.  If the
                            first byte is 260 octal, execute the stand-alone volume loader,
                            using the selected device as input.

                                .ENDR
```

```
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                            ;;;;
                        ;;;;            EQUATES USED ONLY BY BOOTSTRAPS                 ;;;;
                        ;;;;                                                            ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                        .SBTTL BOOTS-RX Controller Definitions

                        ; RX01/RX02 (RXV11,RXV21) Register Definitions

        177170          RXCS=   177170                          ;Control and Status
        177172          RXDB=   RXCS+2                          ;Data Buffer

                        ; RX Control and Status Bits

        100000          RX$$ER= 100000                          ;Error
        040000          RX$$IN= 040000                          ;Initialize controller
        030000          RX$$XA= 030000                          ;Extended address bits
        004000          RX$$02= 004000                          ;1 if RX02; 0 if RX01
        003000          RX$$XX= 003000                          ;Unused bits
        000400          RX$$DE= 000400                          ;Density (1=double,0=single)
        000200          RX$$TR= 000200                          ;Transfer function
        000100          RX$$IE= 000100                          ;Interrupt enable
        000040          RX$$DN= 000040                          ;Done
        000020          RX$$UN= 000020                          ;Unit select
        000016          RX$$FN= 000016                          ;Function select
        000001          RX$$GO= 000001                          ;GO

                        ; RX Function Codes (in RX$$FN) with GO bit preset

        000001          RX$FIL= 0*2+RX$$GO                      ;Fill buffer
        000003          RX$EMP= 1*2+RX$$GO                      ;Empty buffer
        000005          RX$WRT= 2*2+RX$$GO                      ;Write sector
        000007          RX$RED= 3*2+RX$$GO                      ;Read sector
        000011          RX$STD= 4*2+RX$$GO                      ;Set media density
        000013          RX$RST= 5*2+RX$$GO                      ;Read status
        000015          RX$WDD= 6*2+RX$$GO                      ;Write sector with deleted data
        000017          RX$REC= 7*2+RX$$GO                      ;Read error code

                        ; RX Error Codes

        000400          RXE$UN= 000400                          ;Unit selected
        000200          RXE$DR= 000200                          ;Drive ready
        000100          RXE$DD= 000100                          ;Deleted data
        000040          RXE$DN= 000040                          ;Drive density
        000020          RXE$DE= 000020                          ;Density error
        000004          RXE$ID= 000004                          ;Initialize done
        000001          RXE$CR= 000001                          ;CRC error

                        ; Miscellaneous Definitions

        000010          RETRY=  8.                              ;Number of retries

                                .SBTTL BOOTS-TU58 Definitions and Protocol Equates

```

```
                        ; Absolute address definitions
                        
000002                  FILNAM  = 000002                ;Address of RAD50 filename for
                                                        ; stand-alone program loading
001000                  DIRBUF  = 001000                ;Start of 512. word buffer used
                                                        ; for RT-11 directory operations
                                                        ;  in stand-alone loading
                        
                        ; TU58 Address definitions
                        
176540                  TI$CSR  = RCSR$2                ;DL receiver control and status
176542                  TI$BFR  = RBUF$2                ;DL receiver data buffer
176544                  TO$CSR  = XCSR$2                ;DL transmitter control and status
176546                  TO$BFR  = XBUF$2                ;DL transmitter data buffer
                        
                        
                        ; TU58 Radial Serial Protocol codes
                        
                        ; Flag Byte Definitions:
                        
000001                  R$$DAT  = ^B<00001>              ;Data message flag
000002                  R$$CTL  = ^B<00010>              ;Control message flag
000004                  R$$INI  = ^B<00100>              ;Initialize flag
000020                  R$$CON  = ^B<10000>              ;Continue flag
000023                  R$$XOF  = ^B<10011>              ;XOFF
                        
                        ; Control packet operation codes:
                        
000000                  R$NOP   = 0.                    ;No-operation
000001                  R$INIT  = 1.                    ;Initialize
000002                  R$READ  = 2.                    ;Read operation
000003                  R$WRIT  = 3.                    ;Write operation
000004                  R$COMP  = 4.                    ;Compare (NOP on TU58)
000005                  R$POSI  = 5.                    ;Position operation
000006                  R$ABRT  = 6.                    ;Abort (NOP on TU58)
000007                  R$DIAG  = 7.                    ;Diagnose
000010                  R$GETS  = 8.                    ;Get status
000011                  R$SETS  = 9.                    ;Set status (NOP on TU58)
000012                  R$GETC  = 10.                   ;Get characteristics
000013                  R$SETC  = 11.                   ;Set characteristics (NOP on TU58)
000100                  R$END   = ^B<01000000>          ;*END message
                        
                        ; END packet success codes:
                        
000000                  S$NORM  = 0.                    ;Normal success
000001                  S$RETR  = 1.                    ;Success but with retries
177776                  S$PART  = -2.                   ;Partial operation (end of medium)
177770                  S$UNIT  = -8.                   ;Invalid unit number
177767                  S$CART  = -9.                   ;No cartridge
177765                  S$WPRT  = -11.                  ;Cartridge write protected
177757                  S$DCHK  = -17.                  ;Data check error
177740                  S$SEEK  = -32.                  ;Seek error (block not found)
177737                  S$MOTR  = -33.                  ;Motor stopped
177720                  S$UFCD  = -48.                  ;Invalid operation code
177711                  S$RECN  = -55.                  ;Invalid record number
                        
                                .SBTTL BOOTS-RT11 Definitions and Equates
```

```
                        ; RT-11 Directory Structure Definitions
                        
001000          SEGALO  = DIRBUF                ;Number of segments allocated
001002          NXTSEG  = DIRBUF+2              ;Number of next logical segment
001004          HGHSEG  = DIRBUF+4              ;Highest segment in use
001006          XTRBYT  = DIRBUF+6              ;Number of extra bytes per entry
001010          STRBLK  = DIRBUF+10             ;Starting block# for files
                                                ; in this segment
000016          ENTSIZ  = 7*2                   ;Size of a directory entry
000010          D.FLEN  = 10                    ;Offset to file length in entry
000400          TENTAS  = 000400                ;Flag for tentative file entry
001000          EMPTYS  = 001000                ;Flag for empty area entry
002000          PERMFS  = 002000                ;Flag for permanent file
004000          ENDSGS  = 004000                ;Flag for end of segment
                        
                        ; RT-11 System Communications Area Definitions
                        
000040          RT$STA  = 000040                ;Start address for program
000042          RT$ISP  = 000042                ;Initial stack pointer
000044          RT$JSW  = 000044                ;Job status word
000046          RT$USR  = 000046                ;USR load address
000050          RT$HGH  = 000050                ;Job high memory limit
000052          RT$EMT  = 000052                ;(Byte) EMT error code
000053          RT$UER  = 000053                ;(Byte) User error code
000054          RT$RMN  = 000054                ;Base address of resident monitor
000056          RT$FCH  = 000056                ;(Byte) Console fill character
000057          RT$FCT  = 000057                ;(Byte) Console fill count
```

```
                                          .SBTTL BOOTS-Program entry point

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                   ;;;;
                                  ;;;;  BOOTSTRAP INITIALIZATION AND COMMAND INTERPRETER  ;;;;
                                  ;;;;                                                   ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; A 'D' was entered in response to the ODT prompt, so we get
                                  ; here and expect 'D','X' or 'Y' next, followed by a CR or a
                                  ; unit number.  We set bits up in B.CNTL as follows:
                                  ;
                                  ;    BIT  7: 0 = TU58
                                  ;            1 = RX01/02
                                  ;            Used by stand-alone volume loader to select proper
                                  ;            read routine.
                                  ;    BIT  0: Device number
                                  ;
                                  ; Note: if no memory was found at 000000, bit 15 of B.CNTL,
                                  ;   called "NO.LOW" will be set and the bootstraps will be
                                  ;   disabled.

172164                            BOOTS::
172164  012737  000072  176544            MOV     #TUBAUD,@#TO$CSR        ;Set TU58 Baud Rate

                                  ; Jump here with ODT if booting TU58's at other than default baud rate

172172                            STTUBD::
172172  010667  175566                    MOV     SP,IN.USR               ;Permit HALTs and BREAKS
                                                                          ; by making IN.USR non-zero
172176  005004                            CLR     R4                      ;Assemble new B.CNTL here
172200  004767  177304                    CALL    GETCHR                  ;Keyboard character in R2
172204  120227  000104                    CMPB    R2,#'D                  ;DD = TU58 cassette
172210  001412                            BEQ     1$                      ;R4 is clear for DD
172212  012704  000200                    MOV     #DEVBIT,R4              ;R4 bit 7 is set for DX, DY
172216  020227  000130                    CMP     R2,#'X                  ;DX = RX01 or RX02
172222  001405                            BEQ     1$                      ;DY = RX01 or RX02, the code's
172224  020227  000131                    CMP     R2,#'Y                  ; the same- it knows both den-
172230  001402                            BEQ     1$                      ;  sities, DMA, non-DMA
172232                                    ABORT   <Illegal device name>

172236  004767  177246            1$:     CALL    GETCHR                  ;Get device number or CR
172242  022702  000015                    CMP     #15,R2                  ;Is it CR?
172246  001410                            BEQ     3$                      ;CR means drive 0
172250  162702  000060                    SUB     #'0,R2                  ;Drive 0?
172254  001405                            BEQ     3$                      ;Yup.
172256  005302                            DEC     R2                      ;Drive 1?
172260  001402                            BEQ     2$                      ;Yes, skip the ABORT
172262                                    ABORT   <Illegal unit number>

172266  005204                    2$:     INC     R4                      ;For unit 1.
172270  110467  175474            3$:     MOVB    R4,B.CNTL               ;Set device, unit information.
172274  005767  175470                    TST     B.CNTL                  ;Test NO.LOW
172300  100002                            BPL     4$                      ;We have low memory
                                                                          ;We don't, so go to ODT
```

-----> HALT AT PC=172264 INDICATES "ILLEGAL UNIT NUMBER"

```
                                        ABORT   <No low memory, can't boot>

                                ; Before proceeding, we set up the bus timeout trap vector, enable
                                ; trap to 4 emulation and reset the bus.  We do a delay (see
                                ; explanation below) and set up the stack so the stand-alone booter and
                                ; device primary bootstraps can get the information they need passed
                                ; to them in R0 and R1 (see CHK240, below).

172306  012737  172370  000004  4$:     MOV     #BADBOT,@#4             ;If we time out, we want to re-
172314  012737  000300  000006          MOV     #PRIO,@#6               ;initialize everything.
172322  000005                          RESET                           ;For now, init. the bus.

                                ;       Note: the previous instruction also screws up some devices
                                ;       which perform a long initialization sequence, such as RX02's,
                                ;       which do an automatic boot from drive 0.  The long delay below
                                ;       is necessary in order to assure drive 1 is ready if a boot
                                ;       is desired from it.

172324                                  DELAY   R0,R1,9.                ;Delay 2 seconds
172336  012706  167644                  MOV     #$$TACK,SP              ;Initialize the stack.
172342  010667  175430                  MOV     SP,TRAP4                ;Set up trap-to-4 emulation
                                                                        ;by making TRAP4 non-zero
172346  012716  037776                  MOV     #37776,(SP)             ;Some boots need a memory-top
                                                                        ; address here, so 8K will do
172352  010402                          MOV     R4,R2                   ;Boot control word here
172354  042702  177776                  BIC     #^C<DEVNUM>,R2          ;Want only unit no. in R2
172360  010246                          MOV     R2,-(SP)                ;And we'll save it too.

172362  105704                          TSTB    R4                      ;Bit 7 set for RX01/02
172364  100405                          BMI     RXBOOT                  ;Go to floppy boot
172366  000436                          BR      TUBOOT                  ;Go to TU58 boot

172370                          BADBOT:
172370  012706  167644                  MOV     #$$TACK,SP              ;Restore the stack
172374                                  ABORT   <Unexpected timeout during boot>
```

```
                                        .SBTTL BOOTS-RX01/RX02 Bootstrap

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;                FLOPPY BOOTSTRAP                    ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; This routine will bootstrap either floppy drive, at the density of the
                                ; media mounted in that drive.

172400                          RXBOOT:
172400  012746  177170                  MOV     #RXCS,-(SP)             ;Need floppy CSR for CHK240
172404  005737  177170                  TST     @#RXCS                  ;If not there, time out via 4
172410  000240                          NOP                             ;to ST173 and reset the world
172412  012701  001000                  MOV     #512.,R1                ;Byte count
172416  005000                          CLR     R0                      ;Starting block number
172420  005004                          CLR     R4                      ;RAM buffer address = 000000
172422  004767  000402                  CALL    DREAD                   ;LOAD IT ALL IN
```

```
                                                 .SBTTL BOOTS-Distinguishing type of boot block

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                              ;;;;
                                        ;;;;      DISTINGUISH STANDARD FROM STAND-ALONE FROM              ;;;;
                                        ;;;;                NON-BOOTABLE VOLUMES.                         ;;;;
                                        ;;;;                                                              ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                        ; The CHK240 routine will repeat powerup sequence if location 0 does not
                                        ; contain a valid secondary bootstrap (i.e., does not have a 240 or 260
                                        ; in it).  It starts execution of the booted program if there's a 240,
                                        ; and goes to the stand-alone program loader if there's a 260.

172426                                  CHK240:
172426  022737  000240  000000                  CMP     #240,@#0                ;Did we read a valid bootstrap?
172434  001410                                  BEQ     1$
172436  022737  000260  000000                  CMP     #260,@#0
172444  001447                                  BEQ     STANDB                  ;Stand-alone volumes start with 260
172446  004767  175766                          CALL    VECSET                  ;Restore wiped-out vectors
172452                                          ABORT   <No boot block on volume>

172456  012601                          1$:     MOV     (SP)+,R1                ;Unit CSR address
172460  012600                                  MOV     (SP)+,R0                ;Unit number
172462  005007                                  CLR     PC                      ;Standard secondary boots
```

```
                                            .SBTTL BOOTS-TU58 Bootstrap

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                            ;;;;
                                ;;;;            TU58 TAPE CASSETTE BOOTSTRAP                    ;;;;
                                ;;;;                                                            ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                        .ENABL  LSB
172464                          TUBOOT:
172464  012746  176540                  MOV     #TI$CSR,-(SP)           ;CHK240 wants TU58 CSR
172470  012701  176544                  MOV     #TO$CSR,R1              ;R1 => output CSR for TU58 serial line
172474  005003                          CLR     R3                      ;Set R3 = 0 (Two NULLs)
172476  005211                          INC     @R1                     ;Start transmitting BREAK to TU58
172500  004767  001132                  CALL    CH8OUT                  ;Send eight NULLS
172504  105711                  1$:     TSTB    @R1                     ;Is transmitter ready again yet?
172506  100376                          BPL     1$                      ;If PL no - wait
172510  042711  000001                  BIC     #XC.BRK,@R1             ;Else stop sending BREAK now
172514  012703                          MOV     (PC)+,R3                ;Get two INIT commands for TU58
172516     004     004                  .BYTE   R$$INT,R$$INT
172520  004715                          CALL    @R5                     ;And transmit them
172522  005741                          TST     -(R1)                   ;Dump any garbage char in TI$BUF
172524  105737  176540          2$:     TSTB    @#TI$CSR                ;Is character available from the TU58?
172530  100375                          BPL     2$                      ;If PL, no - wait in loop
172532  121127  000020                  CMPB    @R1,#R$$CON             ;If so, was it a CONTINUE flag?
172536  001402                          BEQ     3$                      ;If EQ, yes- go ahead
172540                                  ABORT   <TU58 initialization error>

                                ; TU58 is now initialized.  Prepare to read block #0.

172544  005000                  3$:     CLR     R0                      ;Block number = 0
172546  012701  001000                  MOV     #512.,R1                ;Byte count = one block
172552  004767  000212                  CALL    READZU                  ;Attempt to read the block
172556  100323                          BPL     CHK240                  ;If PL, read was successful
172560                                  ABORT   <TU58 block 0 read error>
                                        .DSABL  LSB
```

```
                                        .SBTTL BOOTS-Stand-alone volume bootstrap

                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;                                                    ;;;;
                               ;;;;            STAND-ALONE-VOLUME BOOTSTRAP            ;;;;
                               ;;;;                                                    ;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                               ; This routine loads stand-alone programs (assumed to be in RT-11 .SAV
                               ; file format) from an RT-11 file structured TU58 cartridge.  It is
                               ; invoked if the first word in block 0 of the cartridge is a 260.

172564                         STANDB:
172564  012700  000001                 MOV     #1,R0                    ;Set directory segment #1
172570  006300                 1$:     ASL     R0                       ;Two blocks per segment
172572  022020                         CMP     (R0)+,(R0)+              ;Add 4 to R0, as directory starts
                                                                        ; in block#6
172574  012701  002000                 MOV     #1024.,R1                ;Prepare to read two blocks
172600  012704  001000                 MOV     #DIRBUF,R4               ;Into the directory buffer
172604  004767  000162                 CALL    READU                    ;Read the segment
172610  100002                         BPL     2$                       ;If PL, read was successful
172612                                 ABORT   <Directory read error>
172616  012704  001010         2$:     MOV     #STRBLK,R4               ;Else prepare to pick up
                                                                        ; starting block
172622  012400                         MOV     (R4)+,R0                 ;R0 = starting block for files
172624  010403                 3$:     MOV     R4,R3                    ;Save pointer to current entry
172626  032724  002000                 BIT     #PERMFS,(R4)+            ;Is this a permanent file?
172632  001010                         BNE     4$                       ;If bit set, yes - check if it matches
172634  022744  004000                 CMP     #ENDSGS,-(R4)            ;Else is this the end-of-segment
                                                                        ; marker?
172640  001015                         BNE     5$                       ;If NE, no - go skip this entry
172642  013700  001002                 MOV     @#NXTSEG,R0              ;Else get number of next segment
172646  001350                         BNE     1$                       ;If NE, there is one - go read it
172650                                 ABORT   <File not found>

172654  012705  000002         4$:     MOV     #FILNAM,R5               ;Point to RAD50 name of desired file
172660  022425                         CMP     (R4)+,(R5)+              ;Check file name, first word
172662  001004                         BNE     5$                       ;If NE not desired file
172664  022425                         CMP     (R4)+,(R5)+              ;...Check second word of filename
172666  001002                         BNE     5$                       ;If NE not desired one
172670  022425                         CMP     (R4)+,(R5)+              ;...Finally, check extension
172672  001410                         BEQ     LOAD                     ;If EQ, got it - go load this
                                                                        ; one into memory
172674  010304                 5$:     MOV     R3,R4                    ;Get entry pointer back
172676  062704  000010                 ADD     #D.FLEN,R4               ;Advance to file size of entry
172702  062400                         ADD     (R4)+,R0                 ;Update current file base
172704  022424                         CMP     (R4)+,(R4)+              ;And skip to next file entry
172706  063704  001006                 ADD     @#XTRBYT,R4              ;Plus any extra bytes in each entry
172712  000744                         BR      3$                       ;Continue file search
```

```
                                                    .SBTTL BOOTS-Load Stand-Alone Program File
                
172714  011401                  LOAD:   MOV     @R4,R1                  ;R1 = size of file in blocks
172716  000301                          SWAB    R1                      ; * 256. = word count
172720  006301                          ASL     R1                      ; * 2 = byte count
172722  004767  000042                  CALL    READZU                  ;Read the program file into memory
172726  100002                          BPL     1$                      ;If MI, error in read-ABORT
172730                                  ABORT   <Stand-alone file read error>
172734  013705  000040          1$:     MOV     @#RT$STA,R5             ;Get program start adrs
172740  032705  000001                  BIT     #1,R5                   ;Is adrs even?
172744  001402                          BEQ     START$                  ;If EQ yes - okay
172746                                  ABORT   <Illegal transfer address>
                
172752                          START$:
172752  012601                          MOV     (SP)+,R1                ;Pass the CSR address
172754  112600                          MOVB    (SP)+,R0                ;Get unit number booted
172756  013706  000042                  MOV     @#RT$ISP,SP             ;Load program's stack pointer
172762  005067  175010                  CLR     TRAP4                   ;Disable trap to 4 feature
172766  000115                          JMP     @R5                     ;Go start program execution
                
172770                          READZU:
172770  005004                          CLR     R4                      ;Load at 0
172772  016602  000004          READU:  MOV     4(SP),R2                ;Get unit number
172776  000407                          BR      AROUN3                  ; SKIP OVER THE ENTRY POINT
```

```
                                                .SBTTL 173000G ENTRY POINT
                                ;
                        173000                  .=173000
173000                  S1173::
173000  106427  000340          MTPS    #PR17           ;Can't assume anything here.
173004  000005                  RESET                   ;But PWRSUP usually does.
173006  005000                  CLR     R0              ;DELAY for the sake of DLART.
173010  077001                  SOB     R0,.            ;(Maint. bit cleared by RESET
                                                        ;just a little too long).
173012  000167  175242          JMP     PWRSUP
```

```
                                                .SBTTL BOOTS-Continued
173016                                  ARDUN3:
173016  105767  174746                        TSTB    B.CNTL
173022  100402                                BMI     DREAD                   ;Bit 7 set for RX01/RX02
173024  000167  000370                        JMP     TREAD                   ;Read from tape
```

```
                                        .SBTTL BOOTS-RX01/RX02 Read routines

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;            FLOPPY DISK READ ROUTINES               ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; With registers set up as below, read the appropriate number of
                                ; full sectors from the floppy, at either density, with either
                                ; RXV21 DMA or RXV11 Programmed I/O interface.
                                ;
                                ; R0: Starting block number for transfer.
                                ; R1: Byte count for transfer
                                ; R2: Unit number
                                ; R4: Address of buffer to receive data

                                .ENABL LSB
173030  010446                  DREAD:  MOV     R4,-(SP)                ;Save buffer address
173032  010046                          MOV     R0,-(SP)                ;Save starting LBN
173034  010146                          MOV     R1,-(SP)                ;Save byte count

                                ; Check status and media density of selected drive

173036  012701  177172                  MOV     #RXDB,R1                ;Set up R1 for benefit of RXGO
173042  005000                          CLR     R0                      ;Initialize current unit/density word
173044  006002                          ROR     R2                      ;Bit 0 set = unit 1
173046  103002                          BCC     1$
173050  052700  000020                  BIS     #RX$$UN,R0              ;Set unit 1
173054  004567  000312          1$:     JSR     R5,RXGO                 ;Start a read status operation
173060  000013                          .WORD   RX$RST                  ; to determine status and density
173062  111102                          MOVB    @R1,R2                  ;Pick up low byte of status
173064  100402                          BMI     2$                      ;If PL, drive not ready
173066                                  ABORT   <Floppy drive not ready>
173072  032702  000040          2$:     BIT     #RXE$DN,R2              ;Check media density
173076  001411                          BEQ     3$                      ;If EQ, single density

                                ; Double density.
                                ; Logical sector number = logical block number * 2
                                ; Sector count = byte count/256.

173100  052700  000400                  BIS     #RX$$DE,R0              ;Set double density in command
173104  012602                          MOV     (SP)+,R2                ;Byte count
173106  000302                          SWAB    R2                      ;Divide by 256
173110  012603                          MOV     (SP)+,R3                ;LBN
173112  006303                          ASL     R3                      ;Multiply by 2
173114  012704  000200                  MOV     #128.,R4                ;Words per sector
173120  000410                          BR      4$

                                ; Single density.
                                ; Logical sector number = logical block number * 4
                                ; Sector count = byte count/128.

173122  012602                  3$:     MOV     (SP)+,R2                ;Byte count
173124  000302                          SWAB    R2                      ;Divide by 256
```

-----> HALT AT PC=173070 INDICATES "FLOPPY DRIVE NOT READY"

```
 58 173126  006302                        ASL     R2                      ;And multiply by 2
 59 173130  012603                        MOV     (SP)+,R3                ;LBN
 60 173132  006303                        ASL     R3
 61 173134  006303                        ASL     R3                      ;Multiply by 4
 62 173136  012704  000100                MOV     #64.,R4                 ;Words per sector
 63
 64                                ; Set up stack as follows:
 65                                ;   0(SP) = Logical Sector Number
 66                                ;   2(SP) = Sector count
 67                                ;   4(SP) = words per sector
 68                                ;   6(SP) = Buffer address
 69
 70 173142  010446                4$:     MOV     R4,-(SP)                ;Words per sector
 71 173144  010246                        MOV     R2,-(SP)                ;Sector count
 72 173146  010346                        MOV     R3,-(SP)                ;Logical Sector Number
 73
 74                                ; Start the read operation.
 75                                ; This is the top of the loop.
 76
 77 173150  004567  000216        5$:     JSR     R5,RXGO                 ;Start a sector read
 78 173154  000007                        .WORD   RX$RED
 79
 80                                ; Convert Logical Sector Numbers to Physical tracks and sectors.
 81
 82 173156  011603                        MOV     @SP,R3                  ;Get Logical Sector Number
 83 173160  012702  000010                MOV     #8.,R2                  ;Loop count
 84 173164  022703  006400        6$:     CMP     #26.*200,R3             ;Does 26 go into dividend?
 85 173170  101002                        BHI     7$                      ;Branch if not, C clear (BHI => BCC)
 86 173172  062703  171400                ADD     #-26.*200,R3            ;Subtract 26 from dividend (C set)
 87 173176  006103                7$:     ROL     R3                      ;Shift dividend and quotient
 88 173200  005302                        DEC     R2                      ;Decrement loop count
 89 173202  003370                        BGT     6$                      ;Branch till divide done
 90 173204  110302                        MOVB    R3,R2                   ;Copy track number
 91 173206  105003                        CLRB    R3                      ;Remove track number from remainder
 92 173210  000303                        SWAB    R3                      ;Get remainder
 93 173212  022703  000014                CMP     #12.,R3                 ;C=1 if 13<=R3<=25, else C=0
 94 173216  006103                        ROL     R3                      ;Sector*2 (2:1 interleave)
 95                                                                        ;[+1 (C) if sector 13-25]
 96 173220  006302                        ASL     R2                      ;Double the track number
 97 173222  060203                        ADD     R2,R3                   ;Skew the sector
 98 173224  060203                        ADD     R2,R3                   ; by adding in
 99 173226  060203                        ADD     R2,R3                   ;  6 * track number
100 173230  006202                        ASR     R2                      ;Undouble the track number
101 173232  005202                        INC     R2                      ; and make it 1-76 (Skip track 0
102                                                                       ;  for ANSI)
103 173234  162703  000032        8$:     SUB     #26.,R3                 ;Put sector
104 173240  002375                        BGE     8$                      ; into range
105 173242  062703  000033                ADD     #27.,R3                 ;  1-26
106
107                               ; Read the sector
108
109 173246  010311                        MOV     R3,@R1                  ;Set sector number
110 173250  004514                        JSR     R5,@R4                  ;
111 173252  010211                        MOV     R2,@R1                  ;Set track number
112 173254  004514                        JSR     R5,@R4                  ;Perform a sector read
113 173256  100002                        BPL     9$                      ;If MI, error
114 173260                                ABORT   <floppy read error>
```

```
-----> HALT AT PC=173262 INDICATES "FLOPPY READ ERROR"

                                        ; Empty RXV11/RXV21 buffer into RAM
                                        ;
173264  004567  000102          9$:     JSR     R5,RXGO                 ;Start empty buffer function
173270  000003                          .WORD   RXSEMP                  ; and wait for TR
173272  032737  004000  177170          BIT     #RXSS02,@#RXCS          ;Is DMA available?
173300  001407                          BEQ     10$                     ;If EQ no - handle as RX01
                                        ;
                                        ; RX02 DMA Operation
                                        ;
173302  016611  000004                  MOV     4(SP),@R1               ;Else load word count
173306  004514                          JSR     R5,@R4                  ;Wait for TR
173310  016611  000006                  MOV     6(SP),@R1               ;And load current bus address
173314  004514                          JSR     R5,@R4                  ;wait for DONE
173316  000410                          BR      12$
                                        ;
                                        ; RX01 Programmed I/O Operation
                                        ;
173320  016603  000004          10$:    MOV     4(SP),R3                ;Get word count
173324  006303                          ASL     R3                      ;Turn word count into byte count
173326  016602  000006                  MOV     6(SP),R2                ;Get starting bus address
173332  111122          11$:    MOVB    @R1,(R2)+               ;Move one byte from buffer to memory
173334  004514                          JSR     R5,@R4                  ;Wait for TR or DONE
173336  077303                          SOB     R3,11$                  ;Loop for all bytes in first sector
                                        ;
                                        ; Loop back if not yet finished
                                        ;
173340  016603  000004          12$:    MOV     4(SP),R3                ;Get word count
173344  006303                          ASL     R3                      ;Turn into byte count
173346  060366  000006                  ADD     R3,6(SP)                ;update bus address
173352  005216                          INC     @SP                     ;Update Logical Sector Number
173354  005366  000002                  DEC     2(SP)                   ;Decrement Sector Count
173360  001273                          BNE     5$                      ;Read another sector
173362  062706  000010                  ADD     #8.,SP                  ;Pop the stack
173366  000257                          CCC                             ;Clear condition codes
                                        ;                               ; to show success.
173370  000207                          RETURN                          ;All done
                                .DSABL LSB
```

-----> HALT AT PC=173262 INDICATES "FLOPPY READ ERROR"

```
                                ; The main subroutine for sending disk commands and waiting for
                                ; their completion.
                                ;
                                ; Register usage:
                                ;       R0 = density bit ! unit select bit (proto for commands)
                                ;       R1 = RXDB address
                                ;       R4 = RXGO IR/DONE test routine pointer
                                ;

173372  012504          RXGO:   MOV     (R5)+,R4                ;Copy command word to use
173374  050004                  BIS     R0,R4                   ;Set unit # and density
173376  010437  177170          MOV     R4,@#RXCS               ;Start operation
173402  010704                  MOV     PC,R4                   ;Copy adrs for later calls
173404  005741                  TST     -(R1)                   ;R1 -> RXCS
173406  032711  000240  1$:     BIT     #RXS$TR!RXS$DN,@R1      ;wait for TR or DONE
173412  001775                  BEQ     1$                      ;If EQ, neither are true yet
173414  005721                  TST     (R1)+                   ;Reset R1 -> RXDB and check for errors
173416  000205                  RTS     R5                      ;Return to caller
```

```
                                      .SBTTL BOOTS-TU58 Read routines

                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;                                                    ;;;;
                              ;;;;          TU58 DECtape II READ ROUTINES             ;;;;
                              ;;;;                                                    ;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                              ; Starts a read operation on the TU58 by transmitting a command packet
                              ;
                              ; Inputs:
                              ;       R0 = starting block # for transfer
                              ;       R1 = byte count for transfer
                              ;       R2 = unit number
                              ;       R4 = address of buffer to receive data
                              ; Outputs:
                              ;       R0, R1, R2 unchanged
                              ; Destroys:
                              ;       R3, R4, R5

                              .ENABL LSB
173420  010446                TREAD:  MOV     R4,-(SP)                ;Save buffer address
173422  005004                        CLR     R4                      ;Init checksum
173424  012703  005002                MOV     #10.*400+RS$CTL,R3      ;Set command flag and length
173430  004767  000206                CALL    CH2OUT                  ;Output two chars and set R5
173434  012703  000002                MOV     #RSREAD,R3              ;Send read command and modifier=0
173440  004715                        CALL    @R5
173442  010203                        MOV     R2,R3                   ;Then unit number and switches=0
173444  004715                        CALL    @R5
173446  005003                        CLR     R3                      ;Plus a zero sequence number
173450  004715                        CALL    @R5
173452  010103                        MOV     R1,R3                   ;Followed by the byte count
173454  004715                        CALL    @R5
173456  010003                        MOV     R0,R3                   ;And the block number
173460  004715                        CALL    @R5
173462  010403                        MOV     R4,R3                   ;Finally, transmit the checksum
173464  004715                        CALL    @R5
```

```
                                ; Now ready to accept data messages from the TU58

173466  012600                  MOV     (SP)+,R0        ;R0 -> data buffer
                        ;       CLC                     ;(CH2OUT leaves C clear)
173470  006001                  ROR     R1              ;R1 = word count for transfer
173472  004767  000116    1$:   CALL    7$              ;Get first word of packet
173476  122703  000001          CMPB    #R$SDAT,R3      ;Is this indeed a data message?
173502  001017                  BNE     3$              ;If NE no - may be END message
173504  105003                  CLRB    R3              ;Else clear flags
173506  000303                  SWAB    R3              ;Move packet byte count to low byte
173510  106003                  RORB    R3              ;And convert to word count
173512  160301                  SUB     R3,R1           ;Remove from transfer count
173514  010305                  MOV     R3,R5           ;And copy for loop counter
173516  004767  000102    2$:   CALL    9$              ;Get next two words
173522  010320                  MOV     R3,(R0)+        ;Store in buffer
173524  077504                  SOB     R5,2$           ;Loop for entire data message
173526  004767  000044          CALL    5$              ;Get checksum and compare
173532  005701                  TST     R1              ;Have all data records been
                                                        ; transferred?
173534  001356                  BNE     1$              ;If NE no
173536  004767  000052          CALL    7$              ;And get prospective
                                                        ; END packet start
173542  004767  000056    3$:   CALL    9$              ;Get opcode/success bytes
                                                        ; of END packet
173546  122703  000100          CMPB    #R$END,R3       ;Is this an END packet?
173552  001402                  BEQ     4$              ;If NE no - ABORT
173554                          ABORT   <TU58 END packet missing>
173560  010300            4$:   MOV     R3,R0           ;Save success code in R0
173562  004767  000032          CALL    8$              ;Read remainder of END packet
173566  004767  000004          CALL    5$              ;And check its checksum
173572  000300                  SWAB    R0              ;Set CC's on success code of transfer
173574  000207                  RETURN                  ;Return to caller

173576  004767  000064    5$:   CALL    CH2IN           ;Get two checksum bytes
173602  020403                  CMP     R4,R3           ;Does it match calculated value?
173604  001402                  BEQ     6$              ;If NE no - ERROR
173606                          ABORT   <TU58 checksum error>

173612  000207            6$:   RETURN                  ;Else return with success

173614  005004            7$:   CLR     R4              ;Init checksum
173616  000402                  BR      9$              ;And get the first word

173620  004717            8$:   CALL    @PC             ;Read 4 words
173622  004717                  CALL    @PC
173624  004767  000036    9$:   CALL    CH2IN           ;Read next two bytes
173630  060304                  ADD     R3,R4           ;Add into checksum
173632  005504                  ADC     R4              ; with end-around carry
173634  000207                  RETURN                  ;And back to caller
                        .DSABL LSB
```

-----> HALT AT PC=173610 INDICATES "TU58 CHECKSUM ERROR"

```
                                ; CH2OUT -- Write two bytes to the TU58
                                ;
                                ; Writes two bytes to interface and updates checksum.
                                ;
                                ; Inputs:
                                ;       R3 = two bytes to be output; low byte first
                                ;       R4 = current checksum word
                                ; Outputs:
                                ;       R3 unchanged
                                ;       R4 updated to new checksum
                                ;       R5 pointing to CH2OUT routine for easier future CALLs

173636                          CH8OUT:
173636  004717                        CALL    @PC                     ;Entry point to output 8 characters
173640  004717                        CALL    @PC
173642                          CH2OUT:
173642  010705                        MOV     PC,R5                   ;Set R5 to following routine adrs
173644  060304                        ADD     R3,R4                   ;Update checksum word
173646  005504                        ADC     R4                      ; with end-around carry
173650  004717                        CALL    @PC                     ;Repeat for both characters
173652  105737  176544          1$:   TSTB    @#TOSCSR                ;Is interface ready for output?
173656  100375                        BPL     1$                      ;If PL no - wait
173660  110337  176546                MOVB    R3,@#TOSBFR             ;Else transmit character to TU58
173664  000407                        BR      CHRET                   ;Merge with other routine to return

                                ; CH2IN -- Read two bytes from the TU58
                                ; CHIN  -- Read a single byte from the TU58
                                ;
                                ; Inputs:
                                ;       none.
                                ; Outputs:
                                ;       R3 = character(s) read

173666  004717                  CH2IN:  CALL    @PC                     ;Read two, not one
173670  105003                  CHIN:   CLRB    R3                      ;And zero out space for new one
173672  105737  176540          1$:     TSTB    @#TISCSR                ;Is a character available?
173676  100375                        BPL     1$                      ;If PL no
173700  153703  176542                BISB    @#TISBFR,R3             ;Else set into register
173704  000303                  CHRET:  SWAB    R3                      ;Move current character over
173706  000207                        RETURN                          ;And return to caller
```

```
                                                        .SBTTL END STATEMENT
000001                  .END
```

SYMBOL TABLE

```
AROUN2  172010          E.PAR = 000001          PP.B16= 000014          RT$USR= 000046          R.PC  = 167766 G
AROUN3  173016          FAKOUT  170424          PP.B17= 000016          RXBOOT  172400          R.STAK= 000001
AUTOBA  170472 G        FILNAM= 000002          PP.C  = 176204          RXCS  = 177170          R.TYPE= 167762 G
BADBOT  172370          GETCHR  171510          PP.CHI= 000010          RXDB  = 177172          R.STRT  170036 G
BAUDR$= 000032          GETNUM  171612          PP.CLU= 000001          RXE$CR= 000001          SAVPC = 167752 G
BD.003= 000000          HGHSEG= 001004          PP.CWR= 176206          RXE$DD= 000100          SAVPS = 167754 G
BD.006= 000010          HKBDQ   171140          PP.DRA= 000020          RXE$DE= 000020          SEGALO= 001000
BD.012= 000020          HKBD$   171142          PP.DRB= 000002          RXE$DN= 000040          SPACE = 000040
BD.024= 000030          HVBAUD  170556          PP.MDA= 000040          RXE$DR= 000200          SRET    171636
BD.048= 000040          INBYTE  170550          PP.MDB= 000004          RXE$ID= 000004          S1ANDB  172564
BD.096= 000050          INBYT$  170556          PP.MD2= 000100          RXE$UN= 000400          START   172000 G
BD.192= 000060          INITS   171750          PP.MOD= 000200          RXGO    173372          START$  172752
BD.384= 000070          IN.USR= 167764 G        PRINT   170706          RXSEMP= 000003          STRBLK= 001010
BIT0  = 000001          KBDQ    170674          PRI6  = 000300          RXSFIL= 000001          STTUBD  172172 G
BIT1  = 000002          KBD$    170702          PRI7  = 000340          RXSREC= 000017          ST173   173000 G
BIT10 = 002000          LCSET   171246          PUTCHR  171522          RXSRED= 000007          SWCMD   171212
BIT11 = 004000          LEDOFF= 000017          PUTCLF  171554          RXSRST= 000013          S$CART= 177767
BIT12 = 010000          LF    = 000012          PUTLF   171564          RXSSTD= 000011          S$DCHK= 177757
BIT13 = 020000          LOAD    172714          PUTSTR  171542          RXS$DD= 000015          S$MOTR= 177737
BIT14 = 040000          LOCDSP  171252          PWRSUP  170260 G        RXS$RT= 000005          S$NORM= 000000
BIT15 = 100000          MODE  = 000221          QODT    170666          RXS$DE= 000400          S$OPCD= 177720
BIT2  = 000004          MSGQ    171730          RAMBOT= 160010          RXS$DN= 000040          S$PART= 177776
BIT3  = 000010          MSG$    171731          RAMTOP= 167776          RXS$ER= 100000          S$RECN= 177711
BIT4  = 000020          NEXNUM  171600          RBUFS1= 177562          RXS$FN= 000016          S$RETR= 000001
BIT5  = 000040          NOCT    171642          RBUFS2= 176542          RXS$GO= 000001          S$SEEK= 177740
BIT6  = 000100          NO.LOW= 100000          RB.BRK= 004000          RXS$IE= 000100          S$UNIT= 177770
BIT7  = 000200          NXTSEG= 001002          RB.ERR= 100000          RXS$IN= 040000          S$WPRT= 177765
BIT8  = 000400          OCTSTR  171656          RB.FRM= 020000          RXS$TR= 000200          TENTA$= 000400
BIT9  = 001000          OCTSTO  171652          RB.OVR= 040000          RXS$UN= 000020          TI$BFR= 176542
BOOTS   172164 G        ODT     170602 G        RCMD    171144          RXS$XA= 030000          T1$CSR= 176540
BRKNOO  170010 G        ODTFLG= 167750 G        RCMD1   171232          RXS$XX= 003000          TO$BFR= 176546
B.CNTL= 167770 G        ODTLOC= 167746 G        RCSR$1= 177560          RXS$02= 004000          TO$CSR= 176544
CHIN    173670          ODTSTK= 167744 G        RCSR$2= 176540          RSABRT= 000006          TRAP4 = 167776 G
CHK240  172426          ODTWHY= 167774 G        RC.ACT= 004000          RSCOMP= 000004          TREAD   173420
CHRET   173704          ONENUM  171576          RC.DUN= 000200          RSDIAG= 000007          TUBAUD= 000072
CH2IN   173666          O.CNTL= 167772 G        RC.IEN= 000100          RSEND = 000100          TUBOOT  172464
CH2OUT  173642          PATERN  171750          READU   172772          RSGETC= 000012          T.BIT = 000020
CH8OUT  173636          PBR0  = 000010          READZU  172770          RSGETS= 000010          USERSP= 167760 G
CR    = 000015          PBR1  = 000020          REGOUT  171240          RS1NIT= 000001          VECSET  170440 G
DEVBIT= 000200          PBR2  = 000040          RESTAR  172004 G        RSNOP = 000000          XBUF$1= 177566
DEVNUM= 000001          PCMD    171044          RETRY = 000010          RSPOSI= 000005          XBUFS2= 176546
DIAGNO  171754          PERMF$= 002000          RFLAG = 000200          RSREAD= 000002          XCSR$1= 177564
DIRBUF= 001000          PP.A  = 176200          RPOINT= 167756 G        RSSETC= 000013          XCSR$2= 176544
DONE    171574          PP.B  = 176202          RT$EMT= 000052          RSSETS= 000011          XC.BRK= 000001
DREAD   173030          PP.BIC= 000000          RT$FCH= 000056          RSWRIT= 000003          XC.IEN= 000100
D.FLEN= 000010          PP.BIS= 000001          RT$FCT= 000057          RS$CON= 000020          XC.MNT= 000004
EMPTY$= 001000          PP.BI0= 000000          RT$HGH= 000050          RS$CTL= 000002          XC.PBE= 000002
ENDSG$= 004000          PP.BI1= 000002          RT$ISP= 000042          RS$DAT= 000001          XC.RDY= 000200
ENTSIZ= 000016          PP.BI2= 000004          RT$JSW= 000044          RS$INT= 000004          XIRBYT= 001006
ERRBIT  171742          PP.BI3= 000006          RT$RMN= 000054          RS$XOF= 000023          $$TACK= 167644 G
E.EXT = 000100          PP.BI4= 000010          RT$STA= 000040          R.HALT= 100000          $$$BRK  170000 G
E.INT = 000010          PP.B15= 000012          RT$UER= 000053          R.NXM = 000200          $$$LTC  170006 G

. ABS.  174000     000
        000000     001
ERRORS DETECTED:  0
```

SYMBOL TABLE

VIRTUAL MEMORY USED: 9216 WORDS (36 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 46 PAGES
,FALCON/C=FALCON

CROSS REFERENCE TABLE (CREF V04.00)

```
$$$BRK  14-16#   27-35
$$$LTC  14-20#   27-37
$$TACK  13-30#   24-13    27-17    49-77    49-91
AROUN2  44-47    46-2#
AROUN3  54-24    56-2#
AUTOBA  27-13    28-43#
B.CNTL  13-14#   26-49*   49-54*   49-55    56-3
BADBOT  49-66    49-90#
BAUDR$   9-14#   28-44
BD.003   7-21#
BD.006   7-22#
BD.012   7-23#
BD.024   7-24#    9-14
BD.048   7-25#
BD.096   7-26#
BD.192   7-27#
BD.384   7-28#    9-18
BIT0     5-5#     7-43     8-29     8-47     10-5     10-13    35-50
BIT1     5-6#     7-36     8-27     8-38     8-40     8-42     8-44
BIT10    5-15#
BIT11    5-16#    6-16     6-37
BIT12    5-17#
BIT13    5-18#    6-35
BIT14    5-19#    6-32
BIT15    5-20#    6-30     10-3     10-9
BIT2     5-7#     7-30     8-25     8-38     8-39     8-42     8-43
BIT3     5-8#     7-15     8-23     8-38     8-39     8-40     8-41
BIT4     5-9#     7-16     8-21     9-34
BIT5     5-10#    7-17     8-18     38-11
BIT6     5-11#    6-23     7-8      8-17     38-9
BIT7     5-12#    6-19     7-3      8-15     9-32     10-4     10-11
BIT8     5-13#
BIT9     5-14#
BOOTS   33-21    49-25#
BRKNOO  14-18    14-26#   21-17
CH21N   61-34    61-46    62-34#
CH2OUT  60-27    62-16#
CH8OUT  52-17    62-13#
CHIN    62-35#
CHK240  51-17#   52-36
CHRET   62-24    62-39#
CR       5-25#   37-26    38-21    40-24    41-20    43-13
D.FLEN  48-125#  53-47
DEVBIT  10-11#   49-37
DEVNUM  10-13#   49-83
DIAGNO  33-25    44-37#
DIRBUF  48-62#   48-118   48-119   48-120   48-121   48-122   53-21
DONE    40-17    40-31#
DREAD   50-21    56-4     57-21#
E.EXT   10-17#   44-17
E.INT   10-18#   44-18
E.PAR   10-19#   46-6
EMPTY$  48-127#
ENDSG$  48-129#  53-31
ENTSIZ  48-124#
ERRBIT  44-19#   46-11
```

CROSS REFERENCE TABLE (CREF V04.00)

```
FAKOUT  27-12    27-15#
FILNAM  48-60#   53-38
GETCHR  33-18    36-30    37-25    39-15#   39-17    41-19    49-34    49-44
GETNUM  33-36    37-33    41-22#
HGHSEG  48-120#
HKBD$   35-57#   37-27    38-15
HKBDQ   35-56#   37-21    37-31    37-37    37-39    38-18    38-20
HVBAUD  29-10    29-31#
IN.USR  13-16#   14-17    14-31*   19-36*   20-31    21-11*   22-13*   27-16*   35-52*   49-31*
INBYT$  29-11    29-27#
INBYTE  29-8     29-20#   29-32
INITS   44-27#   46-14
KBD$    32-44    32-51#   35-57    46-45
KBDQ    32-48#   33-32    33-34    33-37    33-45    35-37    35-56    36-16    36-20    36-22    36-32
LCSET   33-43    37-17#
LEDOFF   9-10#   26-27    44-39
LF       5-24#   37-36    38-13    38-17    40-29    43-13
LOAD    53-44    54-3#
LOCDSP  36-39    37-18#   38-28
MODE     9-5#    24-20    44-38
MSG$    32-52    43-13#
MSGQ    32-49    43-12#
NEXNUM  41-18#   41-30
NO.LOW  10-9#    26-49
NOCT    41-25    41-35#
NXTSEG  48-119#  53-34
O.CNTL  13-10#   37-48    38-5
OCTST0  32-40    42-14#   46-44
OCTSTR  37-22    38-25    42-15#
ODT     14-32    22-19    27-18    32-11#
ODTFLG  13-26#   32-57*   36-14*   37-38    38-19
ODTLOC  13-27#   36-38*   37-17*   37-45    38-12*   38-23*   38-24
ODTSTK  13-29#   13-30    32-21
ODTWHY  13-7#    32-16*   35-35*
ONENUM  36-15    41-16#
PATERN  44-33#   46-19
PBR0     7-15#    7-22     7-24     7-26     7-28
PBR1     7-16#    7-23     7-24     7-27     7-28
PBR2     7-17#    7-25     7-26     7-27     7-28
PCMD    33-28    35-22#
PERMF$  48-128#  53-29
PP.A     8-6#    46-4
PP.B     8-7#    46-3*
PP.BI0   8-45#
PP.BI1   8-44#
PP.BI2   8-43#
PP.BI3   8-42#
PP.BI4   8-41#
PP.BI5   8-40#
PP.BI6   8-39#
PP.BI7   8-38#    9-10
PP.BIC   8-48#
PP.BIS   8-47#    9-10
PP.C     8-8#
PP.CHI   8-23#
PP.CLO   8-29#    9-5
```

```
PP.CWR   8-5#    24-20*   26-27*   44-38*   44-39*
PP.DRA   8-21#    9-5
PP.DRB   8-27#
PP.MD2   8-17#
PP.MDA   8-18#
PP.MDB   8-25#
PP.MOD   8-15#    9-5
PR16     9-27#   27-11    32-55    49-67
PR17     9-28#   27-36    27-38    35-48    55-5
PRINT   32-50    32-54#
PUTCHR  37-24    38-22    38-27    39-19#   39-21    40-18    40-25    40-30    42-22
PUTCLF  40-23#   42-14
PUTLF   40-29#
PUTSTR  32-56    40-15#   40-19
PWRSUP  24-12#   45-5     55-10
QUOT    32-37    32-41#
RS$CON  48-81#   52-27
RS$CTL  48-79#   60-26
RS$DAT  48-78#   61-7
RS$INT  48-80#   52-22    52-22
RS$XOF  48-82#
RSABRT  48-92#
RSCOMP  48-90#
RSDIAG  48-93#
RSEND   48-98#   61-25
RSGETC  48-96#
RSGETS  48-94#
RSINIT  48-87#
RSNUP   48-86#
RSPOSI  48-91#
RSREAD  48-88#   60-28
RSSETC  48-97#
RSSETS  48-95#
RSWRIT  48-89#
R.STRT  19-11#   45-7
R.HALT  10-3#    14-29    21-46
R.NXM   10-4#    19-34    22-18    35-35
R.PC    13-15#   21-35*   21-39*   21-41
R.STAK  10-5#    20-11    35-35
R.TYPE  13-18#   14-29*   19-32    19-34*   20-11*   20-17*   21-46*   22-18*   32-16    32-36    32-42    32-54*
RAMBOT   9-22#   25-19
RAMTOP   9-23#   25-24
RB.BRK   6-37#
RB.ERR   6-30#
RB.FRM   6-35#
RB.OVR   6-32#
RBUFS1   6-6#    24-35    29-3     29-7     32-12    39-18
RBUFS2   6-10#   48-69
RC.ACT   6-16#
RC.DUN   6-19#   24-36
RC.IEN   6-23#   24-36
RCMD    33-30    36-13#
RCMD1   36-23    36-36#
RCSRS1   6-5#    24-36    29-5     39-16
RCSRS2   6-9#    46-12    48-68
READU   53-22    54-23#
```

```
READZU  52-35    54-6     54-21#
REGOUT  36-25    36-34    36-38#
RESTAR  21-27    45-6#
RETRY   48-54#
RFLAG    9-32#   36-14
RPOINT  13-22#   32-32*   36-37
RTSEMT  48-138#
RTSFCH  48-141#
RTSFCT  48-142#
RTSHGH  48-137#
RTSISP  48-134#  54-17
RTSJSW  48-135#
RTSRMN  48-140#
RTSSTA  48-133#  54-9
RTSUER  48-139#
RTSUSR  48-136#
RXSSO2  48-21#   58-5
RXSSDE  48-23#   57-44
RXSSDN  48-26#   59-15
RXSSER  48-18#
RXSSFN  48-28#
RXSSGO  48-29#   48-33    48-34    48-35    48-36    48-37    48-38    48-39    48-40
RXSSIE  48-25#
RXSSIN  48-19#
RXSSTR  48-24#   59-15
RXSSUN  48-27#   57-31
RXSSXA  48-20#
RXSSXX  48-22#
RXSEMP  48-34#   58-4
RXSFIL  48-33#
RXSREC  48-40#
RXSRED  48-36#   57-78
RXSRST  48-38#   57-33
RXSSTD  48-37#
RXSWDD  48-39#
RXSWRT  48-35#
RXBOUT  49-87    50-14#
RXCS    48-13#   48-14    50-15    50-16    58-5     59-12*
RXDB    48-14#   57-27
RXESCR  48-50#
RXESDD  48-46#
RXESDE  48-48#
RXESDN  48-47#   57-37
RXESDR  48-45#
RXESID  48-49#
RXESUN  48-44#
RXGO    57-32    57-77    58-3     59-10#
S$CART  48-106#
S$DCHK  48-108#
S$MOTR  48-110#
S$NORM  48-102#
S$OPCD  48-111#
S$PART  48-104#
S$RECN  48-112#
S$RETR  48-103#
S$SEEK  48-109#
```

CROSS REFERENCE TABLE (CREF V04.00)

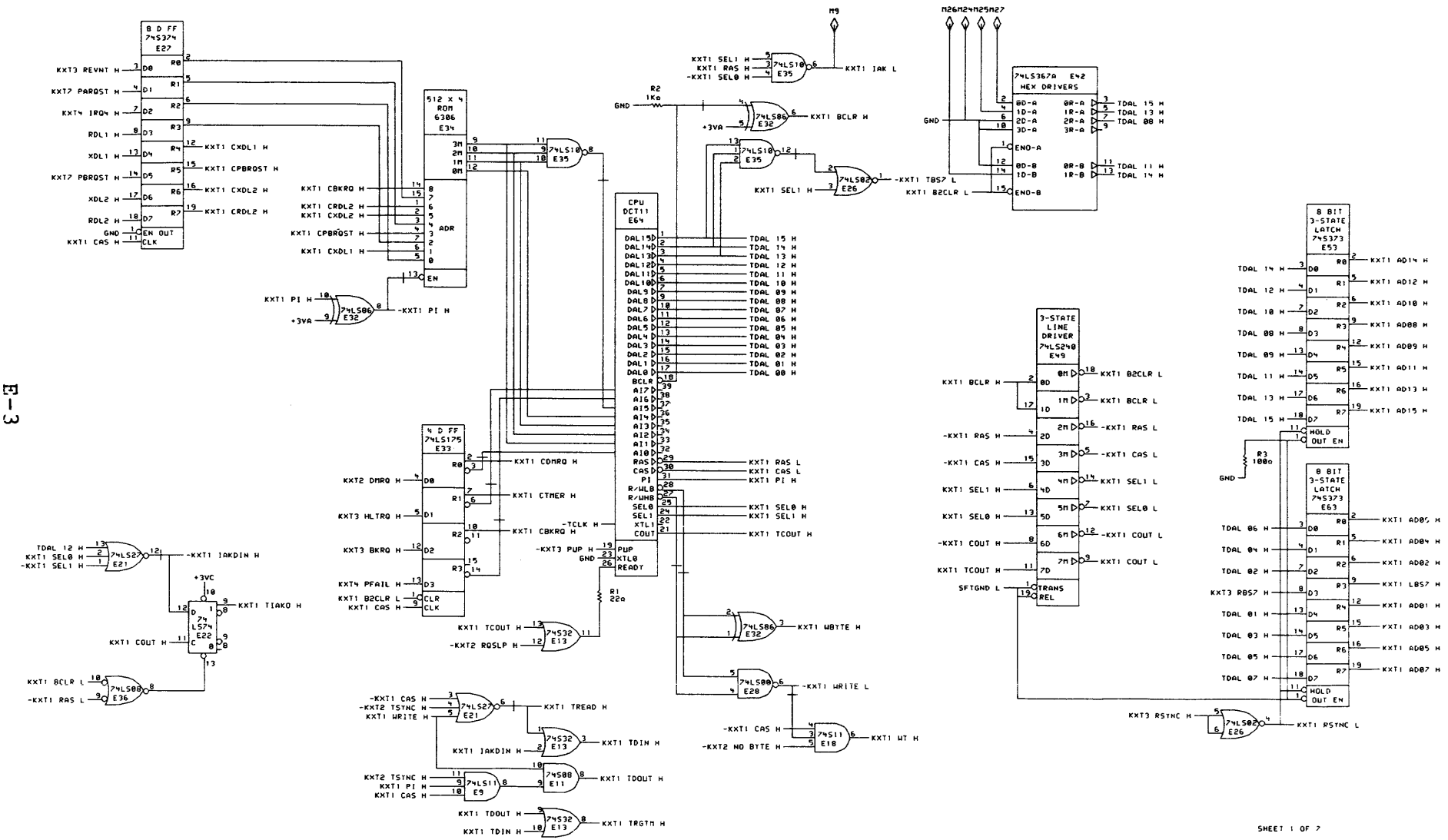
```
S$UNIT  48-105#
S$WPRT  48-107#
SAVPC   13-25#   14-27*   21-37*   27-12*   32-39    35-11*   35-54    36-24
SAVPS   13-24#   14-28*   21-36*   27-11*   35-16*   35-53    36-33    37-45
SEGALO  48-116#
SPACE    5-26#   37-23
SRET    41-21    41-32#
ST173   55-4#
STANDB  51-21    53-15#
START   45-4#
START$  54-11    54-14#
STRBLK  48-122#  53-25
STTUBD  49-30#
SWCMD   36-18    36-29#
T.BIT    9-34#   37-50
TENTA$  48-126#
TISBFR  48-69#   62-38
TISCSR  48-68#   52-13    52-25    62-36
TOSBFR  48-71#   62-23*
TOSCSR  48-70#   49-26*   52-14    62-21
TRAP4   13-5#    22-11    49-78*   54-18*
TREAD   56-5     60-24#
TUBAUD   9-18#   49-26
TUBOOT  49-88    52-12#
USER$P  13-20#   32-20*   32-25
VECSET  26-46    27-34#   51-22
XBUF$1   6-8#    39-22*
XBUF$2   6-12#   48-71
XC.BRK   7-43#   52-20
XC.IEN   7-8#
XC.MNT   7-30#   44-26
XC.PBE   7-36#    9-14     9-18    44-25    44-26
XC.RDY   7-3#    24-39
XCSR$1   6-7#    24-32*   24-39    28-44*   29-37*   39-20
XCSR$2   6-11#   48-70
XTRBYT  48-121#  53-50
```

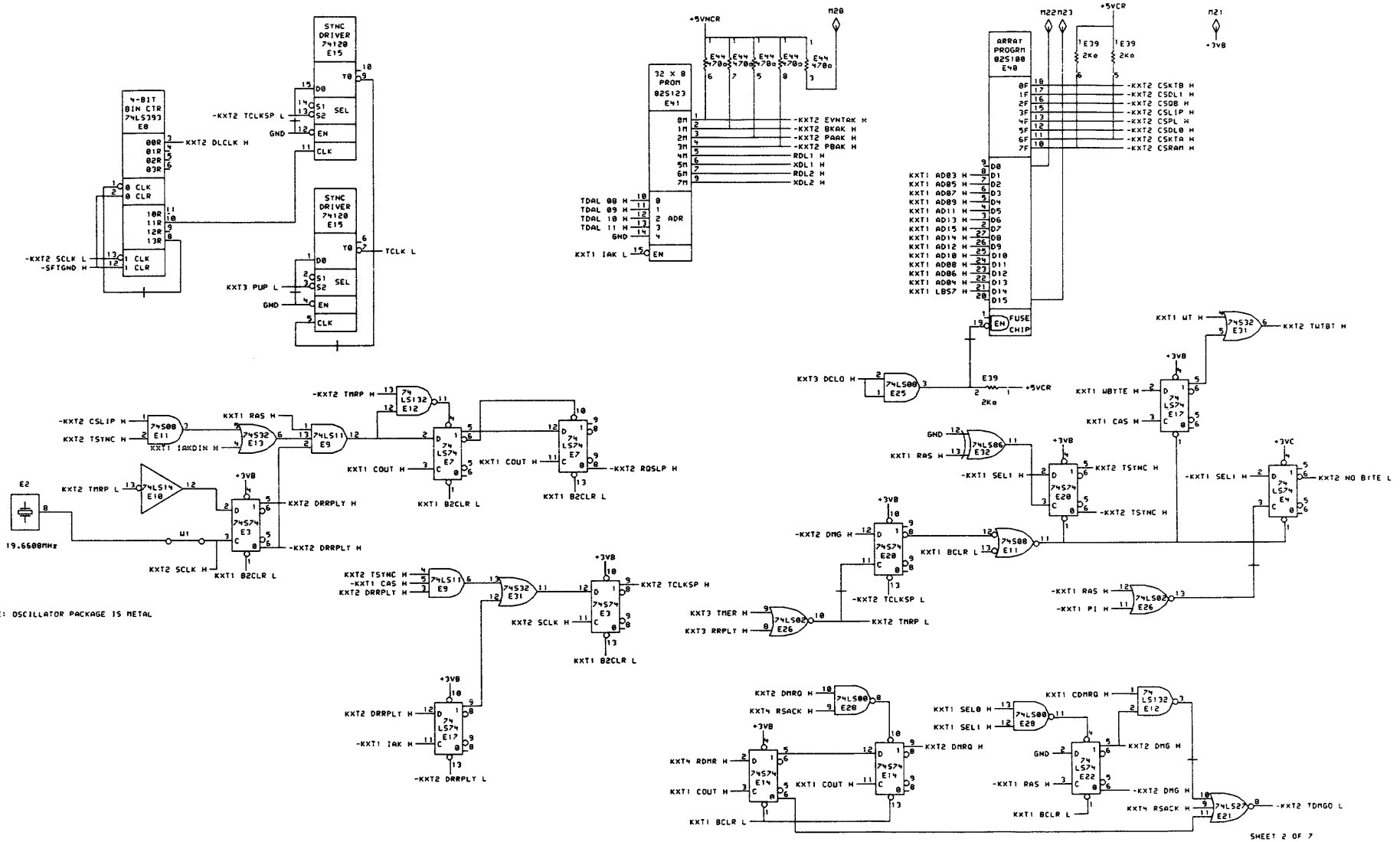
CROSS REFERENCE TABLE (CREF V04.00)

ABORT 11-15# 49-42 49-51 49-58 49-92 51-23 52-29 52-37 53-24 53-36 54-8 54-12 57-36 57-114
       61-27 61-37
DELAY 12-23# 49-76

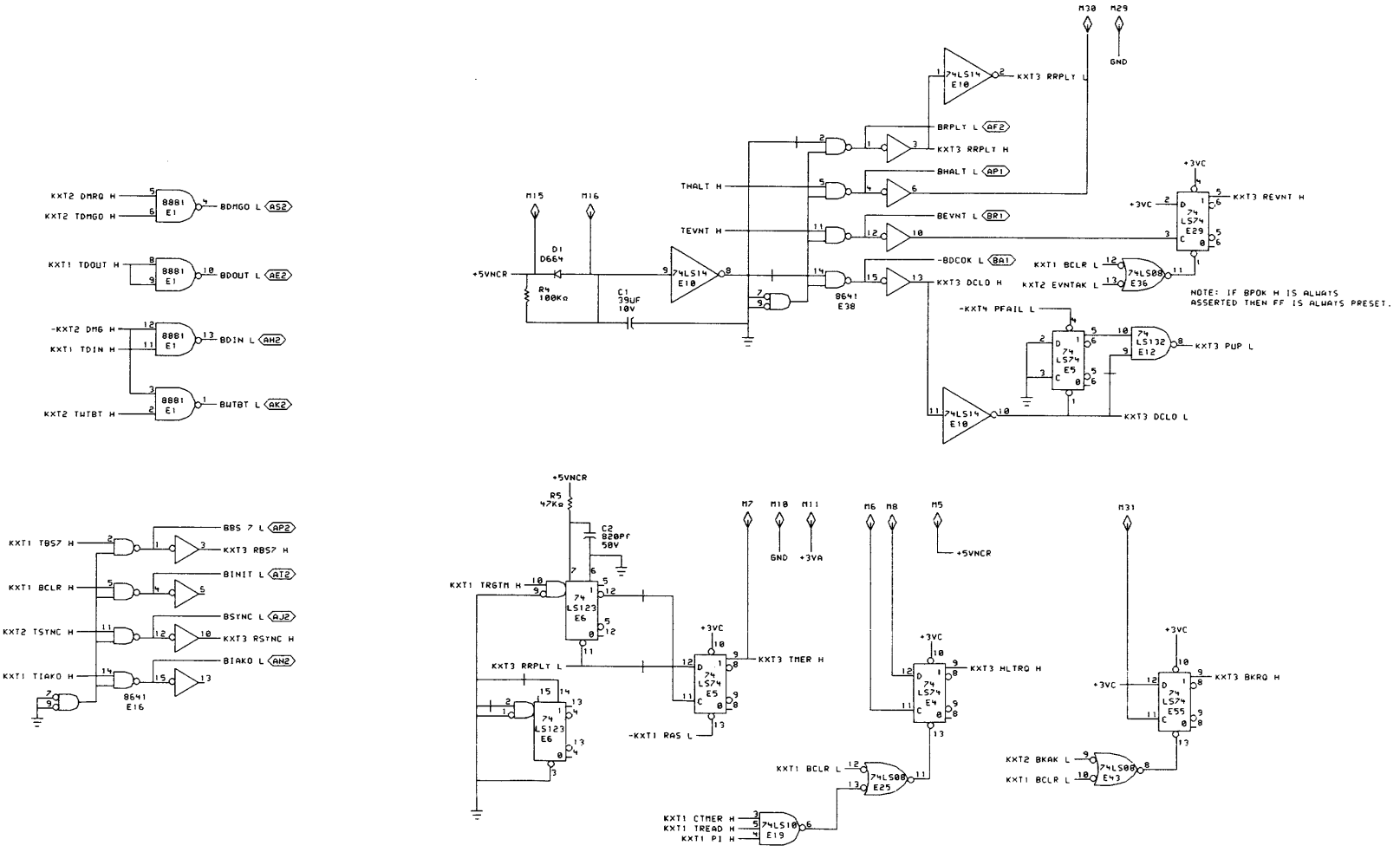
APPENDIX E
SBC-11/21 SCHEMATICS

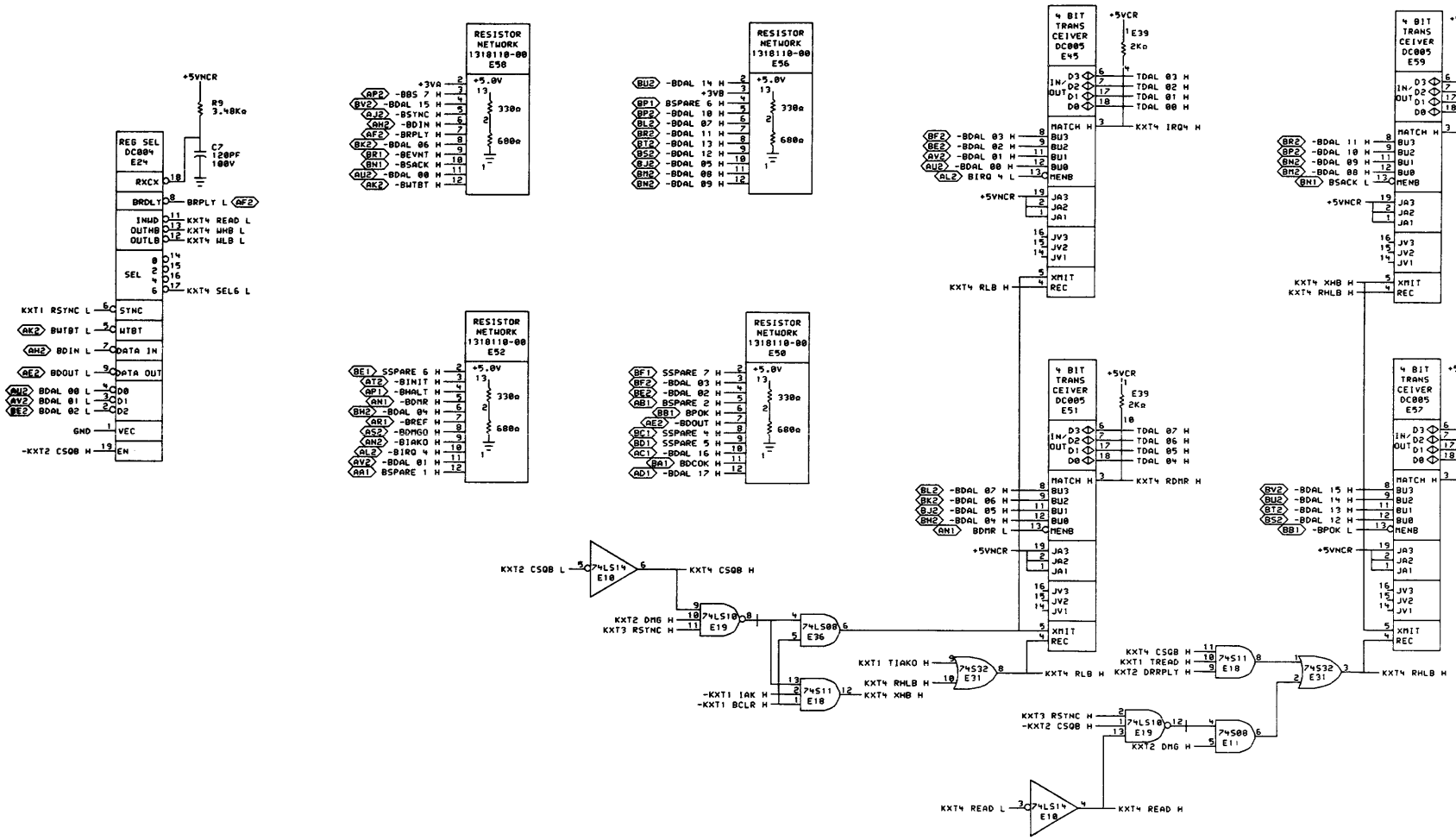
This appendix provides the user with the electrical schematics for the SBC-11/21 module.

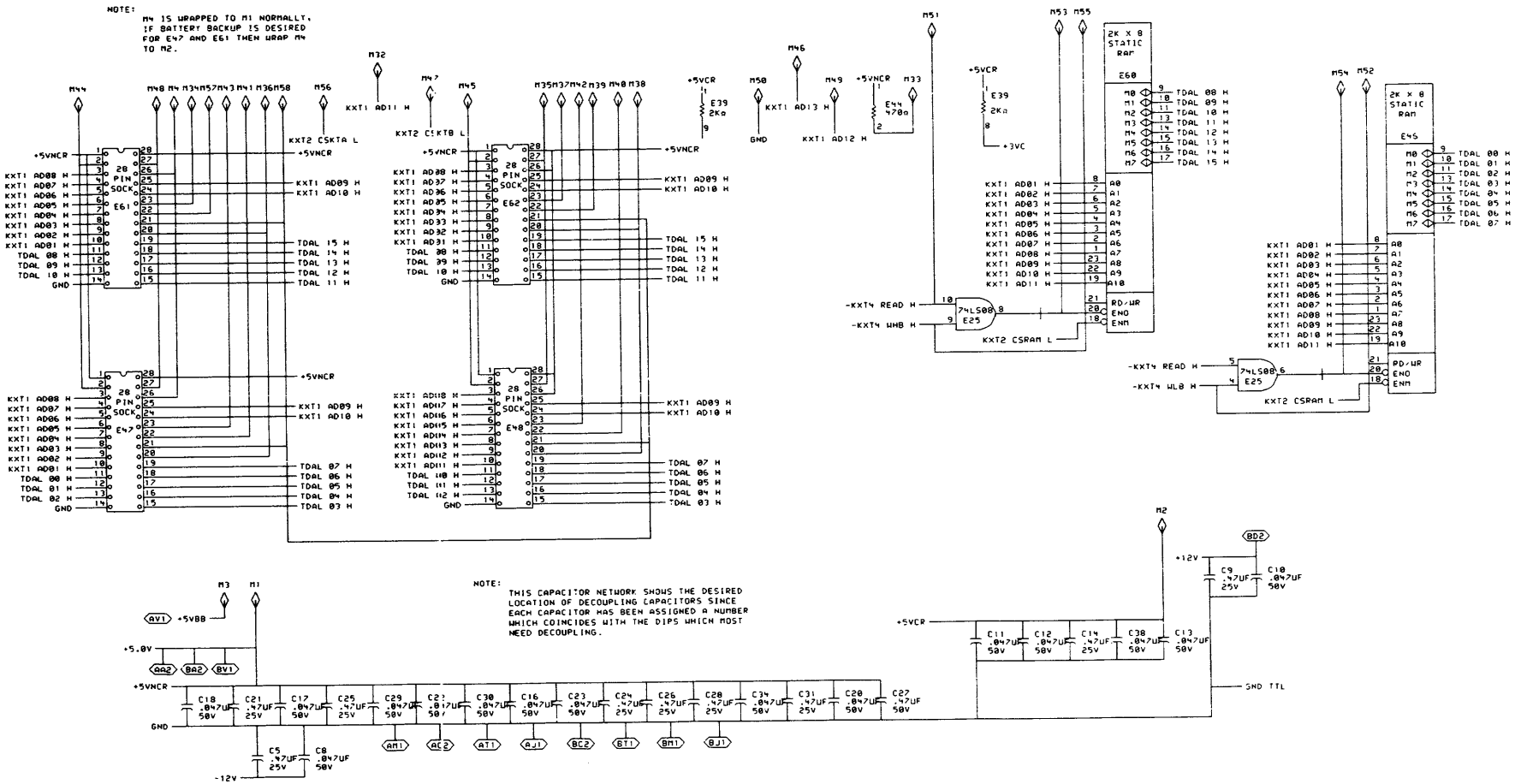

8 D FF
74S374
E27
D0
D1
D2
D3
D4
D5
D6
D7
EN OUT
CLK
R0
R1
R2
R3
R4
R5
R6
R7
KXT3 REVNT H
KXT7 PAROST H
KXT4 IRQN H
RDL1 H
XDL1 H
KXT7 PBROST H
XDL2 H
RDL2 H
GND
KXT1 CAS H
KXT1 CXDL1 H
KXT1 CPBROST H
KXT1 CXDL2 H
KXT1 CRDL2 H
512 X 4
ROM
6306
E34
3M
2M
1M
0M
ADR
EN
KXT1 CBKRQ H
KXT1 CRDL2 H
KXT1 CXDL2 H
KXT1 CPBRQST H
KXT1 CXDL1 H
74LS10
E35
KXT1 PI H
74LS86
E32
+3VA
-KXT1 PI H
CPU
DCT11
E64
DAL15
DAL14
DAL13
DAL12
DAL11
DAL10
DAL9
DAL8
DAL7
DAL6
DAL5
DAL4
DAL3
DAL2
DAL1
DAL0
BCLR
AI7
AI6
AI5
AI4
AI3
AI2
AI1
AI0
RAS
CAS
PI
R/WL0
R/WH0
SEL0
SEL1
XTL1
COUT
PUP
XTL0
READY
TDAL 15 H
TDAL 14 H
TDAL 13 H
TDAL 12 H
TDAL 11 H
TDAL 10 H
TDAL 09 H
TDAL 08 H
TDAL 07 H
TDAL 06 H
TDAL 05 H
TDAL 04 H
TDAL 03 H
TDAL 02 H
TDAL 01 H
TDAL 00 H
KXT1 RAS L
KXT1 CAS L
KXT1 PI H
KXT1 SEL0 H
KXT1 SEL1 H
KXT1 TCOUT H
-TCLK H
-KXT3 PUP H
GND
R1
22Ω
4 D FF
74LS175
E33
KXT2 DMRQ H
KXT3 HLTRQ H
KXT3 BKRQ H
KXT4 PFAIL H
KXT1 B2CLR L
KXT1 CAS H
KXT1 CDMRQ H
KXT1 CTMER H
KXT1 CBKRQ H
KXT1 TCOUT H
-KXT2 RQSLP H
74LS32
E13
R2
1Ko
GND
+3VA
74LS08
E32
KXT1 BCLR H
KXT1 SEL1 H
KXT1 RAS H
-KXT1 SEL0 H
74LS10
E35
KXT1 IAK L
M9
M26M24M25M27
74LS367A E42
HEX DRIVERS
GND
ENO-A
ENO-B
TDAL 15 H
TDAL 13 H
TDAL 08 H
TDAL 11 H
TDAL 14 H
-KXT1 TBS7 L
KXT1 B2CLR L
74LS08
E26
KXT1 SEL1 H
TDAL 12 H
KXT1 SEL0 H
-KXT1 SEL1 H
74LS27
E21
-KXT1 IAKDIN H
+3VC
74
LS74
E22
KXT1 TIAKO H
KXT1 COUT H
KXT1 BCLR L
-KXT1 RAS L
74LS08
E36
-KXT1 CAS H
-KXT2 TSTNC H
KXT1 WRITE H
KXT1 TREAD H
KXT1 IAKDIN H
KXT1 TDIN H
KXT2 TSTNC H
KXT1 PI H
KXT1 CAS H
74LS11
E9
74S08
E11
KXT1 TDOUT H
KXT1 TDOUT H
KXT1 TDIN H
KXT1 TRGTN H
74LS86
E32
KXT1 WBYTE H
74LS08
E28
-KXT1 WRITE L
-KXT1 CAS H
-KXT2 NO BYTE H
74S11
E18
KXT1 WT H
3-STATE
LINE
DRIVER
74LS240
E49
KXT1 BCLR H
-KXT1 RAS H
-KXT1 CAS H
KXT1 SEL1 H
KXT1 SEL0 H
-KXT1 COUT H
KXT1 TCOUT H
SFTGND L
0M
0D
1M
1D
2M
2D
3M
3D
4M
4D
5M
5D
6M
6D
7M
7D
TRANS
REL
KXT1 B2CLR L
KXT1 BCLR L
-KXT1 RAS L
-KXT1 CAS L
KXT1 SEL1 L
KXT1 SEL0 L
-KXT1 COUT L
KXT1 COUT L
8 BIT
3-STATE
LATCH
74S373
E53
TDAL 14 H
TDAL 12 H
TDAL 10 H
TDAL 08 H
TDAL 09 H
TDAL 11 H
TDAL 13 H
TDAL 15 H
HOLD
OUT EN
KXT1 AD14 H
KXT1 AD12 H
KXT1 AD10 H
KXT1 AD08 H
KXT1 AD09 H
KXT1 AD11 H
KXT1 AD13 H
KXT1 AD15 H
R3
100Ω
GND
8 BIT
3-STATE
LATCH
74S373
E63
TDAL 06 H
TDAL 04 H
TDAL 02 H
KXT3 RBS7 H
TDAL 01 H
TDAL 03 H
TDAL 05 H
TDAL 07 H
KXT1 AD06 H
KXT1 AD04 H
KXT1 AD02 H
KXT1 LBS7 H
KXT1 AD01 H
KXT1 AD03 H
KXT1 AD05 H
KXT1 AD07 H
KXT3 RSTNC H
74LS08
E26
KXT1 RSTNC L

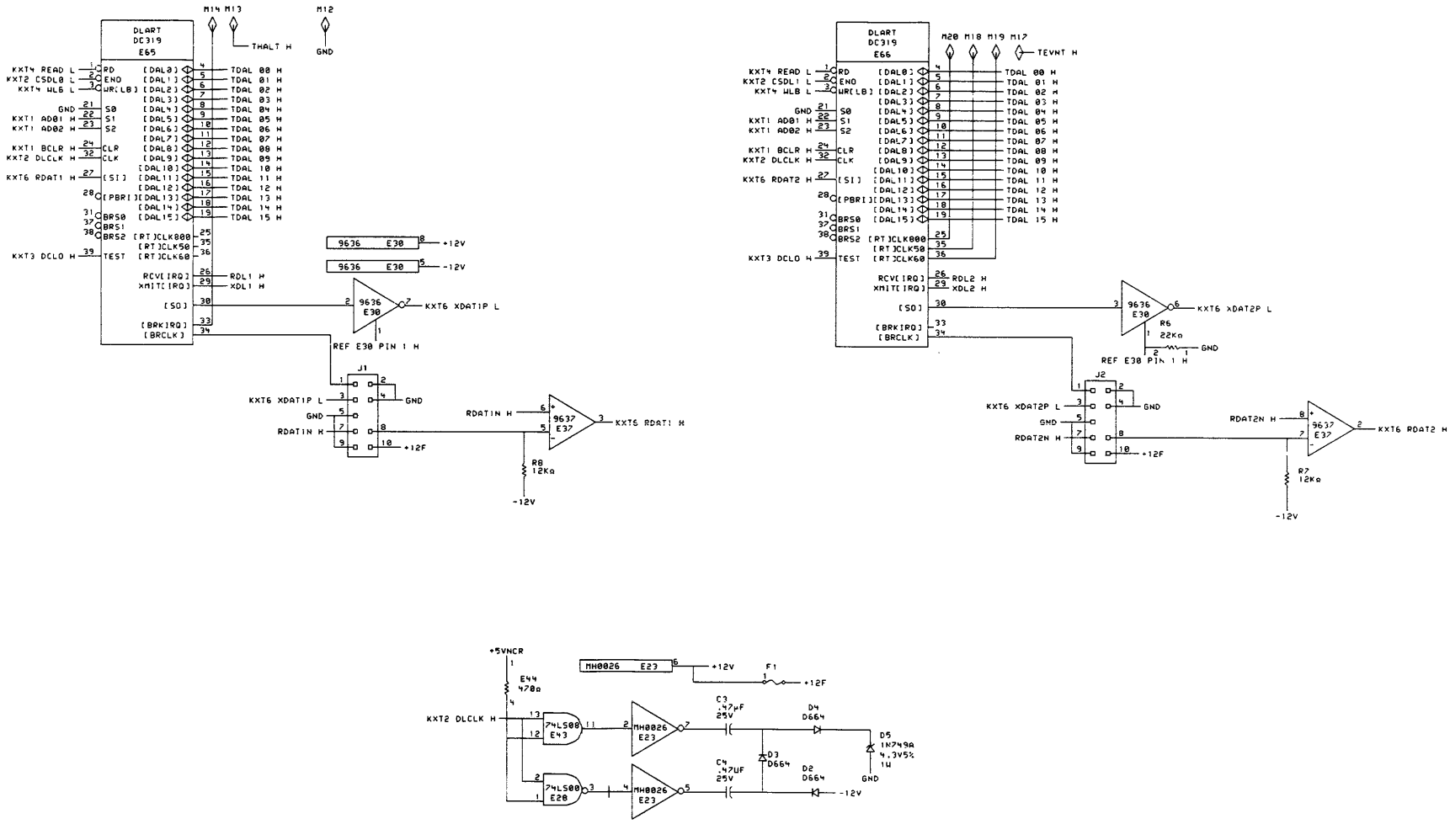

SYNC DRIVER 74120 E15
4-BIT BIN CTR 74LS393 E8
-KXT2 TCLKSP L
KXT2 DLCLK H
-KXT2 SCLK L
-SFTGND H
KXT3 PUP L
TCLK L
32 X 8 PROM 82S123 E41
TDAL 08 H
TDAL 09 H
TDAL 10 H
TDAL 11 H
KXT1 IAK L
-KXT2 EVNTAK H
-KXT2 BKAK H
-KXT2 PAAK H
-KXT2 PBAK H
RDL1 H
XDL1 H
RDL2 H
XDL2 H
+5VNCR
M20
M22M23
+5VCR
M21
+3VB
ARRAY PROGRM 82S100 E40
-KXT2 CSKTB H
-KXT2 CSDL1 H
-KXT2 CSOB H
-KXT2 CSLIP H
-KXT2 CSPL H
-KXT2 CSDL0 H
-KXT2 CSKTA H
-KXT2 CSRAM H
KXT1 AD03 H
KXT1 AD05 H
KXT1 AD07 H
KXT1 AD09 H
KXT1 AD11 H
KXT1 AD13 H
KXT1 AD15 H
KXT1 AD14 H
KXT1 AD12 H
KXT1 AD10 H
KXT1 AD08 H
KXT1 AD06 H
KXT1 AD04 H
KXT1 LBS7 H
FUSE CHIP
KXT3 DCLO H
KXT1 WT H
KXT2 TWTBT H
KXT1 WBYTE H
KXT1 CAS H
KXT1 RAS H
KXT1 SEL1 H
KXT2 TSTNC H
KXT2 NO BYTE L
-KXT2 CSLIP H
KXT2 TSTNC H
-KXT2 TMRP H
KXT1 RAS H
KXT1 IAKDIN H
KXT1 COUT H
KXT1 B2CLR L
-KXT2 RQSLP H
E2
19.6608MHz
KXT2 TMRP L
KXT2 DRRPLY H
-KXT2 DRRPLY H
W1
KXT2 SCLK H
NOTE: OSCILLATOR PACKAGE IS METAL
-KXT2 DMG H
KXT1 BCLR L
-KXT2 TCLKSP L
KXT3 TMER H
KXT3 RRPLY H
KXT2 TMRP L
-KXT1 RAS H
-KXT1 PI H
KXT2 TCLKSP H
KXT2 DRRPLT H
-KXT1 IAK H
-KXT2 DRRPLY L
KXT2 DMRQ H
KXT4 RSACK H
KXT4 RDMR H
KXT1 SEL0 H
KXT1 CDMRQ H
KXT2 DMG H
-KXT2 TDMGO L
SHEET 2 OF 7

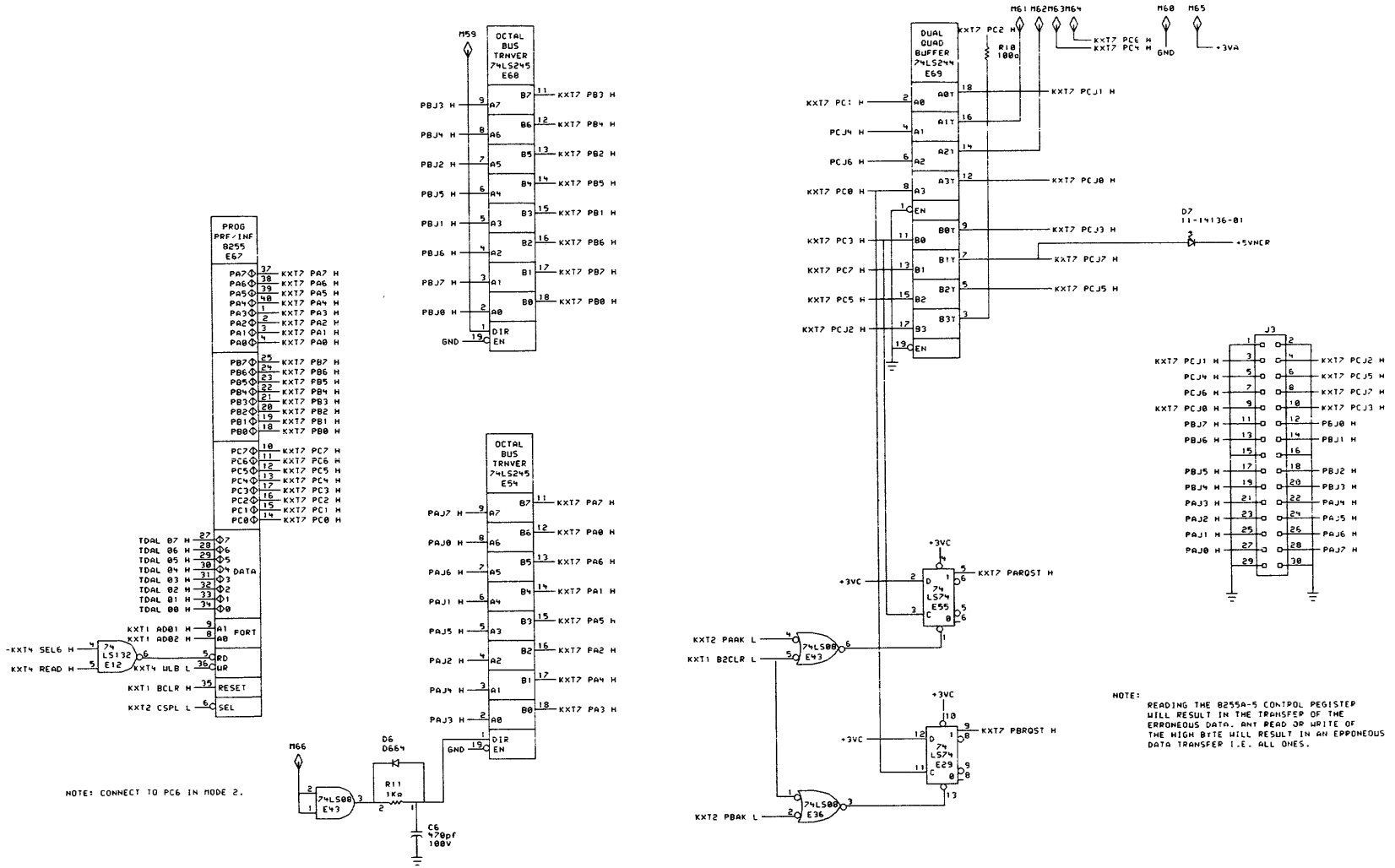
E-5


KXT2 DMRQ H
KXT2 TDMGO H
BDMGO L AS2
KXT1 TDOUT H
BDOUT L AE2
-KXT2 DMG H
KXT1 TDIN H
BDIN L AH2
KXT2 TWTBT H
BWTBT L AK2
KXT1 TBS7 H
BBS 7 L AP2
KXT3 RBS7 H
KXT1 BCLR H
BINIT L AT2
BSYNC L AJ2
KXT2 TSYNC H
KXT3 RSYNC H
KXT1 TIAKO H
BIAKO L AN2
8641 E16
+5VNCR
R5 47KΩ
C2 820PF 50V
KXT1 TRGTM H
74 LS123 E6
KXT3 RRPLT L
KXT3 TMER H
-KXT1 RAS L
+3VC
74 LS74 E5
M7
M10
M11
GND
+3VA
M6
M8
M5
+5VNCR
KXT3 MLTRQ H
KXT1 BCLR L
74LS08 E25
KXT1 CTMER H
KXT1 TREAD H
KXT1 PI H
74LS10 E19
M31
KXT3 BKRQ H
KXT2 BKAK L
KXT1 BCLR L
74LS08 E43
M15
M16
D1 D664
+5VNCR
R4 100KΩ
C1 39UF 10V
74LS14 E10
THALT H
TEVNT H
8641 E38
BRPLT L AF2
KXT3 RRPLT H
BHALT L AP1
BEVNT L BR1
-BDCOK L BA1
KXT3 DCLO H
KXT3 RRPLT L
M30
M29
GND
KXT3 REVNT H
KXT1 BCLR L
KXT2 EVNTAK L
74LS08 E36
NOTE: IF BPOK H IS ALWAYS ASSERTED THEN FF IS ALWAYS PRESET.
-KXT4 PFAIL L
74 LS132 E12
KXT3 PUP L
KXT3 DCLO L


REG SEL DC004 E24
RXCX
BRDLY
INWD
OUTHB
OUTLB
SEL
STNC
WTBT
DATA IN
DATA OUT
VEC
EN
+5VNCR
R9 3.48Kα
C7 120PF 100V
BRPLT L
KXT4 READ L
KXT4 WHB L
KXT4 WLB L
KXT4 SEL6 L
KXT1 RSYNC L
BWTBT L
BDIN L
BDOUT L
BDAL 00 L
BDAL 01 L
BDAL 02 L
-KXT2 CSQB H
RESISTOR NETWORK 1318110-00 E58
RESISTOR NETWORK 1318110-00 E56
RESISTOR NETWORK 1318110-00 E52
RESISTOR NETWORK 1318110-00 E50
+5.0V
330α
680α
4 BIT TRANS CEIVER DC005 E45
4 BIT TRANS CEIVER DC005 E59
4 BIT TRANS CEIVER DC005 E51
4 BIT TRANS CEIVER DC005 E57
E39 2Kα
MATCH H
XMIT
REC
MENB
KXT4 IRQ4 H
KXT4 RSACK H
KXT4 RDMR H
KXT4 PFAIL H
KXT4 RLB H
KXT4 XHB H
KXT4 RHLB H
BIRQ 4 L
BSACK L
BDMR L
BPOK L
74LS14 E10
74LS10 E19
74LS08 E36
74S32 E31
74S11 E18
74S08 E11
KXT2 CSQB L
KXT4 CSQB H
KXT2 DMG H
KXT3 RSYNC H
KXT1 TIAKO H
KXT4 RHLB H
KXT4 XHB H
-KXT1 IAK H
-KXT1 BCLR H
KXT4 CSQB H
KXT1 TREAD H
KXT2 DRRPLY H
-KXT2 CSQB H
KXT4 READ L
KXT4 READ H


NOTE:
M4 IS WRAPPED TO M1 NORMALLY.
IF BATTERY BACKUP IS DESIRED
FOR E47 AND E61 THEN WRAP M4
TO M2.
28 PIN SOCK
E61
E62
E47
E48
+5VNCR
KXT1 AD08 H
KXT1 AD07 H
KXT1 AD06 H
KXT1 AD05 H
KXT1 AD04 H
KXT1 AD03 H
KXT1 AD02 H
KXT1 AD01 H
TDAL 00 H
TDAL 01 H
TDAL 02 H
TDAL 03 H
TDAL 04 H
TDAL 05 H
TDAL 06 H
TDAL 07 H
TDAL 08 H
TDAL 09 H
TDAL 10 H
TDAL 11 H
TDAL 12 H
TDAL 13 H
TDAL 14 H
TDAL 15 H
KXT1 AD09 H
KXT1 AD10 H
KXT1 AD11 H
KXT1 AD12 H
KXT1 AD13 H
KXT2 CSKTA L
KXT2 CSKTB L
GND
E39 2Ka
E44 470a
2K X 8 STATIC RAM
E60
E45
74LS08 E25
-KXT4 READ H
-KXT4 WHB H
KXT2 CSRAM L
RD/WR
ENO
ENM
+3VC
NOTE:
THIS CAPACITOR NETWORK SHOWS THE DESIRED LOCATION OF DECOUPLING CAPACITORS SINCE EACH CAPACITOR HAS BEEN ASSIGNED A NUMBER WHICH COINCIDES WITH THE DIPS WHICH MOST NEED DECOUPLING.
+5VBB
+5.0V
-12V
+12V
SND TTL
C18 .047UF 50V
C21 .47UF 25V
C17 .047UF 50V
C25 .47UF 25V
C29 .047UF 50V
C22 .047UF 50V
C30 .047UF 50V
C16 .047UF 50V
C23 .047UF 50V
C24 .47UF 25V
C26 .47UF 25V
C28 .47UF 25V
C34 .047UF 50V
C31 .47UF 25V
C20 .047UF 50V
C27 .47UF 50V
C5 .47UF 25V
C8 .047UF 50V
C9 .47UF 25V
C10 .047UF 50V
C11 .047UF 50V
C12 .047UF 50V
C14 .47UF 25V
C38 .047UF 50V
C13 .047UF 50V


DLART
DC319
E65
M14 M13
THALT H
M12
GND
KXT4 READ L
KXT2 CSDL0 L
KXT4 WLB L
KXT1 AD01 H
KXT1 AD02 H
KXT1 BCLR H
KXT2 DLCLK H
KXT6 RDAT1 H
KXT3 DCLO H
TDAL 00 H
TDAL 01 H
TDAL 02 H
TDAL 03 H
TDAL 04 H
TDAL 05 H
TDAL 06 H
TDAL 07 H
TDAL 08 H
TDAL 09 H
TDAL 10 H
TDAL 11 H
TDAL 12 H
TDAL 13 H
TDAL 14 H
TDAL 15 H
9636 E30
+12V
-12V
REF E30 PIN 1 H
KXT6 XDAT1P L
J1
KXT6 XDAT1P L
GND
RDAT1N H
+12F
9637 E37
KXT6 RDAT1 H
R8
12Kα
-12V
DLART
DC319
E66
M20 M18 M19 M12
TEVNT H
KXT4 READ L
KXT2 CSDL1 L
KXT4 WLB L
KXT1 AD01 H
KXT1 AD02 H
KXT1 BCLR H
KXT2 DLCLK H
KXT6 RDAT2 H
KXT3 DCLO H
TDAL 00 H
TDAL 01 H
TDAL 02 H
TDAL 03 H
TDAL 04 H
TDAL 05 H
TDAL 06 H
TDAL 07 H
TDAL 08 H
TDAL 09 H
TDAL 10 H
TDAL 11 H
TDAL 12 H
TDAL 13 H
TDAL 14 H
TDAL 15 H
RDL2 H
XDL2 H
9636 E30
KXT6 XDAT2P L
R6
22Kα
GND
REF E30 PIN 1 H
J2
KXT6 XDAT2P L
GND
RDAT2N H
+12F
RDAT2N H
9637 E37
KXT6 RDAT2 H
R7
12Kα
-12V
+5VNCR
E44
470α
MH0026 E23
+12V
F1
+12F
KXT2 DLCLK H
74LS00 E43
74LS00 E28
MH0026 E23
MH0026 E23
C3
.47μF
25V
C4
.47UF
25V
D4
D664
D3
D664
D2
D664
D5
1N749A
4.3V5%
1W
GND
-12V


PROG PRF/INF 8255 E67
PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0
TDAL 07 H TDAL 06 H TDAL 05 H TDAL 04 H TDAL 03 H TDAL 02 H TDAL 01 H TDAL 00 H
DATA
KXT1 AD01 H
KXT1 AD02 H
A1
A0
PORT
-KXT4 SEL6 H
KXT4 READ H
74 LS132 E12
KXT4 WLB L
RD
WR
KXT1 BCLR H
RESET
KXT2 CSPL L
SEL
NOTE: CONNECT TO PC6 IN MODE 2.
M59
OCTAL BUS TRNVER 74LS245 E68
PBJ3 H
PBJ4 H
PBJ2 H
PBJ5 H
PBJ1 H
PBJ6 H
PBJ7 H
PBJ0 H
KXT7 PB3 H
KXT7 PB4 H
KXT7 PB2 H
KXT7 PB5 H
KXT7 PB1 H
KXT7 PB6 H
KXT7 PB7 H
KXT7 PB0 H
GND
DIR
EN
OCTAL BUS TRNVER 74LS245 E54
PAJ7 H
PAJ0 H
PAJ6 H
PAJ1 H
PAJ5 H
PAJ2 H
PAJ4 H
PAJ3 H
KXT7 PA7 H
KXT7 PA0 H
KXT7 PA6 H
KXT7 PA1 H
KXT7 PA5 H
KXT7 PA2 H
KXT7 PA4 H
KXT7 PA3 H
M66
D6 D664
R11 1Ko
74LS08 E43
C6 470pf 100V
M61 M62 M63 M64
M60 M65
GND
+3VA
DUAL QUAD BUFFER 74LS244 E59
KXT7 PC2 H
KXT7 PC6 H
KXT7 PC4 H
R10 100Ω
KXT7 PC1 H
PCJ4 H
PCJ6 H
KXT7 PC0 H
KXT7 PC3 H
KXT7 PC7 H
KXT7 PC5 H
KXT7 PC2 H
A0T A1T A2T A3T B0T B1T B2T B3T
KXT7 PCJ1 H
KXT7 PCJ0 H
KXT7 PCJ3 H
KXT7 PCJ7 H
KXT7 PCJ5 H
D7 11-141136-01
+5VNCR
J3
KXT7 PCJ1 H
PCJ4 H
PCJ6 H
KXT7 PCJ0 H
PBJ7 H
PBJ6 H
PBJ5 H
PBJ4 H
PAJ3 H
PAJ2 H
PAJ1 H
PAJ0 H
KXT7 PCJ2 H
KXT7 PCJ5 H
KXT7 PCJ7 H
KXT7 PCJ3 H
PBJ0 H
PBJ1 H
PBJ2 H
PBJ3 H
PAJ4 H
PAJ5 H
PAJ6 H
PAJ7 H
+3VC
74 LS74 E55
KXT7 PARQST H
KXT2 PAAK L
KXT1 B2CLR L
74LS08 E43
74 LS74 E29
KXT7 PBRQST H
KXT2 PBAK L
74LS08 E36
NOTE:
READING THE 8255A-5 CONTROL REGISTER WILL RESULT IN THE TRANSFER OF THE ERRONEOUS DATA. ANY READ OR WRITE OF THE HIGH BYTE WILL RESULT IN AN ERRONEOUS DATA TRANSFER I.E. ALL ONES.

Digital Equipment Corporation • Bedford, MA 01730