AIM 65/40

System user's manual

microcomputer system

Rockwell
International

...where science gets down to business

WARNING: The equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

This equipment has been tested and certified with the Rockwell model A65/40-0004 Series power supplies.

INFORMATION TO USER: This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- reorient the receiving antenna
- relocate the computer with respect to the receiver
- move the computer away from the receiver
- plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

"How to Identify and Resolve Radio-TV Interference Problems."

This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4.

169R12-006

AIM 65/40

System user's manual

microcomputer system

Rockwell
International

...where science gets down to business

$15.00
Document No.
29650N86
Order No. 280
Rev. 1 Jan. 1982

NOTICE

Rockwell International does not assume any liability arising out of the application or use of any products, circuit, or software described herein, neither does it convey any license under its patent rights nor the patent rights of others. Rockwell International further reserves the right to make changes in any products herein without notice.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

SECTION 6. USING THE I/O ROM

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF TABLES (continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

SECTION 1

INTRODUCTION

Congratulations on your selection of the Rockwell AIM 65/40 Microcomputer System. The AIM 65/40 is a professional 8-bit microcomputer designed for flexible installation and easy operation in a wide range of industrial, scientific and education applications. Standalone peripherals with independent operation allow individual subassemblies or the integrated system to be installed in a variety of hardware and software/firmware configurations to satisfy your application requirements. The AIM 65/40 you are using may consist of one or more of the standard or optional AIM 65/40 components.

This manual describes how to install and operate a standard configuration AIM 65/40 Microcomputer System, part number A65/40-5000. This system (shown in Figure 1-1) consists of the following modular components:

o AIM 65/40 Graphics Printer	A65/40-0600
o AIM 65/40 40-Character Display	A65/40-0400
o AIM 65/40 Keyboard	A65/40-0200
o AIM 65/40 Single Board Computer with I/O ROM	A65/40-1000
o AIM 65/40 Debug Monitor and Text Editor ROMs	A65/40-7000

A variety of hardware and firmware options extend the development and implementation capabilities of the AIM 65/40. These include:

o AIM 65/40 Expanded Keyboard	A65/40-0800
o AIM 65/40 CRT Controller Module	A65/40-0210
o Symbolic Assembler	A65/40-7010
o BASIC Interpreter	A65/40-7020
o FORTH System	A65/40-7050
o PL/65 Compiler	A65/40-7030
o Instant Pascal System	A65/40-7060

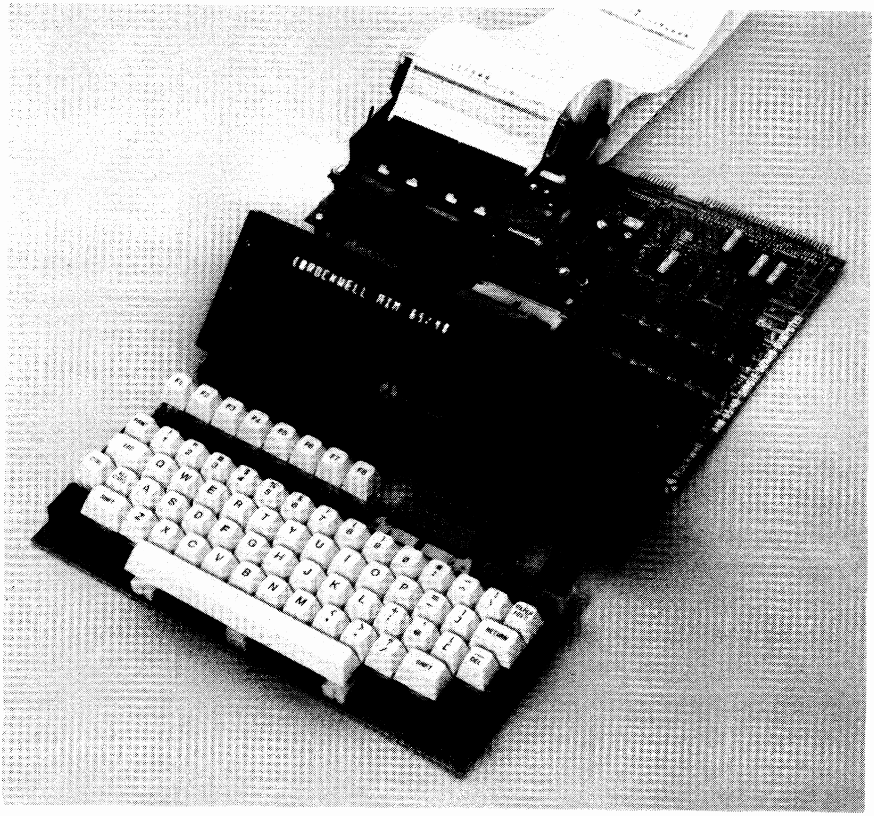


Figure 1-1. AIM 65/4Ø Microcomputer System

Other functions may easily be added to the AIM 65/4Ø by the connecting Rockwell RM 65 modules, such as PROM/ROM, RAM, FDC and CRTC peripheral controllers, IEEE-488 interface, RS-232C serial, parallel I/O, and analog I/O. These modules connect directly to the AIM 65/4Ø Expansion connector through a user-installed receptacle (for a single add-on module) or through an RM 65 adapter to an RM 65 card cage (for a multiple module expansion). Separate user manuals describing operation and language capabilities of the optional AIM 65/4Ø hardware and firmware, as well as the expansion RM 65 modules, complement this manual.

1.1 USER MANUAL DESCRIPTION

Section 1, Introduction, introduces the AIM 65/4Ø Microcomputer System, explains the structure of this manual and identifies other reference documents.

Section 2, Installation and Startup, tells you how to unpack, set-up and turn-on the AIM 65/4Ø.

Section 3, Operating the AIM 65/4Ø, describes the keyboard functions and tells you how to operate the Text Editor and Debug Monitor. A simple example illustrates mnemonic operation code (op-code) entry into the Text Editor, elementary text editing, automatic op-code translation using the resident mnemonic entry function and program debugging using Debug Monitor functions.

Section 4, Debug Monitor Commands, describes in detail how each command operates along with all options and associated prompts and messages.

Section 5, Text Editor Commands, explains the operation of each command along an description of the Text Buffer operation.

Section 6, Using the I/O ROM Firmware Description, defines the structure of the I/O ROM firmware and describes input, output and general purpose subroutines available for application use.

Section 7, Using the Parallel Application Interface, describes the capabilities of the R6522 Versatile Interface Adapter (VIA) with interface examples.

Section 8, Using the RS-232-C Interface, describes how to use the RS-232C serial interface through the R6551 Asynchronous Communications Interface Adapter (ACIA).

Section 9, Using the Audio Recorder Interface, explains how to connect the AIM 65/4Ø SBC module to one or two audio cassette recorders and how to read data from a recorder and to write data to a recorder.

Section 1Ø, Using the Teletype Interface, describes how to interface to and operate with a teletype.

Section 11, AIM 65/4Ø SBC Module Description, describes the hardware operation of the SBC module.

Section 12, AIM 65/4Ø Graphics Printer Description, explains the hardware and firmware operation of the printer and associated controller.

Section 13, AIM 65/4Ø 4Ø-Character Display Description, tells how the hardware and firmware associated with the 4Ø-Character vacuum fluorescent display and controller operate.

Section 14, AIM 65/4Ø Keyboard Description, describes the keyboard operation and associated interfaces.

Section 15, Firmware Description, describes the structure of the I/O ROM, Debug Monitor and Text Editor firmware.

Section 16, Troubleshooting and Adjustments, helps you isolate and correct certain problems. Should any uncorrectable problems occur, follow instructions in this section, in the warranty card or packing slip for repair information.

Appendices A through N contain often used-information in summary form for quick and easy reference.

An assembly listing of the AIM 65/4Ø I/O and Debug Monitor/Text Editor ROMs is provided in a separate volume, 2965ØN86L. This listing includes the entire annotated source code in R65ØØ assembly language for these programs, along with the generated machine code. Design techniques and general purpose subroutines included in these programs may be used as a model for your application program.

1.2 REFERENCE DOCUMENTATION

The following Rockwell documents describe how the R65ØØ microcomputer devices and the R65ØØ CPU programming instructions operate. Other general books that explain elementary 65ØØ assembly language programming and optional AIM 65/4Ø hardware and firmware operation are also listed. These documents may be ordered directly from:

Rockwell International
P. O. Box 3669, RC55
Anaheim, Ca. 928Ø3
Attn: Sales Support Services

<u>Rockwell</u>

2965ØN3Ø	R65ØØ Microcomputer System Programming Manual (details the instruction set of the R65ØØ CPU)
2965ØN31	R65ØØ Microcomputer System Hardware Manual (describes detail operation of R65ØØ devices)
2965ØN5Ø	R65ØØ Programming Reference Card (summarizes R65ØØ CPU assembly language and machine instruction operation)
2965ØN69	AIM 65/4Ø I/O ROM Program Listing (contains the complete assembly listing of the I/O ROM computer program instructions)
2965ØN92	AIM 65/4Ø Monitor Program Listing (contains complete assembly listing of the the Monitor and Editor computer program instructions)
2965ØN93	AIM 65/4Ø Summary Card (summarizes AIM 65/4Ø switch settings, Debug Monitor and Text Editor commands and I/O ROM subroutines)

29650N94 AIM 65/40 SBC Module Wall Schematic (includes complete logic, internal routing and external interface signals for the SBC module)

Other

Scanlon, L. J., *6502 Software Design*, Howard W. Sams & Co. Inc., Indianapolis, IN, 1980

Camp, R. C., T.A. Smay, and C. J. Triska, *Microprocessor Systems Engineering*, Matrix Publishers, Inc., Portland, OR 1979

Additional available reference information is listed in Appendix M.

SECTION 2

INSTALLATION AND START-UP

2.1 INSTALLATION

2.1.1 Unpacking and Set-Up

a. Unpacking

CAUTION

Microcomputer devices are manufactured using the Metal-Oxide Semiconductor (MOS) process. The inadvertent application of high voltages may damage MOS devices installed on any of the AIM 65/4Ø subassemblies. Be sure to take the following precautions before handling or using the AIM 65/4Ø:

- Discharge any static electrical charge accumulated on your body by touching a ground connection (e.g., a grounded equipment chassis) before touching the AIM 65/4Ø. This precaution is especially important if you are working in a carpeted area or in an environment with low relative humidity.

- Make sure all test equipment, interfacing hardware, and electrical tools (e.g., soldering irons) are properly grounded before using them with the AIM 65/4Ø.

(1) Carefully remove the AIM 65/4Ø, documentation and loose equipment from the shipping container. You may wish to save the shipping container and packing material in case you need to ship or store the AIM 65/4Ø at some future date.

(2) Verify that all parts listed on the packing slip are included.

WARNING

The bottom of the printed circuit boards have components mounted through feedthrough holes that may protrude past the solder cap. These clipped leads may be sharp and could puncture skin. To avoid injury, handle the modules by their edges or place fingers between the mounted components.

(3) Remove any conductive foam or foreign material from beneath or around each module.

b. Supporting Feet Installation

(1) Attach the supporting rubber feet on the bottom of the SBC and keyboard modules if the AIM 65/40 is to be operated on a flat surface, e.g. a table or desk top rather than installed in an enclosure. Remove the protective film from pad's sticky surface before attaching it lightly to one of the approximate locations shown in Figure 2-1.

(2) After all pads are attached, set the modules down on a flat surface and press down firmly over each foot to permanently affix it to the module.

c. SBC Module Inspection

(1) Inspect the socketed components on the SBC module. If any socketed devices have loosened during shipment, reseat them by firmly and evenly pressing down on the top of the device with one hand while supporting the SBC module under the device socket with the other hand in order to prevent flexing of the printed circuit board.

(2) Make sure that the bare removeable jumper posts are not bent and touching each other.

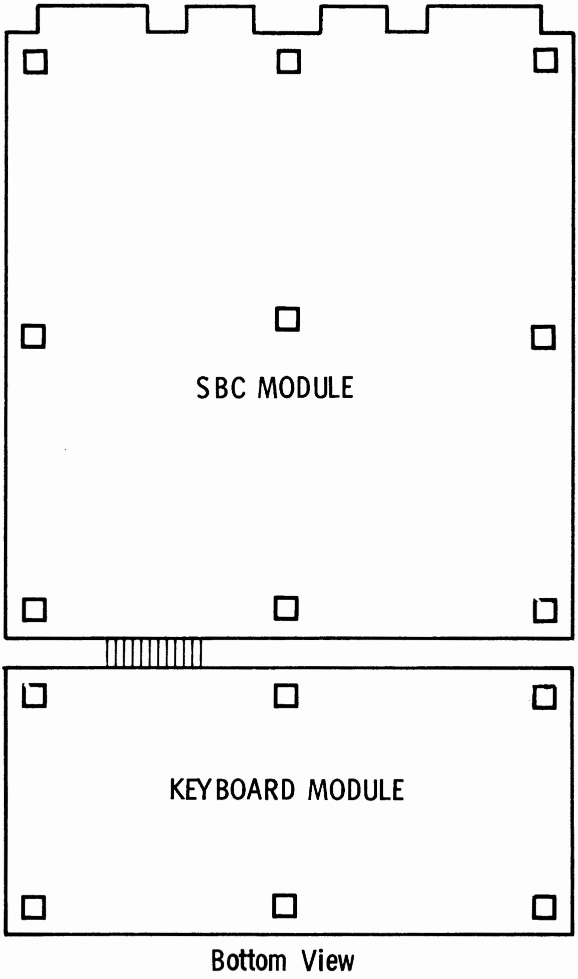


Figure 2-1. Supporting Feet Installation

d. Printer Paper Installation

(1) Remove the tear bar by lifting it straight up (see Figure 2-2).

(2) Take the roll of paper from the shipping container and separate the start of the printer paper from the roll.

(3) Tear or cut the paper from each edge at about a 45` angle to form a pointed "V" behind any adhesive or foreign material.

(4) Slide the roll of paper onto the paper holder rod and install the rod into the supporting slots. The paper should feed from under the roll toward the printer.

(5) Pull the printer head release lever toward the keyboard edge of the SBC module to release the printer thermal head from the platen.

(6) Insert the paper into the back of the printer under the platen until it can be grasped from above. Pull the paper up slightly until the entire leading edge is past the tear bar edge.

CAUTION

If printer power is on, the regulator transistors on the printer module may be hot.

(7) Push the lever on the top of the printer toward the back of the Printer Module to position the printer thermal head against the paper on the platen.

(8) Install the tear bar and tear off the excess paper.

CAUTION

Any adhesive or foreign material that comes in contact with the printer thermal elements may damage the printer.

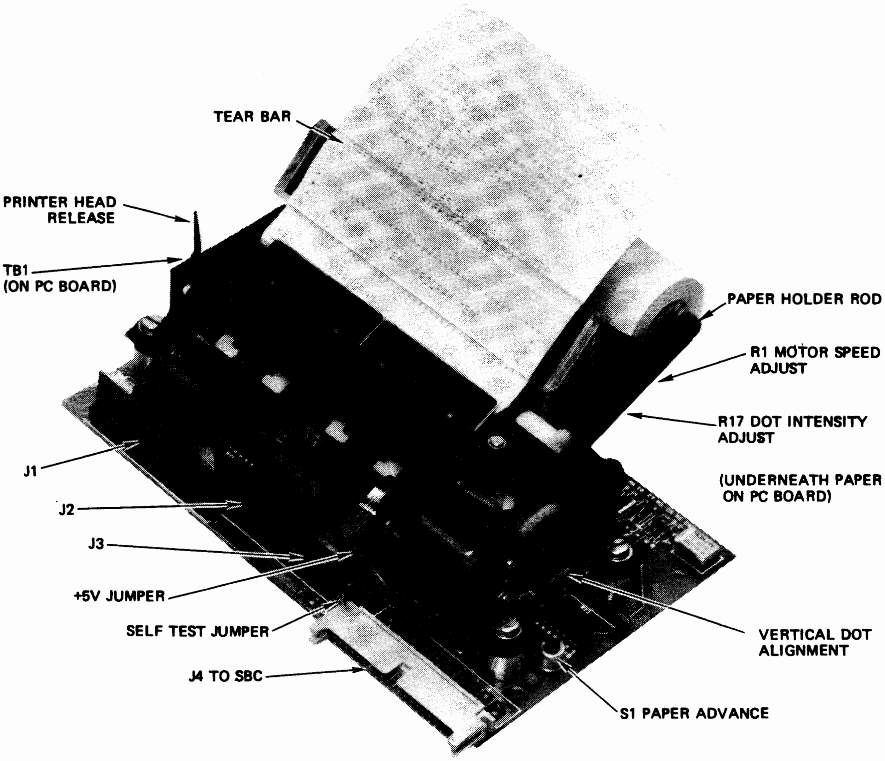


Figure 2-2. AIM 65/40 Printer Detail

2.1.2 AIM 65/4Ø Peripheral Connection

Connect the AIM 65/4Ø perpherals as follows to the SBC module. These steps connect the peripherals to the normal connectors as addressed by the AIM 65/4Ø I/O and Debug Monitor firmware. Refer to Figure 2-3 for connector location.

a. Keyboard Connection

Connect one end of the loose cable to J1 on the AIM 65/4Ø keyboard and the other end to J7 on the AIM 65/4Ø SBC module.

b. Display Connection

Verify that one end of the display interconnect cable is connected to J1 on the AIM 65/4Ø Display and the other end is connected to J6 on the AIM 65/4Ø SBC module.

c. Printer Connection

Verify that one end of the printer interconnect cable is connected to J4 on the AIM 65/4Ø Printer module and the other end to J5 on the AIM 65/4Ø SBC module.

2.1.3 Power Supply Connection

The AIM 65/4Ø system requires only two voltages to operate: +5V and +24V. The power supply connection is shown in Figure 2-4.

The +5V supplies power to the SBC, Graphics Printer and 4Ø-character Display modules. The +5V requirements are:

+5V ± 5% (4.75V to 5.25V)
Regulated
4.9A maximum

The +24V supplies power to the printer.

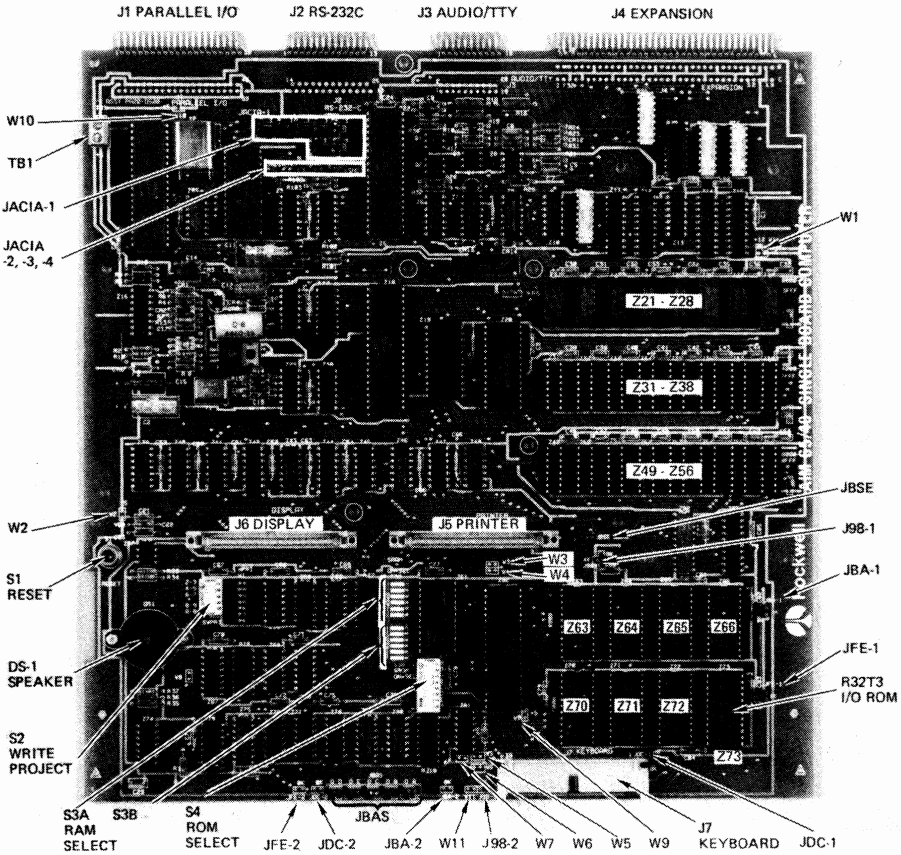


Figure 2-3. AIM 65/4Ø SBC Module Detail

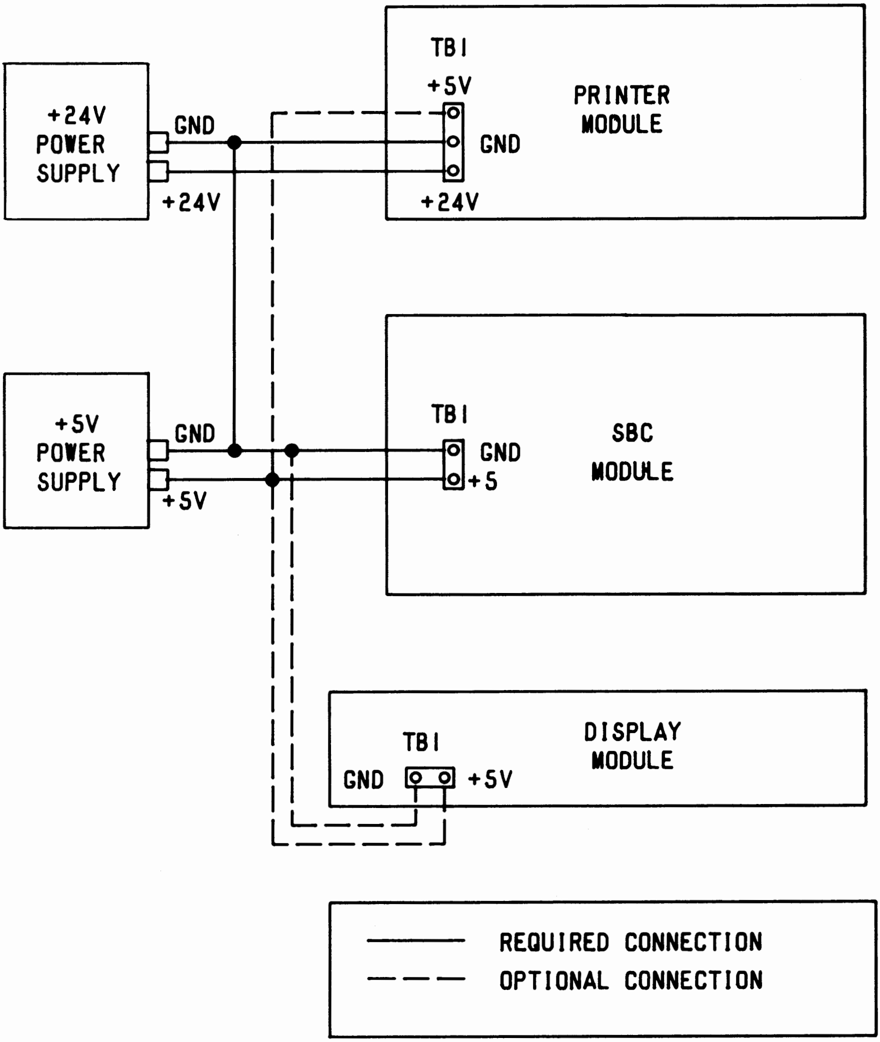


Figure 2-4. Power Supply Connection

The +24V requirements are:

+24V (21.4V to 27.6V)
Unregulated
4.ØA maximum (6.3A peak)

The +24V current may vary from less than Ø.4A, when the printer is not activated, to a peak of 6.3A during a print cycle. Note that the peak current is of short duration and therefore may not appear this high when monitoring with a slow response meter. A 4.ØA power supply is adequate.

CAUTION

Ensure that the power is <u>TURNED OFF</u> before continuing!

Prepare leads to connect the power supplies to the AIM 65/4Ø modules. Use at least 2Ø gauge wire to minimize voltage drop between the power supplies and the AIM 65/4Ø. Trim 3/16" of the insulation off the end that connects to the AIM 65/4Ø and tin the exposed wire using a soldering iron and rosin core solder.

NOTE

When connecting a power lead to the AIM 65/4Ø power terminals, first loosen the terminal post screw. Insert the tinned lead into the appropriate clamp down hole on the side of the terminal strip. (This is easily done by grasping the lead just behind the tinned area with needle nose pliers and using the pliers to insert the lead.) Then tighten the terminal post screw.

a. <u>SBC Module Power Connection</u>

(1) Connect the +5V power supply GROUND lead to TB1-1 (GND).

(2) Connect the +5V power supply +5V lead to TB1-2 (+5V).

CAUTION

Improperly connecting power leads may damage the SBC and interfacing modules connected to the power lines. Be sure to recheck the connections before proceeding.

b. Graphics Printer Power Connection

(1) Connect the GROUND lead from the power supply to the printer TB1 GND terminal.

(2) Connect the +24V lead from the power supply to the printer TB1 +24V terminal.

c. Alternate Power Connections for Printer Display

The printer and display are factory-configured to take their +5V power from the SBC module, through their respective interface cables. However, some users may wish to drive these components directly from the power supply. Here's how to do this.

To drive the printer from the power supply:

(1) Connect the power supply +5V lead to the printer TB1 +5V terminal.

(2) On the printer module, remove (cut or unsolder) jumper wire W1 (located 1/2 inch behind connector J3, labeled +5V) to disconnect +5V from connector J4.

To drive the display from the power supply:

(1) Connect the +5V GROUND lead from the power supply to the display TB1 GND terminal.

(2) Connect the +5V lead from the power supply to the display TB1 +5V terminal.

(3) On the display module, remove (cut or unsolder) jumper wire W1 to disconnect +5V from connector J1.

2.1.4 Initial Switch and Jumper Positions

The AIM 65/40 is now ready to be turned on. Before you turn power on, however, verify the following:

- o I/O and Debug Monitor ROMs are installed.
- o 16K of on-board RAM is installed in sockets Z21 through Z28.
- o No off-board expansion memory or I/O is connected.

Consult Sections 2.2, 2.3 and 2.4 for different PROM/ROM/RAM configurations and general use of the switches, removable jumpers and permanent jumpers, respectively. Figures 2-5 and 2-6 show the memory maps for firmware and memory banking respectively. Refer to Section 4.2.10 for a description of memory banking use.

a. ROM Installation

Verify that the I/O and Debug/Monitor ROMs in the following sockets:

ROM ID	ROM Function	Socket
R32T3	I/O	Z73
R32U5	Debug Monitor	Z65
R32U6	Debug Monitor	Z66

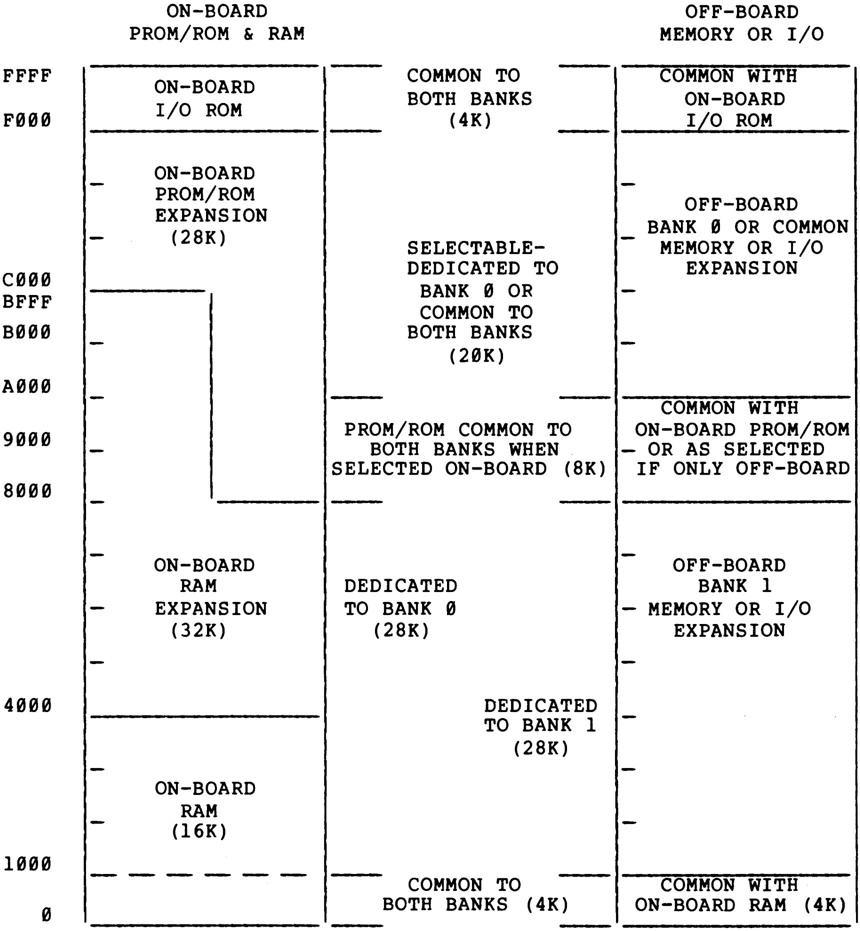
b. Switches

(1) Set the ROM Select switches to select on-board operation for the I/O and Debug Monitor ROM addresses as follows:

S4-1 open	S4-5 open
S4-2 open	S4-6 open
S4-3 closed	S4-7 open
S4-4 closed	S4-8 closed

	AIM 65/40 (Bank Ø)	RM 65 (Bank 1)
FFFF FØØØ	AIM 65/4Ø I/O ROM and on-board I/O (see Section 6.1.5)	Shared with Bank Ø
EFFF EØØØ	Optional AIM 65/4Ø Math Pack ROM	RM 65 PROM Programmer Module ROM
DFFF DØØØ	Optional AIM 65/4Ø High Level Language ROMs	User Available Off-Board
CFFF CØØØ	BASIC PL/65 FORTH	
BFFF AØØØ	AIM 65/4Ø Debug Monitor & Text Editor ROMs	
9FFF 9ØØØ	Optional AIM 65/4Ø Assembler	RM 65 CRTC Module ROM
8FFF 8ØØØ	User Available On-Board RAM	RM 65 FDC Module ROM
7FFF 7ØØØ		RM 65 IEEE-488 Module ROM
6ØØØ		User Available Off-Board
5ØØØ		
4ØØØ		
3ØØØ		
2ØØØ		
1ØØØ		
8ØØ		Shared with Bank Ø
7FF 4AØ	Reserved for Optional Languages and RM 65 Modules	
49F 2ØØ	AIM 65/4Ø System Constants & Variables	
1FF 1ØØ	Page One R65Ø2 CPU Stack	
FF Ø	Page Zero User and System Variables	

Figure 2-5. AIM 65/4Ø Firmware Memory Map



ON-BOARD PROM/ROM & RAM

OFF-BOARD MEMORY OR I/O

FFFF

F000

C000

BFFF

B000

A000

9000

8000

4000

1000

0

ON-BOARD I/O ROM

COMMON TO BOTH BANKS (4K)

COMMON WITH ON-BOARD I/O ROM

ON-BOARD PROM/ROM EXPANSION (28K)

SELECTABLE- DEDICATED TO BANK 0 OR COMMON TO BOTH BANKS (20K)

OFF-BOARD BANK 0 OR COMMON MEMORY OR I/O EXPANSION

PROM/ROM COMMON TO BOTH BANKS WHEN SELECTED ON-BOARD (8K)

COMMON WITH ON-BOARD PROM/ROM OR AS SELECTED IF ONLY OFF-BOARD

ON-BOARD RAM EXPANSION (32K)

DEDICATED TO BANK 0 (28K)

DEDICATED TO BANK 1 (28K)

OFF-BOARD BANK 1 MEMORY OR I/O EXPANSION

ON-BOARD RAM (16K)

COMMON TO BOTH BANKS (4K)

COMMON WITH ON-BOARD RAM (4K)

Figure 2-6. AIM 65/40 Memory Banking Map

Refer to Section 2.2.1 to select other configurations of ROM operation.

(2) Set RAM Select switches (S3-A1 through S3-A4) to the CLOSED (CL) position and S3-A5 through S3-B6 to the OPEN (OP) position to select on-board operation for 16K of RAM (see Section 2.2.2).

(3) Set RAM Write Protect switches S2-1 through S2-6 to the OPEN (OP) position to write-enable the lower 16K bytes ($Ø-$3FFF) of on-board RAM (See Section 2.2.3).

c. Jumpers

(1) Verify that 4K/8K ROM size jumpers JBA-1 and JFE-1 are installed in the 4K position.

(2) Verify that jumpers JBA-2 and JFE-2 are removed.

NOTE

These jumpers are factory installed and should not require changing unless they have been moved (see Section 2.3.5).

2.1.5 AIM 65/4Ø TURN-ON

a. Turn on the +5V and +24V power supplies. If they are separate, turn on the +5V supply first. If +24V is turned on first, the printer paper will continuously advance until +5V is applied. If +5V is turned on first, PRINTER DOWN will be displayed and the speaker will beep momentarily.

Shortly after power is turned on, the following message will be displayed and printed:

```
ROCKWELL AIM 65/4Ø
```

The Debug Monitor prompt, a left brace ({) in the left-most position, will then be displayed as well as a blinking cursor (all character segments illuminated) in position two.

NOTES

1. If +24V turn-on is delayed too long after +5V is applied, the printer may turn off automatically, and PRINTER DOWN will be displayed. Press PRINT to verify that the printer is working. If this happens, press the P key while the CTRL key is pressed to toggle the printer on or off. AUTO-PRINT ON or AUTO-PRINT OFF will be printed after each command.

2. If the SBC Display or Printer do not appear to operate, the power supply lines are probably incorrectly connected. Turn off the power and verify Section 2.1.3. If the display and printout still do not appear properly, refer to the troubleshooting procedures in Section 16.

b. The AIM 65/40 System is now awaiting entry of a Monitor command from the keyboard.

```
*****************************************************************
*                                                               *
*   Go directly to Section 3 and follow the description of      *
*   the fundamental operation of the system.  Refer back to     *
*   the Section 2.2 only when you need to change a switch or    *
*   jumper position on the SBC module.                          *
*                                                               *
*****************************************************************
```

2.2 SWITCHES

The AIM 65/4Ø contains three DIP switch groups that select/deselect on-board ROM, select/deselect on-board RAM and write protect RAM. This section describes these switches with tables to show functions selected by the switch positions.

2.2.1 PROM/ROM Select Switches (SROM)

Switches S4-1 through S4-8 select the on-board operation of AIM 65/4Ø PROM/ROM devices in 4K byte increments (see Table 2-1). When one of these switches is closed, the corresponding address range is selected for on-board PROM/ROM operation. When a switch is open, the corresponding address range is selected for off-board memory or I/O operation through the RM 65 Expansion Bus interface. The AIM 65/4Ø operates with all 4K ROMs, all 8K ROMs or a mix of 4K and 8K installed. A maximum of 32K bytes of PROM/ROM may be installed on-board. Be sure that no other memory or I/O (off-board or on-board) is assigned the same address area as selected for PROM/ROM.

AIM 65/4Ø peripheral addresses for keyboard, display, printer, parallel I/O and serial I/O are permanently assigned to $FF8Ø - $FFDF. This address range is always selected on-board and can not be selected off-board.

NOTE

PROM/ROM devices installed at $8ØØØ-$FFFF address must have the following information stored in the first three bytes of each 4K-byte boundary, e.g. $8ØØØ-$8ØØ2, in order to identify the program and to perform an Auto-Start initialization upon reset (see Section 6.2):

$XØØØ	$XX	Program ID
$XØØ1	$5A	Auto-Start Constant
$XØØ2	$A5	Auto-Start Constant

The program ID can be any value (Ø through $FF).

Table 2-1. PROM/ROM Select Switches (S4)

4K Byte PROM/ROM			8K Byte PROM/ROM		
Address	Socket	Switch	Address	Socket	Switch
8ØØØ - 8FFF	Z63	S4-1			
9ØØØ - 9FFF	Z64	S4-2	8ØØØ - 9FFF	Z64	S4-2
AØØØ - AFFF	Z65	S4-3			
BØØØ - BFFF	Z66	S4-4	AØØØ - BFFF	Z66	S4-4
CØØØ - CFFF	Z7Ø	S4-5			
DØØØ - DFFF	Z71	S4-6	CØØØ - DFFF	Z71	S4-6
EØØØ - EFFF	Z72	S4-7			
FØØØ - FFFF	Z73	S4-8	EØØØ - FFFF	Z73	S4-8

NOTES

1. Switch positions:
   CLOSED (CL) or ON = On-board PROM/ROM is selected.
   OPEN (OP) or OFF = On-board PROM/ROM is deselected.
2. Compatible 2K byte PROM/ROMs may be used in all 4K byte PROM/ROM sockets except Z73.

CAUTION

Use only the Z64, Z66, Z71 and Z73 slots for 8K ROMs. Otherwise, your PROM/ROM device(s) may be damaged.

2.2.2 RAM Select Switches (SRAM)

Switches S3-A1 through S3-B6 select the AIM 65/4Ø RAM in 4K byte increments. When a switch is closed, the on-board RAM is selected. When the switch is open, the on-board RAM is deselected, the off-board memory or I/O is selected and the access of data is external to the AIM 65/4Ø through the RM 65 System Expansion Bus interface. (See Table 2-2).

Table 2-2. RAM Select Switches (S3)

Address	Switch	RAM
0000 - 0FFF	S3-A1	Row 0 (Z21-Z28)
1000 - 1FFF	S3-A2	
2000 - 2FFF	S3-A3	
3000 - 3FFF	S3-A4	
4000 - 4FFF	S3-A5	Row 1 (Z31-Z38)
5000 - 5FFF	S3-A6	
6000 - 6FFF	S3-B1	
7000 - 7FFF	S3-B2	
8000 - 8FFF	S3-B3	Row 2 (Z49-Z56)
9000 - 9FFF	S3-B4	
A000 - AFFF	S3-B5	
B000 - BFFF	S3-B6	

NOTES

1. CLOSED (CL) or ON = On-board RAM is selected.
2. OPEN (OP) or OFF = On-board RAM is deselected.
3. A full row at a time must be installed for proper operation.

2.2.3 RAM Write Protect Switches (SWPRT)

Switches S2-1 through S2-5 write-protect the AIM 65/40 on-board RAM memory, in 8K byte blocks. If you want to protect a program in RAM from being inadvertantly changed or destroyed, just close the SWPRT switch for the applicable address range to be protected (See Table 2-3).

RAM addresses from 2000 to BFFF can be protected via the switches. The lower 8K (0000 - 1FFF) of RAM cannot be write protected because the system variables and R6502 CPU stack require writing into RAM.

Table 2-3. RAM Write Protect Switches (S2)

Address	Switch
2000 - 3FFF	S2-1
4000 - 5FFF	S2-2
6000 - 7FFF	S2-3
8000 - 9FFF	S2-4
A000 - BFFF	S2-5

NOTES

1. Switch Position:
   CLOSED (CL) or ON = Write Protected.
   OPEN (OP) or OFF = Not Write Protected.
2. Address 0 - 1FFF may not be write protected.

2.2.4 RESET Button

Pressing the RESET pushbutton on the SBC module causes a low $\overline{\text{RES}}$ signal to be transmitted to all interfacing devices, on-board and off-board. All interfacing devices, (R6551 ACIA, System R6522 VIA, User R6522 VIA, Keyboard R6522 VIA) and the R6502 CPU will be initialized to their reset state. Refer to the individual device description for the definition of the hardware reset operations.

The reset function is also initiated by power turn-on. The RESET key on the Keyboard is connected in parallel with the RESET button on the SBC module. Either a "cold" or "warm" RESET will be performed, as described in Section 3.1.1.

2.3 REMOVEABLE JUMPERS

The AIM 65/40 contains removeable jumpers that allow you to easily reconfigure the system for different PROM/ROM sizes, memory bank selection and RS-232-C serial port operation.

2.3.1 PROM/ROM Size Jumpers (JXX-1 and JXX-2)

The AIM 65/4Ø operates with either 4K- or 8K-byte PROM/ROM devices up to a maximum of 32K bytes on-board. PROM/ROM size jumpers configure the PROM/ROM sockets and chip select circuitry for either 4K or 8K device operation.

There are two types of PROM/ROM size jumpers, JXX-1 and JXX-2. The JFE-1, JDC-1, JBA-1 and J98-1 double-position jumpers must be set to either the 4K position (for 4K devices) or to the 8K position (for 8K devices). The JFE-2, JDC-2, JBA-2 and J98-2 single position jumpers must be installed only for 8K devices. Refer to Table 2-4 for the PROM/ROM Size Jumper identification and the corresponding address range and sockets. Install and remove the jumpers as shown in this table.

Table 2-4. PROM/ROM Size Jumpers (JXX-1 and JXX-2)

4K Byte PROM/ROM					8K Byte PROM/ROM				
		Install		Remove			Install		Install
Address	Ref	JXX-1	Pos	JXX-2	Address	Ref	JXX-1	Pos	JXX-2
8ØØØ-8FFF	Z63	J98-1	4K	J98-2					
9ØØØ-9FFF	Z64	J98-1	4K	J98-2	8ØØØ-9FFF	Z64	J98-1	8K	J98-2
AØØØ-AFFF	Z65	JBA-1	4K	JBA-2					
BØØØ-BFFF	Z66	JBA-1	4K	JBA-2	AØØØ-BFFF	Z66	JBA-1	8K	JBA-2
CØØØ-CFFF	Z7Ø	JDC-1	4K	JDC-2					
DØØØ-DFFF	Z71	JDC-1	4K	JDC-2	CØØØ-DFFF	Z71	JDC-1	8K	JDC-2
EØØØ-EFFF	Z72	JFE-1	4K	JFE-2					
FØØØ-FFFF	Z73	JFE-1	4K	JFE-2	EØØØ-FFFF	Z73	JFE-1	8K	JFE-2

NOTES

1. Compatible 2K byte PROM/ROMS may be installed in all 4K byte PROM/ROM sockets except Z73.
2. Ref = Socket reference no.

2.3.2 PROM/ROM Bank Select Jumpers (JBAS-X)

The addresses A000 to EFFF for on-board PROM/ROM are either permanently assigned to bank 0 or may be assigned to both bank 0 and bank 1, under jumper control (See Table 2-5).

The 12K bytes of on-board PROM/ROM at 8000-9FFF and F000-FFFF, when installed and selected (see Section 2.2.1), are always common to both banks. Note that the memory bank for installed and selected RAM (see Section 2.2.2) is controlled by jumper JBSE (see Section 2.3.3).

Five jumpers assign 20K bytes of on-board PROM/ROM (A000-EFFF) to just bank 0 or to both bank 0 and 1 in 4K blocks (see Table 2-5). Install these jumpers in the 0 position to select bank 0 only operation or in the C position to select bank 0 and bank 1 operation.

Table 2-5. PROM/ROM Bank Select Jumpers (JBAS-X)

		Jumper Position	
Address	Jumper	ROM Bank 0 (Dedicated)	ROM Banks 0 and 1 (Common)
A000 - AFFF	JBAS-A	0	C
B000 - BFFF	JBAS-B	0	C
C000 - CFFF	JBAS-C	0	C
D000 - DFFF	JBAS-D	0	C
E000 - EFFF	JBAS-E	0	C

NOTES

1. 8000 - 9FFF are common to both banks.
2. F000 - FFFF are common to both banks (System I/O).

2.3.3 RAM Bank Select Jumpers (JBSE)

Addresses Ø to $BFFF for on-board RAM are either permanently assigned common to both banks (Ø - $ØFFF) or may be assigned to bank Ø or to both bank Ø and 1 ($1ØØØ-$BFFF) as shown in Figure 2-6.

The 4K bytes at Ø - $ØFFF are always common to both banks so program data on page Ø (Ø - $ØØFF) and the R65Ø2 CPU stack on page 1 ($Ø1ØØ - $Ø1FF) can be used in either bank. The rest of the on-board RAM ($1ØØØ - $BFFF) may be assigned to bank Ø or to both bank Ø and 1 under jumper control.

When jumper JBSE is installed (the factory delivered configuration), the RAM is dedicated to bank Ø (see Table 2-6).

When jumper JBSE is removed, the RAM is comon to both banks Ø and 1.

$\overline{\text{BSE}}$, the bank address signal, is connected to the system R6522 VIA (Z5) port PB2 and may be driven under software control using the ZROBANK, BANKØ and BANK1 subroutines (see Section 6.5) or by directly storing into the VIA B port data register (Section 11.2.5). The $\overline{\text{BSE}}$ line may also be controlled under Monitor control using the O command to toggle the memory bank (see Section 4.2.1Ø).

Table 2-6. RAM Bank Select Jumper (JBSE)

Address	Jumper	RAM Bank Ø Only (Dedicated)	RAM Bank Ø and 1 (Common)
1ØØØ-BFFF	JBSE	Install	Remove

2.3.4 Data Terminal/Set Operation Jumpers (JACIA-1)

The seven JACIA-1 double-position jumpers configure the AIM 65/40 RS-232-C serial channel as a Data Set or as a Data Terminal (see Figure 2-6). The R6551 ACIA device provides the interface between the R6502 CPU and the RS-232-C interface circuitry (see Section 11.10.1).

The seven RS-232-C I/O signals on connector J2 that are controlled by the JACIA-1 jumpers are:

- Clear to Send (CTS)
- Received Data (RD)
- Transmitted Data (TD)
- Request to Send (RTS)
- Data Carrier Detect (DCD)
- Data Terminal Ready (DTR)
- Data Set Ready (DSR)

To configure the AIM 65/40 for Data Terminal/Data Set operation, install jumpers JACIA-1A through JACIA-1G as shown in Table 2-7:

a. For Data Terminal Operation

Place the JACIA-1 jumpers on the C (Common) and T (Terminal) pins.

b. For Data Set Operation

Place the JACIA-1 jumpers on the C (common) and S (Set) pins.

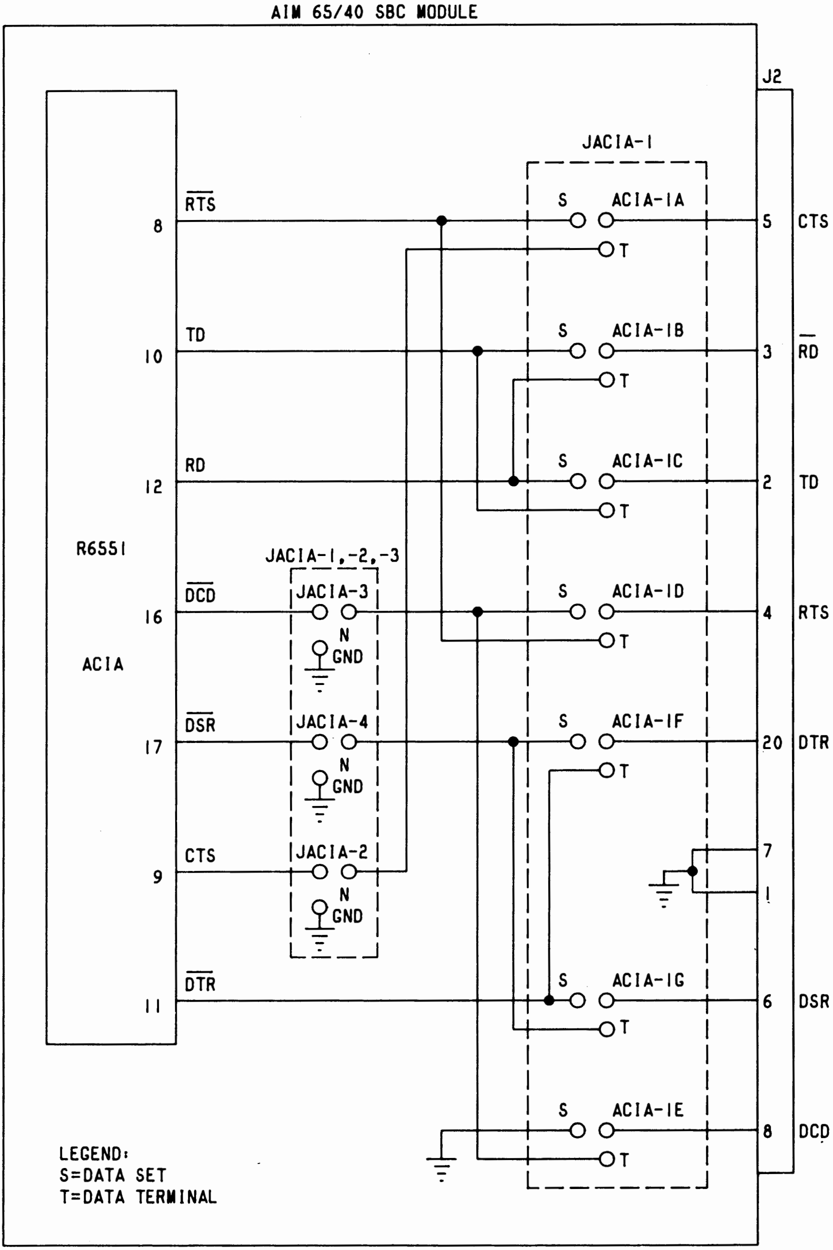


Figure 2-7. RS-232C Jumpers

Table 2-7. Data Terminal/Set Jumpers (JACIA-1)

Connector J2 Signal	Jumper	Jumper Position	
		Data Terminal Operation	Data Set Operation
CTS	JACIA-1A	T	S
$\overline{\text{RD}}$	JACIA-1B	T	S
$\overline{\text{TD}}$	JACIA-1C	T	S
RTS	JACIA-1D	T	S
DCD	JACIA-1E	T	S
DTR	JACIA-1F	T	S
DSR	JACIA-1G	T	S

2.3.5 Data Terminal Control Jumpers (JACIA-2,-3,-4)

The JACIA-2, -3, and -4 double-position jumpers allow the R6551 ACIA to operate when the user Data terminal output signals (CTS, DCD and DSR) are not supplied or polled low.

When any of the three following signals are not low active, install the JACIA-2, -3 and -4 jumpers per Table 2-8.

Table 2-8. Data Terminal Control Jumpers (JACIA-2, -3, -4)

Signal Source	Signal	Jumper	Jumper Position	
			Interface Not Supplied	Interface Supplied
Data Terminal	CTS	JACIA-2	GND	N
Data Terminal	DCD	JACIA-3	GND	N
Data Terminal	DSR	JACIA-4	GND	N

NOTE: If any of these signals are supplied, they should not be jumpered to ground (GND) for simulation.

2.4 WIRE JUMPERS

The AIM 65/4Ø SBC has provisions for installation of several wire jumpers. These semi-permanent jumpers allow you to reconfigure the Keyboard R6522 VIA control lines and to route +5V power to off-board peripherals such as AIM 65/4Ø display, printer or keyboard. Most of the wire jumpers on the SBC module are utilized in pairs. It is assumed these jumpers will be permanently connected in one position at any specific installation. See to Table 2-9 for all of the jumper wire numbers, connection information and functional use.

2.4.1 RAM VRA (W1) and RAM VBB (W2)

Jumpers W1 and W2 are factory installed to either connect -12V to VRA and -5V to VBB if triple voltage RAM devices are installed or to connect +5V to VRA and an open to VBB if single voltage RAM devices are installed.

Jumper W1 connects RAM VRA to either -12V or to +5V. Jumper W2 connects RAM VBB to either -5V or to an open.

Jumpers W1 and W2 may be user modified to operate with either type of RAM devices, however either all triple-voltage or all single voltage RAMs must be installed, i.e. triple-voltage and single-voltage RAM may not be installed simultaneously.

To operate with triple-voltage RAMs, install jumper W1 in the -12V position and install jumper W2.

To operate with single-voltage RAMs, install jumper W1 in the +5V position and remove jumper W2.

Table 2-9. Wire Jumpers (WX)

Jumper	Function	Reference
W1	+12V Position: Connects -12V to RAM VRA +5V Position: Connects +5V to RAM VRA.	Section 2.4.1
W2	Installed: Connects -5V to RAM VBB. Removed: Disconnects -5V from RAM VBB.	
W3	Installed: Connects PAPER FEED to the Keyboard to the printer. Removed: Disconnects PAPER FEED from the Keyboard to the printer.	Section 2.4.2
W4	Installed: Connects the Keyboard key depression signal to the Keyboard VIA port CA1. Removed: Disconnects the Keyboard key depression signal from the Keyboard VIA port CA1.	
W5	Installed: Connects Keyboard VIA port CB2 to the speaker circuit. Removed: Disconnects Keyboard VIA port CB2 from the speaker circuit.	Section 2.4.3
W6	Installed: Connects the Keyboard Connector J7 pin 1 to the RESET circuit. Removed: Disconnects the Keyboard Connector J7 pin 1 from the RESET circuit.	
W7	Installed: Connects the Keyboard Connector J7 pin 3 to the Keyboard VIA port CB1. Removed: Disconnects the Keyboard Connector J7 pin 3 from the Keyboard VIA port CB1.	Section 2.4.4 Section 2.4.4
W8	Installed: Connects the Keyboard Connector J7 pin 3 to the CPU $\overline{\text{NMI}}$ input. Removed: Disconnects the Keyboard Connector J7 pin 3 from the CPU $\overline{\text{NMI}}$ input.	

Table 2-9. Wire Jumpers (WX) (Continued)

Jumper	Function	Reference
W9	Installed: Connects +5V to Keyboard Connector J7 pins 2 and 4Ø. Removed: Disconnects +5V from Keyboard Connector J7 pins 2 and 4Ø.	Section 2.4.5
W1Ø	Installed: Connects +5V to Parallel Application Connector J1-pins 2 and 4Ø. Removed: Disconnects +5V from Parallel Application Connector J1-pins 2 and 4Ø.	Section 2.4.6
W11	Installed: Allows single step operation up to address $8FFF. Removed: Allows single step operation up to address $9FFF.	Section 2.4.7

2.4.2 Paper Feed (W3) and Key Depressed (W4)

Factory installed jumper W3 connects the PAPER FEED signal from Connector J7 pin 39 to the PAPER FEED signals on connector J5 pin 15 and Connector J6 pin 15. As configured, the PAPER FEED key on the keyboard can issue PAPER FEED to the Graphics Printer connected to J5 and to an optional printer connected to J6. The 4Ø-character Display normally connected to J6 does not use the PAPER FEED signal.

Factory installed jumper W4 routes the key depressed signal from Z81-8 to the Keyboard VIA port CA1. As configured, the Keyboard VIA issues an Interrupt Request ($\overline{\text{IRQ4}}$) whenever it detects that a key has been depressed.

Jumpers W3 and W4 can be user modified to route connector J7 pin 39 to the Keyboard VIA port CA1 for user defined operation so this modification will disconnect the key depressed signal from the Keyboard VIA and the PAPER FEED signal from the Keyboard. If this modification is made, the AIM 65/4Ø I/O ROM firmware will not detect depression of a key on the keyboard.

If the keyboard is used in this configuration, a user provided keyboard scan function must be provided. Also in this configuration, the PAPER FEED key on the keyboard will be disabled. PAPER FEED to the printer can then only be issued under program control by sending out repetitive line feeds or by depressing the PAPER FEED pushbutton on the printer.

To connect J7-39 to the Keyboard VIA CA1 port, perform the following:

a. Remove W3 and W4 jumpers.

b. Add a jumper from W3 (connector J7-39 side) to W4 (Z62-40 side).

2.4.3 Speaker Driver (W5) and Keyboard RESET (W6)

Factory installed Jumper W5 routes the Keyboard R6522 VIA (Z62) port CB2 to the on-board speaker circuit to drive the speaker.

Factory installed jumper W6 routes the RESET key signal (RESET SW) from connector J7 pin 1 to pin 2 of the Reset Timer (Z57). As configured, this connects the RESET key on the keyboard to the on-board Reset circuit, in parallel with the on-board RESET pushbutton.

Jumpers W5 and W6 can be user modified to route connector J7 pin 1 to the Keyboard R6522 VIA port CB1 for user defined operations. This modification of course, disconnects the speaker circuit from the Keyboard VIA and also disconnects the external RESET key signal from the on-board Reset circuit.

To connect J7-1 to the Keyboard VIA CB2 port, perform the following:

a. Remove W5 and W6 jumpers.

b. Add a jumper from W6 (connector J7-1 side) to W5 (Z62-19 side).

2.4.4 ATTN Key (W7) and Keyboard VIA CB1 (W8)

Factory installed jumper W8 connects pin 3 of the Keyboard Connector J7 to the Non-Maskable Interrupt ($\overline{\text{NMI}}$) input to the R6502 CPU. J7-3 is normally connected to the Keyboard $\overline{\text{ATTN SW}}$ signal originating from the ATTN key. In this configuration, pressing the ATTN key causes a non-maskable interrupt in the CPU processing. The AIM 65/40 Monitor firmware default linkage and $\overline{\text{NMI}}$ interrupt processing causes the AIM 65/40 Monitor command level to be entered thus allowing Monitor or user function interruption without requiring a RESET.

Jumper W7 allows pin 3 of connector J7 to be connected to the Keyboard VIA port CB1, for user defined operation. This jumper is not factory installed since the ATTN key signal is routed directly to the CPU $\overline{\text{NMI}}$ input.

Jumpers W7 and W8 can be user modified to connect J7-3 to port CB1 of the Keyboard VIA rather than the CPU. If this is done, pressing of the ATTN key will not cause an NMI interrupt. User provided software could, however, generate an Interrupt Request ($\overline{\text{IRQ4}}$) upon detecting the key. Normally, this modification would be done to route a user defined signal to the CB1 port. To connect J7-3 to the Keyboard VIA port CB1, remove jumper W8 and add jumper W7.

2.4.5 +5V to Connector J7 (W9)

The Jumper W9 routes the on-board +5V logic Voltage to the Keyboard Connector J7, pins 2 and 40 for interface equipment use. This jumper is not factory installed. To install, add a wire jumper.

WARNING

Maximum +5V current through J7 should not exceed 200 mA per pin.

2.4.6 +5V to Connector J1 (W1Ø)

Jumper W1Ø routes the on-board +5V logic voltage to the Parallel I/O J1 Connector, pins 2 and 4Ø for use by the interface eauipment. This jumper is not factory installed. To install, add a wire jumper.

WARNING

> Maximum +5V current through J1 should not exceed 2ØØmA per pin.

2.4.7 Single Step Limit (W11)

Jumper W11 assigns the upper address limit for single step program execution. When removed (the factory configuration), single step operation is enabled up to address $8FFF. When installed, single step execution will extend up to $9FFF.

2.5 PROM/ROM INSTALLATION

2K-, 4K- and 8K-byte PROM/ROM devices that are compatible with the pin assignments shown in Table 2-1Ø may be installed on-board. The installation procedure is:

a. Turn power off.

b. Install the PROM/ROM devices in sockets Z63-Z73 according to the address ranges shown in Table 2-4.

c. Install jumper(s) JXX-1 per Table 2-4.

d. Install jumper(s) JXX-2 per Table 2-4.

e. Set on-board PROM/ROM select switches S4-X per Table 2-1.

f. Install on-board PROM/ROM bank select jumpers JBAS-X per Table 2-5.

Table 2-1Ø. PROM/ROM Pin Assignments

Pin No.	Signal Name	Signal Description
1	A7	Address Input A7
2	A6	Address Input A6
3	A5	Address Input A5
4	A4	Address Input A4
5	A3	Address Input A3
6	A2	Address Input A2
7	A1	Address Input A1
8	AØ	Address Input AØ
9	DØ	Data Out DØ
1Ø	D1	Data Out D1
11	D2	Data Out D2
12	GND	VSS Ground
13	D3	Data Out D3
14	D4	Data Out D4
15	D5	Data Out D5
16	D6	Data Out D6
17	D7	Data Out D7
18	A11	Address Input A11
19	A1Ø	Address Input A1Ø
2Ø	$\overline{\text{OE}}$	Chip Select (deselected when high)
21	A12 or +5V	Address Input A12 or +5V
22	A9	Address Input A9
23	A8	Address Input A8
24	+5V	VCC Power

2.6 RAM INSTALLATION

Triple or single voltage RAM devices that are compatible with the pin assignments listed in Table 2-11 may be installed on-board in sockets.

a. Turn power off.

b. Install the RAM devices in sockets Z21-Z28, Z31-Z38, or in Z49-Z56 according to the address ranges shown in Table 2-2.

c. Install wire jumpers W1 and W2 depending on RAM type per Table 2-9.

d. Set on-board RAM select switches S3-XX per Table 2-2.

e. Set write protect switches S2-X per Table 2-3.

f. Install or remove the RAM bank select jumper JBSE per Table 2-6.

2.7 INTERRUPT REQUEST PRIORITY OPERATION

On-board circuitry and software control assign up to seven $\overline{\text{IRQ}}$ interrupts to prioritized $\overline{\text{IRQ}}$ lines to the R65Ø2 CPU and inhibit interrupts below a desired level.

2.7.1 $\overline{\text{IRQ}}$ Priority Routing Header (HPRI)

The 14-pin $\overline{\text{IRQ}}$ Routing Header connects the $\overline{\text{IRQ}}$ source lines to prioritized $\overline{\text{IRQ}}$ enable gates. It plugs into socket H1 to route the six interrupt request ($\overline{\text{IRQ}}$) lines from the on-board peripheral, RM 65 expansion bus interface and write protect circuitry to prioritized $\overline{\text{IRQ}}$ level lines $\overline{\text{IRQ7}}$ to $\overline{\text{IRQ1}}$. The $\overline{\text{IRQ7}}$ (highest priority) to $\overline{\text{IRQ1}}$ (lowest priority) lines are connected to the R65Ø2 CPU $\overline{\text{IRQ}}$ input line through the interrupt request priority circuit (see Section 11.6) which inhibits $\overline{\text{IRQ}}$ signals below a selected level from reaching the CPU.

Table 2-11. RAM Pin Assignments

Pin No.	MM529Ø (Tri-Voltage) Signal	NMC5295 (Single-Voltage) Signal	Pin Signal Name
1	VBB (-5V)	NC	Supply Voltage
2	DI	DI	Data Input
3	$\overline{\text{WE}}$	$\overline{\text{WE}}$	Write Enable
4	$\overline{\text{RAS}}$	$\overline{\text{RAS}}$	Row Address Strobe
5	AØ	AØ	Address Input AØ
6	A2	A2	Address Input A2
7	A1	A1	Address Input A1
8	VDD (+12V)	VCC (+5V)	Supply Voltage
9	VCC (+5V)	NC	Supply Voltage
1Ø	A5	A5	Address Input A5
11	A4	A4	Address Input A4
12	A3	A3	Address Input A3
13	A6	A6	Address Input A6
14	DO	DO	Data Output
15	$\overline{\text{CAS}}$	$\overline{\text{CAS}}$	Column Address Strobe
16	VSS	VSS	Ground (GND)

NOTES

1. The designated National RAM parts or equivalent may be used.
2. Tri-voltage and single-voltage RAM devices may not be mixed.
3. The RAM must be added a physical row (16K bytes) at a time.

Table 2-12 lists the input signals to the header (pins 1-7) and the output signals (pins 4-14) with their corresponding prioritized IRQ lines. Note that the factory installed header routes the input signals straight across to the opposite pins. This configuration assigns the IRQ from the RM 65 Expansion Bus interface (see Section 11.11) to the highest interrupt priority (IRQ7). The write protect interrupt (IRQWP) is assigned the lowest IRQ priority (IRQ1).

You may re-assign the input IRQ lines to meet specific application requirements. For example, if the User R6522 VIA interrupt (IRQU) is assigned highest priority in your installation, you will wire it to IRQ7 (pin 13) rather than IRQ5 (pin 8).

2.7.2 IRQ Priority Level Mask

A 6-bit mask written out on the data lines to the mask latch (address $FF80) in the IRQ Request Priority circuit controls the level below which IRQ interrupts are inhibited (see Table 2-13). The I/O ROM initializes this mask to $00 to allow all IRQ interrupts (see Section 6.2.1).

You can change the mask by writing a value of $01 to $20 to address $FF80. Refer to Table 2-13 to determine which value you need based on your application needs, i.e. what IRQ lines are used and how they are routed thorugh the HPRI header.

Table 2-12. Interrupt Request Header Connections (H1)

Input Signal	I/P Pin	Factory Routing	O/P Pin	Output Signal	Interrupt Level	Z59 Latch Bit
User ACIA ($\overline{\text{IRQA}}$)	1	<———>	14	$\overline{\text{IRQ3}}$	5	Q2
RM 65 Bus ($\overline{\text{BIRQ}}$)	2	<———>	13	$\overline{\text{IRQ7}}$	1	None
Write Prot ($\overline{\text{IRQWP}}$)	3	<———>	12	$\overline{\text{IRQ1}}$	7	Q0
Keyboard VIA ($\overline{\text{IRQK}}$)	4	<———>	11	$\overline{\text{IRQ4}}$	4	Q3
No connection	5	<———>	10	$\overline{\text{IRQ2}}$	6	Q1
System VIA ($\overline{\text{IRQS}}$)	6	<———>	9	$\overline{\text{IRQ6}}$	2	Q5
User VIA ($\overline{\text{IRQU}}$)	7	<———>	8	$\overline{\text{IRQ5}}$	3	Q4

Table 2-13. Interrupt Request Priority Mask

Mask Pattern	Priority Level	Interrupts Allowed	Interrupts Inhibited
00	1	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ1}}$	None
01	2	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ2}}$	$\overline{\text{IRQ1}}$
02	3	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ3}}$	$\overline{\text{IRQ1}}$ - $\overline{\text{IRQ2}}$
04	4	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ4}}$	$\overline{\text{IRQ1}}$ - $\overline{\text{IRQ3}}$
08	5	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ5}}$	$\overline{\text{IRQ1}}$ - $\overline{\text{IRQ4}}$
10	6	$\overline{\text{IRQ7}}$ - $\overline{\text{IRQ6}}$	$\overline{\text{IRQ1}}$ - $\overline{\text{IRQ5}}$
20	7	$\overline{\text{IRQ7}}$	$\overline{\text{IRQ1}}$ - $\overline{\text{IRQ2}}$

NOTE

The mask pattern is to be written to the $\overline{\text{IRQ}}$ Priority Latch PRIRTY ($FF80), which is write only, i.e. it cannot be read.

SECTION 3

OPERATING THE AIM 65/40 SYSTEM

The section describes the general operation of the AIM 65/40 system, hardware (keyboard, printer, display and SBC module) and firmware (Debug Monitor/Text Editor and the I/O ROM).

While not all the commands and options are illustrated, the purpose and operating principle of most commands will become evident as you try them. As each command is encountered, you may want to refer to the applicable Section 4 or 5 detail description for more information.

3.1 AIM 65/40 KEYBOARD FUNCTIONS

Operation of the AIM 65/40 system is simple and straightforward. The system prompts you with symbols, abbreviations or words to indicate when a command, code or data is to be entered. The system is therefore quite interactive in a command/response manner. When a command is entered, the system performs the associated function and informs you whether the command is successfully completed or an error condition is encountered.

As mentioned in Section 2.1, the Monitor command level is active when the Monitor prompt is displayed in the left-most position and the cursor is blinking in position two. Whatever else is displayed is left over from the previous command and will disappear when a new Monitor command is entered.

Before using any Monitor commands, however, read and try the following to become familiar with the keyboard operation as well with the controls on the printer and SBC modules.

3.1.1 Reset Key

a. Warm RESET

During the course of AIM 65/4Ø system operation or program development, you may do something that hangs up the system, auses an unexpected result or uses an unknown operation. In most cases, you can recover and return control to the Monitor by just pressing RESET on the keyboard (or on the SBC module).

Press RESET and hold it down and note that the display will blank. Release the RESET key and observe display of the warm RESET message after a short delay.

```
AIM 65/4Ø
```

The message is also printed if the printer is turned on (see Section 3.1.12). This delay allows time for the display peripheral to initialize after RESET is released.

This is called a "warm" RESET because not all system variables are initialized. The I/O ROM constants and variables described in Section 6.1 are system variables. System variables that may be altered by you are sometimes called user variables. User variables are not initialized by the warm reset in order to allow system recovery without requiring you to reload them with your values. If, however, any variables which affect system operation are inadvertently altered to an invalid value, the AIM 65/4Ø system may not cover properly with a warm RESET thus necessitating a cold RESET.

b. Cold RESET

A cold RESET initializes all system variables necessary for AIM 65/4Ø system operation. Default values are stored in user variables that you may later alter (see Sections 6.1.2

and 6.1.3). If the system does not appear to operate correctly, you can always recover by performing a cold RESET.

Use the CTRL key in conjunction with the RESET key to command a cold RESET. Press RESET now but before you release it, press and hold the CTRL key down. Now that RESET is released, release the CTRL key and note display of the cold reset message:

```
ROCKWELL AIM 65/40
```

3.1.2 ATTN Key

The ATTN key allows a user-function to immediately be performed (i.e. attention is given to that function) by the R6502 CPU upon command from the keyboard without going through a cold or warm RESET. Pressing the ATTN key normally causes a Non-Maskable Interrupt (NMI) to occur in the R6502 CPU (see Section 2.4.4). User-provided vectoring in the I/O ROM NMI processing (see Section 6.1.1) links to the application processing.

Unless the ATTN key signal routing on the SBC module is altered or a user function changes the NMI interrupt linkage, the Monitor uses ATTN key to interrupt the current operation and return control to the Monitor command level.

The Monitor processes (the NMI interrupt the NMI vector is altered, see Section 6.1.1) as part of the single step execution function. The speaker beeps to indicate ATTN key depression until the key is released. Upon key release, the current displayed line is printed. Next, a single instruction is disassembled, displayed/printed, and executed then control is returned to the Monitor command level. The exact instruction displayed is not important, it merely indicates ATTN key depression. One of the instructions in the I/O ROM will probably be displayed. The Monitor prompt and cursor are then displayed.

If system variables have been altered to invalid values, a warm or cold RESET may be required to return control to the Monitor.

3.1.3 ESC Key

Pressing the ESC (escape) key causes the Monitor or Editor to terminate most functions currently in process and return to the Monitor or Editor command level, respectively. This key is acted upon only at certain points, e.g. at the end of a displayed line in a list function, and may not be acted at all in some functions, e.g. dumping to audio tape.

When ESC is pressed, any partially completed command that is displayed will be printed (if the Auto-Print is on) along with

```
(ESC)
```

3.1.4 PRINT Key

Pressing the PRINT key causes the contents of the entire display line (up to 8Ø characters), to be printed. This is especially handy when the Auto-Print is off (See Section 3.1.11) if you want to see a portion of text that is beyond the display boundary. Use it whenever you want.

3.1.5 PAPER FEED Key

The PAPER FEED Key advances the printer paper as long as it is held down. The paper feed button on the printer (to the right of the printer motor) can also be used. Try both of them.

3.1.6 RETURN Key

The RETURN (carriage return) key terminates command or data entry into the system. It tells the Monitor or Editor to act on a command or data (either just entered or a default value) as defined for specific command. In the Editor data input mode, it also terminates a line of entered text.

During some functions which output multiple lines, e.g. list from the Text Editor or disassemble memory, RETURN is used to command one line of output.

3.1.7 SPACE Key

The SPACE key inputs a SPACE character during text entry and is also used like RETURN to indicate completion of command or data entry in some commands.

At the Monitor command level, the SPACE key commands display of the next eight memory locations (see Section 4.4.2) while at the Editor command level it commands display of the current line (see section 5.3.12).

The SPACE key is also used in some Monitor and Editor functions to indicate continuous output rather than a single line or specified line counts, e.g. list form the Editor or disassemble memory.

3.1.8 DEL Key

The DEL (delete) key is used to delete erroneous characters by back spacing over them. The DEL key can only be used to correct an entry in certain modes of operation, e.g. text entry or address entry.

To delete or change data after it has been entered (after pressing RETURN), escape from the command back to the Monitor (or Editor), re-enter the command and then correct the data.

3.1.9 SHIFT Key

Either of the two SHIFT keys may be used to enter an upper case letter or the character indicated on the top half of a key cap. The SHIFT key must be held down while the other key is typed.

3.1.1Ø ALL CAPS

The ALL CAPS key is a locking key which allows all letters from the keyboard to be entered in upper case rather than lower case. Upper/lower case distinction is important in text entry, however the Monitor generally ignores the difference in command entry and responds to either upper or lower case as upper case.

When the ALL CAPS key is in the up position, letters are entered in lower case. Press the ALL CAPS key to lock it in the down position to enter upper case letters. Release the key by pressing it again.

3.1.11 CTRL Key

The CTRL (Control) key allows additional control over AIM 65/4Ø system operation. Selected critical functions requiring increased user awareness are often commanded by simultaneous depression of the CTRL key and another key to minimize inadvertent initiation. Some of these commands are:

CTRL RESET	Cold RESET
CTRL P	Toggle Auto-Print On/Off
CTRL Z	Direct Peripheral Command Mode

You have already tried CTRL RESET to command a cold RESET. Now look at the other two commands.

3.1.12 CTRL P - Toggle Auto-Print On/Off.

As previously mentioned, pressing P while CTRL is depressed toggles the Auto-Print On/Off state. This turns the printer on or off relative to system commands and responses that are displayed and normally printed.

When Auto-Print is on, all displayed prompts, commands and messages are automatically printed to provide a complete audit trail of AIM 65/4Ø operation. When Auto-Print is off, the commands and responses are not printed unless specifically directed to the printer. The state of the Auto-Print is printed upon toggle command:

```
        AUTO-PRINT ON
or
        AUTO-PRINT OFF
```

Try it by pressing CTRL and P until AUTO-PRINT OFF is printed. Press the SPACE bar a few times and notice that the displayed data is not printed. Turn the printer back on by again pressing CTRL and P, then press SPACE once more. Refer to Section 4.4.2 to see what the Monitor SPACE command is doing.

3.1.13 CTRL Z - Direct Peripheral Command

Pressing the Z key while the CTRL key is depressed commands the Direct Peripheral Control mode. In this mode, all characters entered on the keyboard are directly transferred to both the printer and display characters. These peripherals act upon the characters as described in Section 12 and 13, respectively.

The normal Monitor processing is by-passed in this mode so use it cautiously. Since the Monitor is not active, you must press ATTN, RESET or CTRL RESET to return to the Monitor.

Two specific CTRL Z commands are mentioned here because their special interest in the AIM 65/4Ø operation. they are the peripheral self test command and the AIM 65/4Ø RAM test command. Section 3.2 discusses both of these commands.

3.1.14 Number Keys

The number keys (Ø-9) enter numbers in text and data entry modes and are also used to command major and special functions. Keys 1 and 2 toggle recorder remote control lines in the Monitor and Editor (see Sections 4.9.4 and 5.6, respectively) while keys 3 and 4 command verify tape checksum and toggle breakpoints on/off functions in the Monitor (see Sections 4.9.5 and 4.7.3, respectively).

Pressing the numbered keys during listing of text from the Editor (see Section 5.3.9) or disassembling memory (see Section 4.5.2) in the Monitor speeds up or slows down the output. This is useful when scanning data using the display while the printer is off.

3.1.15 Function Keys

The function keys (F1 - F8) can be assigned to command user functions from the Monitor or Editor (see Sections 4.1Ø.1 and 5.7). A user vector associated with each key points to the starting address of each user provided subroutine.

These user vectors are initialized to default values during cold RESET to cause return to the Monitor command level when a function key is pressed. Warm RESET does not alter the values so pressing a function key now will cause a "?" to be displayed, the speaker to beep and the function key number to be displayed as the command.

3.2 SELF TEST COMMANDS

Two types of self test functions are provided in the AIM 65/4Ø system:

- Peripheral Self Test
- SBC Module RAM Test

Both of these functions are commanded from the Direct Peripheral Command mode (CTRL Z, see Section 3.1.13).

3.2.1 Peripheral Self Test

The printer and display peripherals enter self-test upon receipt of an ESC T command sequence (see Sections 12.2.1 and 13.2.1, respectively.

Since printer and display are intelligent peripherals, each unit performs its self test independently and simultaneously.

Rapidly press and release the Z key while the CTRL key is pressed. Release the CTRL key. Ensure the ALL CAPS key is depressed (locked down). Press and release the ESC key. Now press and release the T key.

The printer prints the RAM and ROM test results:

```
RAM OK
ROM=8544
TEST COUNT=00
```

then prints the character set and stops. If left in the test mode, the printer test will repeat once every hour and will increment the printed test count.

NOTE

Auto-Print must be on (see Section 3.1.12) or the AIM 65/40 SBC will not send the self test command to the printer.

The display reports the RAM and ROM test results:

```
RAM OK
ROM=D57D
```

then displays the cursor in each character position from left to right. The periods associated with each character are all displayed as the cursor is moved through each position. The cursor is then displayed in all positions. The character set is then displayed one line at a line bracketed by cursor characters.

Press RESET to return control to the Monitor.

3.2.2 SBC Module RAM Test

A special case for the Direct Peripheral Control mode is the SBC Module RAM test. When another CTRL Z is received while in this mode (before peripheral test is commanded), SBC RAM test is entered. The ARE YOU SURE? prompt is displayed to prevent accidental initiation of this test. You can also specify the upper limit of the memory to prevent running the test on uninstalled memory or to prevent storing of the RAM test pattern in upper RAM.

WARNING

> The RAM test stores test patterns in all memory locations tested. Be sure to save any required program or data on permanent media before commanding the test.

The test operates in four passes. Pass one writes a bit pattern into each byte. The page number is stored in the first byte of the first page. This value is incremented by one and stored in the next byte. This continues for each page for all of the RAM being tested, e.g.:

address	pattern
Ø2ØØ	Ø2
Ø2Ø1	Ø3
.	.
.	.
.	.
Ø2FF	Ø1
Ø3ØØ	Ø3
Ø3Ø1	Ø4
.	.
.	.
.	.
Ø3FF	Ø2

Pass two reads each byte and compares it to the expected value. If it does not agree, the number 4 is displayed and printed along with the address of the detected error e.g.:

```
4  4ØØØ                (Pass 2 error at address $4ØØØ)
```

Pass three writes the complement of the pattern into the same bytes.

Pass four compares the pass three written pattern with data read from memory. If an error is detected during this pass, the number 1 and the failed address is displayed and printed, e.g.

```
        1  4000                    (Pass 4 error at address $4000)
```

After these four passes are complete, the cycle number is displayed and the test starts over. The test continues until RESET is pressed. A cold reset is then automatically performed.

Go ahead and try it

```
        CTRL Z
        CTRL Z
        ARE YOU SURE?  TO= XXXX   (Enter 4000, 8000 or C000)
          .
          .
          .
        RESET
```

SECTION 4

AIM 65/4Ø DEBUG MONITOR DESCRIPTION

The AIM 65/4Ø Debug Monitor (commonly just called Monitor) controls the AIM 65/4Ø system operation in a program development mode of operation. The Monitor is a computer program that contains powerful debug facilities to speed assembly language program checkout as well as to support program development using higher level languages.

The Monitor can be used to load, verify and dump executable object code, establish breakpoints, execute application programs under single step or run control, trace instructions and registers in single step execution mode, and to examine and alter both memory and register values. Function key linkage allows easily initiation of application programs.

The Monitor functions are invoked by single keystroke commands. Table 4-1 summarizes the commands by functional grouping. Many commands issue subprompts to request further information entry before execution. Upon completion of a Monitor function, control returns to the Monitor command level, to accept entry of another command.

The Monitor is contained in two 4K-byte R2332 ROMs located at $AØØØ-$BFFF (see the Memory map in Figure 2-4) and installed in sockets Z65 and Z66 (see the installation procedure in Section 2.1.4). The software structure of the Monitor is described in Section 15.1.

Table 4-1. AIM 65/40 Debug Monitor Commands

Category	Command	Function
Control Commands	CTRL RESET RESET ATTN ESC E T C + & O CTRL Z CTRL C CTRL N @	Enter and Initialize the Monitor Enter Monitor Non-Maskable Interrupt Escape to Monitor Command Level Initialize Text Buffer and Enter Editor Re-enter Editor Recover Text Buffer and Enter Editor Repeat Last Command Execute Command String Toggle Memory Bank Direct Peripheral Control Clear Display and Home Cursor Home Cursor Enter Data Output Rate
Display/Alter Registers	R A P S X Y *	Display Register Contents Display/Alter Accumulator Display/Alter Processor Status Display/Alter Stack Pointer Display/Alter X Register Display/Alter Y Register Display/Alter Program Counter
Display/Alter Memory	M SPACE - /	Display Selected Memory Contents Display Higher Memory Contents Display Lower Memory Contents Alter Memory Contents
Enter/Disassemble Instructions/ Symbols	I K ;	Enter Mnemonic Instruction Disassemble Memory Enter Symbol Value
Execution/Trace	G Z J H V	Execute Program Toggle Step Trace On/Off Display Register Header Display Jump and Branch History Toggle Symbol Table On/Off
Breakpoints	? # 4 B	Display Breakpoints Clear Breakpoints Toggle Breakpoint Enable On/Off Set Breakpoint
Load/Dump Memory	F D L	Verify Object Code Dump Object Code Load Object Code

Table 4-1. AIM 65/40 Debug Monitor Commands (Continued)

Category	Command	Function
Peripheral Control	PRINT	Print Display Contents
	CTRL P	Toggle Auto-Print On/Off
	1	Toggle Recorder 1 Control On/Off
	2	Toggle Recorder 2 Control On/Off
	3	Verify Tape Checksum
User Function Keys	F1	Enter Function 1
	F2	Enter Function 2
	F3	Enter Function 3
	F4	Enter Function 4
	F5	Enter Function 5
	F6	Enter Function 6
	F7	Enter Function 7
	F8	Enter Function 8

4.1 AIM 65/4Ø DEBUG MONITOR FEATURES

The features of the Monitor include:

- Display and alter memory - Any memory location may be displayed and altered.

- Display and alter any register - Any of the six registers may be displayed and altered.

- Trace of the last 16 jump or branch (branch taken) instructions executed - The single step mode maintains a list of the last 16 breaks in sequential instruction execution where the next address was greater than three bytes away from the current address.

- Eight conditional software breakpoints - Conditional software breakpoints work in conjunction with both the single step and run modes to stop the processor prior to executing an instruction at an address which corresponds to an enabled breakpoint. Software breakpoint ability can be disabled without destroying the breakpoint information.

- Symbolic disassembly of current location - When a breakpoint is encountered, the system displays a symbolic disassembly of the instruction at the current location, showing a label (if any), an opcode and the symbolic operand field (if symbols are available). This powerful capability ties the symbol table created at program assembly time or manually entered with the debug software so you can debug at the assembly language level and do not have to decode hexadecimal printouts. The register contents are also displayed.

- Complete program trace - When in single step mode, the next instruction to be executed may be shown, with registers, to create a complete program trace.

- I/O device flexibility - Object code may be loaded from, or dumped to, any system device. The dump command also provides the capability to dump several sections of code to the same file so that fragmented memory may be reloaded automatically. Both ASCII and binary formats are available to dump object code to mass storage.

- I/O independent system functions - System functions are, in general, I/O independent so that data can be taken from, or directed to, any system device. For example, object code can be dumped to the serial port, the parallel port, or to audio tape by means of simple commands before invoking the Assembler. In addition, provision is made to invoke user supplied input and/or output routines.

- Command file - A series of Monitor commands can be operated on by the command file interpreter to provide automatic Debug Monitor and Editor operations. These commands can be entered into memory using the Editor then executed from memory.

- Built-in pseudo assembler - Allows instructions to be entered in symbolic form, with three-letter mnemonics instead of hexadecimal op codes.

- Mnemonic entry - Direct entry of R65ØØ object code into memory using symbolic operation codes allows easy program correction or patching without reassembling.

- Disassemble memory - Memory can be disassembled into R65ØØ mnemonic operation codes with hexadecimal or symbolic operands and labels (if symbols are defined) to assure proper program loading and instruction sequences.

- Symbolic debugging - A symbol table is used by the Monitor to equate two byte values with six byte symbols. This symbol table is used to set breakpoints symbolically (see Section 4.7.5), to start execution with a label (see Section 4.6.1) to dump between symbols (see Section 4.8.2) and to disassemble instructions with labels as operands (see Section 4.5.2).

4.2 CONTROL COMMANDS

Four commands are available to enter the Monitor command level from either a cold RESET, a warm RESET, a non-maskable interrupt or escape from a current Monitor function. Three other commands provide Editor initial entry, re-entry or text buffer recovery. Two additional other control commands are available which repeat the last Monitor command or invoke a series of Monitor commands.

4.2.1 CTRL RESET - Enter and Initialize Monitor (Cold Reset)

The CTRL RESET (see Section 3.1.1) initializes Monitor constants and variables (see Section 15.2) and enters the Monitor command level. The Monitor cold RESET message

```
ROCKWELL AIM 65/40
```

is displayed and printed (the printer AUTO-PRINT is turned on, see Section 3.1.12) followed by steady display of the Monitor command prompt ({), and blinking display of the input character cursor in the first two positions of the display.

4.2.2 RESET - Enter Monitor (Warm Reset)

The RESET enters the Monitor at the command level without initializing any Monitor constants and variables. The Monitor warm RESET message

```
AIM 65/40
```

is displayed (and printed, if AUTO-PRINT is turned on) followed by display of the Monitor command prompt and input character cursor.

NOTE

> If the Monitor constants or variables are altered to invalid values, it may be necessary to perform a CTRL RESET to restore the Monitor command level.

4.2.3 ATTN - Non-Maskable Interrupt

The ATTN key is usually wired to the R6502 CPU $\overline{\text{NMI}}$ input (see Section 2.4.4) which causes a non-maskable interrupt upon depression. The I/O ROM provides user alterable $\overline{\text{NMI}}$ interrupt handling linkage (see Section 6.3) which is initialized by the Monitor upon a cold RESET to point to a single step execution (see Section 4.6.1). Unless this linkage is altered by your application program, pressing ATTN will interrupt execution at the end of the current instruction. The next instruction to be executed will be disassembled and displayed along with the register values. Control then returns to the Monitor command level.

For example, press ATTN while the Monitor prompt is being displayed. An instruction similar to this one will be displayed:

```
 = 63 00 FF 03 F0 F63F      LDY $0358
```

Now enter any valid Monitor command.

Pressing ATTN is a quick way to return to the Monitor command level without resetting anything.

WARNING

> If an unknown operation is being performed and ATTN does not return control to the Monitor command level, or Monitor operation appears incorrect, perform a CTRL RESET. The unknown operation may have altered Monitor variables (see Section 6.2.2), necessitating a cold RESET.

4.2.4 ESC Command - Escape to Monitor Command Level

The ESC command escapes from most Monitor functions and returns control to the Monitor command level. ESC is operative only in the functions that examine inputs from the keyboard.

For example, ESC will not cause the Monitor to return to the command level until a dump to audio tape is completed, in which case it would return anyway. ESC is used most often to terminate a partial complete function when an erroneous input has caused an unexpected response or if it is not desired to complete the function.

Note that ESC may be used by other major functions, such as the Editor, to return to that function's command level rather than to the Monitor. The user-defined vector ESCIV (see Section 6.1.3) points to the appropriate ESC key processing.

Try it by quickly pressing and releasing the ESC key. The message

```
  (ESC)
```

will be displayed (and printed if AUTO-PRINT is on; see Section 3.1.12) on either the same or next line followed by display of the Monitor command prompt and input character cursor.

4.2.5 E Command - Initialize Text Buffer and Enter Editor

The Monitor E command initializes the AIM 65/40 Editor text buffer and enters the Editor to receive input text. Refer to Section 5.2.1 for a detailed description.

4.2.6 C Command - Create Text Buffer and Re-enter Editor

The Monitor C command can recover an old text buffer or create a new one when ASCII characters are already in ROM. Refer to Section 5.2.2 for details.

4.2.7 T Command - Re-Enter Editor

The Monitor T command re-enters the AIM 65/40 Editor at the top of the existing text buffer (established by a prior E or C command). Refer to Section 5.2.3 for details.

4.2.8 + Command - Repeat Last Command

The + command repeats the last entered Monitor command. This is useful when a long Monitor command, such as dump to audio tape recorder, has been commanded but needs repeating (perhaps because the recorder was not connected the first time).

4.2.9 & Command - Execute Command String

The & command inputs a string of Monitor or Editor commands from a file in memory, rather than from the keyboard. This allows automatic debug and test sequences to be pre-programmed.

Use the & command as follows:

a. Initialize a text buffer using the Monitor E command (see Sections 5.2.1).

b. Build a command file by entering a consecutive series of Monitor commands after an initial blank character ($20). Envision what the Monitor response or prompt is for each command. Enter "!" or RETURN for each carriage return. The "!" is preferred for readability and to prevent two consecutive RETURNs from terminating the reading of text into the text buffer. Note that a RETURN terminating a line of text in the text buffer is also interpreted as RETURN command to the Monitor.

NOTE

The "!" character is also used for other purposes. Two consecutive "!!" characters cause termination of the command string interpretation and control is returned to the Monitor command level. Three consecutive "!!!" characters cause the next command processed from the keyboard. Inputs will continue to be accepted from the keyboard until a single "!" character is input, at which time control reverts back to the command string interpreter.

c. Terminate the command file with two consecutive "!!" characters. Note that a RETURN immediately preceding command file termination should be indicated with a RETURN rather than a "!" character to prevent the switching to the manual input mode (see step b.)

d. Type & to return control from the Editor to the Monitor.

e. Type & to enter the command file. AIM 65/4Ø will respond with

```
{&}    FROM=XXXX
```

f. Enter the starting address of the command file (the displayed address will correspond to the start of the text buffer) and end the input with RETURN.

g. Upon completion of the command file processing, control will be returned to the Monitor command level.

NOTE

Automatic chain of command files can be performed by including a "&" character followed by the address of the new command file data.

Examples:

1. Load an application program (file name=TST3) from an audio recorder and start execution at $Ø8ØØ.

   (a) Initialize the Text Buffer to load the command file

```
{E}
 EDIT FROM=2ØØØ TO=3FFF IN=<RETURN>
```

   (b) Enter the consecutive Monitor commands in the command file. Be sure the first character is a blank (enter with the SPACE bar).

```
 LØ!T1TST3
GØ8ØØ!!
<RETURN>
 *END*
```

   NOTE: <RETURN> means RETURN key depression.

   (c) Return to the Monitor.

```
={Q}
```

   (d) Invoke the command file

```
{&} FROM=2ØØØ
{L} OFFSET=Ø   IN=T UNIT=1 FILE=TST3
{G} Ø8ØØ
```

2. This second example, while appearing somewhat complicated, illustrates the power of the command file to assemble (using the optional symbolic assembler), establish breakpoints and perform execution, all under automatic control with symbolic linkage. This technique speeds up debugging when source code is being updated often and repetitive debug testing is required.

   (a) Initialize Text Buffer No. 1 in the Editor to load the program source code.

```
{E}
 EDIT FROM=2ØØØ<RETURN> TO=27FF IN=<RETURN>
```

(b) Initially enter the source code.

```
   .PAG 'DEMO PROGRAM'
   *=$4Ø
   CTR=*+1
   *=$Ø8ØØ
   START NOP
   CLD
   LDA #Ø
   STA CTR
   LDA CTR
   LDX #$FE
   LDY #Ø2
   XLOOP INX
   BNE XLOOP
   YLOOP DEY
   BNE YLOOP
   INC CTR
   JMP CYCLE
   .END
   <RETURN>

    *END*
```

(c) Return to the Monitor.

```
   ={Q}
```

(d) Initialize Text Buffer No. 2 in the Editor to load the command file.

```
   {E}
    EDIT FROM=28ØØ<RETURN> TO=29FF<RETURN>
```

(e) Enter the consecutive Monitor commands.

```
    4#8BØ;CYCLE!B1;XLOOP!B2;YLOOP!?J
   4G;START!G!G!G!G!G
   !!
```

where:

```
   4 disables the breakpoints
   # cleans all breakpoints
   8 performs subsequent assembly
   BØ;CYCLE sets breakpoint Ø symbolically
   B1;XLOOP sets breakpoint 1
   B2;XLOOP sets breakpoint 2
   ? displays the breakpoints
   J displays the register headings
   4 enables the breakpoints
```

```
G;START! executions in run mode at the starting
  address
G! reserves execution in run mode after breakup
! are carriage returns
```

(f) Return to the Monitor.

```
={Q}
```

(g) Restore Text Buffer No. 1.

```
{C}
 EDIT FROM=2ØØØ<RETURN> TO=27FF<RETURN>
 .PAG 'FILE DEMO PROGRAM'
```

(h) Return to the Monitor.

```
={Q}
```

(i) Set-up and perform the initial assembly.

```
{7}ASSEMBLER V1.Ø
 FROM=18ØØ<RETURN> TO=1FFF<RETURN>
 IN=M
 OBJ TO MEM?Y
 LIST?Y   OUT=<RETURN>
PASS 1
PASS 2

PAGE ØØØ1    DEMO PROGRAM

ADDR  .OBJECT.  SOURCE

ØØØØ                 *=$4Ø
ØØ4Ø          CTR=*+1
ØØ4Ø                 *=$Ø8ØØ
Ø8ØØ EA       START  NOP
Ø8Ø1 D8              CLD
Ø8Ø2 A9 ØØ           LDA #Ø
Ø8Ø4 85 41           STA CTR
Ø8Ø6 A5 41           LDA CTR
Ø8Ø8 A2 FE           LDX #$FE
Ø8ØA AØ Ø2           LDY #Ø2
Ø8ØC E8       XLOOP  INX
Ø8ØD DØ FD           BNE XLOOP
Ø8ØF 88       YLOOP  DEY
Ø81Ø DØ FD           BNE YLOOP
Ø812 E6 41           INC CTR
Ø814 4C EA EA JMP CYCLE
**ERROR Ø1
Ø817                 .END
 ERRORS= ØØØ1
```

(j) Correct the source code by adding the missing CYCLE label.

```
{T}
 PAG 'DEMO PROGRAM'
={C} OLD=STA <RETURN> NEW=CYCLE STA <RETURN>/<SPACE>
START NOP <SPACE>
STA CTR <RETURN>
CYCLE STA CTR

 *END*
```

(k) Return to Monitor.

```
={Q}
```

(l) Be sure breakpoints are initially enabled (the command file will turn them off, then on).

```
{4} ON
```

(m) Invoke the command file

```
{&} FROM =28ØØ
{4} OFF
{#} OFF
{8}ASSEMBLER V1.Ø
PASS 1
PASS 2

PAGE ØØØ1    DEMO PROGRAM


ADDR  .OBJECT.  SOURCE


ØØØØ                  *=$4Ø
ØØ4Ø           CTR=*+1
ØØ4Ø                  *=$Ø8ØØ
Ø8ØØ EA        START  NOP
Ø8Ø1 D8               CLD
Ø8Ø2 A9 ØØ            LDA #Ø
Ø8Ø4 85 41            STA CTR
Ø8Ø6 A5 41     CYCLE  LDA CTR
Ø8Ø8 A2 FE            LDX #$FE
Ø8ØA AØ Ø2            LDY #Ø2
Ø8ØC E8        XLOOP  INX
Ø8ØD DØ FD            BNE XLOOP
Ø8ØF 88        YLOOP  DEY
Ø81Ø DØ FD            BNE YLOOP
```

```
0812 E6 41           INC CTR
0814 4C 06 08        JMP CYCLE
0817                 .END
 ERRORS= 0000

{B}BRK/0=;CYCLE <RETURN> 0806   CYCLE  LDA CTR
{B}BRK/1=;XLOOP <RETURN> 080C   XLOOP  INX
{B}BRK/2=;YLOOP <RETURN> 080F   YLOOP  DEY
{?}0    1    2    3    4    5    6    7
 0806 0800 080F 0000 0000 0000 0000 0000
{4} ON
{G} ;START   0800
{J}
 PP AA XX YY SS
= 32 00 XX XX XX 0806 CYCLE  LDA CTR
{G} 0806

= 30 00 FE 02 XX 080C XLOOP  INX
{G} 080C

= B0 00 FF 02 XX 080C XLOOP  INX
{G} 080C

= 32 00 00 02 XX 080F YLOOP  DEY
{G} 080F

= 30 00 00 01 XX 080F YLOOP  DEY
{G} 080F

= 30 00 00 00 XX 0806 CYCLE  LDA CTR
{G} 0806
```

NOTE

Only the zero flag is pertinent in the processor status register.

(n) Evaluate the debug printout and continue manual debug as required.

4.2.10 O Command - Toggle Memory Bank

The O command toggles the memory bank for Monitor access. This allows you to address either bank 0 or bank 1 from the Monitor command level, in order to perform such functions as display/alter memory, execute programs in Step or Run mode and to use the function keys to link to user defined functions in either bank.

If memory is installed off-board and selected for bank 1 operation, be sure the JBSE jumper is installed to dedicate on-board RAM to bank Ø (see Section 2.3.3). Note also that only addresses $1ØØØ through $BFFF may be assigned unique to either bank, i.e. $Ø-$FFF and $CØØØ-$FFFF are always common to both banks.

Type O to toggle the memory bank. AIM 65/4Ø will display/print the selected bank, such as:

```
  {O} MEMORY BANK 1
  {O} MEMORY BANK Ø
```

Example 1: Display and alter one byte of memory (using SPACE after the M command) on-board in bank Ø and off-board in bank 1.

```
  {O} MEMORY BANK Ø
  {M}1ØØØ 67 g
  {/}1ØØØ 32 g
  {M}1ØØØ 32 2

  {O} MEMORY BANK 1
  {M}1ØØØ 51 Q
  {/}1ØØØ 45 Q
  {M}1ØØØ 45 E
```

Example 2: Command two user functions (each of which returns to the Monitor, see Section 4.11), one in bank Ø and the other in bank 1. Note that eight additional other functions in bank 1 can be linked to using the function keys. However, the starting address for the user functions in bank 1 must be the same for the user functions in bank Ø.

Examples:

```
  {O} MEMORY BANK Ø
  {1} MEMORY BANK Ø

  {O} MEMORY BANK 1
  {2} MEMORY BANK 1
```

4.2.11 CTRL Z Command - Direct Peripheral Control

The CTRL Z command exits the normal Monitor command level mode and sends keyboard entries directly to the on-board printer and display peripherals (see Section 3.1.13). One exception is the performing of the AIM 65/4Ø SBC module RAM test (see Section 3.2.2). Press RESET to return to the Monitor command level.

4.2.12 CTRL C Command - Clear Display and Home Cursor

The CTRL C command clears the display and homes the cursor to position one.

4.2.13 CTRL N Command - Home Cursor

The CTRL N command homes the cursor to position one without clearing the display.

4.2.14 @ Command - Enter Data Output Rate

The @ command inputs a number which determines the rate at which data is output to the display (and printer, if auto-print is on) by a subsequent command that has a variable output rate, e.g. the disassembly memory function (K command, see Section 4.5.2). The number may vary from Ø (fastest) to 9 (slowest).

Example:

```
  {@}1
```

4.3 DISPLAY/ALTER REGISTERS

Seven commands display or alter the contents of the six registers (program counter, processor status, accumulator, X register, Y register and stack pointer). These commands are often used to establish initial register values for checkout purposes. During normal program operation, the register contents would be initialized by previously executed instructions.

4.3.1 R Command - Display Register Contents

The R command displays the current contents of the six registers.

To display the contents of the registers, type R. AIM 65/4Ø will print one line. The register values will be preceded by a letter identifying the register.

Example:

```
  {R}*=E5D2 P=ØØØØØØØØ A=ØD X=ØØ Y=Ø2 S=FF
```

In this example, the registers and their contents are:

Program Counter	(*)=E5D2
Processor Status	(P)=ØØØØØØØØ
Accumulator	(A)=ØD
X Register	(X)=ØØ
Y Register	(Y)=Ø2
Stack Pointer	(S)=FF (which is equivalent to Ø1FF since the stack is always in Page 1.)

4.3.2 A Command - Display/Alter Accumulator

The A command displays and alters the contents of the accumulator.

To display the accumulator register, type A. AIM 65/4Ø will respond with:

```
  {A}=XX
```

To change the value, enter the new value of the accumulator register as a two digit hexadecimal number. A leading zero must be entered in the left digit if the left digit value is zero. The value is not changed until both digits are entered. Press ESC to return to the Monitor command level before the second digit is entered.

Example:

```
  {A}=Ø1
```

In this example, the existing value of A was changed to $Ø1.

4.3.3 P Command - Display/Alter Processor Status

The P command displays and alters the contents of the processor status register.

To display the contents of processor status register, type P. AIM 65/4Ø will display the current value:

```
  {P}=XX
```

To change the contents, enter the new value as a two digit hexadecimal number. A leading zero must be entered in the left digit position if the left digit value is zero. The value is not changed until both digits are entered. Press ESC to return to the Monitor command level before the second digit is entered.

Example:

```
  {P}=ØØ
```

In this example, the value of the processor status register was changed to $ØØ.

4.3.4 S Command - Display/Alter Stack Pointer

The S command displays and alters the value of the stack pointer.

To display the value of the stack pointer, type S. AIM 65/4Ø will display the current value:

```
  {S}=XX
```

To change the value, enter the new value as a two digit hexadecimal number. A leading zero must be entered in the left digit if the left digit value is zero. The value is not changed until both digits are entered. Press ESC to return to the Monitor command level before the second digit is entered.

Example:

```
  {S}=FF
```

In this example, the value of the stack pointer was changed fom $8F to $FF. Note that the stack is always in page one of memory, so the new address of the stack is therefore $Ø1FF.

4.3.5 X Command - Display/Alter X Register

The X command displays and alters the contents of the X register. To display the contents of the X register, type X. AIM 65/4Ø will display the current value:

```
  {X}=XX
```

To change the value, enter the new value as a two digit hexadecimal number. A leading zero must be entered in the left digit if the left digit value is zero. The value is not changed until both digits are entered. Press ESC to return to the Monitor command level before the second digit is entered.

Example:

```
  {X}=Ø2
```

In this example, the value of the X register was changed to $Ø2.

4.3.6 Y Command - Display/Alter Y Register

The Y command displays and alters the contents of the Y register.

To display the contents of the Y register, type Y. AIM 65/40 will respond with:

```
[Y]=XX
```

To change the value, enter the new value as a two digit hexadecimal number. A leading zero must be entered in the left digit if the left digit value is zero. The value is not changed until both digits are entered. Press ESC to return to the Monitor command level before the second digit is entered.

Example:

```
[Y]=03
```

In the above example, the value of the Y register was changed to $03.

4.3.7 * Command - Display/Alter Program Counter

The * command displays and alters the value of the program counter. This command can be used to initialize the value of the program counter for the G command (Execute Program) or the I command (Mnemonic Instruction Entry).

Use the * command as follows:

a. Type * . AIM 65/40 will respond with:

```
{*}=XXXX
```

b. Enter the new hexadecimal value of the program counter. End the input with RETURN or a SPACE.

   Example:

   ```
   {*}=0500
   ```

   In the example above, the program counter was changed to $0500.

4.4 DISPLAY/ALTER MEMORY

4.4.1 M Command - Display Selected Memory Contents

The M command displays the hexadecimal and ASCII contents of one to 18 consecutive memory locations, starting at the specified address.

Use the M command as follows:

a. Type M. AIM 65/40 will respond with:

```
{M}
```

b. Press SPACE if you want to display the contents of one memory location only, otherwise continue to the next step.

c. Enter the hexadecimal address of the first memory location to be displayed.

d. Press RETURN to display the number of memory locations specified in variable MEMCNT ($023D); (see Section 6.1.2). This number is initialized to eight upon a cold RESET.

e. AIM 65/40 will display the contents of the memory locations and the corresponding decoded ASCII character. Non-printable ASCII characters are indicated by ".".

Example 1: Display the contents of eight memory locations.

```
{M}500 <RETURN> 46 46 46 46 46 0D 0D 0D FFFFF...
```

In this example, the memory locations and their contents are:

Address	Contents	ASCII Character
5ØØ	46	F
551	46	F
552	46	F
553	46	F
554	46	F
555	ØD	none (carriage return)
556	ØD	none (carriage return)
557	ØD	none (carriage return)

Example 2: Display the contents of one memory location.

```
{M} <SPACE> 5ØØ <RETURN> 46 F
```

4.4.2 SPACE Command - Display Higher Memory Contents

The SPACE command displays the contents of the next set of memory locations, after the initial address value has been entered using the M or - commands or another SPACE command. The contents of the memory locations are displayed in the same format as the last M command.

Example:

```
{M}5ØØ  47 47 2Ø 2Ø ØD 2Ø 47 47 GG....GG
{ }Ø5Ø8 47 47 47 47 52 52 52 52 GGGGRRRR
{ }Ø51Ø 52 52 52 52 52 52 52 52 RRRRRRRR
```

4.4.3 - Command - Display Lower Memory Contents

The minus command displays the contents of the preceding set of memory locations, after the initial location value has been displayed using the M or SPACE commands or another - command. The contents of the memory locations is displayed in the same format as the last M command.

Example:

```
    {M}53Ø  46 46 46 46 46 46 46 46 FFFFFFFF
    {-}Ø528 54 54 ØD 46 46 46 46 46 TT.FFFFF
    {-}Ø52Ø 54 54 54 54 54 54 54 54 TTTTTTTT
```

4.4.4 / Command - Alter Memory Contents

The / command alters any memory location displayed with the M, SPACE or - commands. The contents of memory are first displayed in the same format as the prior display memory command.

Use the / command as follows:

a. Display the memory location to be altered using M, SPACE or - commands.

b. Press the / key. AIM 65/4Ø displays the address of the first memory location. The cursor prompts for an input on the first byte at the displayed location.

c. If the first memory location is to be altered, enter the new contents in hexadecimal. Two valid hexadecimal digits must be entered. If the location is to be left as is, SPACE to the proper byte. If you SPACE past the last byte, the system will advance to the next line.

d. Proceed to the next location and alter it, if required.

While operating under the / command, a DEL moves the cursor to the left and a SPACE moves the cursor to the right. Holding either of these keys down will continuously advance or backspace the memory cursor. When the cursor wraps around in either direction, the program counter is updated.

When the altering is complete, type RETURN. If the last memory location on the line was altered, no RETURN is necessary. If only the first of a byte has been changed, press ESC to return to the Monitor command level.

Example: Display, alter and verify changes to addresses 8Ø1, 8Ø3 and 8Ø5.

```
{M}8ØØ  4Ø 41 42 43 44 45 46 47 @ABCDEFG
{/}Ø8ØØ 4Ø 51 42 53 44 55 46 47
{M}Ø8ØØ 4Ø 51 42 53 44 55 45 47 @QBSDUFG
```

In this example,

```
  Location 0800 was unchanged.  Enter SPACE.
  Location 0801 changed from 41 to 51.
  Location 0802 was unchanged.  Enter SPACE.
  Location 0803 changed from 43 to 53.
  Location 0804 was unchanged.  Enter SPACE.
  Location 0805 changed from 45 to 55.  Enter RETURN to
                terminate the line since the last two bytes are
                unchanged.
```

The changes were then verified using another M command.

4.5 ENTER/DISASSEMBLE INSTRUCTIONS AND SYMBOLS

Two commands allow R6500 instructions to be entered into memory and instructions already in memory to be disassembled . The I command encodes (or assembles) mnemonic CPU instruction into directly executable object code (machine language) and stores them in memory. The K command decodes (or disassembles) object code from memory into CPU instructions in mnemonic form. The ; command enters symbol values into memory which can be used for symbolic command specification and tracing.

4.5.1 I Command - Enter Mnemonic Instruction

The I command enters R6500 instructions directly into memory as object code from mnemonic instructions entered from the keyboard. Starting from a entered address, operation codes (op codes) are entered using the standard three character instructions mnemonic. Operands, if required, are entered in hexadecimal in accordance with the addressing mode formats. Invalid op codes and operands cause an ERROR message to be displayed.

Use the I command as follows:

NOTE

Toggle off the symbol table (V command, see Section 4.6.5) if it is on, otherwise the operand will be displayed as a symbol if a match is found in the symbol table.

a. Type I. AIM 65/4Ø will respond with the present program counter address and a blinking corsor; for example:

```
16ØØ
```

b. If you now wish to change the program counter, type *. AIM 65/4Ø will display:

```
16ØØ     =
```

Enter the program counter value in the hexadecimal and end the input with RETURN. Correct the address using DEL before RETURN is pressed. If address ØØ was entered, AIM 65/4Ø responds with:

```
Ø8ØØ
```

c. Enter the three character mnemonic for the operation code. An input error in either of the first two characters may be corrected by typing DEL and the correct character.

d. If the entered op code does not require an operand, the object code is computed, stored in memory and displayed in object code form along with the program counter address and the symbolic op code. The program counter is incremented by one. If you want to enter additional instructions in successive addresses, return to Step c.

If the op code is invalid, ERROR will be displayed. The correct op code may then be re-entered without altering the program counter address, since it has not been incremented.

If a valid but undesired op code was entered, it may be corrected in either of two ways:

(1) If the op code requires an operand, enter RETURN before entering an operand or deliberately enter an invalid operand. ERROR will be displayed the complete instruction can be re-entered since the program counter address was not changed.

(2) If the op code does not require an operand, the object code is entered into memory and the program counter is incremented. In this case, re-establish the previous program counter address as in Step c.

e. Enter the operand in hexadecimal in accordance with the addressing mode formats. Refer to the R6500 Programming Manual for a complete description of the addressing formats. Valid operand formats are (H is a hexadecimal digit):

Addressing Mode	Operand Format
Accumulator	A
Immediate	#HH
Zero Page	HH
Zero Page, X	HH,X or HHX
Zero Page, Y	HH,Y or HHY
Absolute	HHHH
Absolute, X	HHHH,X or HHHHX
Absolute, Y	HHHH,Y or HHHHY
Relative	HH or HHHH
(Indirect, X)	(HH,X) or (HHX)
(Indirect, Y)	(HH),Y or (HH)Y
(Indirect)	(HHHH)

Note that for conditional branch instructions, the branch address may be entered as a two-digit relative displacement or as a four-digit absolute address.

End the operand entry with RETURN or SPACE. The op code and operand are computed and stored in memory. The program counter address, the op code object code and the symbolic form of the op code and operand are displayed.

If the operand is invalid, ERROR is displayed and the entire instruction must be re-entered.

An error in operand entry before RETURN or SPACE is entered may be corrected by entering DEL and re-entering the correct data. An error in operand entry after RETURN or SPACE is entered may be corrected by re-establishing the correct program counter address and re-entering the complete instruction.

When entering additional instructions, return to Step d. If instruction entry is complete, end with a RETURN.

**** IMPORTANT ****

Do not use memory locations below $Ø8ØØ. These locations are reserved for the AIM 65/4Ø Monitor and optional software (see Figure 2-4).

Example:

```
{I}

Ø8Ø2                  *=8ØØ
Ø8ØØ EA               NOP
Ø8Ø1 A2 FE            LDX #$FE
Ø8Ø3 E8               INX
Ø8Ø4 DØ Ø7            BNE Ø8ØD
Ø8Ø6 4C 1Ø Ø8         JMP Ø81Ø
Ø8Ø9                  *=81Ø
Ø81Ø AØ Ø2            LDY #$Ø2
Ø812 88               DEY
Ø813 DØ FB            BNE Ø81Ø
Ø815 4C Ø1 Ø8         JMP Ø8Ø1
Ø818
```

4.5.2 K Command - Disassemble Memory

Beginning at the program counter address, the K command disassembles (translates) object code in memory into symbolic R65ØØ instructions, for a specified number of instructions. If an invalid (untranslatable) op code is encountered, that byte will be displayed as "??". Appendix F lists the valid op codes.

Use the K command as follows:

a. Type K. AIM 65/4Ø will display the current program counter value:

```
{K}*=XXXX
```

b. If you want to start at a different address, enter the new value in hexadecimal or as a defined symbol (preceded by a ;), then press RETURN or SPACE. Otherwise just press RETURN or SPACE. If Ø9ØØ is entered, AIM 65/4Ø will respond with:

```
{K}*=Ø9ØØ/
```

If a symbol (e.g. LABEL1) with a value of Ø9ØØ in the symbol table (see Section 4.1.1) is specified, the response will be:

```
{K}*=;LABEL1  Ø9ØØ/
```

c. Specify the number of instructions to be disassembled, as a decimal count from 1 to 9999 then press RETURN. RETURN with no decimal count means one instruction, a "." or SPACE means continuous disassembly, and Ø means 1ØØ instructions.

d. If five instructions are desired, enter 5 and press RETURN. The system responds with:

```
{K}*=Ø9ØØ/5      OUT=
```

e. Enter the output device code (see Appendix A) and respond to subprompts.

f. If Memory (M) is selected as the output device, the AIM 65/4Ø asks where the disassembled source code is to be stored. For example,

```
{K}*=Ø9ØØ/5   OUT=M   FROM=XXXX
```

Enter the starting address and end it with RETURN. The system will then ask for the last address.

```
{K}*=Ø9ØØ/5   OUT=M   FROM=XXXX   TO=XXXX
```

Enter the last address in memory. Remember that source code will be several times larger than the object code so allow about 15 bytes per object code instruction in the destination memory.

g. AIM 65/4Ø responds by disassembling the specified number of instructions. To abort the disassembly operation at any time, press the ESC key. To stop the disassembly temporarily, press SPACE; to resume the disassembly, press SPACE again. If the operands correspond to symbols defined in the symbol table, the symbols will be displayed.

If the output is directed to memory (OUT=M), the disassembled source code will be stored starting at the first address plus one. A carriage return character ($ØD) is stored in the first byte.

If the address limits provide sufficient space, all the disassembled source code will be stored in RAM as ASCII characters and terminated with $ØØ to indicate end of text. If insufficient room is available to hold all the source code, a memory fail indication is displayed along with the last address plus one, e.g.

```
MEM FAIL 1Ø41  (if TO=1Ø4Ø)
```

After the source code has been stored in memory, a text buffer can be created around the code using the C command (see Section 5.2.2).

Example 1: Disassemble the 13-byte example program shown in Section 4.2.9 and output to the printer:

```
{K}*=0800/13 <RETURN> OUT=P
0800 EA        START  NOP
0801 D8               CLD
0802 A9 00            LDA #0
0804 85 41            STA CTR
0806 A5 41     CYCLE  LDA CTR
0808 A2 FE            LDX #$FE
080A A0 02            LDY #02
080C E8        XLOOP  INX
080D D0 FD            BNE XLOOP
080F 88        YLOOP  DEY
0810 D0 FD            BNE YLOOP
0812 E6 41            INC CTR
0814 4C 06 08         JMP CYCLE
```

Example 2: Disassemble the same program to an audio cassette recorder.

```
{K}*=0800/13 <RETURN>  OUT=T UNIT=1 FILE=DEMO <RETURN>
```

Example 3: Disassemble the same program to memory, add assembler directives and variables, and assemble it to another area in memory using the optional assembler.

```
{K}*=0800/13 <RETURN> OUT=M FROM=2500   TO=2600

{C}
 EDIT FROM=2500 <RETURN> TO=2600 <RETURN>

={L}/.    OUT=

 START  NOP
        CLD
        LDA #$00
        STA CTR
 CYCLE  LDA CTR
        LDX #$FE
        LDY #$02
 XLOOP  INX
        BNE XLOOP
 YLOOP  DEY
        BNE YLOOP
        INC CTR
        JMP CYCLE
**END**

={T}

={R} IN=
.PAG . DEMO PROGRAM
*=$40
CTR=*+1
*=$0900
<RETURN>
={B}
```

```
={D}/<RETURN>
={I}
.END
={Q}


{7}ASSEMBLER V1.Ø
 FROM=18ØØ  TO=1FFF
 IN=M
 OBJ TO MEM?Y
 LIST?Y   OUT=
PASS1
PASS2


PAGE ØØØ1     DEMO PROGRAM


ADDR  .OBJECT.  SOURCE

ØØØØ                  *=$4Ø
ØØ4Ø           CTR=*+1
ØØ4Ø                  *=$Ø9ØØ
Ø9ØØ EA        START  NOP
Ø9Ø1 D8               CLD
Ø9Ø2 A9 ØØ            LDA #$ØØ
Ø9Ø4 85 41            STA CTR
Ø9Ø6 A5 41     CYCLE  LDA CTR
Ø9Ø8 A2 FE            LDX #$FE
Ø9ØA AØ Ø2            LDY #$Ø2
Ø9ØC E8        XLOOP  INX
Ø9ØD DØ FD            BNE XLOOP
Ø9ØF 88        YLOOP  DEY
Ø91Ø DØ FD            BNE YLOOP
Ø912 E6 41            INC CTR
Ø914 4C Ø6 Ø9         JUMP CYCLE
Ø914                  .END
 ERRORS= ØØØØ
```

4.5.3 ; Command - Enter Symbol Value

The ; command enters either an initial or new value for a symbol into the symbol table. Once entered, the symbol can be used to specify breakpoints, set from and to addresses, specify start of execution and trace instruction labels and operands symbolically.

Use the ; command as follows:

a. Type ; the AIM 65/4Ø will display the character position cursor after

```
{;}
```

b. Type the symbol characters. Up to six characters may be entered. While any printable characters may be used, it is suggested that characters meeting the optional assembler label requirements be used, especially if you are going to disassemble programs then later assemble from the source code.

c. End the symbol entry by pressing RETURN or SPACE. The AIM 65/40 will display, e.g.

```
{;}LABEL=
```

If the value has been previously defined, the value will also be displayed, e.g.

```
{;}LABEL=125A
```

d. Enter the symbol value in hexadecimal (0 to $FFFF, the dolar sign is optional), e.g.

```
{;}LABLE=4621
```

e. End the input by pressing RETURN or SPACE.

Examples:

```
{;}SYN1 = 0E00
{;}SYN2 = 1000
```

The symbol values in RAM may be examined using the M command at the location of the symbol table (default location = $1800 as specified by Monitor constant STSAVE, see Table 15-1). Look for the symbol name in the ASCII field and symbol value in the hexadecimal field.

4.6 EXECUTION/TRACE COMMANDS

Three commands allow execution and detailed examination of an application or user program. The G command executes the program in single Step mode (non-real-time) where each instruction can be displayed and examined, or in Run mode (real-time) where complete control of the CPU turned over to the application program.

The Z command controls the instruction trace. After execution is terminated and control returned to the Monitor, a history trace of branch and jump instructions may be obtained using the H command. The breakpoint control is described in Section 4.7.

4.6.1 G Command - Execute Program

The G command starts execution of a program at the current or specified program counter value. The starting address may be specified in hexadecimal or as a defined symbol. Execution is also initiated in single step mode (non-real-time) with optional register and instruction tracing or in Run mode (real-time).

Use the G command as follows:

a. If Step execution is planned, set up the instruction trace toggle (Z command, see Section 4.6.2) as desired.

b. Clear, established, check, and enable/disable breakpoints using the #, B, ? and 4 commands, respectively (see Section 4.7). Note that breakpoints operate in either Step or Run execution modes.

c. Type G. AIM 65/4Ø displays the current program counter value.

```
    {G} XXXX
```

d. Enter the starting address if it is different from that displayed. Enter it in hexadecimal or as a symbol (precede the symbol with a ";"). Note that the symbol must be defined in the symbol table, either manually (see Section 4.1Ø.2) or by prior use of an optional assembler or compiler.

e. Initiate execution in either the Run or Step mode.

(1) To execute in the Run mode, press RETURN. The display will blank then execution start.

```
{G} XXXX  RETURN
```

In the Run mode, complete CPU control is turned over to the executing program. The four ways to return to the Monitor command level are:

(a) Executing a JMP instruction to COMIN1 ($A3ØE).

(b) Encountering an enabled breakpoint.

(c) Executing a BRK instruction.

(d) Pressing ATTN or RESET.

If a breakpoint or BRK instruction is encountered or ATTN is pressed, one disassembled instruction is displayed followed by the Monitor prompt.

(2) To execute in the Step mode, press SPACE. The system will ask how many instructions to execute.

```
{G} XXXX <SPACE> /
```

Enter a number from 1 to 9999, followed by RETURN or SPACE to execute a specified number of instructions. Otherwise, press RETURN to execute one instruction, or SPACE or "." to execute continuously.

In the Step mode, the Monitor examines each instruction as it is executed. If the instruction trace toggle (Z command, see Section 4.6.2) is on, the register values and the disassembled instruction to be executed are displayed. If the operands match any symbols in the symbol table, the corresponding symbol is displayed as an operand and/or as a label.

The ESC key can be used to terminate execution in the Step mode in addition to the ways to terminate the Run mode (see above).

NOTE

Execution will halt a BRK instruction placed by the Breakpoint function or as part of the entered code is encountered. Execution may be continued by commanding the G function which displays the address of the next instruction after the BRK. If execution is stopped by a BRK instruction not caused by a breakpoint in Step (not run), the program counter must be set to the address following the BRK instruction before continuing execution.

Examples:

The following program segment illustrates display of register contents, disassembled instructions, and symbol linkage. This is the same program used in Section 4.2.9 to illustrate command file linkage.

If the program object code is not in memory as shown in the disassembly, enter it using the I command (see Section 4.5.1) then enter the symbols with the ";" command (see Section 4.1Ø.2). Alternatively, use the optional assembler.

```
{K}*=Ø8ØØ/13 <RETURN> OUT=<RETURN>
Ø8ØØ EA         START  NOP
Ø8Ø1 D8                CLD
Ø8Ø2 A9 ØØ             LDA #Ø
Ø8Ø4 85 41             STA CTR
Ø8Ø6 A5 41      CYCLE  LDA CTR
Ø8Ø8 A2 FE             LDX #$FE
Ø8ØA AØ Ø2             LDY #Ø2
Ø8ØC E8         XLOOP  INX
Ø8ØD DØ FD             BNE XLOOP
Ø8ØF 88         YLOOP  DEY
Ø81Ø DØ FD             BNE YLOOP
Ø812 E6 41             INC CTR
Ø814 4C Ø6 Ø8          JMP CYCLE
```

Example 1: Step mode, instruction trace on, no breakpoints, execute six instructions. Note that the initial instruction is not disassembled and displayed. This allows subsequent single step instruction tracing to continue without the current instruction being displayed twice.

```
  {#} OFF
  {4} ON
  {4} OFF
  {Z} ON
  {G} Ø8ØØ/6  <RETURN>

   AØ ØØ FE ØØ FE Ø8Ø1        CLD
   AØ ØØ FE ØØ FE Ø8Ø2        LDA #$ØØ
   22 ØØ FE ØØ FE Ø8Ø4        STA CTR
   22 ØØ FE ØØ FE Ø8Ø6 CYCLE  LDA CTR
   22 ØØ FE ØØ FE Ø8Ø8        LDX #$FE
 = AØ ØØ FE ØØ FE Ø8ØA        LDY #$Ø2
```

Example 2: Step mode, instruction trace off, one breakpoint, execute four times.

```
  {B}BRK/Ø=;CYCLE   28Ø6   CYCLE  LDA CTR
  {?}Ø    1    2    3    4    5    6    7
   Ø8Ø6 ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ ØØØØ
  {4} ON
  {Z} OFF
  {G} Ø8ØØ <SPACE> /.

  = 22 ØØ FE ØØ FE Ø8Ø6 CYCLE  LDA CTR
  {G} Ø8Ø6 <SPACE> /.


  = 2Ø ØØ ØØ ØØ FE Ø8Ø6 CYCLE  LDA CTR
  {G} Ø8Ø6 <SPACE> /.

  = 2Ø Ø1 ØØ ØØ FE Ø8Ø6 CYCLE  LDA CTR
  {G} Ø8Ø6  /.

  = 2Ø Ø3 ØØ ØØ FE Ø8Ø6 CYCLE  LDA CTR
```

Example 3: Step mode, instruction trace on, no breakpoints, terminate with ESC.

```
  {Z} ON
  {$} OFF
  {G} Ø8ØØ <SPACE> /.
```

```
22 00 00 00 FE 0801        CLD
22 00 00 00 FE 0802        LDA #$00
22 00 00 00 FE 0804        STA CTR
22 00 00 00 FE 0806 CYCLE  LDA CTR
22 00 00 00 FE 0808        LDX #$FE
(ESC)
```

Example 4: Run mode, two breakpoints, execute two times.

```
{B}BRK/1=;XLOOP   080C   XLOOP  INX
{?}0    1    2    3    4    5    6    7
 0806 080C 0000 0000 0000 0000 0000 0000
{4} ON
{G} 0800 <RETURN>

= 32 00 FE 00 FE 0806 CYCLE  LDA CTR
{G} 0806 <RETURN>

= 30 00 FE 02 FE 080C XLOOP  INX
```

4.6.2 Z Command - Toggle Instruction Trace

The Z command toggles the flag which controls the single step instruction trace. If the flag is on and instructions are being executed in the Step mode (see Section 4.6.1) an instruction trace will be printed. If the symbol table toggle is on (see Section 4.10.3) and a symbol values is found, the symbol will be displayed as the operand or lable.

NOTE

A cold RESET will disable symbol linkage until the symbols are re-entered or the symbol linkage toggle is turned on (see Section 4.10.3).

To toggle the instruction trace toggle, Type Z. AIM 65/40 responds with the status of the instruction trace mode:

```
{Z} ON
{Z} OFF
```

4.6.3 J Command - Display Register Heading

The J command displays the register heading. This heading provides an easy register reference for subsequent single step program execution with instruction and register trace enabled (see Section 4.6).

To show the register header, type J. AIM 65/4Ø will respond with:

```
  {J}
    PP AA XX YY SS
```

4.6.4 H Command - Display Jump and Branch History

The H command displays the addresses of the last sixteen JMP or branch instructions that were executed. This trace capabilty exists only after the AIM 65/4Ø has been executing instructions in the Step mode.

Example:

Using the example program in Section 4.6.1, execute 1ØØØ instructions in Step mode with the instruction trace off.

```
{Z} OFF
{G} Ø8ØØ <SPACE> /1ØØØ  <RETURN>

= 2Ø 4C ØØ Ø1 FF 281Ø        BNE $28ØF
{H}
 Ø8ØD Ø814 Ø81Ø Ø8ØD Ø814 Ø81Ø Ø8ØD Ø814
 Ø81Ø Ø8ØD Ø814 Ø81Ø Ø8ØD Ø814 Ø81Ø Ø8ØD
```

Note the value of the counter in the A register ($4C) which indicates 76 cycles were performed.

4.6.5 V Command - Toggle Symbol Table on/off

The V command controls the flag that tells the Monitor to search the symbol table for a symbol value during debug functions. If the flag is on, the symbol table will be searched during single step trace (when step trace toggle is on, see Z command - Section 4.6.2), disassemble memory (K command see Section 4.5.2), memory address entry (FROM=, TO=) or breakpoint entry (B command, see Section 4.7.5) or breakpoint occurrance (G command, see Section 4.6.1). Any symbol found for the look-up value will be displayed as the operand symbol or address label.

NOTE

> In some cases, a symbol may be improperly displayed due to an operand equating to a symbol value that should not equate to the symbol. You can usually determine if the symbol is proper for the value in question based on your knowledge of the program under test.

If the flag is off, the table will not be searched. This saves considerable time when the symbol table is large and either a symbol does not exist or it is not necessary to display a symbol value.

Type V to use the command. The current state of the flag will be displayed.

```
{V}ON
{V}OFF
```

4.7 BREAKPOINTS

AIM 65/4Ø breakpoints help you to debug programs quickly and effortlessly. The breakpoint feature essentially allows you to specify up to eight different instructions in your program as points at which the R65Ø2 will stop, or "break", before executing that instruction.

What does this do for you? It allows you to find out when a certain instruction is being executed (or whether it is being executed at all!) and gives you the opportunity to examine the contents of any memory location or register just prior to that instruction. That is, breakpoints give you the opportunity to look at the progress of your program at several points in its execution sequence.

The AIM 65/40 Debug Monitor has four breakpoint commands:

- o The ? command allows you to display the currently-assigned breakpoint addresses.
- o The # command allows you to clear all eight breakpoints (set them to address $0000) upon completion of debug or in preparation for reassignment.
- o The 4 command allows you to enable (turn on) or disable (turn off) the breakpoint feature.
- o The B command allows you to set any of the eight breakpoint addresses.

4.7.1 How to Use Breakpoints

As mentioned previously, the AIM 65/40 provides eight breakpoints which you can assign to up to eight different instruction addresses in your program. When you turn power on or perform a CTRL RESET, the breakpoints are cleared. Once you decide which program instructions you would like execution to stop, these instructions should be assigned as breakpoints using the B command. Breakpoints 0 through 7 operate independently and can be assigned in any order; that is, AIM 65/40 does not care whether only breakpoint 3 carries an assigned address, nor does it care about the order in which the addresses are assigned.

With breakpoints assigned you are ready to run your program. Start execution by using the G command (see Section 4.6.1) in either the Step or Run mode. The program will begin executing and will run until a breakpoint address is encountered.

Upon encountering a breakpoint, the AIM 65/40 will stop then display the disassembled instruction at that address. This is the instruction that is about to be executed by the R6502. At this point, you can examine the status of the program. You may want to display the contents of the registers (with the R command) to see if they contain what you think they'll contain, or display a few memory locations (with the M command), for content. If everything is as expected, you may continue program execution by again using the G command. The breakpoint assignments will remain intact until you either clear them, perform a CTRL RESET or turn power off. You can clear them at any time using the # command or assigning a zero to that breakpoint.

NOTE

> When a program in RAM is executed using the G or function keys, the instructions at the breakpoint addresses set by the B command (see Section 4.7.5) are replaced by BRK instructions (i.e. op code ØØ) if the # command (see Section 4.7.3) has previously toggled breakpoint enable on. When a ØØ op code is encountered, the instruction at the address is disassembled and displayed by the Monitor NMI interrupt processing. The BRK instructions at the breakpoint addresses are also replaced with the regular instructions at this time.
>
> Should a program executed by the G or F1-F8 keys be stopped by a RESET or any other abnormal termination which does not go through NMI interrupt processing, the instructions replaced by BRK the Monitor breakpoint set-up will not be restored until a subsequent $\overline{\text{NMI}}$ is processed. Should this situation occur, the proper instructions can be easily restored by pressing the ATTN key once after a RESET.

4.7.2 ? Command - Display Breakpoints

The ? command displays the current address of each of the eight breakpoints. The leftmost, four-digit hexadecimal value is the address of breakpoint Ø, the rightmost value is the address of breakpoint 7. $ØØØØ indicates that a breakpoint is unassigned.

To use the ? command, type ?. AIM 65/40 will respond with the breakpoint identifiers and the corresponding breakpoint addresses, e.g.:

```
{?}0    1    2    3    4    5    6    7
 0800 0806 080C 080F 0000 0000 0000 0000
```

In this example, the breakpoint numbers and their corresponding addresses are:

Breakpoint Number	Breakpoint Address
0	0800
1	0806
2	080C
3	080F
4	Not Set
5	Not Set
6	Not Set
7	Not Set

4.7.3 # Command - Clear Breakpoints

All eight breakpoints may be cleared by using the # command. All breakpoints are also cleared when AIM 65/40 power is turned on or upon a cold RESET. Warm RESET does not alter the breakpoint addresses.

To use the # command, type #. AIM 65/40 will respond with:

```
{#} OFF
```

This indicates that all the breakpoints have been cleared. You may verify it with the ? command.

4.7.3 4 Command - Toggle Breakpoint Enable On/Off

The 4 command toggles the breakpoint enable ON or OFF. This allows all breakpoint checking to be disabled temporarily without requiring the breakpoints to be re-entered later. When the breakpoint enable is ON the breakpoints in the program are checked during execution in single step or run mode.

Normally, breakpoint addresses are assigned with the B command and then enabled with the 4 command.

Examples:

```
{4} ON
{4} OFF
```

In the above, the breakpoints were enabled (toggled ON) when the first 4 command was entered. The breakpoints were disabled (toggled OFF) when the second 4 command was entered.

NOTE

> Breakpoints are disabled by a cold RESET or upon power-up. Breakpoint addresses are not initialized by a cold reset. Random data will therefore by stored in the breakpoint addresses upon power turn-on. Before enabling the breakpoints after a power turn-on, be sure to first use the # command (see Section 4.7.3) to initially clear them and use the B command (see Section 4.7.5) to set the proper breakpoint addresses.

4.7.5 B Command - Set Breakpoint

The B command allows you to specify the address for any of the eight breakpoints (breakpoint Ø through breakpoint 7). The breakpoint must be an address in the application program. The breakpoint can be specified as an absolute hexidecimal address or as a symbol. If entered as a symbol, the symbol must be assigned a value in the symbol table, either manually using the ";" command (see Section 4.1Ø.2) or automatically by the optional assembler or other compiler. Symbolic entry is convenient since you do not have to look up the absolute address to enter the breakpoint.

Anytime an assembly is performed since the last entry of breakpoints, you will have to update the breakpoints if any of the absolute or symbolic specified addresses have changed (which they probably will).

Use the B command as follows:

a. Type B. AIM 65/4Ø will respond with:

```
{B}BRK/
```

b. After the / prompt, specify the breakpoint to be set by entering a digit between Ø and 7. AIM 65/4Ø will respond by printing the number of the breakpoint entered and an = prompt. For example, if Ø is entered:

```
{B}BRK/Ø=
```

c. To set a breakpoint, enter the hexadecimal address or ";" followed by the symbol where the program is to halt.

To clear a breakpoint, enter Ø for the address. Although cleared, the disassembled instruction at address Ø will be displayed.

d. After the address has been entered, type RETURN or SPACE. control will return to the Monitor. If entered as an address, the instruction at that address will be disassembled and displayed. If entered as a symbol, the symbol value (now the address) will be displayed in addition to the disassembled instruction. If the symbol is not found in the symbol table, the typed symbol will be blanked to request re-entry of the sumbol name. Upon completion, re-enter the B command to set or clear additional breakpoints.

Example:

```
{B)BRK/Ø=Ø8ØØ   START   NOP
{B)BRK/1=;CYCLE   Ø8Ø6   CYCLE  LDA CTR
{B)BRK/2=Ø              BRK
```

In the above example, breakpoint Ø was set to location $Ø8ØØ, breakpoint 1 was set to location $Ø8Ø6 and breakpoint 2 was set to location $ØØØØ (i.e., cleared).

4.8 LOAD/DUMP MEMORY

Two commands allow R65ØØ object code to be loaded into memory from an input device or dumped from memory to an output device.

4.8.1 L Command - Load Memory

The L command loads object code from any system device into memory.

Use the L command as follows:

a. Type L. AIM 65/4Ø will respond with:

```
{L} OFFSET=ØØØØ
```

b. If an offset other than Ø is desired, enter the offset in hexadecimal then RETURN; otherwise just press RETURN. The offset will be added to the address on the input media to specify its address in memory (with overflow ignored to allow wraparound to lower addresses). The AIM 65/4Ø will respond with:

```
IN=
```

c. Type the code of the input device from which the object code is to be loaded:

Input Device	Input Device Code	Subprompt Reference
Keyboard	RETURN or SPACE	
Memory	M	
Floppy disk (user defined)	F	
User defined	U	
User Defined	V	
Audio Tape, AIM 65/4Ø Format	T	9.3.2
Serial (user defined)	S	

d. Respond to subprompts.

e. AIM 65/40 will load the object code from the selected device into memory. When all the code has been loaded, the AIM 65/40 will display DONE and return to the Monitor command level.

If any of the records being read contain a checksum error, or if any part of the memory fails to write, an error message will be printed (see Appendix D), indicating the first addresss of the record which caused the error.

Example: Load object code from an audio cassette recorder (optionally connected to recorder remote control line 1) and previously dumped with a file name of SEQ1

```
{L} OFFSET=0000 <RETURN> IN=T UNIT=1 FILE=SEQ1 <RETURN>
 DONE
```

4.8.2 D Command - Dump Memory

The D command dumps the contents of a specified portion of memory to an output device in either binary or ASCII format (see Appendix I.2).

Use the D command as follows:

a. Type D. AIM 65/40 will respond by asking for the beginning address:

```
FROM=0000
```

b. Enter the beginning address to be dumped, in hexadecimal or as a defined symbol (after a preeceding ;). An input error may be corrected using DEL, then continue to enter a number up to four digits or a symbol up to six digits. End the input with RETURN or SPACE.

If 0800 was entered, AIM 65/40 will respond by asking for the dump ending address:

```
FROM=0800  TO=0000
```

c. Enter the ending address to be dumped, in hexadecimal, or as a defined symbol (following a ;). An input error may be corrected in the same manner as in the beginning address. End the input with a RETURN or SPACE. If 0840 was entered, AIM 65/40 will respond with:

```
FROM=0800  TO=0840  OFFSET=0000
```

d. Enter the offset, if any then press RETURN. An offset allows the output starting address to differ from the FROM= location of the data dumped from memory by a specified amount. The system responds with:

```
FROM=0800  TO=0840  OFFSET=0000  MORE?
```

NOTE

The offset value entered in step d does not change the starting location from which the data is dumped; it changes only the starting address on the output data:

If OFFSET=0 and FROM=2000, the output starting address on the media = 2000.

If OFFSET=1000 and FROM=2000, the output starting address on the media = 3000.

If OFFSET=F000 and FROM=2000, the output starting address on the media = 1000.

e. If another section of memory is to be dumped, enter Y and refer to step a. If N or RETURN is entered the system responds with:

```
TYPE=
```

f. Enter the type of dump format:

(1) Enter A if the data is to be dumped in ASCII format (see Appendix I.2). This format is required if the dumped data is to be displayed. The system will respond with

```
TYPE=A  OUT=
```

(2) Enter B if the data is to be dumped in the binary format (see Appendix I.2). This format is best for mass storage since it is about 8Ø% more dense than ASCII. The data cannot be displayed or printed in this format. The system will respond with

```
TYPE=B DUMP SYMBOLS?
```

Type Y if the symbol table is to be dumped after the other data is output. Type N (or anything else besides Y) if it is not to be dumped.

g. Enter the code of the output device where the dump is to be directed:

Desired Output Device	Output Device Code	Subprompt Reference
System display printer	SPACE or RETURN	
AIM 65/4Ø printer	P	
Floppy disk (user defined)	F	
User defined	U	
User defined	V	
Dummy	X	
Audio Tape, AIM 65/4Ø format	T	9.3.1
Serial (user defined)	S	

h. The memory contents will be dumped to the specified output device in R65ØØ object code format. Upon completion, the AIM 65/4Ø will display DONE and return to the Monitor command level.

Example 1: Dump memory locations $8ØØ to $966 to an audio tape file called DUMP1 located on the tape recorder number 1.

```
    {D}
     FROM=Ø8ØØ <RETURN> TO=Ø966 <RETURN> OFFSET=ØØØØ <RETURN>
  MORE?N
     TYPE=A  OUT=T UNIT=1 FILE= DUMP1
```

Example 2: Dumps memory locations $8ØØ to $966 to tape file called DUMP2 located on tape recorder number 2. In addition, dump from location $ØAØØ to location $ØA8Ø.

```
    {D}
     FROM=8ØØ <RETURN> TO=966 <RETURN> OFFSET=ØØØØ <RETURN>
     MORE?Y
     FROM=AØØ <RETURN> TO=A8Ø <RETURN> OFFSET=ØØØØ <RETURN>
     MORE?N
     TYPE=A  OUT=T UNIT=2 FILE=DUMP2
```

Example 3: Dump from locations $FØØØ to $FØ1Ø and from $F8ØØ to $F82Ø with no offset. Direct the output to the display/printer. Note that the dump type must be ASCII (TYPE=A) for display or printing.

```
    {D}
     FROM=FØØØ <RETURN> TO=FØ1Ø <RETURN> OFFSET=Ø <RETURN>
     MORE?Y
     FROM=F8ØØ <RETURN> TO=F82Ø <RETURN> OFFSET=Ø <RETURN>
     MORE?N
     TYPE=A  OUT=
    ;1EF8ØØ7EØ21ØØ52C7FØ21ØF66Ø2Ø25F9A91B2Ø5
    DF3A9582Ø5DF3A94C2Ø5DF3AØØØØCA6
    ;Ø3F81E8CB3FFØ357

    ;11FØØØFØØØØØ4CØ3FØ5CFØ85FØ62FØ8BFØ56FØ9
    DØAA1
    ;ØØØØ4ØØØ4
     DONE
```

Example 4: Dump from locations $4ØØØ to $FØ2Ø with an offset of $4ØØØ. Direct the output to the printer.

```
    {D}
     FROM=FØØØ TO=FØ2Ø OFFSET=4ØØØ MORE?N
     TYPE=A   OUT=P
    ;1E3ØØØFØØØØØ4CØ3FØ5CFØ85FØ62FØ3BFØ56FØ9
    DFØ79FØ79FØ79FØ79FØ79FØ79FØ1354
    ;Ø33Ø1E79FØ79Ø233
    ;ØØØØ3ØØØ3
     DONE
```

4.8.3 F Command - Verify Memory

The F command verifies the contents of memory (object code) with the contents of a file on a system device. The offset capability allows memory that was loaded with an offset also to be verified. The offset is additive in the same manner as described for the L command (see Section 4.8.1). The contents of both memory and the reference file are displayed along with the address if a mismatch in value is detected. Memory can be verified against a file recorded in, either the binary format or ASCII format.

Use the F command as follows:

a. Type F. AIM 65/4Ø will respond with:

```
{F} OUT=
```

b. Enter the single character output device code specifying where the list of verify errors are to be displayed. For example, if the output is to the display/printer, press RETURN, AIM 65/4Ø responds with:

```
{F} OUT=   OFFSET=ØØØØ
```

c. Enter the offset value (if applicable) to four digits in hexadecimal and end with a RETURN. A RETURN or SPACE defaults to Ø when no value is entered. AIM 65/4Ø responds with:

```
{F} OUT=  OFFSET=ØØØØ  IN=
```

d. Enter the single character input device code (see Appendix A) and respond to subprompts.

e. The AIM 65/4Ø verifies the contents of each data byte read from the input media to the data in memory at the corresponding address. If any errors are detected, the address of the error, the memory contents and the input media value are displayed in this order with up to three addresses on a line.

f. After all input bytes are verified, the AIM 65/40 displays:

```
DONE
```

and returns to the Monitor command level.

Example: Verify object code from file SEQ4 on an audio recorder connected to remote control line no. 1. Direct the error list to the display/printer. Assume errors at $1000 through 1005.

```
{F} OUT=<RETURN> OFFSET=0000 <RETURN> IN=T UNIT=1
    FILE=SEQ1 <RETURN>

SEQ1 00 R
1000 83 C3    1001 03 43    1002 DE 5E
1003 82 C2    1004 90 D0    1005 D5 55
```

4.9 PERIPHERAL CONTROL

The peripheral control commands allow you to control the printer and audio tape recorders.

4.9.1 CTRL P Command - Toggle Auto-Print On/Off

The CTRL P command turns the auto-print control on if it is off, and off if it is on.

The command is entered by pressing the P key while the CTRL key is depressed. The status of the printer control will be indicated by an AUTO-PRINT OFF or AUTO-PRINT ON printout.

When the auto-print is ON, it will print everything that is displayed upon a carriage return and line feed. If the printer control is OFF, information will be printed only when the PRINT key is pressed or when data is directed to the printer (OUT=P).

NOTE

> The printer control is turned ON when the AIM 65/40 power is applied or a CTRL RESET is commanded. Subsequent warm RESETs will not change the active state of the printer control.

4.9.2 PRINT Command - Print Display Contents

The PRINT command causes the contents of the display line (including non-displayed characters) to be printed. Print will occur when the PRINT key is pressed, regardless of the auot-print state.

4.9.3 1 or 2 Command - Toggle Recorder 1 or 2 Control On/Off

Two commands control the audio tape devices. The tape recorder has a control number identifying the device. Tape Control 1 should be connected to tape recorder number 1 for simplicity.

a. Type 1, to toggle Tape Control 1 ON or OFF.

Tape 1 Control is normally connected to tape recorder number 1 remote jack. If so, the tape recorder will not record, play, advance of rewind until Tape 1 control is ON.

The Monitor and Editor commands requiring tape recorder number 1 operation will command the Tape 1 control ON when required and turn OFF upon completion of the command.

These commands are L (Load), D (Dump) and F (Verify) in the Monitor and R (Read) and L (List) in the Editor. To manually operate tape recorder number 1, the Tape 1 Control must be turned ON using the 1 command.

NOTE

> The Tape controls are turned off when AIM 65/40 power is turned on or a CTRL RESET is commanded. Warm RESET does not change the active state of the Tape controls.

b. Type 2, to toggle tape control 2 On or Off.

Tape 2 Control operates in the same manner as Tape 1 Control. Refer to Step a. for the explanation.

4.9.4 3 Command - Verify Tape Checksum

The 3 command is used to verify that the block checksum for either source or object code was properly recorded on audio tape when using the Dump Command. Tape verification should be performed before the contents of memory are altered, in case the data was not properly recorded.

Use the 3 command as follows:

Type 3, AIM 65/4Ø will respond with:

a. {3} UNIT=

b. Type 1 or 2. For example, AIM 65/4Ø will respond with:

```
{3} UNIT=1  FILE=
```

Enter the file name as shown in Section 9.1.5. The operation will continue as described in that section.

Example:

```
{3} UNIT=1  FILE=OBJ
```

4.1Ø F1-F8 KEYS - USER FUNCTION COMMANDS

The preceding portions of this section describe a large number of single-keystroke commands and how the AIM 65/4Ø Monitor responds to them. All of those commands perform very specific pre-defined functions -- alter a memory location or a register, initiate program execution, set or clear breakpoints, and so on.

This section describes how to link the eight function keys (F1-F8) to user defined functions. This linkage allows you to command any one of these functions by pressing the assigned function key from the Monitor command level. (Note, however, that the eight function keys are assigned specific commands in the Editor, see Section 5.6).

Associated with each of the eight keys is a two-byte vector (see Monitor constants in Table 15-2) that points to the starting address of the corresponding user function. This vector is formatted in low-byte, high-byte order (e.g., 5008 = address $0850) that can be generated by the optional assembler .WORD directive.

Use the F1-F8 keys as follows:

a. Enter the user function in memory. The routine must end with a JMP to COMIN1 ($A314) or an BRK instruction to return to the Monitor command level upon completion. The return to CONJN1 causes only the Monitor prompt to be displayed whereas the BRK instruction will display one line of disassembled machine code.

b. Store a vector pointing to the entry address of the user function in the associated function key vector location (see Table 15-2):

Key	Vector Location	Displayed Number
F1	$250-$251	{1}
F2	$252-$253	{2}
F3	$254-$255	{3}
F4	$256-$257	{4}
F5	$258-$259	{5}
F6	$25A-$25B	{6}
F7	$25C-$25D	{7}
F8	$25E-$25F	{8}

c. Press the appropriate function key --- F1 through F8. AIM 65/40 will display the number (in superscript format) corresponding to the key and will jump to the associated user function routine.

Example 1: Enter a user function starting at $Ø8ØØ which returns to the Monitor. Also, link to the F1 key using mneumonic instruction entry (I command, see Section 4.5.1) and alter memory (/ command, see Section 4.4.4) functions.

```
{I}
 XXXX                  *=Ø8ØØ
 Ø8ØØ                  (Enter user function instructions)
   .
   .
   .
 XXXX  4C  Ø3  A3      JMP A314
 XXXX                  (ESC)
{M}25Ø  22 A3 22 A3 22 A3 22 A3 ".".".".
{/}Ø25Ø ØØ Ø8 22 A3 22 A3 22 A3 ".".".".
```

Example 2: Assemble (using the optional assembler) a user function starting at $ØAØØ which returns fo the Monitor. Link the entry to the F2 key. The source code fragment is:

```
COMIN1=$A314
*=$252
.WOR F2ENT

*=$ØAØØ
F2ENT                  (Enter user function instructions)
  .
  .
  .
JMP COMIN1
.END
```

SECTION 5

AIM 65/40 TEXT EDITOR DESCRIPTION

The AIM 65/40 Text Editor provides you with the capability of manipulating, or editing, blocks of text -- called files -- that reside in the AIM 65/40 RAM memory and inputting and outputting these files. What is text? Text is simply one or more strings of ASCII characters that represent data, messages or program instructions in memory. A file is, then, a series of ASCII strings that represents a data table, a set of messages, or a program in memory.

The Text Editor is primarily used to enter source code programs that will be "assembled" or "compiled" (translated into R6502-compatible instructions) by the optional AIM 65/40 software or firmware. The Text Editor can also be used to enter data tables and message information that can be output to ASCII-based peripheral devices such as the AIM 65/40 display and printer. In either application, the Text Editor acts as nothing more than a simple input/editing/output program. The Text Editor makes no qualitative decisions as to what kind of information it is processing -- data characters, message characters and program instruction characters are all treated as "text", nothing more, nothing less.

The Text Editor allows program and text data in ASCII character coding to be entered and manipulated using various line, string and screen oriented commands. Line oriented functions can enter, delete, list and display data at the single or multiple line level. String oriented commands locate or change variable length character strings. Screen oriented commands replace, insert and delete characters at the character level within a line of text. Single character or block change operations may be performed. Table 5-1 lists the Editor commands.

Table 5-1. AIM 65/40 Editor Commands

Category	Command	Function
Editor Entry from Monitor	E C T	Initialize Text Buffer and Enter Editor Recover Text Buffer and Re-enter Editor Re-enter Editor
Editor Control	S ESC Q + CTRL C CTRL N @	Enter Screen Edit Mode Escape to Editor Command Level Quit Editor and Enter Monitor Repeat Last Command Clear Display and Home Cursor Home Cursor Enter Data Output Rate
Line Oriented	R I U O K D T B L ? G SPACE	Read Multiple Lines Insert One Line Go Up Multiple Lines Overlay Current Line Delete (Kill) Multiple Lines Go Down Multiple Lines Go to Top Line Go to Bottom Line List Multiple Lines Display Line Addresses Go To Line Number Display Current Line
String Oriented	F C	Find Character String Change Character String
Screen Oriented	F1/CTRL Q F2/CTRL R F3/CTRL S F4/CTRL T F5/CTRL U F6/CTRL V F7/CTRL W F8/CTRL X CTRL A CTRL A CTRL D	Home Cursor On Line Clear Line To Right Toggle Insert Mode On/Off Delete Character At Cursor Move Cursor Left Move Cursor Right Move Line/Cursor Down Move Line/Cursor Up Add a Line Break a Line Delete a Line
Peripheral Control	CTRL P 1 2	Toggle Auto-Print On/Off Toggle Recorder 1 Control On/Off Toggle REcorder 2 Control On/Off

5.1 AIM 65/40 TEXT EDITOR FEATURES

The Editor features are summarized below.

- Complete User Control of Memory - Text is edited in system memory, with complete user control of which memory locations are used by the Editor.

- I/O Device Flexibility - Input to the Editor may come from any system device. Output from the Editor may be directed to any system device.

- Convenient Line Oriented Editing Commands are provided to:

  - Go to the top or the bottom of the text
  - Go to any given line of the text
  - Go up or down any number of lines
  - Find a given string
  - Change a given string to another string
  - Change multiple occurrences of a character string to another string
  - List one or more lines to any system device
  - Insert one or more lines at the current location of the text pointer, with input coming from any system device
  - Replace a line
  - Delete any number of lines
  - Insert/delete characters at the current line
  - Show the addresses of the active line and the last line of the text

- Flexible Screen Oriented character commands are provided which:

  - Move the cursor right and left
  - Move the line (cursor up and down)
  - Replace and insert characters
  - Delete under and to the left of the cursor
  - Delete under and to the left of the cursor

    - Insert carriage returns to split text lines
    - Home cursor and clear the display

- Multiple String Changes - Automatic and selective character string block change capability allows easy altering of labels, symbols and other character string data. A user specified limit to the number of lines to scan allows absolute control over change incorporation. Display of text lines before and after change incorporation allows operator verification.

5.1.1 Text Buffer

The text is stored in an area of RAM called the Text Buffer. Upon initial entry into the Editor, the user must define the starting and ending limits of the Text Buffer. The default limits in AIM 65/40 memory can be used to allocate all available memory to the Text Buffer. The default lower limit is $2000 and default upper limit is $3FFF. This allows $0800 - $17FF to be used for application program object code and $1800 - $1FFF to be used for the symbol table. All of these default locations are user alterable.

Data is stored in the Text Buffer in ASCII format (see Appendix E for the ASCII character codes). Each character entered requires one byte (8 bits) of RAM. The text is stored in variable length lines; each text line ends with a carriage return ($0D) after the last text character. Line feed ($0A) characters are not stored.

A null character ($00) is stored after the last text line to indicate the end of active text. Be careful not to store a $00 in the active text area of the Text Buffer; otherwise, all text past the $00 will not be accessible using normal Editor commands.

To estimate the amount of RAM required for the Text Buffer, allow one byte for each text character, one byte for each line ending carriage return ($ØD) character and one byte for the text ending $ØØ. Additional memory should be allocated to allow for text additions and changes.

The address limits of the Text Buffer as well as the ending address of the active text can be determined at any time be examining the following dedicated memory locations:

		Example	
Address	Parameter	Contents	Address
ØØFA-ØØFB	Text Current Line	$3Ø25	$253Ø
Ø326-Ø327	Text Ending Address	$4235	$2542
Ø26Ø-Ø261	Text Buffer Starting Address	$ØØ2Ø	$2ØØØ
Ø262-Ø263	Text Buffer Ending Address	$FF3F	$3FFF

See Figure 5-1 for an illustration of the text buffer.

5.1.2 Line Pointer

All line oriented Editor operations begin from the start of the active line. The active line is identified by the line pointer, which is always positioned in front of the first character of the active line. Following an Editor operation, the line pointer is positioned to the start of either the last line operated on or one line down from the last line operated on, depending upon the command.

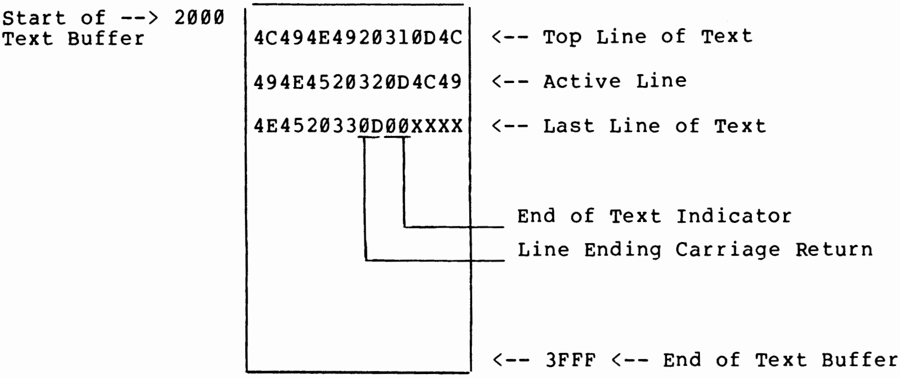
The active line is displayed at the completion of most Editor commands, depending on output device selection. If there is any doubt where the line pointer is positioned, press the SPACE bar to display the active line.

Line pointer positioning commands allow easy manipulation of the line pointer. Using these commands, the line pointer can be easily and quickly moved to the top of the text, to the bottom of the text, up one line or down one line.

```
{E}
 EDIT FROM=2000 TO=3FFF IN=
LINE 1    (Top Line of Text)
LINE 2
LINE 3    (last Line of Text)


 *END*
=(T)
LINE 1
={D}/1
LINE/2    (Active Line of Text)
```

a. Example Text



b. Text Buffer Illustration

Figure 5-1. Example Text Buffer

5.1.3 Dummy Line

A dummy line is provided after the last active line to allow new text to be added at the end of the existing text. If the line pointer is positioned on the last active line of text in the Text Buffer, it must be moved down one line to the dummy line using the D or F7 commands in order to then read or insert new text after the last line of the active text.

When the line pointer is positioned on the dummy line, *END* will be displayed.

5.2 EDITOR ENTRY COMMANDS

Three commands permit the Editor to be entered from the Monitor. One command initializes the Text Buffer upon entry, the second allows re-entry to the Editor after relocating the Text Buffer and the third allows re-entry to Text Buffer.

While these commands can be classified technically as Monitor commands, they are explained here since they are associated with the Editor operation.

The Editor command level prompt is = (equal sign). When this prompt is blinking in position one, an Editor command (see Table 5-1) may be entered. Many commands respond with subprompts to request entry of additional information. Enter the requested data as described in the following sections.

Note that the * (asterisk) cursor indicates that an input character string is required in the line oriented read or insert commands, in the find or change character string commands and in the screen edit character replace mode. The "Y" cursor indicates character string input is required in the screen edit character insert mode. Refer to Section 5.6 for more information.

5.2.1 Monitor E Command - Initialize the Text Buffer and Enter the Editor

The Monitor E command enters the Editor, initializes the Text Buffer limits then automatically enters the text input mode. Use this command to enter the Editor for the first time prior to reading text into the text buffer or when the existing text is no longer needed (and has been saved if needed in the future) and you want to re-initialize the Editor to accept other text.

NOTE

> If you want to recover existing text, use the Monitor command (see Section 5.2.2).

Use the E command as follows:

a. After the Monitor prompt, type E. The Editor will display the last entered starting address of the Text Buffer (the cold RESET value is $2ØØØ):

```
{E}
 EDIT FROM=2ØØØ
```

b. If you want to change the starting address, enter the new value as either a hexadecimal number or as a symbol (preceded by ;). End the entry with RETURN or SPACE. RETURN or SPACE without entering an address will default to the displayed value.

   The Editor will then display the last entered ending address of the text buffer (the cold RESET value is $3FFF):

```
EDIT FROM=2ØØØ TO=3FFF
```

c. If you want to change the ending address, enter the new value and terminate it like you did for the starting address. It will also default to the displayed value if you press RETURN or SPACE without entering a new value.

The Editor will then request entry of the input device code:

```
  EDIT FROM=2ØØØ TO=3FFF IN=
```

e. Enter the code identifying the input device from which the text is to be entered (see Appendix A).

f. After entering the input device code and responding to any subprompts, the Editor automatically enters the read text into the Text Buffer function (R command). Refer to Section 5.4.1 step(a) for instructions on how to enter text in this function.

Example 1: Enter the Editor, set up different text buffer starting and ending addresses and input text from an audio cassette recorder.

```
 {E}
  EDIT FROM=1ØØØ <RETURN> TO=1FFF <RETURN> IN=T UNIT=1 FILE
  =TXT1
```

Example 2: Enter the Editor, use the cold RESET text buffer starting and ending values and input text three lines of text from the keyboard.

```
 {E}
  EDIT FROM=2ØØØ<RETURN> TO=3FFF<RETURN> IN=<RETURN>
 THIS IS THE TOP LINE OF TEXT
 THIS IS THE SECOND LINE OF TEXT
 THIS IS THE THIRD LINE OF TEXT
 <RETURN>
  *END*
```

5.2.2 Monitor C Command - Recover the Text Buffer and Re-enter the Editor

The Monitor C command creates a text buffer around ASCII data already in RAM then re-enters in the Editor. This command is useful for recovering or restoring a text buffer that was unintentionally initialized (using the Monitor E command) or deliberately moved while working with two or more text files in RAM. It can also be used to create a text buffer about assembler source code in memory that was disassembled using the K command (see Section 4.5.2).

The ASCII data must meet the requirements of data in the text buffer (see Section 5.1), notably:

- A carriage return ($ØD) must be included within each 8Ø characters (since the line buffer is 8Ø bytes in length.

- A null ($ØØ) must terminate the text after the last carriage return.

Use the C command as follows:

a. After the Monitor prompt, type C. The Editor will display with the Editor prompt and the last entered starting address (the cold RESET value is $2ØØØ):

```
{C}
 EDIT  FROM=2ØØØ
```

b. If you want to change the starting address, enter the new value as either a hexidecimal number or as a symbol (preceded by ;). End the entry with RETURN or SPACE. RETURN or SPACE without entering an address will default to the displayed value.

The Editor will then display the last entered ending address of the text buffer (the cold RESET value is $3FFF):

```
EDIT FROM=2ØØØ TO=3FFF
```

c. If you want to change the ending address, enter the new value and terminate it like you did for the starting address. It will also default to the displayed value if you press RETURN or SPACE without entering a new value.

d. The Editor will be re-entered, the line pointer set to the top line and the top line displayed.

CAUTION

If the data within the enter text buffer limits are not encoded in ASCII (see Appendix F), the results are unpredictable.

NOTE

The new text buffer may be larger than the recovered or current text in the buffer. The text terminating $ØØ can be after any $ØD (carriage return) within the text buffer limits.

Example 1: Recover the Text Buffer at the cold RESET text buffer limits.

```
{C}
 EDIT FROM=2ØØØ <RETURN> TO=3FFF <RETURN>
THIS IS THE TOP LINE OF TEXT
={Q}
```

Example 2: Establish symbolic text buffer limits and recover the text buffer symbolically.

```
{;}B1S     =2ØØØ
{;}B1E     =3FFF
{C}
 EDIT FROM=;B1S <RETURN> 2ØØØ TO=;B1E <RETURN> 3FFF
THIS IS THE TOP LINE OF TEXT
```

5.2.3 Monitor T Command - Re-enter the Editor

The Monitor T command re-enters the Editor at the top of the Text Buffer to allow editing of text previous entered into the buffer and displays the first line of text. The line pointer is positioned automatically at the top line.

The command is used during program development to return to the Editor from the Monitor after an assembly or compilation has detected an error in the source code. With the source code in the Text Buffer, errors can be rapidly corrected then assembly or compilation quickly and easily repeated.

If multiple text buffers are used, the Monitor C command (see Section 5.2.2) can be used to enter and edit any text that is present.

Some assemblers or compilers may process text starting with the line pointed to by the line pointer. In this case, use the Monitor T command to re-enter the Editor then other commands to move the line pointer to the desired position. Exiting the Editor at this point (with the Q command, see Section 5.3.2) will retain the line pointer at its current position.

To re-enter the Editor, type T after the Monitor prompt, e.g.

```
{T}
THIS IS THE DISPLAYED TOP LINE OF TEXT
```

5.3 EDITOR CONTROL COMMANDS

The Editor control commands switch control out of the Editor command level, return control to either the Editor or the Monitor command level, or repeat the previous Editor command.

5.3.1 S Command - Enter the Screen Edit Mode

The S command enters the screen edit mode in the same manner as the F6 key. Refer to Section 5.6 for usage information.

5.3.2 ESC-Escape to Editor Command Level

The ESC command escapes from Editor commands or functions in process and returns to the Editor command entry level. The Editor examines the keystack for entry of the ESC command at various points to determine if the current processing is to be terminated.

For example, a check is made after each line of text is output during the list function (L command, see Section 5.4.9). If ESC has been pressed, the output listing process is terminated and control returns to the Editor command level.

To return to the Editor command mode, type ESC. AIM 65/40 will display:

```
(ESC)
```

either on the same or on the next line followed by display of the Editor command level prompt (=) in position one. Editor commands may be now entered.

5.3.3 Q Command - Quit the Editor and Enter the Monitor

To exit the Editor and return to Monitor, type the Q after the Editor prompt:

```
  ={Q}
```

The AIM 65/40 will return to the Monitor command level and display the Monitor prompt. Note that pressing the ESC key while in Editor does not exit the Editor, it only returns to the Editor command entry level.

The Editor may be re-entered with the T or C command.

5.3.4 + Command - Repeat the Last Command

The plus (+) command repeats the last command. This is helpful when repeating commands with long set-ups but should be used cautiously since some commands may cause different results when repeated depending on initial conditions.

To repeat the prior command, type +. The Editor will display

```
  ={+}
```

followed by the prior command and results as it is repeated.

5.3.5 CTRL C Command - Clear Display and Home Cursor

The CTRL C command clears the dsiplay and homes the cursor to position one. This command is handly in the screen edit mode if you have to start the text entry from position one after partially entering it. This command should be used cautiously since it is active at all times. If commanded when another command is partially entered, the Editor will continue to respond to keyboard inputs, however the prior displayed prompts and status will not be visible.

5.3.6 CTRL N Command - Home Cursor

The CTRL N command homes the cursor to character position one without clearing the display. This command should also be used cautiously since it is active at all times. If commanded when another command is partially entered, the Editor will continue to respond to keyboard inputs, however the character input cursor is displaced from its normal input position.

5.3.7 @ Command - Enter Data Output Rate

The @ command inputs a number which determines the rate at which data is output to the display (and printer, if auto-print is on) by a subsequent command that has a variable output rate; e.g., the list lines of text function (L command, see Section 5.4.9). The number may vary from Ø (fastest) to 9 (slowest).

Example:

```
  ={@}Ø
```

5.4 LINE ORIENTED COMMANDS

5.4.1 R Command - Read Multiple Lines

The R command reads multiple lines of text from an input device into the text buffer. The R command may be used to enter lines into an empty text buffer (i.e., when creating a new program) or to add lines to text already stored in the text buffer. Text inserted into the text buffer is inserted in front of the active line. There may be a noticeable pause at the end of each line if text is being inserted in front of already existing lines. End each line of input by pressing <RETURN>.

Type two consecutive RETURNs to end the text input mode. Control will then return to the Editor command level.

If an attempt is made to read more text than may be stored in the text buffer, the following message will be printed:

```
*END*
```

and control is returned to the Editor command level.

Use the R command as follows:

a. Position the line pointer to the line before which you want to enter the new text.

b. After the Editor prompt, type R. The Editor will ask for the input device code:

```
={R} IN=
```

c. Enter the code of the input device from which the text will be entered (see Appendix A).

    (1) Press RETURN or SPACE to enter text from the keyboard. AIM 65/40 will display a flashing cursor to indicate where the next digit is to be entered:

    ```
    *
    ```

    Enter text from the keyboard, terminating each line with RETURN. An input error may be corrected by entering DEL and re-entering the desired character. Text can be entered in either lower or upper case. Lower case letters are indicated on the 40-character display by a period in the lower right corner of the character position.

Up to 79 characters may be entered on a line. The first 40 characters are entered from left to right as seen on the display. Starting with character 40, the displayed data will scroll to the left one character position as each new character is entered. The cursor will remain in the 80th position upon entry of 79 characters. You can then delete characters but you can not add any more.

Terminate the text entry by pressing RETURN without entering any data on the new line.

(2) Type T to enter text from an audio cassette recorder. Refer to Section 9.1.6 for the set-up instructions. The system will ask which tape drive contains the file to be read into the text buffer.

```
={R} IN=T  UNIT=
```

(a) Enter the number of the tape drive number e.g. 1. The system will ask for entry of the file name:

```
={R} IN=T  UNIT=1  FILE=
```

(b) Enter the name of the file that is to be read into the text buffer. End the file name with a RETURN. For example:

```
={R} IN=T  UNIT=1  FILE=PROG1
```

(c) AIM 65/40 will read the named file into the text buffer. When the read operation has been completed, the Editor will display the prompt =, indicating it is awaiting the next editor command. The block count from the file will be displayed along with an "R" as the tape is being read (see Section 9.3.2).

Example 1: Read text into the text buffer from the keyboard.

```
={R} IN=<RETURN>
.PAG 'THIS IS A TEST PROGRAM'
*=$2800
LABEL1 LDA #$34
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
 *END*
```

5.4.2 I Command - Insert a Line

The I command inserts one line of text ahead of the active line. Input is always from the keyboard.

Use the I command as follows:

a. Position the line pointer to the line before which you want to insert the new text.

b. Type I. The Editor will indicate the text entry mode from the keyboard by a blinking asterisk in position one:

```
*
```

c. Enter the line of text (to 79 characters) into the buffer. End the input with a RETURN.

d. The Editor will display the line after the inserted text.

Example: Suppose the program in the text buffer is:

```
.PAG 'THIS IS A TEST PROGRAM'
*=$2800
LABEL1 LDA #$34
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
```

Assume that the active line was the fourth line and the following text was inserted using the I command: Use the D command to go down four lines.

```
STA $45
={I}
;THIS LINE WAS INSERTED
STA $45
```

After inserting the new line, the new program would read (using the T and L commands):

```
.PAG 'THIS IS A TEST PROGRAM'
*=$28ØØ
LABEL1 LDA #$34
;THIS LINE WAS INSERTED
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
```

5.4.3 O Command - Overlay Current Line

The O command overlays (replaces) the active line of text with a new line of text. Input is always from the keyboard.

Use the O command as follows:

a. Position the line pointer to the line (text) to be replaced.

b. Type O. The Editor will blank the line then display the prompt * to indicate the position of the next character to enter.

c. Enter the new line of text and terminate the entry with a RETURN. The new entry is printed out.

Example: Assume the program in the text buffer is (using the T and L commands):

```
=(T}
.PAG 'THIS IS A TEST PROGRAM'
={L}/.  OUT=<RETURN>
.PAG 'THIS IS A TEST PROGRAM'
*=$2800
LABEL1 LDA #$34
;THIS LINE WAS INSERTED
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
 *END*
```

Locate the line to be replaced (line 4) using the T and the D or F7 commands, then replace it with the O command:

```
={T}
.PAG 'THIS IS A TEST PROGRAM'
={D}/1 <RETURN>
;THIS LINE WAS INSERTED
={O}
*;THIS IS THE REPLACING LINE
```

After replacing the line, the updated program is (using T and L commands):

```
={T}
={L}/.  OUT=<RETURN>
.PAG 'THIS IS A TEST PROGRAM'
*=$2800
LABEL1 LDA #$34
;THIS IS THE REPLACING LINE
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
 *END*
```

5.4.4 K Command - Delete Multiple Lines

The K command deletes (or kills) multiple lines of text starting with the active line.

Use the K command as follows:

a. Position the line pointer to point to the first line you want to delete.

b. After the Editor prompt, type K. AIM 65/4Ø will respond with:

```
={K}/
```

c. Enter the number of lines to be deleted and end the input with a RETURN. Any number between 1 and 9999 may be entered. RETURN without entering a number means one line and a period without entering a number means all lines.

d. The Editor will delete the lines as follows:

   (1) If <u>only one line</u> is to be deleted, the Editor displays the line to be deleted (preceded by a slash). It then deletes it immediately.

   (2) If <u>multiple lines</u> are to be deleted, the Editor displays all of the lines to be deleted (preceded by a slash). It then asks for approval to delete them by displaying the prompt message:

   ```
   ARE YOU SURE?
   ```

   (a) Type Y if you want to delete the displayed lines.

   (b) Type any other key (except ATTN or RESET) to indicate that you do not want to delete them.

e. The new current line is then displayed along with the Editor command level prompt displayed in position one to indicate function completion.

For example, assume the program in the Text Buffer is (using the T, L and SPACE commands):

```
={T}
={L}/.  OUT=<RETURN>
.PAG 'THIS IS A TEST POGRAM'
*=$2800
LABEL1 LDA #$34
;THIS LINE WAS INSERTED
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
*END*
```

Locate the line to be deleted (Line 4) using the B and U commands, then delete it with the K and RETURN:

```
={B}
 *END*
={U}/5 <RETURN>
;THIS LINE WAS INSERTED
={K}/<RETURN>
/;THIS LINE WAS INSERTED
STA $45
```

After deleted the desired line of text, the program now looks like this:

```
={T}
={L}/.  OUT=<RETURN>
.PAG 'THIS IS A TEST POGRAM'
*=$2800
LABEL1 LDA #$34
STA $45
INX
JMP LABEL1 ;TRANSFER BACK
.END
 *END*
```

5.4.5 U Command - Go Up Multiple Lines

The U command moves the text pointer up (toward the beginning of the text) from 1 to 9999 lines from the current line. If you attempt to go past the top line of the text, the Editor will set the text pointer at the top line and display:

```
*TOP*
```

In any case, the new active line will be displayed.

Use the U command as follows:

a. After the Editor prompt, type U. The Editor will respond with:

```
=(U)/
```

b. Enter the number of lines that the text pointer is to be moved up and end the input with a RETURN or SPACE. RETURN without entering a number will move up one line while a period or a SPACE will move to the top line. The Editor will move the pointer up the specified number of lines and print the new active line.

```
={U}/<RETURN>
LABEL1 LDA #$34
={U}/2<RETURN>
.PAG 'THIS IS A TEST PROGRAM.
```

5.4.6 D Command - Go Down Multiple Lines

The D command moves the line pointer down (toward the end of the text) from 1 line to the last line. The text pointer advances the indicated number of lines and displays the new active line. If you attempt to move the pointer past the last line of the text, the line pointer will be positioned one line below the last text line and the following message will be displayed:

```
*END*
```

Use the D command as follows:

a. After the Editor prompt, type D. The Editor will respond with:

```
={D}/
```

b. Enter the number of lines that the text pointer is to be moved down and end the input with a RETURN or SPACE. RETURN with entering a number will move down one line while a period or a SPACE will move down to one line past the bottom, i.e., to the *END*. The Editor will move the pointer down the specified number of lines and display the new active line.

5.4.7 T Command - Go to Top Line

The T command moves the line pointer to the top line of text buffer.

Use the T command as follows:

After the Editor prompt, type T. The Editor will move the text pointer to the top line of the program and display that line. The system responds with:

```
=[T]
DISPLAYED TOP LINE
```

5.4.8 B Command - Go to Bottom Line

The B command moves the line pointer to the last line of text in the text buffer and displays the line contents. If no data has been entered into the text buffer, *TOP* will be displayed.

Use the B command as follows:

a. After the Editor prompt, type B. The Editor will move the text pointer to the last line of text and display of that line.

```
={B}
LAST LINE OF TEXT
```

b. To add text (any number of lines) to the end of the text currently in the text buffer, follow the B command with a D/1 or F7 command. This causes the line pointer to be moved down one line to a dummy line following the last line of actual text and to display *END*. Use the R or I command at this point to insert text ahead of the dummy line.

   Example:

```
=(B)
LAST LINE OF TEXT
={D}/1 <RETURN>
 *END*
={I}
THIS IS THE NEW LAST LINE OF TEXT
 *END*
```

5.4.9 L Command - List Multiple Lines

The L command lists one or more lines of text to any output device starting with the current line, for as many lines as are indicated. The L command can be used to save all or part of the contents of the text buffer on tape or user defined devices. The line pointer moves with the listed text or stays at the current line, depending on the type of command termination.

Use the L command as follows:

a. Position the line pointer to the first line you want to list.

b. After the Editor prompt, type L. The Editor will respond with:

```
={L}/  OUT=
```

c. Enter the number of lines to be listed (from 1 to 9999). End the line count entry with RETURN or SPACE; RETURN will cause the line pointer to be moved along with the listed lines while SPACE will cause the line pointer to remain on the first line to be listed. RETURN without entering a number will list one line and move the line pointer down one line. A . (period) without entering a number will list all the following lines and will move the line pointer to the bottom (*END*). A SPACE without entering a number will list all the following lines but will keep the line pointer at the first line to be entered. The Editor then asks for entry of the output device code:

```
={L}/  OUT=
```

d. Type the output device code (see Appendix A) and respond to any subprompts requesting more information.

e. AIM 65/40 will list the specified number of Text Buffer lines, beginning with the active line and ending with the last specified line, to the output device. If output is to the audio cassette recorder, a count of each 8Ø character block will be indicated as it is listed.

A list operation can be terminated at any time by pressing ESC. Pressing SPACE will stop and resume the printout.

Example: List five lines to the display/printer.

```
.PAG 'THIS IS A TEST PROGRAM'
={L}/5<RETURN>
.PAG 'THIS IS A TEST PROGRAM'
=$28ØØ
LABEL1.  LDA #$34
STA $45
INX
*END*
```

5.3.1Ø ? Command - Display Line Addresses

The ? command shows the addresses of the current line and the last line, in hexadecimal.

Use the ? command as follows:

a. After the Editor prompt, type ? The Editor responds with:

```
=(?}XXXX YYYY
```

b. The first hexadecimal number (XXXX) is the Text Buffer active line address. The second hexadecimal number (YYYY) is the text buffer last line address.

5.4.11 G Command - Go to Line Number

The G command moves the text pointer to a given line number. Even though line numbers are not displayed by the AIM 65/4Ø, each line contains an implicit line number.

Use the G command as follows:

a. After the Editor prompt, type G. The Editor responds with:

```
={G}/
```

b. After the / prompt, enter the number of the lines (1 to 9999) that the text pointer is to be moved downward. A RETURN with no number defaults to 1. A RETURN or SPACE after a line number goes to the specified line. A RETURN or SPACE without a line number goes to the bottom of text (*END*).

c. The system will automatically move the line pointer to the new line and display the line contents.

Example:

```
={T}
.PAG 'THIS IS A TEST PROGRAM'

={G}/4 <RETURN>
INX
```

5.4.12 SPACE Command - Display Current Line

The SPACE command displays the contents of the active line. It is normally used after the line has been edited to see the results of the editing.

To use the SPACE command, press the SPACE bar. The Editor responds by displaying the active line.

```
={ }
```

5.5 STRING ORIENTED COMMANDS

5.5.1 F Command - Find Character String

The F command finds a specified character string of up to 2Ø characters. The search for the string starts at the beginning of the current line and continues until the first occurrence of the string, or until the end of the text is encountered. If the complete text file is to be searched, first move the line pointer to the top of text. This command may be used to locate a particular line of text to delete, to locate a text line for reference prior to an insert or read command or to determine if a certain character string exists in the text buffer.

Use the F command as follows:

a. After the Editor prompt, type F. The Editor responds with:

```
={F}*
```

b. Enter the character string that is to be found. Enter the minimum number of characters to uniquely identify the desired character string. Note that SPACE is a valid text character. End the input with a RETURN.

c. The Editor scans the text looking for the first occurrence of the entered string.

(1) If the string is found, the system displays the line that contains the string and positions the line pointer at the beginning of that line.

(2) If the string is not found, the end of text message is displayed (*END*).

(3) If the string is found, but is not on the desired line, type F and RETURN to resume the search. Upon locating the next occurrence of the entered string, the line containing the string is displayed. If one line contains several occurrences of the string, that line will continue to be displayed until the search continues past that line.

Example 1: Find the line containing the word SECOND (first enter the example text):

```
{E}
 EDIT FROM=2ØØØ<RETURN> TO=3FFF<RETURN> IN=<RETURN>
THIS IS THE TOP LINE OF TEXT
THIS IS THE SECOND LINE
THIS IS THE THIRD LINE



 *END*

={T}
THIS IS THE TOP LINE OF TEXT
={F}SEC<RETURN>
THIS IS THE SECOND LINE
```

Example: Find the third occurrence of the word LINE.

```
={T}
THIS IS THE TOP LINE OF TEXT
={F}LINE<RETURN>
THIS IS THE TOP LINE OF TEXT
={F}<RETURN>
THIS IS THE SECOND LINE
={F}<RETURN>
THIS IS THE THIRD LINE
```

5.5.2 C Command - Change Character String

The C command changes one or more occurrences of a specified character string ("old") to another specified character string ("new"). It operates by scanning the text from the current line for all occurrences of the old string. The number of lines to scan can also be specified in order to limit the search area. Within the scanned area, multiple changes can be made either automatically or upon operator indication that a specific occurrence of the old string is to be changed or skipped. Strings of up to 2Ø characters may be changed to strings of up to 4Ø characters.

Use the C command as follows:

a. Position the line pointer either on or preceeding the line containing the string to be changed.

b. After the Editor prompt, type C. The system prompts to request entry of the character string to change:

```
={C}  OLD=
```

c. Enter the old string (up to 2Ø characters) and end the input with a RETURN. The system will display the old string followed by the prompt to request entry of the new string. If LINE is entered as the old string, the response will be:

```
={C}  OLD=LINE  NEW=
```

d. Enter the new string (up to 4Ø characters) and end the input with a RETURN. The system will display the new string then ask for number of lines to scan and how to make the changes. If TIME is entered as the new string, the response will be:

```
={C}  OLD=LINE  NEW=TIME /
```

e. Enter the number of lines (1 to 9999) to be scanned for the old string. End the entry with a RETURN, SPACE or period (.). Typing <RETURN> without entering a number causes only the current line to be scanned, while typing a SPACE or period causes the current and all following lines to the end of the text to be scanned.

The method of change, i.e. automatic or selective, depends on the line count termination entry.

(1) Press RETURN or period (.) to automatically change all occurrences of the old string to new string.

(2) Press SPACE to selectively change all occurrences of the old string to the new string. Upon locating each occurrence of the old string, the line containing the old string is displayed. The cursor is positioned over the first character of the old string in question. The text is scrolled to position the old string onto the display if it is located more than 4Ø characters from position one.

(a) Press RETURN to change the located old string to the new string and to then advance the character pointer to the next occurrence of the old string.

(b) Press SPACE to skip the located old string and to then advance the character pointer to the next occurrence of the old string.

f. All occurrences of the old string are displayed along with all changes to each located string.

Example 1: Automatically change all occurrences of 3 to 999 on all lines (first enter the initial text into the text buffer):

```
{E}
 EDIT FROM=2ØØØ<RETURN> TO=3FFF<RETURN> IN=<RETURN>
X1 X2 X3 X4 X5
Y1 Y2 Y3 Y4 Y5
Z1 Z2 Z3 Z4 Z5

 *END*
={T}
X1 X2 X3 X4 X5
={L}/.    OUT=<RETURN>
X1 X2 X3 X4 X5
Y1 Y2 Y3 Y4 Y5
Z1 Z2 Z3 Z4 Z5
 *END*
={T}
X1 X2 X3 X4 X5
={C} OLD=3<RETURN> NEW=999<RETURN> /.
X1 X2 X3 X4 X5
X1 X2 X999 X4 X5

Y1 Y2 Y3 Y4 Y5
Y1 Y2 Y999 Y4 Y5

Z1 Z2 Z3 Z4 Z5
Z1 Z2 Z999 Z4 Z5


 *END*
={T}
X1 X2 X999 X4 X5
={L}/.    OUT=<RETURN>
X1 X2 X999 X4 X5
Y1 Y2 Y999 Y4 Y5
Z1 Z2 Z999 Z4 Z5
 *END*
```

Example 2: Selectively change two occurrances of 1 to 111 on three lines:

```
={T}
X1 X2 X999 X4 X5
={C} OLD=1<RETURN> NEW=111<RETURN> /3<SPACE>
X1 X2 X999 X4 X5
<RETURN>
X111 X2 X999 X4 X5

Y1 Y2 Y999 Y4 Y5
<SPACE>

Z1 Z2 Z999 Z4 Z6
<RETURN>
Z111 Z2 Z999 Z4 Z5
```

```
  *END*
 ={T}
 X111 X2 X99 X4 X5
 ={L}/.    OUT=<RETURN>
 X111 X2 X999 X4 X5
 Y1 Y2 Y999 Y4 Y5
 Z111 Z2 Z999 Z4 Z5
  *END*
```

Example 3: Automatically change Z to ZEBRA on three lines.

```
 ={T}
 X111 X2 X999 X4 X5
 ={C} OLD=Z<RETURN> NEW=ZEBRA<RETURN> /3<RETURN>
 Z111 Z2 Z999 Z4 Z5
 ZEBRA111 Z2 Z999 Z4 Z5

 ZEBRA111 Z2 Z999 Z4 Z5
 ZEBRA111 ZEBRA2 Z999 Z4 Z5

 ZEBRA111 ZEBRA2 Z999 Z4 Z5
 ZEBRA111 ZEBRA2 ZEBRA999 Z4 Z5

 ZEBRA111 ZEBRA2 ZEBRA999 Z4 Z5
 ZEBRA111 ZEBRA2 ZEBRA999 ZEBRA4 Z5

 ZEBRA111 ZEBRA2 Z999 ZEBRA4 Z5
 ZEBRA111 ZEBRA2 Z999 ZEBRA4 ZEBRA5

  *END*
```

5.6 SCREEN ORIENTED COMMANDS

Screen editing commands extend the flexibility of the AIM 65/40 Editor beyond just line and character string oriented commands. Horizontal positioning commands allow cursor placement over the characters to be edited by overstrike, adding by insertion and deleting by single key entry (DEL or F5). Vertical cursor positioning commands scroll the text up or down while maintaining cursor positioning to enhance text visability and easy line positioning.

The general procedure for screen editing is:

a. Move the line pointer up or down to display the line to be edited using the line oriented positioning commands (U, D, T, B, F7, F8 or G) or the character string find command (F).

b. Press S or F6 to enter the screen edit mode. The Editor prompt (=) is replaced by the screen edit cursor (*).

c. Use the F6 and F5 keys to position the cursor horizontally.

d. Press F3 to enter the character insert mode (instead of character replace mode), if desired.

e. Replace/insert or delete characters as required.

f. Effect the change (up to this point the changes have not been made in the text buffer; they have been displayed only for editing purposes) as follows:

    (1) Press RETURN. The text buffer is updated, the screen edit mode (and insert mode, if active) is exited and the Edit or command level prompt (=) is displayed in position one.

    (2) Press F7 or F8 to move the cursor down or up. The changes are incorporated in the text buffer, the insert mode terminated (if active), and the screen editor cursor position retained.

g. Press ESC at any time to exit the screen edit mode without effecting the changes typed since entering the screen edit mode.

The individual screen editing commands are described in the following sections.

5.6.1 F1 (or CTRL Q) Command - Home Cursor On Line

When in the screen edit mode, the F1 (or CTRL Q) command moves the cursor to position one (the home position) without altering the display.

5.6.2 F2 (or CTRL R) Command - Clear Line to Right

When in the screen edit mode, the F2 (or CTRL R) command clears all the characters to the right of the cursor.

5.6.3 F3 (or CTRL S) Command - Toggle Insert Mode On/Off

When the Editor is in the screen edit mode, the F3 (or CTRL S) command toggles the character insert mode on or off. When the insert mode is active, the blinking asterisk cursor is replaced by a blinking three segment cursor (-<).

Each typed character in the insert mode is inserted in the position occupied by the cursor. The cursor, the character under the cursor and all characters to the right of the cursor are shifted right one position for each character entered or to the left for each character deleted (using the DEL key). This allows words, symbols or other character strings to be easily inserted in the middle of a line.

Use the F3 command as follows:

a. Position the line pointer to the line where the insertion is desired.

b. Enter the screen edit mode (if not already in it) using the Editor S or F6 commands.

c. Move the cursor to the position in front of which new characters are to be inserted using the F5 (move cursor right) and F6 (move cursor left) commands.

d. Type the characters to be inserted or delete the characters to be removed using the F4 or DEL keys.

e. Exit the insert mode by one of the following means:

(1) Press RETURN to effect the change. The changed text will be permanently added to the current line, the screen edit mode will be terminated, and the Editor command level will be re-entered.

(2) Press F7 or F8 to effect the change then move the cursor down or up one or more lines while staying in the same character position. The insert mode will toggle off but the screen edit mode will continue.

(3) Press F3 to toggle the insert mode off and still stay in the screen edit mode (the cursor will change back to the asterisk symbol but will not change position). Note that the line will not be updated until RETURN, F7 or F8 is pressed however.

(4) Press ESC to exit both the insert mode and the screen edit mode without effecting the changes. The Editor command level cursor will be displayed in position one.

5.6.4 F4 (or CTRL T) Command - Delete Character at Cursor

When in the screen edit mode, the F4 (or CTRL T) command deletes the character at the cursor position then shifts the characters from the right of the cursor over to the left one position.

Note that the DEL key deletes the character to the left of the cursor then shifts the characters from the right of the deleted character along with the cursor over to the left one position.

Use the F4 command as follows:

a. Position the line pointer to the line where deletion is desired.

b. Enter the screen edit mode (if not already in it) using the S or F6 commands.

c. Position the cursor over the first character to delete using the F5 or F6 commands.

d. Press F5 to delete the character under the cursor.

e. Terminate the deletion using the F7, F8 or RETURN keys as described in Section 5.6.

5.6.5 F5 (or CTRL U) Command - Move Cursor Left

While in the screen edit mode, the F5 (or CTRL U) command moves the cursor to the left. When the cursor reaches position one, the displayed data is scrolled to the right until the first character in the text line is in display position one.

The F5 command can be used in the insert mode also.

5.6.6 F6 (or CTRL V) Command - Move Cursor Right

While in the screen edit mode, the F6 (or CTRL V) command moves the cursor to the right. If the screen edit mode is not active when F6 is pressed, the mode is entered automatically. When the cursor reaches positon 4Ø, the displayed data is scrolled to the left until the 79th character in the text line is in display position 4Ø.

The F6 command can be used in the insert mode also.

5.6.7 F7 (or CTRL W) Command - Move Line/Cursor Down

The F7 (or CTRL W) command moves the cursor line pointer down one line at the Editor command level or moves both the line pointer and the character position cursor down one line in the screen edit mode. The active line pointed to by the line pointer is displayed. Holding the key down will repeat the process. This allows quick single line positioning or multiple line scrolling in the down direction.

END is displayed if the end of the text is reached followed by display of the last line of text.

5.6.8 F8 (or CTRL X) Command - Move Line/Cursor Up

The F8 (or CTRL X) command moves the line pointer up one line at the Editor command level mode or moves both the line pointer and the character position cursor up one line in the screen edit mode. The active line pointed to by the line pointer is displayed. Holding the key down will repeat the process. This allows quick single line positioning or multiple line scrolling in the up direction.

TOP is displayed if the top of the text buffer is reached followed by display of the top line of text.

5.6.9 CTRL A Command - Add a Line

The CTRL A command adds a line in the screen edit mode.

Use the CTRL A command as follows:

a. Enter the screen edit mode with the S or F6 command.

b. Position the cursor to the line before which you want to add the new line.

c. Type CTRL A. The Editor will insert a blank line before the line the cursor was on.

d. Continue screen editing.

5.6.1Ø CTRL B Command - Break a Line

When in the screen edit mode, the CTRL B command inserts a carriage return ($ØD) in the text line. This allows long lines to be broken into shorter lines.

Use the CTRL B command as follows:

a. Position the line pointer to the line to be split.

b. Enter the screen edit mode (if not already in it) using the S or F6 command.

c. Move the screen edit cursor to the character position where the carriage return is to be inserted.

d. Type CTRL B. The carriage return symbol (>-) is inserted in the cursor position. The inserted carriage returns can then be edited like any other symbol <u>until</u> (<u>but not after</u>) the changes are effected into the Text Buffer. Repeat steps c and d as required.

e. Type the RETURN, F7 or F8 key to effect the change.

f. The new current line will be displayed as terminated by the new carriage return(s).

NOTE

Once carriage returns have been included in the Text Buffer, they can not be deleted on a character basis since they are a line delimiter.

5.6.11 <u>CTRL D Command - Delete a Line</u>

The CTRL D command deletes a line in the screen edit mode.

Use the CTRL D command as follows:

a. Enter the screen edit mode with the S or F6 command.

b. Position the cursor to the line to be deleted.

c. Type CTRL D. The Editor will delete the line the cursor was on and display the next line.

d. Continue screen editing.

5.7 PERIPHERAL CONTROL

5.7.1 CTRL P Command - Toggle Auto-Print On/Off

The CTRL P command turns the printer auto-print control on or off as described in Section 4.9.1.

5.7.2 PRINT Command - Print Display Contents

The PRINT command causes the contents of the display line (including non-displayed characters) to be printed. Print will occur when the PRINT key is pressed regardless of the auto-print state.

5.7.3 1 or 2 Command - Toggle Recorder 1 or 2 Control On/Off

The 1 and 2 commands control the audio recorder remote control lines in the Editor in the same manner as described for the Monitor in Section 4.9.3.

SECTION 6

USING THE I/O ROM

The 4K-byte I/O ROM at $FØØØ-$FFFF contains initialization, interrupt handling, input/output and general purpose utility routines and subroutines that support user developed programs as well as the AIM 65/4Ø Monitor/Editor and optional software/firmware.

After an application program is entered, assembled/compiled and debugged using the Monitor/Editor and optional language software, e.g. assembler, PL/65 Compiler, BASIC Interpreter, or FORTH System; the program can be run without the Monitor/Editor ROMs installed. This provides the application program with an additional 8K bytes of on-board PROM/ROM space ($AØØØ-$BFFF) at run-time.

This section describes the functions of the I/O ROM and how to use them. Study the structure and capabilities of the data (vectors, constants and variables) and processing (interrupt, I/O and utility) as described in this section in conjunction with the I/O ROM assembly listing (see document no. 2965ØN69) to learn the capabilities of the I/O ROM. This will be useful during program development using both assembly and high level languages. Also, refer to the Monitor and Editor descriptions and the assembly listing as an example of how to interface with the I/O ROM.

6.1 MEMORY MAP

The firmware memory map in Figure 2-6 shows the location of the AIM 65/4Ø I/O ROM, Monitor/Editor and optional firmware along with RM 65 module firmware. Areas of memory used by the I/O ROM and reserved for optional firmware are also specified. Use areas of memory specified as user available for your program and data. Do not use areas reserved for optional firmware or I/O unless you know that there will not be a conflict later.

Figure 6-1 shows the breakdown of the I/O ROM. Refer to the I/O ROM assembly listing for more details. The segmentation of the lower address of RAM is detailed in Figure 6-2.

6.1.1 I/O Vectors

All input/output operations are handled through I/O vectors located in RAM to provide maximum flexibility in the configuring of application program I/O. Two vectors, one for input and one for output, are associated with the following I/O devices (except where only one vector is required, as in the case of the printer):

- o System terminal
- o Serial
- o Audio Tape
- o Memory
- o Floppy Disk
- o User Defined 1
- o User Defined 2
- o Printer (output only)
- o Display

Each vector points to the first of three jump (JMP) instructions located in a corresponding input or output jump table. These instructions jump to consecutive open file, transmit/receive data and close file subroutines. When an input/output subroutine is called through the vector, the associated I/O processing sequences through the three subroutines in consecutive order.

If you do not need to physically open a file (e.g. input from a keyboard), a JMP instruction to a dummy open subroutine must be provided to clear the decimal mode and to return from subroutine (RTS). Similarly, if you do not need to close a file, the same subroutine can be used, e.g. see the IOOK subroutine at $FØGE in the I/O ROM assembly listing.

FFFF FFFE	IRQ Interrupt Vector
FFFD FFFC	RES Interrupt Vector
FFFB FFFA	NMI Interrupt Vector
FFF9 FFEØ	I/O ROM Subroutine
FFDF FFDØ	User R6551 ACIA Registers
FFCF FFCØ	Keyboard R6522 VIA Registers
FFBF FFBØ	System R6522 VIA Registers
FFAF FFAØ	User R6522 VIA Registers
FF7F FF1C	I/O Device Initialization
FF1B FD43	IRQ Interrupt Processing
FD42 F983	Audio Tape I/O Processing
F982 F963	Serial I/O Subroutines
F962 F8B3	Printer Driver Subroutines
F8B2 F7D3	Display Driver Subroutines
F7D2 F55E	Keyboard Input Subroutines
F55D F4E8	System Routines & Messages
F4E7 F3F2	General Purpose Subroutines
F3F1 F312	Output Subroutines
F311 F21D	Input Subroutines
F21C F11D	RES Interrupt & Auto-Start Processing
F11C FØB1	NMI Interrupt Processing
FØBØ FØ56	Input and Output Jump Tables
FØ55 FØØØ	I/O Vectors, Constants & Variables

Figure 6-1. I/O ROM Memory Map

CØØØ	8ØØ	User Available
7FF	4AØ	Reserved for Optional Languages and RM 65 FDC Module (1)
49F	45Ø	Audio Tape Output Buffer (2)
44F	4ØØ	Audio Tape Input Buffer (2)
3FF	273	I/O & Monitor Working Storage
272	26B	Monitor Variables
26A	25Ø	Monitor Constants
24F	246	I/O ROM Variables
245	224	I/O ROM Constants
223	2ØØ	I/O ROM I/O Vectors
1FF	1ØØ	R65Ø2 CPU Stack
FF	FØ	I/O ROM Page Zero
DF	Ø	User Available Page Zero

Notes: 1. User available if optional languages and RM 65 FDC Module are not used.

2. User available if audio tape interface is not used.

Figure 6-2. Low RAM Detail Memory map

Table 6-1 shows the I/O vectors and the values initialized by the I/O ROM during a cold RESET. While any of the vectors may be changed by your application program, you will most likely alter one or two user-defined I/O functions or a floppy disk interface.

CAUTION

> Inadvertently altering the I/O vectors to invalid values may prevent AIM 65/4Ø from operating properly. Should this happen, a cold RESET will be required to recover.

Table 6-1. I/O ROM Vectors

Address	Label	No. Bytes	Cold Reset Value	Parameter
Ø2ØØ	IOVTAB	2	FØ5C	System Input Vector
Ø2Ø2		2	FØ85	System Output Vector
Ø2Ø4	IOVS	2	FØ62	Serial Input Vector
Ø2Ø6		2	FØ8B	Serial Output Vector
Ø2Ø8	IOVT	2	FØ56	Audio Tape Input Vector
Ø2ØA		2	FØ9D	Audio Tape Output Vector
Ø2ØC	IOVM	2	FØ79(1)(2)	Memory Input Vector
Ø2ØE		2	FØ79(1)(2)	Memory Output Vector
Ø21Ø	IOVF	2	FØ79(1)	Floppy Disk Input Vector
Ø212		2	FØ79(1)	Floppy Disk Output Vector
Ø214	IOVU	2	FØ79(1)	User Defined Input Vector
Ø216		2	FØ79(1)	User Defined Output Vector
Ø218	IOVV	2	FØ79(1)	User Defined Input Vector
Ø21A		2	FØ79(1)	User Defined Output Vector
Ø21C		2	FØ79(1)	Undefined Input Vector
Ø21E	IOVP	2	FØ94	Printer Output Vector
Ø22Ø	IOVDT	2	FØ7Ø	Display Input Vector
Ø222	IOVX	2	FØ7F	"X" Input Vector

NOTES

1. Initialized to undefined I/O by the I/O ROM.
2. Initialized to different values in the Monitor during Reset Audio-Start processing (see Section 15.1).
3. Refer to the jump tables at $FØ56 - $FØAE in the I/O ROM assembly listing.

Review the I/O ROM input and output jump tables at $FØ56-$FØA5. Note that while it appears that some JMP instructions are missing, closer inspection will reveal that there are three consecutive JMP instructions associated with each vector; the third one in some cases is the same as the first instruction for another device.

6.1.2 I/O ROM Constants

Program constants are values that, once initialized, normally do not change. AIM 65/4Ø I/O constants that you may alter to meet specific application requirements are located in RAM. Table 6-2 lists the I/O constants and the values initialized by the I/O ROM during cold RESET.

These values can easily be changed by the RESET Auto-Start processing in your application program. Note that when the Monitor/Editor ROMs are installed, three interrupt vectors are changed by the Monitor (see Section 15.2) to point to the Monitor entry point, single step instruction execution and BRK instruction processing. Since the RESET Auto-Start processing in other PROM/ROM areas ($8ØØØ-$9ØØØ and $CØØØ-$EØØØ) is performed after the Monitor, you can re-initialize these constants to other values if needed.

The I/O constants listed in Table 6-1 are described in more detail below. Unless otherwise noted, the initialization is performed by the I/O ROM upon cold RESET.

RESETF - Flag indicating that the last RESET performed was either cold ($Ø) or warm (Ø). This flag can be used in the application program RESET Auto-Start processing to determine whether to perform cold or warm initialization.

MONENT - Vector pointing to the starting address of the application program to which the I/O ROM jumps upon completion of RESET Auto-Start processing. Initialized to point to the I/O ROM cold reset entry. Subsequently initialized by the Monitor to point to the Monitor entry address.

Table 6-2. I/O ROM Constants

Address	Label	No. Bytes	Cold Reset Value	Parameter
Ø224	RESETF	1	8Ø	Cold(8Ø)/Warm(ØØ) RESET Flag
Ø225	MOMENT	2	FF7B(1)	RESET Exit to Monitor
Ø227	UNMIBM	2	FØB4	NMI Before I/O ROM
Ø229	UNMIBR	2	F12Ø(1)	NMI Before Return
Ø22B	UIRQBM	2	FD46	IRQ Before I/O ROM
Ø22D	UIRQAM	2	F5A7	IRQ After I/O ROM
Ø22F	BRKINS	2	FØDC(1)	Break Interrupt Vector
Ø231	TSTKEY	2	F625	Test for Key Down Vector
Ø233	TRBUF	2	Ø4ØØ	Start of Tape Input Buffer
Ø235	TRBEND	2	Ø44E	End of Tape Input Buffer
Ø237	TWBUF	2	Ø45Ø	Start of Tape Output Buffer
Ø239	TWBEND	2	Ø49E	End of Tape Output Buffer
Ø23B	TNAMSZ	1	Ø5	No. of Chars in File Name
Ø23C	NULL	1	ØØ	No. of Nulls at <CR><LF>
Ø23D	MEMCNT	1	Ø8	No. of Bytes to Display
Ø23E	IRGSYN	1	5Ø	No. of Tape Sync Char(/2)
Ø23F	KEYLIM	1	2Ø	No. of Keys on Keystack
Ø24Ø	REPT1	1	14	Delay to Repeat of Key
Ø241	REPT2	1	Ø2	Key Repeat Speed
Ø242	BEEPCY	1	8Ø	No. of Speaker Cycles
Ø243	BEEPON	1	1E	Speaker Cycle On Time
Ø244	BEEPOF	1	1E	Speaker Cyele Off Time
Ø245	TAPSPD	1	C3	Audio Tape Bit Width

NOTES

1. Initialized to different values by the Monitor (see Section 15.1).

If the Monitor is installed and it is desired to start the Monitor but either bypass or replace the ROCKWELL AIM 65/40 and AIM 65/40 reset messages, the assembly code located at MSTART ($A13D-$A169) in the Monitor must be replaced with equivalent code before jumping to the Monitor command entry point at COMIN1 ($A314).

UNMIBM - Vector pointing to the start of the NMI interrupt processing subroutine. Initialized to point to the start of the NMI processing subroutine in the I/O ROM (see Section 6.3). This processing saves the CPU status before releasing control to the user program.

UNMIBR - Vector pointing to the start of the user NMI interrupt processing after initial processing by the I/O ROM (see Section 6.3). Initialized to point to the I/O ROM cold reset processing. Subsequently initialized by the Monitor to point to the Monitor NMI processing subroutine to handle ATTN key depression and single step instruction execution.

UIRQBM - Vector pointing to the start of the IRQ interrupt processing subroutine (see Section 6.4). Initialized to point to the default I/O ROM IRQ interrupt processing subroutine which saves the CPU status.

IURQAM - Vector pointing to the user IRQ interrupt processing to be performed after initial processing by the I/O ROM (see Section 6.4). Initialized to point to an IRQ error message display subroutine.

BRKINS - Vector pointing to the BRK instruction processing by the IRQ interrupt processing subroutine (see Section 6.4). Initialized to point to the I/O ROM NMI processing subroutine return preparation. Subsequently initialized by the Monitor to point to the BRK instruction handing in the Monitor IRQ interrupt processing.

TSTKEY - Vector pointing to the start of the keyboard input processing. Initialized by the I/O ROM to point to the keyboard input processing at ANYSTK which tests for depression of a key as indicated by an entry on the keystack. This vector must be changed if the I/OVTAB vector at $0200 is changed to point to an input device other than the AIM 65/40 keyboard (see the example in Appendix M).

TRBUF - First address of the audio tape input buffer. Initialized to the start of the default 79-byte buffer ($0400).

TRBEND - Last address of the audio tape input buffer. Initialized to the end of the default 79-byte buffer ($044E).

TWBUF - First address of the audio tape output buffer. Initialized to the start of the default 79-byte buffer ($0450).

TWEND - Last address of the audio tape output-buffer. Initialized to the end of the default 79-byte buffer ($049E).

NOTE

> The tape input and output buffers may be located elsewhere in memory. The size of the two buffers must be identical and can be as large as desired (limited by available RAM).

TNAMSZ - Number of characters in the file name (FILE=) used in the audio tape file handling subroutines. Initialized to 5. The value may vary from 1 to 20 ($14).

NULL - Number of null characters ($00) output by the CRLF subroutine in the Monitor. Initialized to 0. The value can vary from 1 to 255 ($FF).

MEMCNT - The number of bytes displayed by the Monitor display selected memory (M), display next memory (SPACE), display prior memory (-) and change memory (/) commands. Initialized to eight bytes. Can vary from 1 to 18 ($12).

IRGSYN - The number of the sync ($16) characters (divided by 2) output prior to the second and subsequent blocks of data output to audio tape. Initialized by the I/O ROM to 160 characters (80=$50). This value can vary from 3 (6 characters) to 255 (510 characters).

KEYLIM - The number of keys that the keystack can hold. Initialized to 32 ($20).

REPT1 - The length of time that a key is initially depressed before the key is repeated. Initialized to about a second ($14). The value can vary from 1 to 254 ($FE). $FF indicates no repeat function.

REPT2 - The length of time between repeat key samples once the initial repeat has been detected. Initialized to about 3 ms ($02). The value can vary from 1 (fastest) to 255 ($FF).

BEEPCY - The number of cycles that the speaker· is turned on by the BEEP subroutine. Initialized to 128 ($80) cycles. The value can vary from 1 to 255 ($FF).

BEEPON - The speaker on-time during one cycle. Initialized to $1E.

BEEPOFF - The speaker off-time during one cycle. Initialized to $1E.

TAPSPD - The audio tape bit width. Initialized to $C3 by the I/O ROM to correspond to 1200 Hz for a logic 0 and 2400 Hz for a logic 1. Can be modified for faster or slower frequency but is not recommended to prevent audio tapes with the recording frequency different from the AIM 65/40 default 1200/2400 Hz from being generated.

6.1.3 I/O ROM VARIABLES

Program variables are located in RAM since the values are updated periodically during processing. The I/O ROM variables are listed in Table 6-3 along with the values initialized at cold RESET. These variables are used by both the I/O ROM and the Monitor and are not normally directly accessed by the application program.

The three variables (INDEV, OUTDEV and ESCIV) included in Table 6-3 are really system variables and are used extensively by the user. The INDEV and OUTDEV variables are often loaded by I/O ROM subroutines but may be set separately by user software before calling any I/O subroutines using them. The escape vector must be loaded by an application program (unless the Monitor is installed, which provides default processing).

INDEV - Index to the active input device

Value	Device
0	System Terminal (Keyboard)
1	Serial
2	Audio Tape (2)
3	Memory (2)
4	Floppy Disk (2)
5	User Defined (2)
6	User Defined

(1) RETURN or SPACE
(2) Non-interactive device

OUTDEV - Index to the active output device

Value	Device
0	System Terminal (display)
1	Serial
2	Audio Tape (2)
3	Memory (2)
4	Floppy Disk (2)
5	User Defined (2)
6	User Defined
7	Printer
8	Null (no output)

(1) RETURN or SPACE
(2) Non-interactive

ESCIV - Vector to the ESC key processing. Upon detecting that the ESC key is down, the I/O ROM will jump to ESC key process through this vector.

The I/O ROM and Monitor/Editor assembly listings list other variables. Those variables are not initialized at RESET and are used internally.

Table 6-3. I/O ROM Variables

Address	Label	No. Bytes	Cold Reset Value	Parameter
Ø246	VSPEED	1	35	Visual Delay Time Flag
Ø247	DATBNO	1	ØØ	Tape Block Number Display Flag
Ø248	INTDEV	1	18	Number of Lines on Display
Ø249	WSPDV	1	CØ	Printer Online Status
Ø24A	FAILUR	1	ØØ	Device Failure Flag
Ø24B	SSDAF	1	ØØ	Single Step Disassembly Flag
Ø24C	FLAGS	1	CØ	Reg & Instruction Trace Flags
Ø24D	BANKFL	1	ØØ	Bank Flag
Ø24E	WARM1	1	AA	Check Location No. 1
Ø24F	WARM2	1	55	Check Location No. 2
Ø273	INDEV	1		Active Input Device Index
Ø274	OUTDEV	1		Active Output Device Index
Ø275	ESCIV	2		ESC Key Processing Vector

6.1.4 I/O ROM Page Zero Usage

A minimum of page zero is used by the I/O ROM to allow most of it to be available to your application program. Some locations are used, however, to provide short instructions and fast execution time in memory and time critical I/O usage. Table 6-4 lists the parameters (constants and variables) used by the I/O ROM and Monitor.

Table 6-4. I/O ROM Page Zero Parameters

Address	Label	No. Bytes	Parameter
ØØFØ	SYMTBL	2	Symbol Table Vector
ØØF2	ZPIV	2	LDA/STA-CMP Vector
ØØF4	ZPTMP	2	Zero Page Temporaries
ØØF6	TRVEC	2	Tape Read Access Vector
ØØF8	TWVEC	2	Tape Write Access Vector
ØØFA	NOWLN	2	Tape Editor Current Line Pointer
ØØFC	VECTOR	2	Indirect Load/Store Address
ØØFE	STAKIV	2	Address of Keystack

6.1.5 On-board Peripheral Data

Tables 6-5 through 6-7 list the I/O parameters for the User, System and Keyboard R6522 VIA peripheral devices. The I/O parameters for the User R6551 ACIA are listed in Table 6-8.

6.2 RESET and Auto-Start

When the $\overline{\text{RES}}$ line to the R65Ø2 CPU is low, the CPU is in a reset state and is not fetching and executing instructions.

When $\overline{\text{RES}}$ goes high, the CPU delays six cycles then fetches the program counter (PC) from $FFFC (PC low byte) and $FFFD (PC high byte). The CPU starts program execution at that PC address. A routine must be provided starting at that address to initialize the CPU registers, I/O devices, internal data and then jumps to, or starts, normal processing. The I/O ROM performs these functions in the AIM 65/4Ø system. These functions include:

Table 6-5. User R6522 VIA Registers

Address	Label	No. Bytes	Cold Reset Value	Parameter
FFAØ	UORB	1	FF	Port B Data Register
FFA1	UORA	1	FF	Port A Data Register
FFA2	UDRB	1	FF	Port B Data Direction Register
FFA3	UDRA	1	ØØ	Port A Data Direction Register
FFA4	UT1CL	1	-	Timer 1 Latch/Counter Low
FFA5	UT1CH	1	-	Timer 1 Latch/Counter High
FFA6	UT1LL	1	-	Timer 1 Latch Low
FFA7	UT1LH	1	-	Timer 1 Latch High
FFA8	UT2CL	1	-	Timer 2 Latch/Counter Low
FFA9	UT2CH	1	-	Timer 2 Counter High
FFAA	USR	1	FF	Shift Register (SR)
FFAB	UACR	1	ØØ	Auxiliary Control Register (ACR)
FFAC	UPCR	1	ØØ	Peripheral Control Register (PCR)
FFAD	UIFR	1	ØØ	Interrupt Flag Register (IFR)
FFAE	UIER	1	8Ø	Interrupt Enable Register (IER)
FFAF	UORAX	1	FF	Port A Data Register (w/o Handshake)

NOTE

Cold RESET value initialized by the R6522 VIA upon hardware RESET.

Table 6-6. System R6522 VIA Registers

Address	Label	No. Bytes	Cold Reset Value	Parameter
FFBØ	SORB	1	FF	Port B Data Register
FFB1	SORA	1	FF	Port A Data Register
FFB2	SDRB	1	FF	Port B Data Direction Register
FFB3	SDRA	1	ØØ	Port A Data Direction Register
FFB4	ST1CL	1	-	Timer 1 Latch/Counter Low
FFB5	ST1CH	1	-	Timer 1 Latch/Counter High
FFB6	ST1LL	1	-	Timer 1 Latch Low
FFB7	ST1LH	1	-	Timer 1 Latch High
FFB8	ST2CL	1	-	Timer 2 Latch/Counter Low
FFB9	ST2CH	1	-	Timer 2 Counter High
FFBA	SSR	1	FF	Shift Register (SR)
FFBB	SACR	1	ØØ	Auxiliary Control Register (ACR)
FFBC	SPCR	1	ØØ	Peripheral Control Register (PCR)
FFBD	SIFR	1	ØØ	Interrupt Flag Register (IFR)
FFBE	SIER	1	8Ø	Interrupt Enable Register (IER)
FFBF	SORAX	1	FF	Port A Data Register (w/o Handshake)

NOTE

Cold RESET value initialized by the R6522 VIA upon hardware RESET.

Table 6-7. Keyboard R6522 VIA Registers

Address	Label	No. Bytes	Cold Reset Value	Parameter
FFCØ	KBORB	1	FF	Port B Data Register
FFC1	KBORA	1	FF	Port A Data Register
FFC2	KBDRB	1	FF	Port B Data Direction Register
FFC3	KBDRA	1	ØØ	Port A Data Direction Register
FFC4	KBT1CL	1	-	Timer 1 Latch/Counter Low
FFC5	KBT1CH	1	-	Timer 1 Latch/Counter High
FFC6	KBT1LL	1	-	Timer 1 Latch Low
FFC7	KBT1LH	1	-	Timer 1 Latch High
FFC8	KBT2CL	1	-	Timer 2 Latch/Counter Low
FFC9	KBT2CH	1	-	Timer 2 Counter High
FFCA	KBSR	1	FF	Shift Register (SR)
FFCB	KBACR	1	ØØ	Auxiliary Control Register (ACR)
FFCC	KBPCR	1	ØØ	Peripheral Control Register (PCR)
FFCD	KBIFR	1	ØØ	Interrupt Flag Register (IFR)
FFCE	KBIER	1	8Ø	Interrupt Enable Register (IER)
FFCF	KBORAX	1	FF	Port A Data Register (w/o Handshake)

NOTE

Cold RESET value initialized by the R6522 VIA upon hardware RESET.

Table 6-8. User R6551 ACIA Registers

Address	Label	No. Bytes	Cold Reset Value	Parameter
FFDØ	ACIADR	1	ØØ	Data Register
FFD1	ACIASR	1	1Ø	Status Register (1)
FFD2	ACIACM	1	ØB	Command Register (2)
FFD3	ACIACN	1	1E	Control Register (2)

NOTE

1. Initialized by the R6551 upon hardware RESET.
2. Initialized by I/O ROM code RESET processing.

- A cold RESET function which initializes all CPU and I/O parameters to their initial (power up or cold start) values.

- A warm RESET function which initializes only the parameters required to regain control of CPU without changing I/O or system constants.

- An Auto-Start function which allows user programs to initialize upon RESET then either return to the I/O ROM for other initialization or start application program execution. A flowchart of the RESET processing is shown in Figure 6-3.

6.2.1 Cold/Warm RESET

The I/O ROM first clears the decimal mode and disables an IRQ interrupt. It then tests the bit pattern in two RAM locations to determine if a cold or warm reset is to be performed. Upon power up, the bit pattern will be random, thus forcing a cold reset. Note that these values can also be altered under operator or user program control to force a cold reset upon the next $\overline{\text{RES}}$ occurrence. To do this, change either variable WARM1 to a value other than $55 or WARM2 (see Section 6.1.3) to a value other than $AA before pressing RESET.

If a cold reset is being performed, the I/O ROM constants and variables are initialized to their cold reset values (see Section 6.1).

The reset of the CPU and the I/O devices are then initialized for further operation. The CTRL key is sampled to determine if a cold reset is to be performed under operator control. If the CTRL key is down, a cold reset is forced and repeated until the CTRL key is released.

The reset function then delays for enough time for the display peripheral to initialize. This time is dependent upon the number of lines on the display. The delay may be up to one second for a multi-line CRT display with 24 lines (see variable INTDEV in Section 6.1.3).

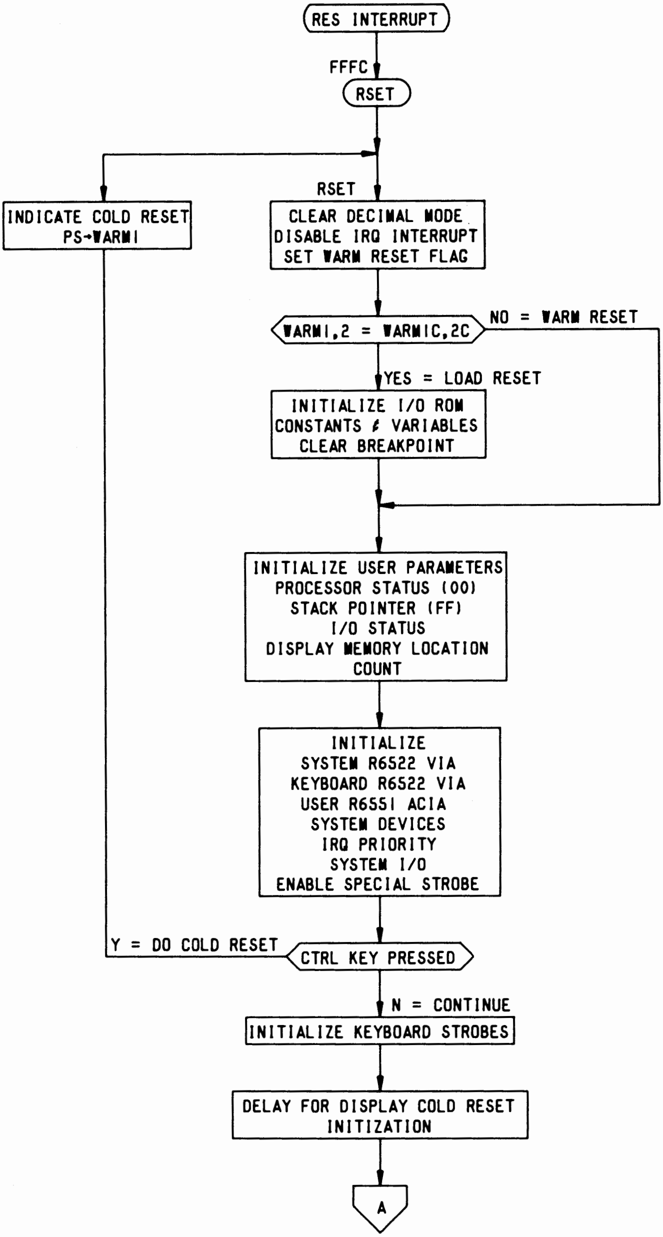


Figure 6-3. I/O ROM RESET and Auto-Start Processing

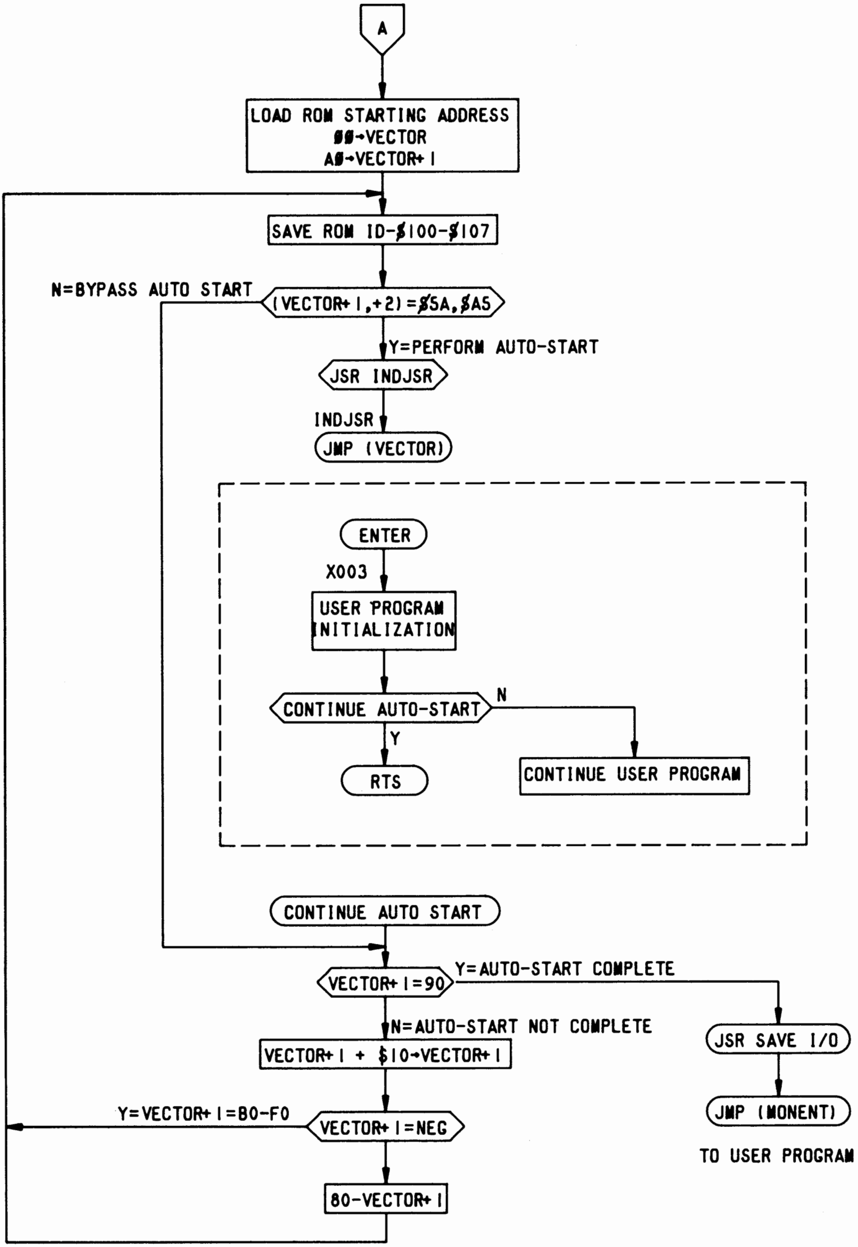


Figure 6-3. I/O ROM RESET and Auto-Start Processing (Continued)

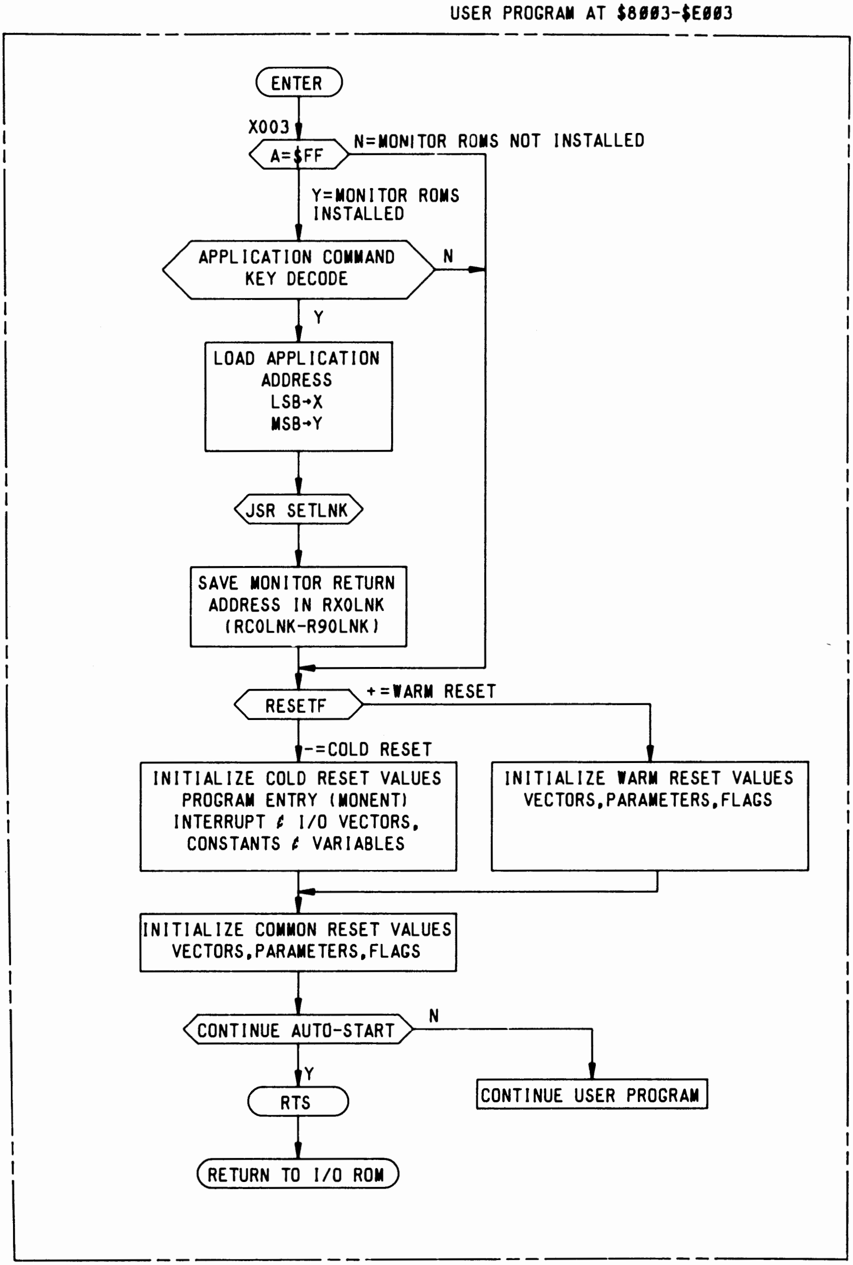

USER PROGRAM AT $8ØØ3-$EØØ3
ENTER
X003
A=$FF
N=MONITOR ROMS NOT INSTALLED
Y=MONITOR ROMS INSTALLED
APPLICATION COMMAND KEY DECODE
N
Y
LOAD APPLICATION ADDRESS LSB→X MSB→Y
JSR SETLNK
SAVE MONITOR RETURN ADDRESS IN RXOLNK (RCOLNK-R9OLNK)
RESETF
+=WARM RESET
-=COLD RESET
INITIALIZE COLD RESET VALUES PROGRAM ENTRY (MONENT) INTERRUPT & I/O VECTORS, CONSTANTS & VARIABLES
INITIALIZE WARM RESET VALUES VECTORS, PARAMETERS, FLAGS
INITIALIZE COMMON RESET VALUES VECTORS, PARAMETERS, FLAGS
CONTINUE AUTO-START
N
Y
RTS
CONTINUE USER PROGRAM
RETURN TO I/O ROM

Figure 6-3. I/O ROM RESET and Auto-Start Processing (Continued)

6.2.2 Auto-Start

The first three bytes of $A000, $B000, $C000, $D000, $E000, $F000, $8000 and $9000 are accessed in this order to store the program ID and to determine if the I/O ROM is to jump to these memory areas for application program initialization.

The first byte at $X000 from each area is stored at $0100-$10F as the program or PROM/ROM identifier (ID). This ID is user decided and can be interrogated under program or operator control to determine what program or PROM/ROM device is installed. For example, the AIM 65/40 I/O ROM version 1.0 ID is $F1 (from $F000) and the AIM 65/40 Monitor/Editor version 1.0 ID is $A1 (from $A000) and $B1 (from $B000).

The next two bytes, $X001 and $X002, are Auto-Start indicators. If the values at these addresses are $5A, and $A5, respectively, the I/O ROM jumps to the address at $X003 for its Auto-Start function. If the values are not $5A and $A5, the I/O ROM bypasses the Auto-Start function for that address range and skips to the next 4K-byte area.

Once the I/O ROM has jumped to the application program for Auto-Start initialization, that program has complete control of the system. It can perform functions such as:

a. If the Monitor ROMs are installed (indicated by $FF in the Accumulator upon entry), linkage can be included to have the Monitor jump to the application program for command key decoding <u>prior</u> to the Monitor decoding and acting on the key. This allows new functions to be added or other functions to be substituted in the Monitor.

   To include this linkage, the application auto-start function must load the entry address of the application decode function into the X (LSP) and Y (MSP) registers and call the SETLNK subroutine in the Monitor. Upon return from this subroutine, the Monitor command decoder entry address in the X (MSP) and Y (LSP) registers must be saved in a user variable. It is recommended that the return link variable corresponding to the 4K-byte address block (i.e. RCØLNK-R9ØLNK at $Ø3F4-$3FE) be used.

After Auto-start completion and during Monitor key command processing, the Monitor will jump to the Application program with the typed key value in the A-Register (in ASCII) before acting on the key itself.

If the application program determines that the typed key is not a valid application command, it should jump indirect through the appropriate return link vector (e.g. R8ØLNK) to allow the Monitor to decode and process the key. If the key is valid for the application, the application program should process the key appropriately then return to the Monitor with an RTS upon completion.

The active input device (INDEV), the active output device (OUTDEV) and the vector to the ESC key processing (ESCIV) must be loaded followed by a call to the SAVIO subroutine either during auto-start or later application program initialization. When the Monitor is installed, these variables can be loaded upon entry from the Monitor (via the G command or a function key (see Sections 4.6.1 and 4.1Ø, respectively) since the Monitor also initializes them upon entry (see MSTART processing in Monitor listing and Section 15.2).

b. It can initialize variables to their cold reset state if RESETF indicates a cold reset in progress ($8Ø).

NOTE

The Z flag in the Processor Status also contains the warm/cold reset status upon Auto-Start entry from the I/O ROM. The Z flag can therefore be tested instead of the RESETF flag until it is altered (+ = cold reset, - = warm reset).

c. It can initialize variables to their warm reset state if RESETF indicates a warm reset in progress (Ø).

d. It can either return to the I/O ROM with an RTS instruction for continued auto-start of other application programs, or it can keep control and continue into run-time operation. If it returns to the I/O ROM, it can also set up the exit from I/O ROM Auto-Start to start program execution at a starting address (MONENT, see Section 6.1.2) after all address areas have been initialized, i.e., at the completion of I/O ROM Auto-Start processing. Study the assembly listing of the AIM 65/40 Monitor Reset processing as an example.

6.3 NMI Interrupt Handling

When the $\overline{\text{NMI}}$ input to the CPU transitions low, a non-maskable interrupt occurs. The CPU completes the instruction currently being executed then loads the program counter with the address stored in the NMI vector at $FFFA and $FFFB. The CPU then continues execution at that address. A subroutine must be included in the application program to respond to the interrupt then return to the main program at the point of interruption to continue operation.

The NMI vectors point to the I/O ROM NMI processing (illustrated in Figure 6-4) which first saves the AIM 65/40 CPU register and memory bank information. It also selects bank 0 for NMI interrupt handler operation. Two I/O constants, UNMIBM and UNMIBR (see Section 6.2.1) allow your NMI handler to perform the total handling function or just the application function, respectively.

6.3.1 NMI Handler Before I/O ROM

If you want your application program to provide the total NMI handling function, load UNMIBM with the vector to the start of your NMI handler. In this case you must save all CPU registers you alter that are not saved on the stack, save the memory bank status and select bank 0 (see the NMI interrupt processing in the I/O ROM assembly listing as an example). Upon completion, restore the registers and memory bank then return from interrupt (RTI).

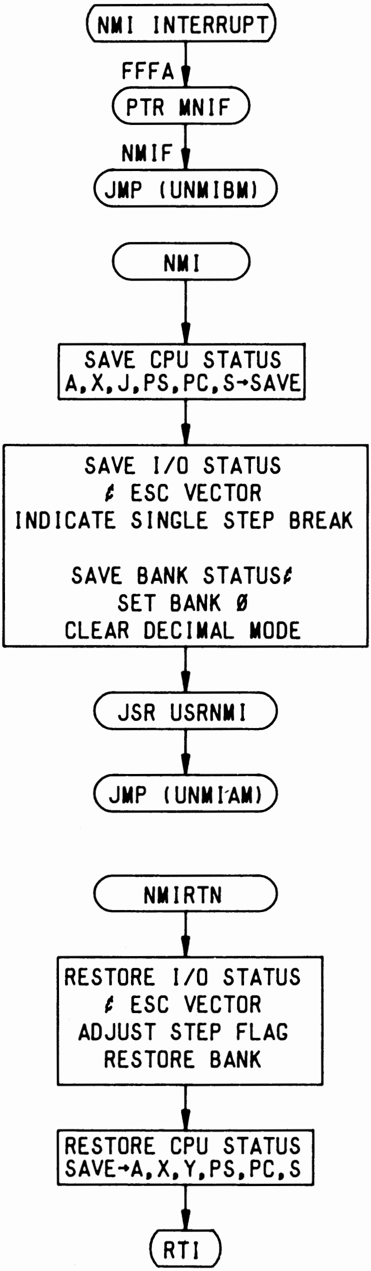


Figure 6-4. I/O ROM NMI Processing

6.3.2 NMI Handler Before Return

You can let the I/O ROM do all the housekeeping (i.e., saving and storing CPU status and memory bank) by loading UNMIBR with a vector to the start of your handler. With the technique, end your handler with a return from subroutine (RTS).

Note that the Monitor uses the NMI interrupt for ATTN key and single step instruction execution processing. It loads UNMIBR during its Auto-Start initialization (see Section 15.2) to point to its NMI Processing subroutine. By loading either UNMIBM or UNMIBR with a vector to your handler, you will bypass Monitor handling of the NMI interrupt.

6.3.3 NMI Return

The processing at NMIRTN restores user I/O status, the system bank and machine status before returning to the point of interruption.

6.4 IRQ INTERRUPT HANDLING

When the $\overline{\text{IRQ}}$ input to the CPU is low and the IRQ Disable bit in the Processor Status register has been cleared, an IRQ interrupt occurs. The CPU completes the instruction currently being executed then loads the program counter with the address stored in the IRQ vector at $FFFE and $FFFF. Like the NMI interrupt handling, a subroutine must be provided to respond to the interrupt and return at the point of interruption to continue operation.

The IRQ vector points to the I/O ROM IRQ interrupt processing (illustrated in Figure 6-5) which jumps to the IRQ processing through the constant UIRQBM.

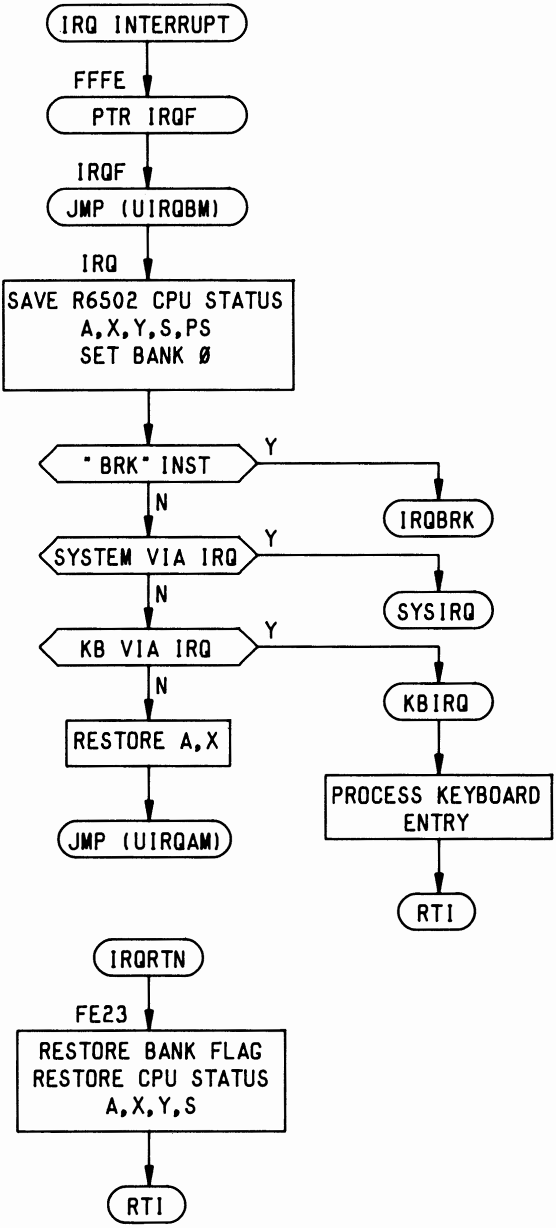


Figure 6-5. I/O ROM IRQ Processing

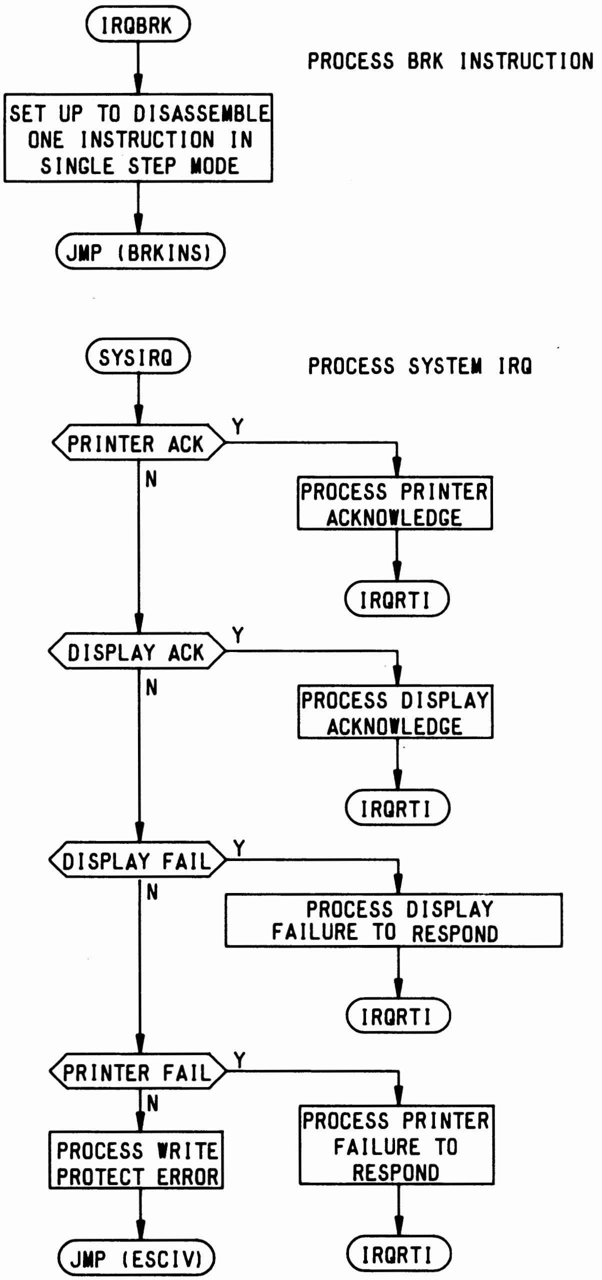

IRQBRK
PROCESS BRK INSTRUCTION
SET UP TO DISASSEMBLE ONE INSTRUCTION IN SINGLE STEP MODE
JMP (BRKINS)
SYSIRQ
PROCESS SYSTEM IRQ
PRINTER ACK
Y
N
PROCESS PRINTER ACKNOWLEDGE
IRQRTI
DISPLAY ACK
Y
N
PROCESS DISPLAY ACKNOWLEDGE
IRQRTI
DISPLAY FAIL
Y
N
PROCESS DISPLAY FAILURE TO RESPOND
IRQRTI
PRINTER FAIL
Y
N
PROCESS WRITE PROTECT ERROR
JMP (ESCIV)
PROCESS PRINTER FAILURE TO RESPOND
IRQRTI

Figure 6-5. I/O ROM IRQ Processing (Continued)

6.4.1 IRQ Handler Before I/O ROM

The I/O ROM loads UIRQBM during a cold reset to point to I/O ROM processing at label IRQ, which saves the CPU status, checks for BRK instruction (op code = ØØ) execution, a system R6522 IRQ interrupt and a keyboard R6522 IRQ interrupt.

If you change the UIRQBM vector to point to an IRQ routine of your own, you must provide similiar checks and linkage to the system and keyboard IRQ processing in the I/O ROM or the display/printer and keyboard interface will not operate properly.

6.4.2 IRQ Handler After I/O ROM

After the processing described in Section 6.4.1 is performed, the I/O ROM jumps indirect though UIRQAM. The I/O ROM loads UIRQAM with a pointer to IRQERR which will display "IRQ ERROR" and will jump to escape processing though vector ESCIV.

An application IRQ handler should normally be linked through UIRQAM. Upon completion of application IRQ processing, that processing should jump back to the IRQ interrupt return at IRQRTI, or perform similiar functions to restore the system bank and CPU status.

6.4.3 BRK Instruction Handling

The I/O ROM also loads constant BRKINS during a cold reset to point to NMIRTN (see Section 6.3) which saves the address and status at the BRK instruction before returning to the point of interruption.

If the Monitor is installed, it loads BRKINS with a pointer to the Monitor BRK instruction processing.

6.5 I/O ROM SUBROUTINES

The I/O ROM contains input, output, peripheral driver and general purpose subroutines in addition to interrupt handlers. These subroutines provide the processing needed to interface with AIM 65/4Ø peripherals independently of the Monitor/Editor firmware. The primary subroutines are summarized alphanumerically in Appendix K along with the calling and return conditions. They are also described by functional area of usage in the following paragraphs.

The assembly and machine instructions for these subroutines are included in the I/O ROM assembly listing (document no. 2965ØN69). This assembly listing shows other subroutines as well as other entry points for these described subroutines that perform a variation in the processing. You will have to study the assembly listing to learn all the capabilities and variations of the subroutines.

The Monitor/Editor ROMs also contain some general purpose I/O and utility subroutines (see Secton 5.3). Some of those I/O subroutines (e.g. WHEREI and WHEREO) are useful when setting up I/O handling in an interactive manner; i.e., by the operator. In most OEM and end user applications, those subroutines are not required since the I/O is pre-determined and can be established using only the subroutines in the I/O ROM.

The subroutines are identified by label, address, type (I = input, O = output, I/O = input or output, B = memory bank, U = utility) and the registers altered (A, X and Y).

a. <u>Setting up the Active Input/Output Device and I/O Vectors</u>

INLOW	F451	I	A

Sets the active input device to the keyboard (INDEV = Ø).

OUTLOW F45B O A

Sets the active output device to the display/printer (OUTDEV = Ø).

LLD F457 I/O A

Sets active input device to the keyboard (INDEV = Ø) and the active output device to the display/printer (OUTDEV = Ø). Calls INLOW.

GETIOV F33D I/O A

Establishes the VECTOR to the open, process or close I/O processing from pointers in the vector table ($2ØØ - $223). A call to GETIOV will cause a JMP indirect through the VECTOR to the jump table.

The Y-Register contains the offset from IOVTAB base address ($2ØØ) to the vector address of the jump table corresponding to the desired input/output device code.

Y		INDEV		
Reg.	Letter	OUTDEV	I/O	
ØØ		Ø	I	Keyboard
Ø2		Ø	O	Display/Printer
Ø4	S	1	I	Serial
Ø6	S	1	O	Serial
Ø8	T	2	I	Audio Tape
ØA	T	2	O	Audio Tape
ØC	M	3	I	Memory
ØE	M	3	O	Memory
1Ø	F	4	I	Floppy Disk
12	F	4	O	Floppy Disk
14	U	5	I	User Defined
16	U	5	O	User Defined
18	V	6	I	User Defined
1A	V	6	O	User Defined
1C	-	7	I	Undefined

Y Reg.	Letter	INDEV OUTDEV	I/O	
1E	P	7	O	Printer
2Ø	-	8	I	Display
22	X	8	O	Null (No Output)

* Entering of this letter in response to IN= or OUT= prompts displayed by the WHEREI or WHEREO subroutines in the Monitor (see Section 15.3) will set up the proper code in the INDEV and OUTDEV variables.

The A-Register contains the offset from the jump table base address to the JMP instruction to the open, process or close processing:

A	JMP to
Ø	Open Processing
3	Input or Output Processing
6	Close Processing

INCVEC F44A I/O

Increment VECTOR by one and, if VECTOR becomes zero, also increment VECTOR+1.

OPENI F21D I A,Y

Opens the active input device corresponding to the device code in INDEV. Uses OPENIO.

OPENO F312 O A,Y

Opens the active output device corresponding to the device code in OUTDEV. Uses OPENIO.

OPENIO F317 I/O A

Opens the active input (or output) device corresponding to the device code in the Y-Register; code format is the same as in INDEV (or OUTDEV + 2). Calls GETIOV with Ø in the A-Register (for jump to open processing) then jumps indirect through VECTOR.

CLOSEI F323 O A,Y

Closes the active input device corresponding to the device code in INDEV and sets it to the keyboard. Calls INLOW. Uses COIF.

CLOSEO F31C O A,Y

Closes the active output device corresponding to the device code in OUTDEV and sets it to the display/printer. Calls OUTLOW. Uses COIF.

COIF F324 I/O A,Y

Closes the active input (or output) device corresponding to the device code in the Y-Register; code format is the same as in INDEV (or OUTDEV + 2). Calls GETIOV with 6 in the A-Register (for JMP to close processing) then jumps indirect through VECTOR.

b. Input from the Active Input Device

INALL F233 I A

Gets one ASCII character from the active input device and returns it in the A-Register and in LASTIN ($Ø2A7).

INPUT F248 I A

Gets one ASCII character from the active input device (calls INALL) and then checks for and processes the following keys (if the input is not from audio tape).

<u>Key</u> <u>Action</u>

ESC Displays (ESC), then jumps to the program identified by the vector ESCIV.

PRINT Prints the entire contents of the displayed line including nonvisible characters, i.e. more than 4Ø characters, then waits for the next input character.

CTRL C Clears the line, homes the cursor to the leftmost position and waits for the next input character.

CTRL N Homes the cursor to the leftmost position and waits for the next input character.

CTRL P Toggles the Auto-Print state and waits for the next input character.

If the input character is none of the above, the subroutine returns after comparing the contents of the A-Register with a carriage return ($ØD).

If the input is from audio tape, the tape read status is checked. If a SYNC, CHECKSUM or BLOCK error is detected, the error is displayed and the subroutine escapes through the ESCIV vector; otherwise it returns. If an error is detected, the abort through the ESCIV vector can be bypassed and input processing continued if bit 5 in variable WSPDV ($249) is set to "1".

REDOUT F29B I A

Inputs one ASCII character from the active input device (calls INPUT) and outputs it to the display/printer if it is a non-CTRL character. Returns with the character (in ASCII) in the A-Register after comparing it to a carriage return ($ØD).

RDRUB F2A8 I A,Y

Inputs one ASCII character from the active input device (calls INPUT) and then tests for the DEL ($79) or CTRL H ($Ø8) which will cause the Y-Register to be decremented (if non-zero) and the cursor to move one space to the left. If the cursor is at position one (Y=Ø), the BEEPER is sounded and the cursor is left there. Returns with the input character (in ASCII) in the A-Register.

Call with the Y-Register containing Ø to $7F.

c. <u>Output to the Active Output Device</u>

OUTALL F32B O

Sends one ASCII character in the A-Register to the active output device. Calls GETIOV with 3 in the A-Register (for jump to character output). Jumps indirect through VECTOR.

ABLK F37E O A

Sends one blank character ($2Ø) to the active output device. Jumps to OUTALL.

ABX F384 O A,X

Sends n blank characters ($2Ø) to the active output device. The number of blanks is contained in the X-Register. Calls ABLK.

ABX3 F382 O A,X

Sends three blank characters ($2Ø) to the active output device. Calls ABLK.

WRAXA F3AØ O A

Converts four hexadecimal numbers in the A-Register and the X-Register to ASCII (A-Register first) and sends them to the active output device. Calls NUMA.

NUMA F3A4 O A

Converts two hexadecimal numbers in the A-Register to ASCII and sends them to the active output device. Sends the most significant portion (bits 4-7) then the least significant portion (bits Ø-3). Uses NOUT to convert each and output each character.

NOUT F3AC O A

Converts the hexadecimal number in bits Ø-3 of the A-Register to ASCII and sends it to the active output device. Calls H2ASCI. Jumps to OUTALL.

SEMI F329 O A

Sends a semicolon ($3B) to the active output device. Uses OUTALL.

d. Output to Display/Printer (System Terminal)

OUTPUT F352 O

Sends the ASCII character in A-Register to the display and to the printer (if auto-print is on, i.e. bit 6 of ACTIVE = 1).

BACK F498 O X

Sends n backspace characters ($Ø8) to the display/printer. The number of characters is contained in the X-Register. Calls BACKSP.

BACKSP F492 O

Sends one backspace character ($Ø8) to the display/printer. Calls OUTPUT.

BLANK F37A O A

Sends one blank character ($2Ø) to the display/printer. Jumps to OUTPUT.

BLANK2 F377 O A

Sends two blank characters ($2Ø) to the display/printer. Uses BLANK.

BLANK3 F374 O A

Sends three blank characters ($2Ø) to the display/printer. Uses BLANK.

BLANK4 F371 O A

Sends four blank characters ($2Ø) to the display/printer. Uses BLANK.

CRLOW F38F O A

Sends a carriage return ($ØD) and line feed ($ØA) to the display/printer. Calls HOME and uses OUTPUT.

CLR2RT F396 O A

Sends a $Ø2 to the display/printer to clear from the cursor to the right. Uses OUTPUT.

HOME F38B O A

Sends a carriage return ($ØD) to the display/printer. Uses OUTPUT.

LFLOW F392 O A

Sends a line feed ($ØA) to the display/printer.

PSLS F34B O A

Sends a slash ($2F) to the display/printer.

NUMABL F3B2 O A

Outputs a blank character ($2Ø) followed by two hexadecimal numbers in the A-Register to the display/printer. Jumps to NUMALO.

WRAX F3BA O A

Converts four hexadecimal numbers in the A-Register and the X-Register to ASCII (A-Register first) and sends item to the display/printer. Calls NUMALO.

NUMALO F3BE O A

Converts two hexadecimal numbers in the A-Register to ASCII and sends them to the display/printer. Calls NOUTLO. Sends the most significant portion (bits 4-7) the least significant portion (bits Ø-3). Calls NOUTLO.

NOUTLO F3C6 O A

Converts the hexadecimal number in bits Ø-3 of the A-Register to ASCII and outputs it to the display/printer. Calls H2ASCI. Jumps to OUTPUT.

CUROFF F3D4 O

Sends a $18 to the display/printer to blank the cursor. Calls OUTPUT.

CURON F3CB O

Sends a $17 to the display/printer to display the cursor. Calls OUTPUT.

CUR2X F3DC O A

Sends a block cursor command ($7F) to the display/printer. Preceded by ESC E S command sequence. Calls CURCNG.

CUR2ST F3EØ O A

Sends an asterisk cursor command ($2A) to the display/printer. Preceded by ESC E S command sequence. Calls CURCNG.

CURCNG F3E2 O

Sends the ASCII character in the A-Register to the display/printer as the cursor. Preceded by ESC E S command sequence.

e. <u>Input from the Display</u>

ODISIN F817 I A

Opens display input. Waits for printer to finish. Outputs Transmit Display Line command (ESC X L character sequence) to the display. Sets System VIA display port to inputs.

GDISIN F83Ø I A

Gets a character from the display when Acknowledge is detected. Returns with ASCII character in the A-Register after issuing Strobe to display.

CDISIN F83C I Y

Closes display input. Sets System VIA display port to outputs.

GETDT F84Ø I A

Gets a character in the A-Register from the display by calling GDISIN though VECTOR IOVDT.

WAITD F8ØC I

Returns when Acknowledge from the display is detected (if the display is active).

f. <u>Input from the Printer</u>

WAITP F935 I

If the printer is active, returns when acknowledge from the printer is detected. If the printer is not active and the printer failure flag is not set, the subroutine returns. If the printer failure flag is set, the PRINTER DOWN message is displayed and the beeper is sounded.

g. Output to the Display

PUTDIS F7D3 O

Sends one ASCII character in the A-Register to the display.

DISPLY F361 O

Sends one ASCII character in the A-Register to the display. Calls PUTDIS though vector IOVTAB +2 (CALLS GETIOV to set up VECTOR to IOVTAB +2 then jumps indirect through VECTOR to PUTDIS).

h. Output to the Printer

OPNPTR F8C8 O A

Opens (enables) printer output. Saves the Auto-Print status. Clears the printer buffer. Call before using PUTPTR.

CLOPTR F8B3 O A

Closes printer output. Restores the Auto-Print to its state before the printer output was opened. Call after using PUTPTR.

PNTKEY F84A O

Sends the contents of the displayed line (including non-visible characters, i.e. more than 4Ø characters) to the printer.

PUTPTR F8F1 O

Sends the ASCII character in the A-Register to the printer. Call OPNPTR before outputting characters with PUTPTR. Call CLOPTR after sending all the print characters.

PRINT F35A O

Sends the ASCII character in the A-Register to the printer by calling PUTPTR through vector IOVP (calls GETIOV to set up VECTOR to IOVP then jumps indirect through VECTOR to PUTPTR).

TOG F2C8 O A,X,Y

Toggles the Auto-Print state. Prints AUTO-PRINT ON or AUTO-PRINT OFF to indicate the current state. Calls OPNPTR and PRINT. Jumps to CLOPTR upon completion.

i. Input from the Keyboard (System Terminal)

READ F22C I A

Sets the active input device to the keyboard and inputs one character via the keystack and returns with the ASCII value in the A-Register. If the keystack is empty, READ waits until a character is entered from the keyboard.

ANYKEY F622 I

Tests the keystack for the presence of a new key and returns the results in the processor status zero flag (Z). Z = 1, a key is available. Z = Ø, no key is available. Jumps indirect through vector TSTKEY.

DMDKEY	F55E	I	A

Strobes the keyboard for a key and returns the ASCII value in the A-Register. (May be used only when the interrupt driven routines are disabled, i.e., no IRQ from the keyboard.)

A2STAK	F593	I	Y

If the keystack is not full, the contents of the A-Register is put onto the keystack and the processor status carry flag (C) is cleared to zero. If the keystack is full, the carry flag is set to 1 and the subroutine returns without putting the contents of the A-Register on the keystack.

KEYDWN	F58D	I	A

Examines the keyboard for a key depression. Returns with the zero flag (Z) indicating the key status. Z = 1, a key is depressed. Z = Ø, no key is depressed.

GETKEY	F5AF	I	A

Gets a key from the keystack and returns it in the A-Register.

GETSTK	F63C	I	A

Gets a key from the stack and returns with it in the A-Register and in CHAR.

j. <u>Output to the Speaker</u>

BEEP	F467	O

Causes the speaker to sound for the number of cycles specified in BEEPCY at the tone specified by the BEEPON and BEEPOF pulse widths.

k. <u>Serial Input</u>

GETSER F963 I A

Tests the RS-232C/TTY serial input port for ready to receive. When the ACIA Receive Data Register is full, the input data byte is loaded into the A-Register and the subroutine returns. Calls TSERI.

TSERI F96C I

Tests the RS-232C/TTY serial input port for ready to receive (R6551 ACIA Status Register Receiver Data Register Full, bit 3) and sets the zero flag accordingly: Z = Ø, receive data register not full, Z = 1, receive data register full.

l. <u>Serial Output</u>

PUTSER F972 O

Tests the RS-232C/TTY serial output port for ready to transmit. When the ACIA Transmit Data Register is empty, the output data byte is stored in the Transmit Data Register and the subroutine returns. Calls TSERO. Call with the output byte in the A-register.

TSERO F97D P A

Tests the RS-232C/TTY serial output port for ready to transmit (R6551 ACIA Status Register Transmitter Data Register Empty, bit 4) and sets the zero flag accordingly: Z = Ø, transmit data register not empty, Z = 1, transmit data register empty.

m. Input from Audio Tape

OPENTI F983 I A,X,Y

Opens audio tape input. Sets active input device to keyboard. Gets unit number and file number. Sets INDEV to audio tape. Sets tape bit frequency.

GETAPE FA63 I A

Gets a byte from the audio tape input buffer and returns it in the A-Register. If the input buffer is empty, a block is loaded from the audio tape input port.

n. Output to Audio Tape

OPENTO FBCC O A,X,Y

Opens the audio tape output. Asks for unit number and file name and then initializes the output routine.

CLOSTO FBØE O A

Writes the final audio tape block and restores the active output device to the display/printer.

PUTAPE FBEE O

Sends the contents of A-Register to the audio tape output buffer. If the output buffer is full, the buffer contents are sent to the audio tape output port.

o. Memory Bank

BANKØ FFE8 B A

Enables Bank zero address range.

BANK1   FFF1   B   A

Enables Bank one address range. Uses SETBNK.

LDAY   F4A6   B   A

Fetches a byte from either RAM bank (Bank 0 or 1) as determined by the Y-Register and the A-Register. The A-Register contains the offset from the variable S1 of the address vector to which the Y-Register is added.

SADDR   F4B7   B

Stores and verifies the contents of the A-Register into either RAM bank as determined by the address vector ADDR and the offset from ADDR contained in the Y-Register.

SETBNK   FFF3   B   A

Enables the memory bank specified in the A-Register contents: A = 0, bank 0 is enables; A=$04, bank is enabled.

ZROBNK   FFE0   B   A

Saves the present bank status and enables bank zero. Uses BANK0.

p. Miscellaneous

H2ASCI   F3F2   U   A

Converts the low order 4 bits of the A-Register to an 8-bit ASCII value.

LEFT   F440   U   A

Shifts the contents of the A-Register 4 bits to the left.

RIGHT   F445   U   A

Shifts the contents of A-Register 4 bits to the right.

GETXY   F41C   U

Pulls the contents of the X and Y Registers from the top two values on the stack, respectively. Use SAVSY to push X and Y onto the stack.

SAVXY   F3FD   U

Pushes the contents of the X and Y Registers onto the stack as the next to top and top values, respectively. Use subroutine GETXY to pull X and Y from the stack.

COLD   F11D   U   A,X,Y

Performs a cold reset.

RSET   F120   U   A,X,Y

If the CTRL key is not pressed, a warm reset is performed. If the CTRL key is pressed, a cold reset is performed when the CTRL key is released.

SECTION 7

USING THE PARALLEL APPLICATION INTERFACE

The parallel I/O interface at the SBC module connector J1 is dedicated to user functions. This interface is connected directly to an R6522 Versatile Interface adapter (VIA) device. This section describes how to use data and control ports, the two internal timers and the serial interface available in the VIA. The information can also be used to program the two VIA devices in the printer/display and keyboard interfaces if they are used for user applications rather the system peripheral ports.

7.1 FEATURES OF THE VERSATILE INTERFACE ADAPTER

The features of the R6522 VIA device include:

- Two 8-bit I/O ports (A and B). Each pin can be individually selected to be either an input or an output.

- Four status and control lines (two associated with each port).

- Two 16-bit counter/timers which can be used to generate or count pulses. These timers can produce single pulses or a continuous series of pulses.

- An 8-bit shift register which can convert data between serial and parallel forms.

- Interrupt logic so that I/O can proceed on an interrupt-driven basis. This logic includes an interrupt flag register that tells whether particular interrupts have occurred and an interrupt enable register which determines whether particular interrupts are allowed.

Figure 7-1 is a block diagram of the R6522 VIA. The VIA operates using 16 data, control and status registers. The addresses for the user parallel I/O VIA are listed in Table 6-5. The input/output lines are defined in Table 7-1.

VIA operation is controlled by the contents of four registers:

- Data Direction Register A (DDRA) determines whether the pins on port A are inputs or outputs.
- Data Direction Register B (DDRB) determines whether the pins on port B are inputs or outputs.
- The Peripheral Control Register (PCR) determines which polarity of transition (i.e., rising edge or falling edge) is recognized on the input status lines (CA1 and CB1) and how the other status lines (CA2 and CB2) operate.
- The Auxiliary Control Register (ACR) determines whether the data ports are latched and how the timers and shift register operate.

Remember, there is a Data Direction Register for each side, but only one pair of control registers. Ports A and B are almost identical. One important difference is that Port B can handle Darlington transistors which are used to drive solenoids and relays. The examples generally use Port A for input and Port B for output.

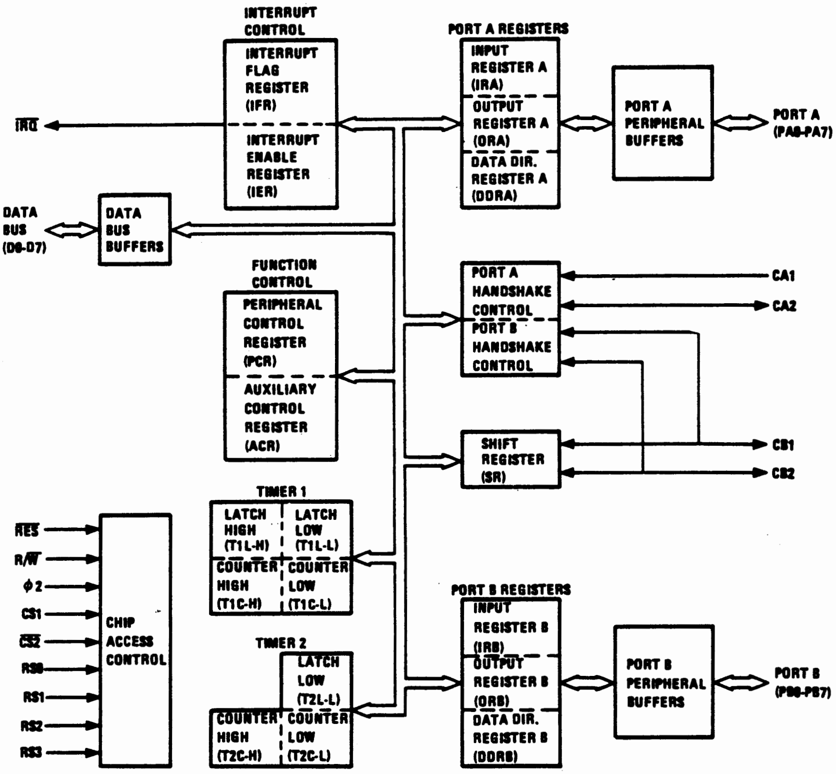


Figure 7-1. R6522 VIA Block Diagram

Table 7-1. SBC Connector J1 (Parallel I/O) Signal Definitions

Mnemonic	Signal Name and Signal Description	Type of Drive
PAØ-PA7	<u>Peripheral A Port, I/O Bits Ø-7</u> These eight bidirectional lines can be individually programmed to act as an input or output under control of the Port A Data Direction Register. The level of the PAØ-PA7 lines can be controlled by the Port A Output Register.	Passive Pull-Up
CA1	<u>Peripheral A Control No. 1</u> This unidirectional line can be used as an interrupt input or can be used in conjunction with CA2 as the input line for handshaking Port A. This line also controls the latching of data on PAØ-PA7 lines.	No Internal Pull-Up
CA2	<u>Peripheral A Control No. 2</u> This bidirectional line can be used as an interrupt input, an output, or can be used in conjunction with CA1 as the output line for handshaking Port A.	Passive Pull-Up

Table 7-1. SBC Connector J1 Signal Definitions (Continued)

Mnemonic	Signal Name and Signal Description	Type of Drive
PBØ-PB7	Peripheral B Port, I/O Bits Ø-7 These eight bidirectional lines can be individually programmed to act as an input or output under control of the Port B Data Direction Register. The level of the PBØ-PB7 lines can be controlled by the Port B Output Register. Lines PB6 and PB7 can also be used with the internal timers.	Active Pull-Up
CB1	Peripheral B Control No. 1 This bidirectional line can be used as an interrupt input, an output, or can be used in conjunction with CB2 as the input line for handshaking Port B. This line can also control the latching of data on PBØ-PB7, or be a clock line for shift register operation.	Active Pull-Up
CB2	Peripheral B, Control No. 2 This bidirectional line can be used as an interrupt input, an output, or can be used in conjunction with CB1 as the output line for handshaking Port B. This is also the serial data line for shift register operation.	Active Pull-Up

7.2 SIMPLE I/O WITH THE VIA

7.2.1 Considerations

Since RESET clears all the VIA registers, disabling all interrupts and clearing all control lines, we'll discuss simple I/O referring only to the data registers and the data direction registers. Simple I/O can be performed with the R6522 VIA as follows:

a. Establish the directions of the pins by storing the proper values in the data direction registers.

b. Transfer data by moving it to or from the data registers.

Note that most programs only have to execute Step a once since the directionality of most input and output devices is fixed (i.e., you never want to read data from a display or printer or write data to a switch or paper tape reader).
Directions can be established as follows:

- A 'Ø' in a bit in a data direction register makes the corresponding pin an input. For example, A 'Ø' in bit 4 of Data Direction Register A makes pin PA4 into an input.

- A '1' in a bit in a data direction register makes the corresponding pin an output. For example, a '1' in bit 6 of Data Direction Register B makes pin PB6 into an output.

For transferring data, remember that the R65Ø2 microprocessor has no specific I/O instructions. Stor.ing data in a VIA port that has been designated for output is equivalent to sending the data to the attached output device. Loading data from a VIA port that has been designated for input is equivalent to reading the data from the attached input device. Any instruction that acts on memory can serve as an I/O instruction if the specified address is actually an I/O device. You must be careful of the exact significance of such instructions in writing, reading and documenting R65Ø2 programs.

7.2.2 Examples

In these examples, we use the hexadecimal addresses in the Mnemonic Entry format (see Section 4.5.1) The examples are shown starting at address Ø8ØØ. The comments to the right of the instructions are explanatory only and may not be input into the Mnemonic Entry function.

a. Fetch data from a simple input port (e.g., from a set of switches or a keypad) and store it in memory location 4Ø.

```
{I}

ØXXX   *=Ø8ØØ
Ø8ØØ A9 ØØ      LDA #$ØØ     Make side A inputs.
Ø8Ø2 8D A3 FF   STA $FFA3
Ø8Ø5 AD F1 FF   LDA $FFF1    Get data.
Ø8Ø8 85 4Ø      STA $4Ø      Store it.
```

b. Send data to a simple output port (e.g., to a set of displays or relays from memory location 4Ø.

```
Ø8ØA A9 FF      LDA #$FF     Make side B all inputs.
Ø8ØC 8D A2 FF   STA $FFA2
Ø8ØF A5 4Ø      LDA $4Ø      Get data.
Ø811 8D AØ FF   STA $FFAØ    Output it.
```

You can mix inputs and outputs on a single port by establishing the directions of individual pins appropriately. Note that you can read the states of data pins (e.g., with LDA $FFA3 or LDA $FFA2) even if they have been designated as outputs. The B side is buffered so that it can always be read correctly, however, the A side is not buffered so that it can only be read correctly if it is lightly loaded (or designated as inputs).

7.3 RECOGNIZING STATUS SIGNALS

7.3.1 Considerations

If the I/O device is more complex, data cannot usually be transferred to or from it at will. In the input case, the processor must know when new data is available (e.g., a key has been pressed on a keyboard or a tape reader has read another character). In the output case, the processor must know whether the device is ready to receive data (e.g., a printer has finished printing the last character or a modem has completed the previous transmission).

Normally, the input or output device provides a status signal. A transition on that line indicates the availability of data or the readiness of the device. The microcomputer I/O section must recognize the transition and allow the processor to determine that it has occurred.

You can handle this kind of I/O with the R6522 Versatile Interface Adapter as follows:

a. Attach the peripheral status input to input CA1 or CB1.

b. Determine which edge on the status line will be recognized by assigning a value to control register bit Ø (CA1) or 4 (CB1). A value of zero in that bit position means that the interrupt flag will be set by a high-to-low transition (of falling edge). A value of one means that the interrupt flag will be set by a low-to-high transition (or rising edge).

c. Determine whether a transition has occurred by examining bit 1 (CA1) or 4 (CB1) of the interrupt flag register. The bit will be one if a transition has occurred.

d. Reset the interrupt flag by reading or writing the corresponding data register. The flag is then ready to be used in the next operation.

7.3.2 Examples

a. Fetch data from an input port with an active high-to-low DATA READY strobe and place the data in memory location $4Ø.

```
Ø814 A9 ØØ        LDA #$ØØ
Ø816 8D A3 FF     STA $FFA3     Make side A inputs.
Ø819 8D AC FF     STA $FFAC     Set CA1 Interrupt Flag
Ø81Ø AD AD FF     LDA $FFAD       on falling edge.
Ø81F 29 Ø2        AND #$Ø2
Ø821 DØ F9        BNE $Ø81C     Data ready?
Ø823 AD A1 FF     LDA $FFA1     Yes, get data.
Ø826 85 4Ø        STA $4Ø       Store it.
```

Clearing the Peripheral Control Register is unnecessary if the routine is starting from a reset. Note that reading the Output Data Register with LDA $FFA1 clears the interrupt flag so that it is available for the next DATA READY signal.

b. Send data to an output port with an active high-to-low PERIPHERAL READY strobe. Get the data from memory location $4Ø and send it when the peripheral is ready.

```
Ø828 A9 FF        LDA #$FF      Make side B outputs.
Ø82A 8D A2 FF     STA $FFA2
Ø82D A9 ØØ        LDA #$ØØ      Set CB1 Interrupt Flag on
Ø82F 8D FC FF     STA $FFFC       falling edge.
Ø832 AD AD FF     LDA $FFAD
Ø835 29 1Ø        AND #$1Ø
Ø837 DØ F9        BNE $Ø832     Peripheral ready?
Ø839 A5 4Ø        LDA $4Ø       Yes, get data.
Ø83B 8D AØ FF     STA $FFAØ     Store it.
```

Note that sending the data to the Output Data Register (STA $FFA2) clears the interrupt flag so that it is available for the next PERIPHERAL READY signal.

c. Fetch data from an input port with an active low-to-high DATA READY strobe and place the data in memory location 4Ø.

```
Ø83E A9 ØØ        LDA #$ØØ
Ø84Ø BD A3 FF     STA $FFA3     Make Side A inputs.
Ø843 A9 Ø1        LDA #$Ø1      Set CA1 Interrupt Flag
Ø845 8D AC FF     STA $FFAC       on rising edge.
Ø848 AD AD FF     LDA $FFAD
Ø84B 29 Ø2        AND #$Ø2
```

```
     Ø84D FØ F7        BEQ $Ø846     Data Ready?
     Ø84F AD A1 FF     LDA $FFA1     Yes, get data.
     Ø852 85 4Ø        STA $4Ø       Store it.
```

d. Send data to an output port with an active low-to-high PERIPHERAL READY strobe. Get the data from memory location 4Ø and send it when the peripheral is ready.

```
     Ø854 A9 FF        LDA #$FF      Make side B outputs.
     Ø856 8D A2 FF     STA $FFA2
     Ø859 A9 1Ø        LDA #$1Ø      Set CB1 Interrupt Flag
     Ø85B 8D FC FF     STA $FFFC       on rising edge.
     Ø85E AD AD FF     LDA $FFAD
     Ø861 29 1Ø        AND #$1Ø
     Ø863 FØ F9        BEQ $Ø85E     Peripheral ready?
     Ø865 A5 4Ø        LDA $4Ø       Yes, get data.
     Ø867 8D AØ FF     STA $FFAØ     Output it.
```

Note that the VIA has both input and output latches. The output latches are always enabled; output data is latched when it is stored in an output data register. The input latches, if they are needed, can be enabled by setting Bit Ø (side A) or Bit 1 (side B) of the Auxiliary Control Register. The input data will then be latched by the active transition on CA1 or CB1.

7.4 PRODUCING OUTPUT STROBES

7.4.1 Considerations

The peripheral may also require notification as to when a transfer has occurred or whether the port is ready to receive data. For example, devices such as digital-to-analog converters commonly require a LOAD pulse to enter data into the converter. A multiplexed display requires an output signal that directs the next output properly. A communications device may need a signal to indicate that an input buffer is available or that an output buffer is full. Output signals may also be needed to turn devices on or off, activate operator displays, or control operating modes.

You can handle this kind of I/O with the R6522 Versatile Interface Adapter as follows:

a. Attach the control output to CA2 or CB2.

b. Make CA2 (CB2) into an output by setting control register bit 3 (7).

c. Make CA2 (CB2) into a pulse by clearing control register bit 2 (6) or into a level by setting that bit.

d. If CA2 (CB2) is a pulse, make it into a handshake signal (low from the time the Output Register is read or written until the next active transition on CA1 (CB1) by clearing control register bit 1 (5) or into a single-cycle strobe by setting that bit.

e. If CA2 (CB2) is a level, determine its value by clearing or setting bit 1 (5).

7.4.2 Port Options

The port options are:

a. CA2 goes low when the processor transfers data to or from Output Register A and goes high when the next active transition occurs on CA1. The signal can indicate that the port is ready for more data or that output data is available. The peripheral's response then indicates that it has sent more data or has processed the previous data.

b. CA2 goes low when the processor transfers data to or from Output Register A and goes high after one clock cycle. This signal indicates that an input or output operation has occurred and can be used for multiplexing.

c. CA2 is a level controlled by the value of control register bit 1. This signal can provide an active-high or low pulse of arbitrary length. It can be used to load registers, turn devices on or off, or control operating modes.

7.4.3 Examples

a. Fetch data from an input device that requires a hand-shake signal and that produces an active high-to-low DATA READY strobe. Place the data in memory location $4Ø.

```
ØB6A A9 ØØ       LDA #$ØØ      Make side A inputs.
Ø86C 8D A3 FF    STA $FFA3
Ø86F A9 Ø8       LDA #$Ø8      Set CA2 handshake output
Ø871 8D AC FF    STA $FFAC       mode with CA1 Interrupt
Ø874 AD AD FF    LDA $FFAD       FLAG set on falling
Ø877 29 Ø2       AND #$Ø2        edge.
Ø879 FØ F9       BEQ $Ø874     Data ready?
Ø87B AD A1 FF    LDA $FFA1     Get and store data.
Ø87E 85 4Ø       STA $4Ø       Send data taken on CA2.
```

The Peripheral Control Register bits are:

bits 4-7 = Ø since CB1 and CB2 are not used

bit 3 = 1 to make CA2 an output

bit 2 = Ø to make CA2 a pulse

bit 1 = Ø to make CA2 a handshake acknowledgement that remains low until the next active transition on CA1

bit Ø = 1 to make the active transition on CA1 a falling edge (high-to-low transition).

b. Fetch data from an input device that requires a brief DATA ACCEPTED strobe for multiplexing or control purposes. Place the data in memory location $4Ø.

```
Ø88Ø A9 ØØ       LDA #$ØØ      Make Side A inputs.
Ø882 8D A3 FF    STA $FFA3
```

```
  Ø885 A9 ØA        LDA #$ØA     Set CA2 pulse output mode
  Ø887 8D AC FF     STA $FFAC      with CA1 Interrupt flag
  Ø88A AD AD FF     LDA $FFAD      set on falling edge.
  Ø88D 29 Ø2        AND #$Ø2
  Ø88F FØ F9        BEQ $Ø88A    Data ready?
  Ø891 AD A1 FF     LDA $FFA1    Yes, get data and send
  Ø894 85 4Ø        STA $4Ø        data taken strobe on
                                   CA2.
```

Bit 1 of the Peripheral Control Register is set to 1 to make CA2 a brief strobe lasting one cycle after the reading of Port A Output Data Register.

c. Send data to an output device that requires a handshake signal and that produces an active low-to-high PERIPHERAL READY strobe. Get the data from memory location $4Ø and send it when the peripheral is ready.

```
  Ø896 A9 FF        LDA #$FF     Make side B outputs.
  Ø898 8D A2 FF     STA $FFA2
  Ø89B A9 9Ø        LDA #$9Ø     Set CB2 handshake output
  Ø89D 8D AC FF     STA $FFAC      mode with CB1 INT FLAG
  Ø8AØ AD AD FF     LDA $FFAD      set on rising edge.
  Ø8A3 29 1Ø        AND #$1Ø
  Ø8A5 FØ F9        BEQ $Ø8AØ    Peripheral ready?
  Ø8A7 A5 4Ø        LDA $4Ø      Get and send data and set
  Ø8A9 8D AØ FF     STA $FFAØ      acknowledge.
```

The Peripheral Control Register bits are:

bit 7 = 1 to make CB2 an output

bit 6 = Ø to make CB2 a pulse

bit 5 = Ø to make CB2 a handshake acknowledgement that remains low until the next active transition on CB1

bit 4 = 1 to make the active transition on CB1 a rising edge (low-to-high transition)

bits Ø-3 = Ø since CA1 and CA2 are not used.

d. Send data to an output device that requires a brief OUTPUT or DATA READY strobe for multiplexing or control purposes. Fetch the data from memory location $4Ø.

```
08AC A9 FF       LDA #$FF     Make side B outputs.
08AE 8D A2 FF    STA $FFA2
08B1 A9 A0       LDA #$A0     Set CB2 Pulse Output
08B3 8D AC FF    STA $FFAC      Mode.  Get and send
08B6 A5 40       LDA $40        data and data strobe.
08B8 8D A0 FF    STA $FFA0
```

Bit 5 of the Peripheral Control Register is set to 1 to make CB2 a brief strobe lasting one cycle after the writing of Port B Output Data Register.

e. Fetch data from an input device that requires an active-high START pulse. The device produces an active high-to-low DATA READY strobe. Place the data in memory location $40.

```
08BB A9 00       LDA #$00     Make side A outputs.
08BD 8D A3 FF    STA $FFA3
08C0 A9 0C       LDA #$0C     Reset Start Pulse.
08C2 8D AC FF    STA $FFAC
08C5 A9 0E       LDA #$0E     Set Start Pulse high on
08C7 8D AC FF    STA $FFAC      CA2.
08CA A9 0C       LDA #$0C     Reset Start Pulse.
08CC 8D AC FF    STA $FFAC
08CF AD AD FF    LDA $FFAD
08D2 29 02       AND #$02
08D4 F0 F9       BEQ $08CF    Data ready?
08D6 AD A1 FF    LDA $FFA1    Yes, Get data.
08D9 85 40       STA $40      Store it.
```

Bit 2 of the Peripheral Control Register is set to 1 to make CA2 a level with the value given by bit 1 of the Peripheral Control Register. This mode can be used to produce pulses of any length and polarity; it is called the manual output mode because there is no automatic pulse information.

In a typical application, an analog-to-digital converter or data acquisition system usually needs a START CONVERSION pulse to begin operations.

f. Send data to an output device that must be turned on before the data is sent and turned off after the data is sent (a logic 1 on a control line turns the device on). The peripheral produces an active low-to-high PERIPHERAL READY strobe. Get the data from memory location $40 and send it when the peripheral is ready.

```
Ø8DB A9 FF      LDA #$FF     Make side B outputs.
Ø8DD 8D A2 FF   STA $FFA2
Ø8EØ A9 FØ      LDA #$FØ     Turn peripheral on by
Ø8E2 8D AC FF   STA $FFAC      setting CB2 high and
Ø8E5 AD AD FF   LDA $FFAD      set CB1 Interrupt Flag
Ø8E8 29 1Ø      AND #$1Ø       on rising edge.
Ø8EA FØ F9      BEQ $Ø8E5
Ø8EC A9 4Ø      LDA #$4Ø     Peripheral ready?
Ø8EE 8D AØ FF   STA $FFAØ    Yes, get data.
Ø8F1 A9 DØ      LDA #$DØ     Send it.
Ø8F3 8D AC FF   STA $FFAC    Turn peripheral off.
```

In many applications, such as portable equipment, the output peripheral is only turned on when data is to be sent to it. In other applications, the processor must issue an OUTPUT REQUEST and receive an acknowledgement before sending the data.

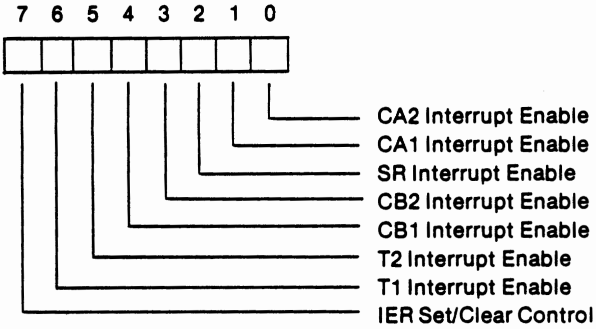
7.5 VIA INTERRUPTS

You can easily use the R6522 Versatile Interface Adapter in an interrupt-driven mode. Figure 7-2 shows the Interrupt Enable Register (IER). Any of the various interrupt sources can be enabled by setting the corresponding enable bit. Note that the most significant bit controls how the other enable bits are affected:

If IER7 = Ø, each '1' in a bit position clears an enable bit and thus disables that interrupt.

If IER7 = 1, each '1' in a bit position sets an interrupt bit and thus enables that interrupt.

Zeroes in the enabling bit positions always leave the enable bits as they were.



INTERRUPT ENABLE BITS (IER0-6)

IERn = 0 Disable interrupt
= 1 Enable interrupt

IER SET/CLEAR CONTROL (IER7)

IER7 = 0 For each data bus bit set to logic 1, clear corresponding IER bit
= 1 For each data bus bit set to logic 1, set corresponding IER bit.

Note: IER7 is active only when $R/\overline{W}$ = L; when $R/\overline{W}$ = H, IER7 will read logic 1.

Figure 7-2. R6522 Interrupt Enable Register (IER)

The following examples should help you see how this works.

a. Enable CA1 interrupt, disable all others.

```
Ø8F6 A9 7D      LDA #$7D     Disable other interrupts.
Ø8F8 8D AE FF   STA $FFAE
Ø8FB A9 82      LDA #$82     Enable CA1 interrupt.
Ø8FD 8D AE FF   STA $FFAE
```

The first operation clears all the interrupt enables, except CA1. The second operation sets the CA1 interrupt enable.

b. Enable CB1 and CB2 interrupts, disable all others.

```
Ø9ØØ A9 67      LDA #$67     Disable other interrupts.
Ø9Ø2 8D AE FF   STA $FFAE
Ø9Ø5 A9 98      LDA #$98     Enable CB1 and CB2
Ø9Ø7 8D AE FF   STA $FFAE      interrupts.
```

Note that we could disable all interrupts in the first step.

c. Disable CA1 interrupt, leave others as they were.

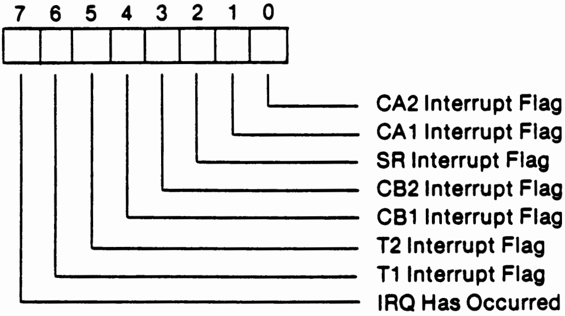
```
Ø9ØA A9 Ø2      LDA #$Ø2     Disable CA1 interrupt.
Ø9ØC 8D AE FF   STA $FFAE
```

d. Disable CB1 and CB2 interrupts, leave others as they were.

```
Ø9ØF A9 18 LDA #$18          Disable CB1 and CB2
Ø911 8D AE FF   STA $FFAE      interrupts.
```

The processor can determine which interrupt has occurred by examining the interrupt flag register (Figure 7-3). Note that examining bit 7 determines if any interrupts have occurred on the VIA. Note also the conditions for clearing the interrupt flags.

R6522 INTERRUPT FLAG REGISTER (IFR)



IFR Bit	Set By	Cleared By
0	Active transition on CA2	Reading or writing the ORA ($A001 or $A00F)
1	Active transition on CA1	Reading or writing the ORA ($A001 or $A00F)
2	Completion of eight shifts	Reading or writing the SR ($A00A)
3	Active transition on CB2	Reading or writing the ORB ($A000)
4	Active transition on CB1	Reading or writing the ORB ($A000)
5	Time-out of Timer 2	Reading T2C-L ($A008) or writing T2C-H ($A009)
6	Time-out of Timer 1	Reading T1C-L ($A004) or writing T1L-H ($A005 or $A007)
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 ($A00D) or IER0-IER6 ($A00E)

Figure 7-3. R6522 Interrupt Flag Register (IFR)

A typical polling sequence would be:

```
     LDA              $FFAD          ;ANY INTERRUPTS ON THIS VIA?
     BPL              NEXTS          ;NO, LOOK AT NEXT SOURCE
     ASL              A              ;IS INTERRUPT FROM T1?
     BMI              TIM1           ;YES, GO SERVICE T1 INTERRUPT
     ASL              A              ;IS INTERRUPT FROM T2?
     BMI              TIM2           ;YES, GO SERVICE T2 INTERRUPT
     ASL              A              ;IS INTERRUPT FROM CB1?
     BMI              CB1            ;YES, GO SERVICE CB1
                                     ;INTERRUPT
      .
      .
      .
     XXX
```

7.6 VIA TIMERS

The two 16-bit timer/counters in the R6522 Versatile Interface Adapter can be used to:

a. Generate a single time interval. The timer must be loaded with the number of clock pulses.

b. Count pulses on pin PB6 (Timer 2 only). The timer must be loaded with the number of pulses to be counted.

c. Generate continuous time intervals (Timer 1 only). The timer must be loaded with the number of clock pulses per interval.

d. Produce a single pulse or a continuous series of pulses on pin PB7 (Timer 1 only). The timer must be loaded with the number of clock pulses per interval.

7.6.1 Timer 2

Let us first look at Timer 2, which can only be used to generate a single time interval (the one-shot mode) or to count pulses on PB6. Bit 5 of the Auxiliary Control Register selects the mode:

Bit 5 = Ø for one-shot mode, 1 for pulse-counting mode

Note that 16-bit Timer 2 occupies two memory locations (see Table 7-1). The first address ($FFA8) is used to read or write the 8 least significant bits; reading this address also clears the Timer 2 Interrupt Flag (Figure 7-3). The second address ($FFA9) is used to read or write the 8 most significant bits; writing this address loads the counters, clears the timer 2 interrupt flag, and starts the timing operation. The completion of the operation sets the Timer 2 Interrupt Flag.

Examples:

a. Generate a delay of 2Ø48 (Ø8ØØ hex) clock pulses.

```
Ø914 A9 ØØ       LDA #$ØØ     Set T2 One-Shot Interval
Ø916 8D AB FF    STA $FFAB    Time Mode.
Ø919 8D A8 FF    STA $FFA8
Ø91C A9 Ø8       LDA #$Ø8     Load and start timer.
Ø91E 8D A9 FF    STA $FFA9
Ø921 A9 2Ø       LDA #$2Ø
Ø923 2C AD FF    BIT $FFAD
Ø926 FØ FB       BEQ $Ø923    T2 counted down?
Ø928 AD A8 FF    LDA $FFA8    Yes, clear T2 Interrupt Flag.
```

Note the final LDA $FFA8. The sole purpose of this instruction is to clear the timer 2 interrupt flag so that it will be available for the next operation.

b. Count 5 input pulses on PB6.

```
Ø92B A9 ØØ       LDA #$ØØ     Make side B inputs.
Ø92D 8D A2 FF    STA $FFA2
Ø93Ø A9 2Ø       LDA #$2Ø     Set T2 Pulse Count Mode.
Ø932 8D AB FF    STA $FFAB
Ø935 A9 Ø5       LDA #$Ø5     Load delay of 5 counts.
Ø937 8D AB FF    STA $FFAB
Ø93A A9 ØØ       LDA #$ØØ
Ø93C 8D A9 FF    STA $FFA9
Ø93F A9 2Ø       LDA #$2Ø
Ø941 2C AD FF    BIT $FFAD
Ø944 FØ FB       BEQ $Ø941    T2 counted down?
Ø946 AD A8 FF    LDA $FFA8    Yes, Clear T2 Interrupt Flag.
```

Note that you must load the least significant bits of the timer first, since loading the most significant bits loads the counters and starts the timing operation.

7.6.2 Timer 1

Timer 1 is somewhat more complex than Timer 2 because it has four operating modes (see Table 7-2). It can be used to generate a single time interval (one-shot mode) or a continuous series of intervals (free-running mode). Furthermore, each loading operation can generate an output pulse on PB7. Bits 6 and 7 of the Auxiliary Control Register determine timer 1 mode.

> Bit 7 = 1 to generate output pulses on PB7, Ø to disable such pulses (in the free-running mode, PB7 is inverted each time the counter reaches zero).
>
> Bit 6 = 1 for free-running mode, Ø for one-shot mode.

Table 7-2. Timer 1 Control

ACR7	ACR6	Mode
0	0	T1 one-shot mode — Generate a single time-out interrupt each time T1 is loaded. Output to PB7 disabled.
0	1	T1 free-running mode — Generate continuous interrupts. Output to PB7 disabled.
1	0	T1 one-shot mode — Generate a single time-out interrupt and an output pulse on PB7 each time T1 is loaded.
1	1	T1 free-running mode — Generate continuous interrupts and a square wave output on PB7.

Timer 1 occupies four memory locations (see Table 7-1). The first two addresses ($FFA4 and $FFA5) are used to read or write the counters. Writing the second address loads the counters, clears the Timer 1 Interrupt Flag, and starts the timing operation. The next two addresses ($FFA6 and $FFA7) are used to read or write the latches without affecting the counters. This allows the generation of complex waveforms in the free-running mode. Writing the most significant bits of the latches also clears the Timer 1 Interrupt Flag.

Examples:

a. Generate a delay of 1024 (0400 hex) clock pulses, no output to PB7.

```
0949 A9 00       LDA #$00       Set T1 One-Shot Mode.
094B 8D AB FF    STA $FFAB
094E 8D A4 FF    STA $FFA4
0951 A9 04       LDA #$04
0953 8D A5 FF    STA $FFA5      Set delay to 0400 counts.
0956 A9 40       LDA #$40
0958 2C AD FF    BIT $FFAD
095B F0 FB       BEQ $0958      T1 counted down?
095D AD A4 FF    LDA $FFA4      Yes, clear T1 Int Flag.
```

This program is the same as the one shown previously for Timer 2 except for the changes in the addresses and the interrupt flag bit position.

b. Generate an interrupt every 4096 (1000 hex) clock pulses and produce a pulse output on PB7 with a pulse width of 4096 clock pulses. Note that Timer 1 is loaded with the desired clock count minus two, e.g. $0FFE.

```
0960 A9 FF       LDA #$FF       Make side B inputs.
0962 8D A2 FF    STA $FFA2
0965 A9 C0       LDA #$C0       Set T1 Free-Run Mode and PB7
0967 8D AB FF    STA $FFAB        Active.
096A 8D AE FF    STA $FFAE      Set Interrupt Enable
096D A9 FE       LDA #$FE       Set delay to 100O counts.
096F 8D A4 FF    STA $FFA4
0972 A9 0F       LDA #$0F       Start T1.
0974 8D A5 FF    STA $FFA5
0977 58          CLI            Enable IRQ Interrupt.
     .
     .
continue
     .
     .
IRQ Interrupt Processing
0978 A9 40       LDA #$40       IRQ Interrupt Processing.
097A 2C AD FF    BIT $FFAD
097D D0 03       BNE $0982      T1 cause IRQ?
097F AD A4 FF    LDA $FFA4      Yes, clear T1 Int Flag.
0982 ....                       Continue.
09XX 40          RTI            Return from interrupt
                                  processing

[M]=FFA0 88 03 7B E0
```

This program will cause the processor to be interrupted every 4Ø96 clock cycles. The level on PB7 will be inverted at the end of each interval (it will go low when the first interval starts). Note that Tl runs continuously; whenever the counters reach zero, they are reloaded with the values in the latches.

7.7 VIA SHIFT REGISTER

The VIA also has a shift register which can be used to convert data between serial (external serial port format) wand parallel (internal format) forms. Auxiliary Control Register bits 2, 3, and 4 control this register as shown in Table 7-3. Loading the register starts the shifting process and an interrupt flag (bit 2 of the Interrupt Flag Register) is set after the completion of eight shifts.

Table 7-3. Shift Register Control

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of ø2.
0	1	1	Shift in under control of external clock.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of ø2.
1	1	1	Shift out under control of external clock.

Examples:

a. Shift in 8 bits under control of Timer 2 and store the data into memory location $4Ø. Subtract 2 from the desired time value (in microseconds) to determine the times 2 count value.

```
Ø983 A9 ØØ       LDA #$ØØ
Ø985 8D AB FF    STA $FFAB     Disable SR.
Ø988 A9 Ø4       LDA #$Ø4
Ø98A 8D AB FF    STA $FFAB     Set SR to shift in by T2.
Ø98D A9 28       LDA #$28
Ø98F 8D A8 FF    STA $FFA8     Set T2 delay to 4Ø counts.
Ø992 A9 ØØ       LDA #$ØØ
Ø994 8D A9 FF    STA $FFA9
Ø997 AD AA       LDA $FFAA     Start SR.
Ø99A AD AD FF    LDA $FFAD
Ø99D 29 Ø4       AND #$Ø4
Ø99F FØ F9       BEQ $Ø99A     Shift in complete?
Ø9A1 A9 ØØ       LDA #$ØØ      Yes, disable SR.
Ø9A3 8D AB FF    STA $FFAB
Ø9A6 AD AA FF    LDA $FFAA     Get data and store it.
Ø9A9 85 4Ø       STA $4Ø
```

b. Shift in 8 bits under control of phase 2 (Ø2) and store

```
Ø9AB A9 ØØ       LDA #$ØØ      Disable SR.
Ø9AD 8D AB FF    STA $FFAB
Ø9BØ A9 Ø8       LDA #$Ø8      Set SR to shift in by Ø2.
Ø9B2 8D AB FF    STA $FFAB
Ø9B5 AD AA FF    LDA $FFAA     Start shift in.
Ø9B8 A2 Ø4       LDX #$Ø4      Delay 18 cycles.
Ø9BA CA          DEX
Ø9BB DØ FD       BNE $Ø9BA     Delay (shift) complete?
Ø9BD 8D AB FF    STA $FFAB     Yes, disable SR.
Ø9CØ AD AA FF    LDA $FFAA     Get and store data.
Ø9C3 85 4Ø       STA $4Ø
```

c. Shift in 8 bits under control of an external clock and store the data into memory location $4Ø.

```
Ø9C5 A9 ØØ       LDA #$ØØ      Disable SR.
Ø9C7 8D AB FF    STA $FFAB
Ø9CA A9 ØC       LDA #$ØC      Set SR to Shift in by
Ø9CC 8D AB FF    STA $FFAB       external clock.
Ø9CF AD AA FF    LDA $FFAA     Start SR.
Ø9D2 AD AD FF    LDA $FFAD
Ø9D5 29 Ø4       AND #$Ø4
Ø9D7 FØ F9       BEQ $Ø9D2     Shift complete?
Ø9D9 AD AA FF    LDA $FFAA     Yes, get and store data.
Ø9DC 85 4Ø       STA $4Ø
```

d. Continually shift out 8 bits from memory location $4Ø at Timer 2 rate.

```
Ø9DE A9 ØØ       LDA #$ØØ      Disable SR.
Ø9EØ 8D AB FF    STA $FFAB
Ø9E3 A9 1O       LDA #$1Ø      Set SR to shift out
Ø9E5 8D AB FF    STA $FFAB       continuously by T2.
Ø9E8 A9 8Ø       LDA #$8Ø      Set delay to 5Ø counts.
Ø9EA 8D A8 FF    STA $FFA8
Ø9ED A9 ØØ       LDA #$ØØ
Ø9EF 8D A9 FF    STA $FFA9
```

```
Ø9F2 A9 4Ø       LDA #$4Ø      Load SR and start shift out
Ø9F4 8D AA FF    STA $FFAA
```

e. Shift out 8 bits from memory location $4Ø under control of Timer 2. Subtract 2 from the desired time value (in microseconds) to determine the times 2 count value.

```
Ø9F7 A9 ØØ       LDA #$ØØ      Disable SR.
Ø9F9 8D AB FF    STA $FFAB
Ø9FC A9 14       LDA #$14      Set SR to shift out by T2.
Ø9FE 8D AB FF    STA $FFAB
ØAØ1 A9 2Ø       LDA #$2Ø      Set delay to 32 counts.
ØAØ3 8D A8 FF    STA $FFA8
ØAØ6 A9 ØØ       LDA #$ØØ
ØAØ8 8D A9 FF    STA $FFA9
ØAØB A5 4Ø       LDA $4Ø       Load data into SR and start
ØAØD 8D AA FF    STA $FFAA       shift out.
```

f. Shift out 8 bits from memory location $4Ø under control of phase 2 (Ø2).

```
ØA1Ø A9 ØØ       LDA #$ØØ      Disable SR.
ØA12 8D AB FF    STA $FFAB
ØA15 A9 18       LDA #$18      Set SR to shift out by Ø2.
ØA17 8D AB FF    STA $FFAB
ØA1A A5 4Ø       LDA $4Ø       Load SR with data and start
ØA1C 8D AA FF    STA $FFAA       shift out.
```

g. Shift out 8 bit from memory location $4Ø under control of an external clock with IRQ interrupt handling.

```
ØA1F A9 ØØ       LDA #$ØØ      Disable SR.
ØA21 8D AB FF    STA $FFAB
ØA24 A9 1C       LDA #$1C      Set SR to shift out by
ØA26 8D AB FF    STA $FFAB       external clock.
ØA29 A9 84       LDA #$84      Enable SR IRQ Interrupt.
ØA2B 8D AE FF    STA $FFAE
ØA2E A5 4Ø       LDA $4Ø       Load SR with data and start
ØA3Ø 8D AA FF    STA $FFAA       shift out.
ØA33 58          CLI           Enable IRQ
           .
           .
           .
Continue processing
           .
           .
           .
IRQ Interrupt Processing

ØA34 A5 4Ø       LDA $4Ø
ØA36 2C AD FF    BIT $FFAD
ØA39 DØ Ø3       BNE $ØA3E     SR cause IRQ Interrupt?
```

```
0A3B AD AA FF   LDA $FFAA     Yes, clear SR Interrupt Flag.
0A3E ....       ...           Continue IRQ processing.
                              Return from interrupt.
0AXX 40         RTI
[M]=FFA0 45 04 7B E0
```

7.8 PARALLEL PRINTER DRIVER EXAMPLE

Figure 7-4 shows an assembly listing of a parallel output driver. This driver can be used to drive a Centronics interface compatible printer connected to the SBC connector J1.

The SBC connector J1 pins should be connected to the printer interface pins as shown for the AIM 65/40 printer interface on SBC connector J5 (see Tables L-1 and L-5).

To use the driver, type U in response to the OUT=prompt.

```
ADDR .OBJECT. SOURCE

0800          ;FOR CENTRONICS PRINTERS
0800          ;WITH NOT DATA STROBE
0800          ;AND NOT DATA ACKNOWLEDGE
0800          ;
0800          ; USER VIA REGISTERS
0800                 *=$FFA0
FFA0          PORTB  *=*+1
FFA1          PORTA  *=*+1
FFA2          DDRB   *=*+1
FFA3          DDRA   *=*+9
FFAC          PCR    *=*+1
FFAD          IFR=*
FFAD          ;
FFAD          ; USER VIA INITIALIZATION
FFAD          DRB=$01
; ONLY THE STROBE (BIT 0) IS USED
FFAD          DRA=$FF
; THIS IS ALL OUTPUTS
FFAD          IPCR=$01
; CA1 RISING EDGE,CLEAR ON WRITE
FFAD          STROB0=$FE
; FORCE BIT 0 LOW (AND)
FFAD          STROB1=$01
; FORCE BIT 0 HI (OR)
FFAD          ;
FFAD          ; I/O JUMP TABLE POINTER
FFAD          ; V ISSUES CR+LF+NULLS
FFAD                 *=$21A
021A 00 08    IOVV   .WOR POUT
021C          ;
021C          ; TABLE OF JUMP VECTORS
021C                 *=$800
0800 4C 09 08  POUT  JMP OPEN
0803 4C 26 08        JMP OUTPUT
0806 4C 48 08        JMP CLOSE
0809          ;
0809          ; OPEN THE PARALLEL OUTPUT
 DEVICE
0809 A9 0D    OPEN   LDA #$0D
080B 8D A1 FF        STA PORTA
080E A9 00           LDA #0
0810 8D A0 FF        STA PORTB
0813 A9 01           LDA #DRB
0815 8D A2 FF        STA DDRB
0818 A9 FF           LDA #DRA
081A 8D A3 FF        STA DDRA
081D A9 01           LDA #IPCR
081F 8D AC FF        STA PCR
0822 20 3A 08        JSR STROBE
0825 60              RTS
```

Figure 7-4. Parallel Output Printer Driver

```
ADDR .OBJECT. SOURCE

0826          ;
0826          ; OUTPUT DATA ROUTINE
0826 48       OUTPUT PHA
0827 20 32 08        JSR WAIT
082A 68              PLA
082B 8D A1 FF        STA PORTA
082E 20 3A 08        JSR STROBE
0831 60              RTS
0832          ;WAIT TILL ACK IS RECEIVED
0832 AD AD FF  WAIT  LDA IFR
0835 4A              LSR A
0836 4A              LSR A
0837 90 F9           BCC WAIT
0839 60              RTS
083A          ; HANDSHAKE OFF CHARACTER
083A A9 FE    STROBE LDA #STROBO
083C 2D A0 FF        AND PORTB
083F 8D A0 FF        STA PORTB
0842 09 01           ORA #STROB1
0844 8D A0 FF        STA PORTB
0847 60              RTS
0848          ;
0848          ; CLOSE THE OUTPUT DEVICE
0848 A9 00    CLOSE  LDA #0
084A 8D A3 FF        STA DDRA
084D 8D A2 FF        STA DDRB
0850 60              RTS
0851                 .END
 ERRORS=0000
```

Figure 7-4. Parallel Output Printer Driver (Continued)

SECTION 8

USING THE RS-232C SERIAL INTERFACE

The SBC module RS-232C interface provides an industry standard asynchronous serial data transmission capability between the AIM 65/40 system and external equipment. This interface is controlled by the programmable R6551 Asynchronous communications Interface Adapter (ACIA) device (Z2), the JACIA-1 through JACIA-4 jumpers (see Section 2.3.4) and a user provided driver program.

This section describes the fundamental operation of the ACIA device and how to configure the AIM 65/40 as a data terminal (a receover)or a data set (i.e. a transmitter). Sample serial driver programs to support these two functions are illustrated.

8.1 FEATURES OF THE R6551 ACIA

The major features of the R6551 ACIA include:

- Full duplex operation with buffered receiver and transmitter
- Data set/modem control functions
- Internal baud rate generator with 15 programmable baud rates (50 to 19,200)
- Program-selectable internally or externally controlled receiver rate
- Programmable word lengths
- Number of stop bits
- Parity bit generation and detection
- Programmable interrupt control
- Program reset
- Program-selectable serial echo mode

Figure 8-1 shows a block diagram of the ACIA device. Four registers are used to process the received and transmitted data, the command and control signals and the status of the operation. The addresses for these registers are listed in Table 6-8.

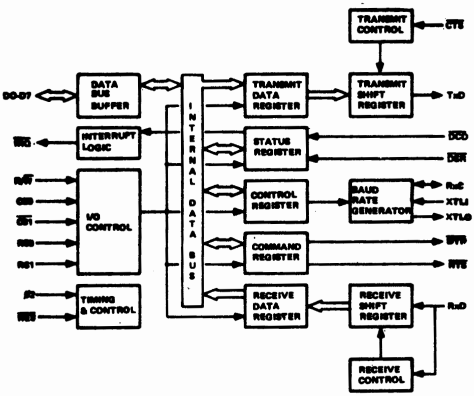


Figure 8-1. R6551 ACIA Block Diagram

Table 8-1 defines the RS-232C signals at the SBC module Connector J2 interface. These definitions apply to the use of the ACIA device since the RS-232C interface circuit (see Section 11.10) is essentially transparent to functional operation.

8.2 R6551 REGISTER AND CONTROL FUNCTIONS

To use the R6551 ACIA device, you must first understand how the internal registers operate. This section summarizes the register contents and bit definitions.

8.2.1 Transmitter and Receiver Data Registers

These registers are used as temporary data storage for the R6551 Transmit and Receive Circuits. Both the Transmitter and Receiver are selected when the R6551 is addressed (i.e. addresses $FF00 - FF03). The R/$\overline{W}$ line determines which actually uses the internal data bus; the Transmitter Data Register is write only and the Receiver Data Register is read only. Figure 8-2 illustrates the data register format and the flow of serial data.

a. Transmitted Data

Bit 0 is the first bit to be transmitted from the transmitter Data Register (least significant bit first). The higher order bits follow in order. Unused bits in this register are "don't care".

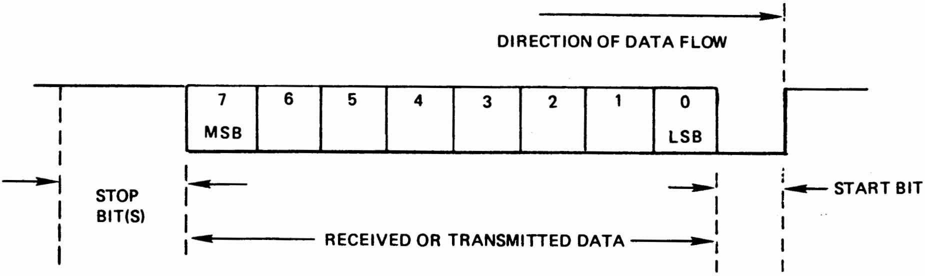


Figure 8-2. R6551 ACIA Data Register Format

Table 8-1. SBC Connector J2 (RS-232C) Signal Definitions

Signal Mnemonic	Signal Name and Signal Description
GND	Signal Ground Ground for logic circuitry associated with RS-232C interface.
$\overline{\text{RD}}$	Receive Data For data terminal operation, serial NRZ data is received on $\overline{\text{RD}}$ from an external device by the SBC module. For data set operation, serial NRZ data is transmitted on $\overline{\text{RD}}$ from the SBC module to an external device. During quiescent (no data) states, the $\overline{\text{RD}}$ line is in a marking (negative voltage level) condition. The start bit (LSB) of a word on the $\overline{\text{RD}}$ line causes $\overline{\text{RD}}$ to go to a space (positive voltage level), followed by a mark or space for each bit of the word, and ending with mark(s) for the stop bit(s). The data rate on RD line is determined by the programmed baud rate.
$\overline{\text{TD}}$	Transmit Data For data terminal operation, serial NRZ data is transmitted on $\overline{\text{TD}}$ from the SBC module to an external device. For data set operation, serial NRZ data received on $\overline{\text{TD}}$ from an external device by the SBC module.

Table 8-1. SBC Connector J2 (RS-232C) Signal Definitions (Continued)

Signal Mnemonic	Signal Name and Signal Description
	During quiescent states, the $\overline{TD}$ line is in a marking condition. The start bit of each word on the $\overline{TD}$ line causes $\overline{TD}$ to go to a space, followed by a mark or space for each bit of the word, and ending with mark(s) for each stop bit(s). The data rate on the $\overline{TD}$ line is determined by the programmed baud rate.
RTS	Request To Send Request to send is generated by the SBC module for an external device. The condition (mark or space) of the RTS signal transmitted from the module is program controlled. RTS is active in the space condition. For data set operation, control signal RTS is received from an external device by the DCD line receiver on the ACIA module. A transition on RTS generates an interrupt request to the Interrupt Request Prioritizer. A space on DCD indicates that the external device is requesting to send.
CTS	Clear To Send For data terminal operation, control signal CTS is received from an external device by the SBC module. Signal CTS controls transmissions from the R6551. The transmission is enabled with a CTS space and is disabled with a CTS mark.

Table 8-1. SBC Connector J2 (RS-232C) Signal Definitions (Continued)

Signal Mnemonic	Signal Name and Signal Description
	For data set operation, control signal CTS is generated by the SBC module for an external device. The condition of CTS is set by the RTS line of the R6551 under program control. CTS is active in the space condition.
DSR	Data Set Ready For data terminal operation, control signal DSR is received from an external device by the the SBC module. A transition on DSR generates an interrupt request to the Interrupt Request Prioritizer. A space condition on DSR indicates that the external device is ready. For data set operation, control signal DSR is generated by the SBC module for an external device. The condition of DSR is set by the DTR line of the R6551 under program control. A space condition on DSR indicates to an external device the SBC module is ready.
DTR	Data Terminal Ready For data terminal operation, control signal DTR is generated by the SBC module for an external device. The condition of DTR is program controlled. A space condition on DTR indicates that the SBC module is ready to the external device. For data set operation, control signal DTR is received from an external device by the DSR line on the SBC module. A transition on DTR generates an interrupt request to the Interrupt Request Prioritizer. A space condition on DTR indicates that the external device is ready.

Table 8-1. SBC Connector J2 (RS-232C) Signal Definitions (Continued)

Signal Mnemonic	Signal Name and Signal Description
DCD	<u>Data Carrier Detected</u> For data terminal operation, control signal DCD is received from an external device by the DCD line on the SBC module. A transition on DCD generates an interrupt request to the RM 65 Bus. A space condition on DCD indicates that the external device has detected a carrier. For data set operation, control signal DCD is generated by the SBC module for an external device. This line is a high (active) level. There is always a space condition on the DCD line.

b. Received Data

The Receiver Data Register holds the first received data bit in bit 0 (least significant bit first). Unused high-order bits are "0". Parity bits are not contained in the Receiver Data Register. They are stripped off after being used for parity checking.

8.2.2 R6551 ACIA Status Register

The Status Register indicates the states of the $\overline{\text{IRQ}}$, $\overline{\text{DSR}}$, and $\overline{\text{DCD}}$ lines, Transmitter and Receiver Data Registers, and Overrun, Framing and Parity Error conditions.

Table 8-3 indicates the format of the R6551 Status Register. A description of each status bit follows.

a. Receiver Data Register Full (Bit 3)

This bit goes to a "1" when the R6551 transfers data from the Receiver Shift Register to the Receiver Data Register, and goes to a "0" when the processor reads the Receiver Data Register.

b. Transmitter Data Register Empty (Bit 4)

This bit goes to a "1" when the R6551 transfers data from the Transmitter Data Register to the Transmitter Shift Register, and goes to a "0" when the processor writes new data onto the Transmitter Data Register.

c. Data Carrier Detect (Bit 5) and Data Set Ready (Bit 6)

These bits reflect the levels of the $\overline{\text{DCD}}$ and $\overline{\text{DSR}}$ inputs to the R6551. A "0" indicates a low level (true condition) and a "1" indicates a high (false). Whenever either of these inputs change state, an immediate processor interrupt occurs, unless the R6551 is disabled (bit 0 of the Command Register is a "0"). When the interrupt occurs, the status

Table 8-3. R6551 ACIA Status Register

Bit No.	Signal Definition
Ø	PARITY ERROR (2) Ø = No Parity Error 1 = Parity Error Detected
1	FRAMING ERROR (2) Ø = No Framing Error 1 = Framing Error Detected
2	OVERRUN (3) Ø = No Overrun 1 = Overrun Has Occurred
3	RECEIVER DATA REGISTER FULL Ø = Not Full 1 = Full
4	TRANSMITTER DATA REGISTER EMPTY Ø = Not Empty 1 = Empty
5	DATA CARRIER DETECT ($\overline{\text{DCD}}$) (3) Ø = $\overline{\text{DCD}}$ Low (Detect) 1 = $\overline{\text{DCD}}$ High (Not Detected)
6	DATA SET READY ($\overline{\text{DSR}}$) (3) Ø = $\overline{\text{DSR}}$ Low (Ready) 1 = $\overline{\text{DSR}}$ High (Not Ready)
7	INTERRUPT (IRQ) Ø = No Interrupt 1 = Interrupt Has Occurred

NOTES

(1) Status Register Reset Condition:

Bit No.								
7	6	5	4	3	2	1	Ø	
Ø	-	-	1	Ø	Ø	Ø	Ø	Hardware Reset ($\overline{\text{RES}}$)
-	-	-	-	-	Ø	-	-	Program Reset (where - = unaffected)

(2) No Interrupt occurs for these conditions.

(3) These interrupts can only be disabled by programming bit Ø in the ACIA command register to a Ø which disables the transmit and receive also.

bits will indicate the levels of the inputs immediately after the change of state occurred. Subsequent level changes will not affect the status bits until the Status Register is interrogated by the processor. At that time, another interrupt will immediately occur and the status bits will reflect the new input levels.

d. Framing Error (Bit 1), Overrun (Bit 2), and Parity Error (Bit 0)

None of these bits causes a processor interrupt to occur, but they are normally checked at the time the Receiver Data Register is read so that the validity of the data can be verified.

e. Interrupt (Bit 7)

The bit goes to a "0" when the Status Register has been read by the processor, and goes to a "1" whenever any kind of interrupt occurs.

8.2.3 R6551 ACIA Control Register

The Control Register selects the desired baud rate, frequency source, word length, and the number of stop bits as shown in Table 8-4.

a. Selected Baud Rate (Bits 0, 1, 2, 3)

These bits, set by the processor, select the Transmitter baud rate, which can be at 1/16 an external clock rate or one of 15 other rates controlled by the internal buad rate generator.

Table 8-4. R6551 ACIA Control Register

Bit No.	Signal Definition		
	<u>SELECTED BAUD RATE (SBR)</u>		
3-Ø	<u>3</u> <u>2</u> <u>1</u> <u>Ø</u>		
	Ø Ø Ø Ø	16x External Clock (2)	
	Ø Ø Ø 1	5Ø	Baud
	Ø Ø 1 Ø	75	Baud
	Ø Ø 1 1	1Ø9.92	Baud
	Ø 1 Ø Ø	134.58	Baud
	Ø 1 Ø 1	15Ø	Baud
	Ø 1 1 Ø	3ØØ	Baud
	Ø 1 1 1	6ØØ	Baud
	1 Ø Ø Ø	12ØØ	Baud
	1 Ø Ø 1	18ØØ	Baud
	1 Ø 1 Ø	24ØØ	Baud
	1 Ø 1 1	36ØØ	Baud
	1 1 Ø Ø	48ØØ	Baud
	1 1 Ø 1	72ØØ	Baud
	1 1 1 Ø	96ØØ	Baud
	1 1 1 1	19,2ØØ	Baud
4	<u>RECEIVE CLOCK SOURCE (RCS)</u>		
	Ø = External Receiver Clock (2)		
	1 = Baud Rate		
6-5	<u>WORD LENGTH (WL)</u>		
	<u>6</u> <u>5</u>		
	Ø Ø	8 Bits	
	Ø 1	7 Bits	
	1 Ø	6 Bits	
	1 1	5 Bits	
7	<u>STOP BIT NUMBER (SBN)</u>		
	Ø = 1 Stop Bit		
	1 = 2 Stop Bit		
	= 1 1/2 Stop Bits (For WL = 5 and No Parity)		
	= 1 Stop Bit (For WL = 8 and Parity)		

NOTES

(1) Control Register Reset Condition:

Bit No.	
7 6 5 5 3 2 1 Ø	
Ø Ø Ø Ø Ø Ø Ø Ø	Hardware Reset (RES)
- - - - - - - -	Program Reset (where - = unaffected)

(2) Not used on AIM 65/4Ø SBC Module.

b. Receiver Clock Source (Bit 4)

This bit controls the clock source to the Receiver. A "0" causes the Receiver to operate a baud rate of 1/16 an external clock. A "1" causes the Receiver to operate at the same baud rate as is selected for the transmitter as shown in Figure 3-3.

c. Word Length (Bits 5, 6)

These bits determine the word length to be used (5, 6, 7 or 8 bits). Figure 3-3 shows the configuration for each number of bits desired.

d. Stop Bit Number (Bit 7)

This bit determines the number of stop bits used. A "0" always indicates one stop bit. A "1" indicates 1-1/2 stop bits if the word length is 5 with no parity selected, 1 stop bit if the word length is 8 with parity selected, and 2 stop bits in all other configuration.

8.2.4 R6551 ACIA Command Register

The Command Register controls parity, receiver echo mode, transmitter interrupt control, the state of the $\overline{\text{RTS}}$ line, receiver interrupt control, and the state of the $\overline{\text{DTR}}$ line (see Table 8-5).

a. Data Terminal Ready (Bit 0)

This bit enables all selected interrupts and controls the state of the Data Terminal Ready ($\overline{\text{DTR}}$) line. A "0" indicates the processor system is not ready by setting the $\overline{\text{DTR}}$ line high. A "1" indicates the processor is ready by setting the DTR line low.

b. Receiver Interrupt Control (Bit 1)

This bit disables the Receiver from generating an interrupt when set to a "1". The Receiver interrupt is enabled when this bit is set to a "Ø" and Bit Ø is set to a "1".

c. Transmitter Interrupt Control (Bits 2, 3)

These bits control the state of the Ready to Send ($\overline{\text{RTS}}$) line and Transmitter interrupt. Table 8-5 shows the various configurations of the $\overline{\text{RTS}}$ line and Transmit Interrupt bit settings.

d. Receiver Echo Mode (Bit 4)

This bit enables the Receiver Echo Mode. Bits 2 and 3 must also be zero. In the Receiver Echo Mode, the Transmitter returns each transmission received by the Receiver delayed by one-half bit time. A "1" enables the Receiver Echo Mode. A "Ø" bit disables the mode.

e. Parity Mode Enable (Bit 5)

This bit enables parity bit generation and checking. A "Ø" disables parity bit generation by the Transmitter and parity bit checking by the Receiver. A "1" bit enables generation and checking of parity bits.

f. Parity Mode Control (Bits 6, 7)

These bits determine the type of parity generated by the Transmitter, (even, odd, mark or space) and the type of parity check done by the Receiver (even, odd, or no check). Table 8-5 shows the possible bit configurations for the Parity Mode Control bits.

Table 8-5. R6551 ACIA Command Register

Bit No.	Signal Definition
Ø	DATA TERMINAL READY (DTR) Ø = Disable Receiver and Transmitter ($\overline{\text{DTR}}$ High) 1 = Enable Receiver and Transmitter ($\overline{\text{DTR}}$ Low)
1	RECEIVER INTERRUPT CONTROL Ø = $\overline{\text{IRQ}}$ Interrupt Enabled from Bit 3 of Status Register 1 = $\overline{\text{IRQ}}$ Interrupt Disabled from Bit 3 of Status Register
3-2	TRANSMITTER INTERRUPT AND REQUEST-TO-SEND CONTROL 3 2 Ø Ø $\overline{\text{RTS}}$ = High = RTS Disabled (except if echo mode enabled). Transmit Interrupt Disabled from Bit 4 of Status Register. (2) Ø 1 $\overline{\text{RTS}}$ = Low = RTS Enabled. Transmit Interrupt Enabled from Bit 4 of Status Register. 1 Ø $\overline{\text{RTS}}$ = Low = RTS Enabled. Transmit Interrupt Disabled from Bit 4 of Status Register. 1 1 $\overline{\text{RTS}}$ = Low = RTS Enabled. Transmit Interrupt Disabled from Bit 4 of Status Register. Transmit BRK on TD output.
4	RECEIVER ECHO MODE Ø = Receiver Normal Mode 1 = Receiver Echo Mode (Bits 2 and 3 must = "Ø") (Note: $\overline{\text{RTS}}$ is enabled to the low (active) state in Echo mode)
5	PARITY MODE ENABLED Ø = Parity Disabled No Parity Bit Generated, Parity Check Disabled. 1 = Parity Enabled
7-6	PARITY CHECK CONTROL (Bit 5 must = 1) 7 6 Ø Ø Odd Parity Transmitted/Received Ø 1 Even Parity Transmitted/Received 1 Ø Mark Parity Bit Transmitted, Parity Check Disabled 1 1 Space Parity Bit Transmitted, Parity check Disabled

Table 8-5. R6551 ACIA Command Register
(Continued)

NOTES

(1) Command Register Reset Condition:

Bit No.

7	6	5	4	3	2	1	Ø	
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Hardware Reset ($\overline{RES}$)
-	-	-	Ø	Ø	Ø	Ø	Ø	Program Reset (where - = unaffected)

(2) Bits 2 and 3 must be zero for receiver echo mode, $\overline{RTS}$ will be low

8.3 PROGRAMMING CONSIDERATIONS

8.3.1 General Considerations

Prior to writing serial channel driver program, determine the serial interface operating characteristics of the external equipment that is to be connected to SBC module, e.g.:

- What type of external RS-232 device is the to communicate with?

- Is the external RS-232 device to operate as a data terminal or a data set?

- Which control signals do the external devices require and supply?

- What baud rate, word length, number of stop bits, and parity does the external device require?

8.3.2 ACIA Module Memory Map

The user R6551 ACIA devices is assigned I/O address space within the SBC module memory map (see Table 6-1). The specific register addresses are:

Address	Register Accessed	
	(R/$\overline{W}$=Low)	(R/$\overline{W}$=High)
FFDØ	Write Transmit Data Register	Read Receive Data Register
FFD1	Program Reset	Read Status Register
FFD2	Write Command Register	Command Register
FFD3	Write Control Register	Control Register

Note that the system can read from, or write into the command and control registers; only write into the transmit register; and only read from the receive register and status register.

8.3.3 Initialization Software

The ACIA initialization software must:

a. Execute a program reset (write operation to address FFD-1 to reset the status, control, and command registers in both ports to the initialization values shown in Tables 8-2 and 8-3, and 8-4. Note that a program reset disables IRQ interrupts.

b. Write the Baud Rate, Word Length, and Numbers of Stop Bits into the R6551 control register.

c. Set R6551 control register bit 4 = 1, because the SBC module does not use an external clock source.

d. Write the required parity control configuration into bits 7-5 in the R6551 command register.

8.3.4 Main-line Software

Main-line software must be designed that will:

a. Set R6551 command register bit Ø=1 when the serial channel is to appear "ready" to the external device.

NOTE

With command register bit Ø=1, the DCD (Data Carrier Detected) and DSR (Data Terminal Ready) IRQ interrupts are enabled.

b. Set R6551 command register bit 1=1 when Receive Register Full IRQ interrupts to the microprocessor are to be disabled.

c. Set R6551 command register bits 3 and 2=01 when Transmit Data Register Empty $\overline{\text{IRQ}}$ interrupts to the microprocessor and Request-to-Send (RTS) responses to the external device are to be enabled. Note that when command bits 3 and 2 are set to 10 or 11 that the transmit interrupt is disabled but RTS (Request to Send) is enabled. Bits 3 and 2=10 can be used for data set operation where the $\overline{\text{RTS}}$ output from the ACIA is used to generate a Clear to Send. Bits 3 and 2=11 can be used to disable Transmit Data Register Empty interrupts when a Break character is transmitted at the end of an output message.

d. If applicable, set R6551 command register bit 4=1 whenever the Echo mode is to be used. The echo mode can be used to monitor data transmitted from the ACIA module .by feeding the transmitted data back to the microcomputer via the R6551 receive data register.

e. If the software is interrupt structured, provide $\overline{\text{IRQ}}$ interrupt recognition and polling routines. The ACIA module may be polled by reading the status register and examining the individual bits. Interrupt recognition and polling maximum response times are predicated on the baud rate.

f. If applicable, check for parity, framing, and/or overrun error conditions in R6551 status register bits 0, 1, and 2 respectively, and provide proper error processing.

g. Provide I/O handlers that are compatible with the types of external devices and the ACIA data set or data terminal channel operation.

8.3.5 ACIA Interrupts

The R6551 device has the capability to generate an interrupt request ($\overline{\text{IRQ}}$) to the CPU for any of the following conditions:

a. A transition on the $\overline{\text{DSR}}$ (Data Set Ready) input to the R6551. This indicates a change in DSR status in the external device when the ACIA channel is configured as a data terminal or indicates a change in DTR status in the external device when the ACIA channel is configured as a data set.

b. A transition on the $\overline{\text{DCD}}$ (Data Carrier Detected) input to the R6551. This indicates a change in DCD status in the external device when the ACIA channel is configured as a data terminal or indicates a change in RTS status in the external device when the ACIA channel is configured as a data set.

c. A Transmit Register Empty condition which indicates that the R6551 transmit register is ready to receive an output character for transmission to an external device.

d. A Receive Register Full condition which indicates that a complete character has been received by the R6551 from an external device.

True Data Set Ready and Data Carrier Detected status are indicated by "1"s in bits 6 and 5, respectively, of the R6551 status register. A status change on $\overline{\text{DSR}}$ or $\overline{\text{DCD}}$ unconditionally sets the Interrupt Flag (bit 7) of the status register to a "1" which generates an interrupt request ($\overline{\text{IRQ}}$) to the processor via the Interrupt Request Prioritizer (see Section 2.7).

A Transmit Register Empty condition is indicated by "1" in bit 4 of the R6551 status register. Providing the R6551 command register has been programmed to a Transmit Interrupt Enable state (bits 3 and 2=01), a Transmit Register Empty condition sets the Interrupt Flag (bit 7) in the status register to a "1". This generates an interrupt request ($\overline{\text{IRQ}}$) to the processor via the Interrupt Request Prioritizer circuit.

A Receive Register Full condition is stored as a 1 bit in bit 3 of the R6551 status register. Providing the R6551 command register has been programmed to a Receiver Interrupt Enable state (bit 1=0), a Receive Register Full condition sets the Interrupt Flag (bit 7) in the status register to a "1" bit. This generates an interrupt request ($\overline{\text{IRQ}}$) to the processor via the Interrupt Request Prioritizer.

8.3.6 Other Considerations

The ACIA module can operate in full-duplex mode or half-duplex mode. In full-duplex mode data can be simultaneously transmitted to an external device and received from an externnal device. In this case, software must handle Transmit Register Empty interrupts that are interleaved with Receiver Register Full interrupts. In the half-duplex operation, system software receives non-interleaved, successive, Transmit Register Empty interrupts throughout the duration of a complete output message, or non-interleaved, successive Receive Register Full interrupts throughout the duration of a complete input message.

An external device may require a high CTS (Clear to Send) signal before it transmits data to the ACIA module. The high CTS signal notifies the external device that the ACIA module is ready to receive data. For data set operation, CTS is produced from an $\overline{\text{RTS}}$ (Request to Send) output from the R6551 device. The $\overline{\text{RTS}}$ signal is generated by programming bits 3 and 2 in the

R6551 command register (see Table 8-4). If the system application allows the ACIA to always appear ready to receive data, $\overline{\text{RTS}}$ can be programmed to the CTS enable state during system initialization and left that way. With $\overline{\text{RTS}}$ programmed to a constant enable state, transmit interrupts are operational and can be included in I/O processing.

For data set operation, it may not be desirable to have the ACIA port appear ready to receive data at all times. In this situation, command register bits 3 and 2 must be periodically programmed to the $\overline{\text{RTS}}$ disabled state ($\overline{\text{RTS}}$ high). Disabling $\overline{\text{RTS}}$ also disables Transmit Register Empty interrupts within the ACIA channel. Thus in full-duplex data set applications where $\overline{\text{RTS}}$ is to be periodically enabled and disabled, the software can not rely on transmit interrupts, but must routinely read the R6551 status register to detect Transmit Register Empty status.

For half-duplex data set operation, data is never transmitted or received simultaneously. In this case, the CTS signal can be program disabled to the external device when the ACIA channel is to appear "not ready" to receive data, and programmed to the enabled state when the ACIA is transmitting data and therefore requires Transmit Register Empty interrupts. System operation is not adversely affected, since transmit and receive data transfers are never interleaved for half-duplex operation.

It should be noted that for data set operation, the "ready to receive"/"not ready to receive" status of the ACIA channel need only be established when a RTS (Request To Send) flag is detected in the DCD bit (bit 5) of the R6551 status register.

8.4 PROGRAMMING EXAMPLES

Two RS-232C driver programs are described below - one for input and one for output. Although each is listed separately, they can be integrated to minimize duplication of code. These examples use a 300 baud device, and a 7-bit word length. They are not interrupt driven and do not check parity.

8.4.1 RS-232C Output Example

This program sets up the serial output device to be a data terminal such as a line printer or CRT terminal. Use this driver as follows:

a. Install JACIA jumpers to operate the AIM 65/4Ø as a data set:

Jumper	Position
JACIA-1 (A-G)	S
JACIA-2	GND
JACIA-3	GND
JACIA-4	GND

b. Connect a 3ØØ baud output device to the RS-232C connector (J2).

c. Load the RS-232C output driver program shown in Figure 8-3. Output data, e.g. text from the Editor List function, or object code from the Monitor Dump function, will be directed to serial output when S is typed in response to the OUT= prompt.

8.4.2 RS-232C Input Example

This program sets up the serial input device to be data terminal such as a keyboard or CRT Terminal. The program is used as follows:

a. Install JACIA jumpers as described in Section 8.4.1, step a.

b. Connect a 3ØØ baud input device to the RS-232C connector (J2).

c. Load the RS-232C Input Driver program shown in Figure 8-4.

Input data will be accepted from the serial input device when S is typed in response to the IN= prompt.

```
ADDR .OBJECT. SOURCE

0800          ;ACIA REGISTER DEFINITIONS
0800                 *=$FFD0
FFD0          ACIA   *=*+1
FFD1          STATUS *=*+1
FFD2          CMND   *=*+1
FFD3          CTRL   *=*+1
FFD4          ;
FFD4          ; ACIA INITIALIZATION
FFD4          CMD=$0B
; DTR=ON,IRQ=OFF,NO ECHO,NO PARITY
FFD4          CTL=$16
; 8-BITS,1 STOP BIT,300 BAUD
FFD4          ;
FFD4          ; I/O TABLE POINTER
FFD4                 *=$206
0206 00 08    IOVS   .WOR SOUT
0208          ; TABLE OF 3 JUMP VECTORS
WHICH MUST OPEN, OUTPUT, AND CLOSE
0208                 *=$800
0800 4C 09 08  SOUT  JMP OPEN
0803 4C 17 08        JMP OUTPUT
0806 4C 28 08        JMP CLOSE
0809          ;
0809          ; OPEN THE SERIAL OUTPUT
DEVICE
0809 A9 0B    OPEN   LDA #CMD
080B 8D D2 FF        STA CMND
080E A9 16           LDA #CTL
0810 8D D3 FF        STA CTRL
0813 AD D0 FF        LDA ACIA
0816 60              RTS
0817          ;
0817          ; OUTPUT DATA ROUTINE
0817 48       OUTPUT PHA
0818 20 22 08  OUT1  JSR OUTYET
081B F0 FB           BEQ OUT1
081D 68              PLA
081E 8D D0 FF        STA ACIA
0821 60              RTS
0822          ; TRANSMIT REGISTER EMPTY?
0822 AD D1 FF  OUTYET LDA STATUS
0825 29 10           AND #$10
0827 60              RTS
0828          ;
0828          ; CLOSE THE SERIAL OUTPUT
DEVICE
0828 60       CLOSE  RTS
0829                  END
 ERRORS=0000
```

Figure 8-3. RS-232C Output Printer·Driver Program

```
ADDR .OBJECT. SOURCE

0800           ;
0800           ;ACIA REGISTER DEINITIONS
0800                  *=$FFD0
FFD0           ACIA   *=*+1
FFD1           STATUS *=*+1
FFD2           CMND   *=*+1
FFD3           CTRL   *=*+1
FFD4           ;
FFD4           ; ACIA INITIALIZATION
FFD4           CMD=$0B
;DTR=ON,IRQ=OFF,NO ECHO,NO PARITY
FFD4           CTL=$36
; 7-BITS, STOP BIT,300 BAUD
FFD4           ;
FFD4           ; SERIAL I/O TABLE POINTER
FFD4                  *=$204
0204 00 09     IOVS   .WOR SINP
0206           ; TABLE OF 3 JUMPVECTORS
 WHICH MUST OPEN, INPUT, AND CLOSE
0206                  *=$900
0900 4C 09 09   SINP  JMP SIOPEN
0903 4C 17 09         JMP SINPUT
0906 4C 26 09         JMP SICLOS
0909           ;
0909           ; OPEN THE SERIAL INPUT
0909 A9 0B     SIOPEN LDA #CMD
090B 8D D2 FF         STA CMND
090E A9 36            LDA #CTL
0910 8D D3 FF         STA CTRL
0913 AD D0 FF         LDA ACIA
0916 60               RTS
0917           ;
0917           ; INPUT DATA ROUTINE
0917 20 20 09   SINPUT JSR INPYET
091A F0 FB            BEQ SINPUT
091C AD D0 FF         LDA ACIA
091F 60               RTS
0920           ; IS THERE A CHARACTER YET
0920 AD D1 FF   INPYET LDA STATUS
0923 29 08            AND #$08
0925 60               RTS
0926           ;
0926           ; CLOSE THE SERIAL INPUT
0926 60        SICLOS RTS
0927                  .END

 ERRORS=0000
```

Figure 8-4. RS-232C Input Driver Program

SECTION 9

USING THE AUDIO RECORDER INTERFACE

The AIM 65/4Ø SBC module may be connected to one or two audio cassette recorders for permanent mass storage of programs (source and object) or data (computational or text).

Permanent storage is desirable because:

- The AIM 65/4Ø RAM is volatile memory--its memory contents are altered to an unpredictable state when RAM power is removed.
- A permanent storage medium is needed to save source programs, object programs and data currently in RAM, to make room for other programs and data. Information recorded on cassette may be read by AIM 65/4Ø as many times as desired.

The audio tape interface allows low cost audio cassette recorders to be used. It is recommended, however, that the highest quality recorders and cassette tapes be used to obtain maximum performance and reliability.

This section describes how to interface to the recorders and how to use the I/O ROM, Monitor and Editor functions with one or two tape recorders.

9.1 INTERFACING WITH AUDIO CASSETTE RECORDERS

Basically, data may output to an audio recorder from a Monitor, Editor or user function. This requires you to first load an 8Ø-byte audio output buffer (see Figure 6-2) with the data to be recorded. This process is transparent at the Monitor/Editor level since you just tell it to dump between addresses, list a number of lines or load memory. When the buffer is full, the block of data is output on the audio output line, which is connected to the recorder MIC input. The audio tape block format is described in Appendix I. This process is repeated until all the data is sent.

A similar process occurs upon loading. The data is read from the recorder (speaker output) into the SBC module (audio input line) into an 8Ø-byte audio input buffer. When the buffer is full, the tape is halted until the application function processes the data, i.e. removes it from buffer or performs any other operation on it. This process is repeated until all the data is read.

The SBC module circuitry that interfaces with tape recorder audio input, output and remote control lines is described in Section 11.7.3. The I/O ROM subroutines to open, close, process and report status as described in Section 6.5.

9.1.1 Recorder Requirements

a. How many recorders are needed?

(1) One recorder is required to read/record object code to/from the Monitor or to read/record text to/from the Editor. This recorder can also be used with the optional assembler when assembling source code from cassette and outputting object code to memory or when assembling from memory and outputting object code to audio cassette.

(2) Two recorders are required when using the optional assembler is used to input source code from one recorder and to output generated object code to the other recorder.

b. Recorder Features

The audio cassette recorders used should be equipped with the following features:

(1) An earphone (EAR) jack. The AIM 65/40 uses it as an audio input line, to read cassette data into RAM.

(2) A microphone (MIC) jack. The AIM 65/40 uses it as an audio output line, to record data from memory onto cassette.

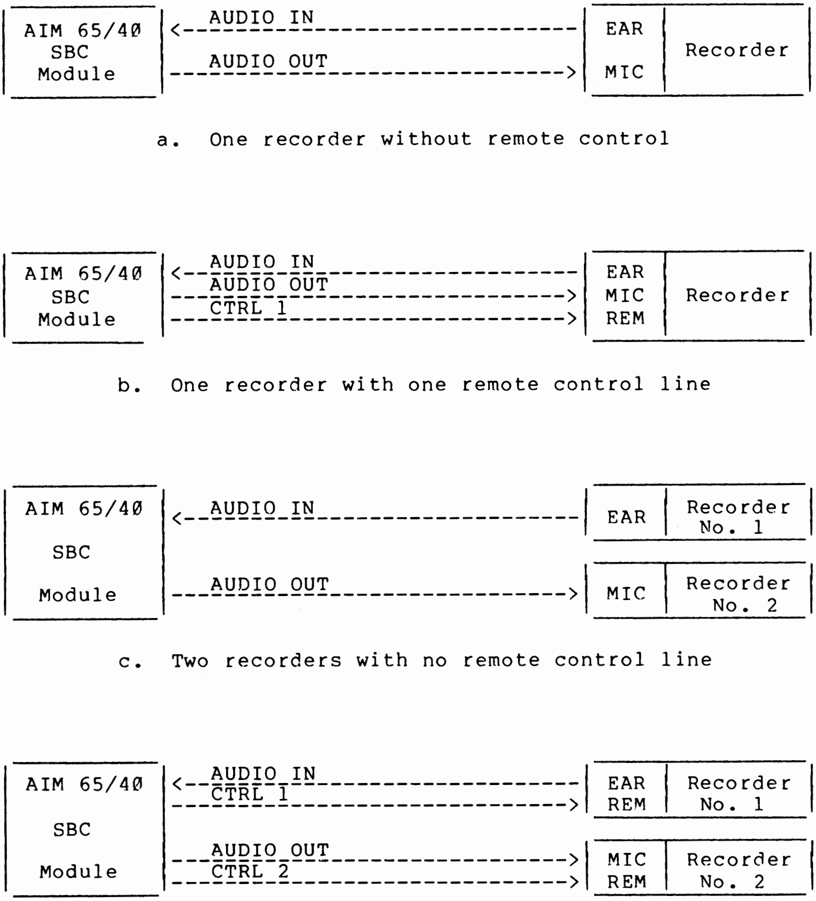
(3) A remote (REM) jack. The AIM 65/40 uses it as a remote control line, to turn the recorder on or off automatically or by user command. While line is not required for many operations, it is recommended that the recorder have this capability.

(4) A tape counter, while not required for operation, provides a convenient reference point for locating programs written on cassette.

9.1.2 Recorder Interface Configurations

The AIM 65/40 SBC module audio interface circuitry connects directly to one or two audio tape recorders, with or without remote control. The SBC module audio input connects to the recorder earphone output to read from tape, whereas the SBC module audio output connects to the recorder microphone input to record on tape. Two other lines allow remote control of two separate records (via the recorder REM jack) using reed relays to be compatible with most recorders. Figure 9-1 illustrates four typical configurations. Table 9-1 defines the interface signals.

A single recorder interface requires both audio input and output lines to be connected if both reading and writing are required the Audio In will plug into recorder's MIC jack. Use of one of the remote control lines is optional except to read text into a partially filled Text Buffer and as required by optional languages, e.g. assembler and Basic.



a. One recorder without remote control

b. One recorder with one remote control line

c. Two recorders with no remote control line

d. Two recorders with two remote control lines

Figure 9-1. Typical Audio Cassette Recorder Hookups

Table 9-1. SBC Connector J3 (Audio) Signal Definitions

Mnemonic	Signal Name and Description	Type of Drive
GND	Signal Ground Connects Audio In and Audio Out shields to signal ground.	
Audio In	SBC Module Audio Tape Input Connects tape recorder earphone (EAR) output to the SBC module audio input line.	
Audio Out	SBC Module Audio Tape Output Connects AIM 65/40 SBC module audio output recorder microphone (MIC) input to the tape line.	
CTRL1	Remote Control 1 Inputs from the recorder REM output to relay 1 input side.	Closure Reed Relay
CTRL1 RTN	Remote Control 1 Return Connects remote control 1 from relay 1 output side to the recorder REM shield.	
CTRL2	Remote Control 2 Input from the recorder REM output to relay 2 input side.	Closure Reed Relay
CTRL2 RTN	Remote Control 2 Return Connects remote control 1 from relay 2 output side to the recorder REM shield.	

When used with one recorder, the remote control allows the recorder to be setup for reading or recording prior to Monitor, Editor or user function initiation. That function then starts and stops the recorder automatically at the proper time under program control.

The remote control lets the SBC module read a block of data (8Ø-bytes) from the tape, stop the tape, process the data, read the next block, and so on. The format of the audio tape is described in Appendix I.

Some operations, such as symbolic assembly with one recorder, require either remote control or a very large gap between blocks (to allow processing to be completed while the tape continues to run). Writing files with long gaps is very time consuming and uses a lot more tape.

When assembling with two recorders, remote control is a necessity, since the assembler reads source from one recorder, processes it, loads the generated output object code in an output audio tape buffer. It may read several blocks of source code before the output buffer is full. If remote control is not used, the subsequent object code load process will be extremely slow.

If two recorders are used, the audio input line should be connected to the EAR jack of one recorder with the audio output line connected to the MIC jack of the other recorder. Use of the remote control lines is optional unless the data is being assembled from cassette or subsequent loading of object code.

Recorders without remote control are connected to AIM 65/4Ø SBC module with just two lines, Audio In and Audio Out.

9.1.3 Audio Recorder Interface Cable Construction

You must provide only the recorder cables needed for your specific application. This depends on factors such as: how many recorders you use, do you need to read and/or record, and do you need remote control. Figure 9-2 shows the typical construction of a four cable assembly.

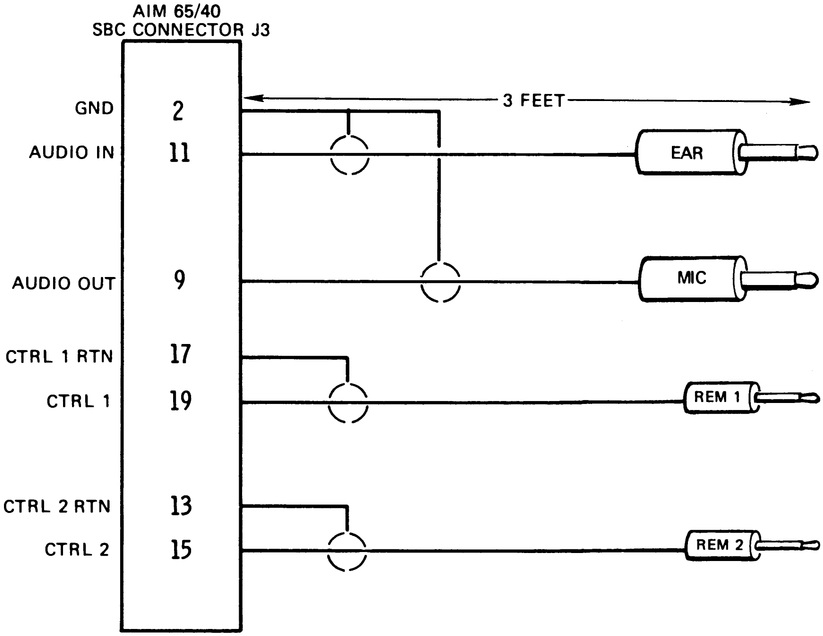


Figure 9-2. Typical Audio Cassette Recorder Interface Cables

a. Parts

You will need the following items to build the total assembly:

(1) A 2Ø-pin edge receptacle (Viking 3VH1Ø/1JN5 or equivalent) with Ø.1ØØ-inch center spacing. The cables will be soldered to their connector then the connector installed on SBC module connector J3.

(2) A six-foot audio data cable assembly that has a miniature (1/8-inch) phono plug at each end. Such a cable is Radio Shack catalog no. 42-242Ø. If you plan to install remote control capability in your system, buy two sets of Radio Shack's "Multi-Purpose Cable Kit" (catalog no. 278-Ø14) instead.

(3) Two earphone/microphone plugs (Switchcraft 75Ø or equivalent) are required if you build your own audio cables.

(4) Two remote plugs (Switchcraft 85Ø or equivalent) are required if you build your own remote control cables.

(5) A short length (two inches) of 26- or 28-gauge hookup wire.

b. <u>Audio Data Cable Assembly</u>

(1) Cut one audio data cable assembly in half and perform these steps at the cut end of each cable (if plugs are already installed at the other end), or at both ends of each cable (if plugs are not installed).

(a) Strip the outer covering back to expose about one inch of the outer braided shield.

(b) Trim off 3/4-inch of the braided shield.

(c) Strip off 1/4-inch of the inner covering.

(d) Solder one end of the hookup wire to the exposed braided shield.

(2) Solder the other end of the hookup wire to Pin 2 (GND) of the edge receptacle.

(3) Solder one end of one cable's inner conductor to Pin 11 of the edge receptacle and the other end to the plug (if no plugs is installed). Mark this plug, "EAR".

7. Solder one end of the other cable's inner conductor to Pin 9 of the edge receptacle and the other end to the plug (if no pins is installed). Mark this plug, "MIC".

(1) Cut the subminiature (3/32-inch) cable assemblies from your "Multi-Purpose Cable Kit" to three-foot lengths and perform preceding Steps (1)(a) through (1)(c) on both of these cables.

(2) Solder one end of one cables' inner conductor to pin 19 (CTRL 1) of the edge receptable, and the other end to the REM 1 plug center connection (If no plug is installed). Mark this plug, "CTRL 1".

(3) Solder the cable shield at the edge connector end of the cable in step b(2) to pin 17 (CTRL 1 RTN) and the other end of the cable shield to the REM 1 plug shield connection (if no plug is installed).

(4) Solder one end of the other cables' inner conductor to pin 15 (CTRL 2) of the edge receptacle and the other end to the REM 2 plug center connection (if no plug is installed). Mark this plug, "CTRL 2".

(5) Solder the cable shield at the edge connector end of the cable in step B(4) to pin 13 (CTRL 2 RTN) and the other end of the cable shield to the REM 2 plug shield connection (if no plug is installed).

(6) Plug the 2Ø-pin edge receptacle onto SBC module connector J3.

(7) Connect the other end of the cables to the recorders as required.

9.2 AUDIO RECORDER ADJUSTMENTS

After installing the audio cassette recorders, run the following SYNC pattern WRITE and READ programs to verify correct recorder interface connection and operation. The SYNC pattern WRITE program writes a continuous stream of SYNC ($16) characters onto a cassette tape with the recorder in the RECORD mode. The SYNC pattern READ program then looks for the SYNC pattern when the tape is read back into the AIM 65/40 SBC module with the recorder in the PLAY mode.

9.2.1 <u>SYNC Test Pattern Program</u>

Enter the following program into AIM 65/40 RAM using the Instruction Mnemonic Entry function or the optional Assembler.

```
ADDR OBJECT   SOURCE

               BEEP    = $F467
               GETBIT  = $FB99
               GETBYT  = $FB87
               ST1CL   = $FFB4
               ST1CH   = $FFB5
               SACR    = $FFBB
               SIFR    = $FFBD
               SIER    = $FFBE
               KBIFR   = $FFCD
               KBIER   = $FFCE
               SETIME  = $FCD9
               SYNCWR  = $FC9B
               TAPSPD  = $0245
               TRT     = $0316
               UIRQBM  = $022B
               BYTE    = $027D
               SETIME  = $FCD9
               SYNCWR  = $FC9B

;SYNC WRITE PROGRAM
0800 78        WRITE   SEI
0801 20 D9 FC          JSR SETIME
0804 A2 00     WRITE1  LDX #0
0806 20 9B FC          JSR SYNCWR
0809 4C 04 08          JMP WRITE1

;SYNC READ PROGRAM
080C 78        READ    SEI
;CLEAR ALL POSSIBLE IRQ'S
080D A9 7F             LDA #$7F
080F 8D BE FF          STA SIER
0812 8D BD FF          STA SIFR
0815 8D CE FF          STA KBIER
0818 8D CD FF          STA KBIFR
;SET IRQ VECTOR TO BEEP
081B A9 62             LDA #<SYNCER
081D 8D 2B 02          STA UIRQBM
0820 A9 08             LDA #>SYNCER
0822 8D 2C 02          STA UIRQBM+1
```

```
;SET T1 TO ONE SHOT MODE OF OPERATION
0825 A9 00            LDA #0
0827 8D BB FF         STA SACR
;SET TI IRQ ENABLE
082A A9 C0            LDA #$C0
082C 8D BE FF         STA SIER
;SET T1 LOW ORDER BYTE
082F A9 00            LDA #$00
0831 8D B4 FF         STA ST1CL
0834 AD 45 02         LDA TAPSPD
0837 4A               LSR A
0838 6D 45 02         ADC TAPSPD
083B 8D 17 03         STA TRT+1
083E A9 40            LDA #$40
0840 2A               ROL A
0841 8D 16 03         STA TRT
0844 58               CLI
; FAILED TO "SYNC"
0845 00        RESTRT BRK
0846 EA               NOP
0847 20 99 FB  TSYNC  JSR GETBIT
084A D0 FB            BNE *-3
084C 66 00            ROR 00
084E A5 00            LDA 00
0850 C9 16            CMP #$16
0852 D0 F3            BNE TSYNC
;GET BYTE FROM TAPE
0854 20 87 FB  SYNCOK JSR GETBYT
0857 C9 16            CMP #$16
0859 D0 EA            BNE RESTRT
;RESET FAILURE TIMER
085B A9 60            LDA #$60
085D 8D B5 FF         STA  ST1CH
0860 D0 F2            BNE SYNCOK

;SYNC SUBROUTINE
0862 48        SYNCER PHA
0863 8A               TXA
0864 48               PHA
0865 98               TYA
0866 48               PHA
0867 20 67 F4         JSR BEEP
086A A9 60            LDA #$60
086C 8D B5 FF         STA ST1CH
086F 68               PLA
0870 A8               TAY
0871 68               PLA
0872 AA               TAX
0873 68               PLA
0874 40               RTI
```

9.2.2 Running the Programs

a. To Record SYNC Characters on Tape

(1) Connect both audio data lines to one recorder as shown in Figure 9-1. The remote control lines are not used.

(2) Install a blank or scratch tape into the recorder. Rewind the tape all the way, and reset the counter to zero.

(3) Start the SYNC WRITE program at address $Ø8ØØ.

```
{G} Ø8ØØ <RETURN>
```

(4) Start the tape recorder in the RECORD Mode.

(5) After several minutes, press RESET to return control to the Monitor command level. Note the end of the recorded SYNC characters on the recorder tape counter.

(6) Stop the record or and rewind the tape.

(7) Verify that the recording was accomplished, by removing the audio input line from the recorder EAR jack, turning down the volume to a comfortable level and starting the recorder in the PLAY mode. The recorded SYNC pattern will provide a steady distinct pitch. If you don't hear this pitch, the recording hook-up is probably incorrect (see Section 9.1.2) or the WRITE program was incorrectly entered. If you hear the SYNC pattern, stop the recorder, rewind the tape, and reconnect the audio input line.

b. To Adjust the Recorder

(1) Start the SYNC READ program at address $Ø8ØC.

```
{G} Ø8ØC <RETURN>
```

The beeper will sound immediately to indicate that SYNC characters are not being read. If the beeping does not occur, you probably made an error in entering the READ program. Disassemble the READ program and verify the instructions are the same as those in Section 9.2.1.

(2) Adjust the recorder tone control to mid-range and the recorder volume to its highest level.

(3) Position the start of the recorded SYNC file in front of the recorder read head, adjust the recorder volume control to maximum, and start the recorder in the PLAY mode.

(4) The beeper will sound if the recorder output at high volume is distorted (probably due to ringing) enough to cause the AIM 65/40 to detect erroneous bits (many recorders will do this). The beeping indicates bad data bits are being received.

Slowly decrease the volume until the beeping stops. Good data bits are now being received. This is the highest setting of the recorder volumne control for reading.

Continue to decrease the volume until the beeping starts again. This is the just below the lowest setting of the recorder volume control for reading. A volume control setting midway between the high and low settings will yield the best reading results. Mark the set point on the volume control.

If the AIM 65/40 is unable to read the tape, check for one of the following conditions:

- poor audio data line connection
- recorder batteries are low
- recorder is malfunctioning
- defective tape
- recorder volume control needs adjustment
- recorder tone control needs adjustment

(5) Determine the best setting for the tone control by first setting the volume near the minimum or maximum setting for reading then varying the tone control until the beeping starts. Set the tone control midway between these points. Mark the set point on the tone control.

(6) Return control to the Monitor by pressing and releasing CTRL RESET (note that a COLD RESET is required to initialize interrupt vectors).

(7) Stop the recorder and rewind the tape.

9.3 USING THE AUDIO TAPE RECORDER

9.3.1 Recording On Audio Tape

Text or object code may be recorded on audio cassette for any Monitor, Editor or optional language function allowing audio tape as the output device (OUT=T) or user defined output functions. Some of these commands and the type and format of recorded data are:

Program	Function	Data Type	Format
Monitor	D - Dump Object Code	Object	Binary or ASCII
Monitor	K - Disassemble Memory	Text	ASCII
Editor	L - List Text Lines	Text	ASCII
Assembler		Object	Binary or ASCII

The recording procedure is:

a. Install the cassette and manually position the tape (past the leader) to where the recording is to start. If a remote control line is installed, the recorder will not start unless the Monitor/Editor remote control toggle command is on.

   Hints for using audio cassette recorders:

   o After inserting a cassette into a recorder, always rewind the cassette until it stops, then reset the recorder counter.

   o When recording the first file on the tape, BE SURE THE CASSETTE HAS ADVANCED BEYOND ANY NON-RECORDABLE LEADER ON THE TAPE. If you cannot see the physical start of the magnetic portion of the tape through a transparent cassette housing, allow at least five counts on the recorder counter or about 1Ø seconds.

- o Allow the tape to run for several seconds between recorded files. Stop the recorder. Enter the tape count on a tape dictionary for future reference. Figure 9-3 shows a typical form, with an example.

CAUTION

> The I/O ROM initializes the interblock gap (Variable IRGSYN, see Section 6.1.2) to 80 ($50) which writes 160 SYNC characters between blocks. The default value of IRGSYN provides sufficient time for operating a recorder using remote control or for performing some processing between reading blocks when remote control is not used.
>
> If remote control is used, the minimum number of SYNC characters required by your recorder may be less, e.g. 20 rather than 160. You may reduce the gap size by storing a smaller number into IRGSYN before recording. Be sure, however, to provide sufficient SYNC characters to allow for recorder degradation and to operate on another (or backup) recorder with unknown remote control performance.

b. If manual recorder control is used, go to step c.

Tape ID: ____________________

Format	File Name	SRC/ OBJ	Tape Count		Address		Program Name	Notes
			Low	High	Low	High		
T	TMR1S	S	010	032	---	---	Timer 1	
A	TMR1L	O	040	047	0200	03D0	Timer 1	
B	MST3S	O	060	071	---	---	Test 3	
B	TST3L	O	080	084	0200	02F2	Test 3	

Figure 9-3. Example Audio Cassette Dictionary Form

If remote recorder control is used, press 1 or 2, depending on which control line is connected to the recorder, until OFF is displayed. Put the recorder in the RECORD mode and verify that the recorder does not start.

c. Command the desired function, e.g. dump object code using the Monitor D command or list text from the Text Buffer using the Editor L command. Respond to subprompts.

d. When OUT= is displayed, press T.

e. When UNIT= is displayed, press 1 or 2 depending on which control line is connected to the recorder. If remote control is not used, enter either number. If the wrong number is entered, press RETURN after FILE= is displayed to back up to UNIT=.

f. When FILE= is displayed, enter the file name (up to five alphanumeric or special characters unless variable TNAMSZ has been changed, see Section 6.1.2). An input error may be corrected by pressing DEL and retyping the correct character. Do not press RETURN to initiate the command until the next step is checked.

g. If remote recorder control is used, go to step h.

If manual recorder control is used, place the recorder in the RECORD mode. Be sure the recorder starts and is in the RECORD mode.

h. Press RETURN or SPACE to initiate the function and the recording process.

If remote recorder control is used, be sure the recorder starts and is in the RECORD mode.

The block count will be displayed followed by the letter "W" as the output progresses. The first block number (ØØ) is displayed about six seconds after the RETURN or SPACE is pressed.

Completion of recording is indicated by return to the command level that existed before the tape record function. The message DONE may be displayed depending on the calling function.

i. If manual recorder control is used, turn the recorder off. If remote recorder control is used, the recorder stops automatically upon function completion.

Record the tape counter value on the tape directory and advance the tape about 5 counts on the recorder counter for subsequent recording.

Example 1:

In the Monitor, dump object code in binary from $Ø8ØØ to $Ø88Ø to file ØBJB1 on a recorder connected to remote control line CTRL 1.

```
{D}
 FROM=Ø8ØØ TO=Ø88Ø OFFSET=ØØØØ MORE?N
 TYPE=A   OUT=T UNIT=1 FILE=OBJB1   Ø2 W
 DONE
```

Example 2:

In the Editor, list text to file SRC1 on a recorder connected to remote control line CTRL 1.

```
={L}/.     OUT=T UNIT=1 FILE=SRC1   Ø3 W
 *END*   Ø4 W
```

Note that the *END* is displayed when the text has been completely dumped to the audio output buffer. However, the actual writing to the recorder is completed when control returns to the Editor command level.

Example 3:

In the optional Assembler, output object code to file PRG⟩ on a recorder connected to remote control line CTRL 2.

```
{7}ASSEMBLER V1. 0
 FROM=1800  TO=1FFF
 IN=M
 OBJ TO MEM?N OFFSET=0000   OUT=T UNIT=2
 FILE=PRGX
 LIST?N   OUT=<RETURN>   03 W
PASS 1
PASS 2

 ERRORS= 0000
```

9.3.2 Reading From a Cassette Tape

Text or object code may be read from audio tape using any Monitor, Editor or optional software function allowing audio tape input device (IN=T). Some of these commands and the type and format of the recorded data are:

Program	Function	Data Type	Format
Monitor	L - Load Object Code	Object	Binary or ASCII
Monitor	F - Verify Object Code	Object	Binary or ASCII
Monitor	3 - Verify Tape	Text or	ASCII
		Object	Binary or ASCII
Editor	R - Read Text	Text	ASCII
Assembler		Text	ASCII

The reading procedure is:

a. Install the cassette and manually position the tape to about five counts or a couple of inches of tape before the start of the desired file. Remember to initialize the tape counter to the start of the cassette tape if not done previously.

b. If manual recorder control is used, go to step c.

If remote recorder is used, press 1 or 2 depending on which control line is used, until OFF is displayed. Put the recorder in the PLAY mode and verify that the recorder does not start. If the tape starts, the remote control line is probably connected incorrectly or is turned ON. If this occurs, turn recorder off. If necessary, rewind the tape to position the start of the file ahead of the read head several counts. Either check the remote recorder control hook-up and repeat this step or continue under manual recorder control.

c. Command the desired function, e.g. load object code using the Monitor L command or read text into the Text Buffer using the Editor R command. Respond to subprompts.

d. When IN= is displayed, press T.

e. When UNIT= is displayed, press 1 or 2 depending on which remote control line is connected to the recorder. If remote recorder control is not used, enter either number. If the wrong number is entered, press RETURN after FILE= is displayed to back up to UNIT=.

f. When FILE= is displayed, enter the file name under which the file was recorded. An input error may be corrected by pressing DEL and retyping the correct character.

Press RETURN or SPACE to initiate the function and the reading process.

If manual recorder control is used, place the recorder in the PLAY mode. The tape will then start reading.

If remote control is used, the remote control line will be turned on automatically to enable tape reading.

i. The AIM 65/40 will search for the commanded file name. Any block numbers read before the first file name is encountered on the tape will be displayed after the commanded file name (e.g. if the tape was initially positioned in the middle of a prior file)..

j. When a file name is read from the tape, it is displayed after clearing the display. The block number is displayed following the name as read from the tape. This block number will be updated as reading progresses.

If the recorded file name matches the commanded file name, the letter "R" (Read) is displayed following the block number and the recorded data is read into the audio tape input buffer. The calling function uses the data as required, e.g. loads it into RAM (Monitor L command) or compares it to RAM (Monitor F command). Data read checks are performed during the reading (see Section 9.3.3).

If the recorded file name does not match the commanded file name, the letter "S" (Search) is displayed following the block number and the data in the recorded file is skipped. The AIM 65/40 continues to search until the commanded file name is located. Data read checks are not performed during the search.

k. Completion of the read is indicated by return to the command level that existed before the tape read function. One exception is the Monitor verify tape function (3 command), which will continue reading all files on the tape until RESET or ATTN is pressed.

Example 1:

In the Monitor, load object code from file OBJB1 on a recorder connected to remote control line CTRL 1.

```
{L} OFFSET=0000 IN=T UNIT=1 FILE=OBJB1
OBJB1   02 R
 DONE
```

Example 2:

In the Editor, read text from file TXT1 from a recorder connected to remote control line CTRL 2.

```
{E}
 EDIT FROM=1ØØØ TO=3FFF IN=T UNIT=1 FILE
=SRC1
SRCØ  Ø4 S
SRC1  Ø4 R

  *END*
```

Example 3:

In the optional Assembler, input source code from file SRC1 on a recorder connected to remote control line CTRL 1.

```
{7}ASSEMBLER V1.Ø
 FROM=1ØØØ TO=1FFF
 IN=T UNIT=1 FILE=SRC1
SRC1  ØØ R
 OBJ TO MEM?N OFFSET=ØØØØ   OUT=X
 LIST?Y   OUT=
PASS 1
PASS 2

ERRORS=ØØØØ
```

9.3.3 Read Error Detections and Recovery

a. Error Detection

If a read error is detected during the loading (i.e. not searching) of a file, an error message will be displayed, the reading stopped and control returned to the calling function. One of three errors may be reported:

(1) SYNC ERROR - A loss of bit sync occurred during reading of any portion of the block (see Appendix I).

(2) CHECKSUM ERROR - The checksum read from the tape did not compare with the checksum computed from data read from the tape.

(3) BLOCK ERROR - The block number read from the tape did not match the expected block number.

NOTE

If the Monitor load function detects a mismatch between the recorded data checksum and the checksum computed from data read from the tape, a LOAD ERROR is displayed (see Section 4.8.2).

b. Recovery

The occurrence of one or more of these errors may be caused by a poorly performing recorder, improperly adjusted volume and tone controls, a bad recording, or an incorrect connection. If the cause of the error cannot be corrected, a change to Monitor variable WSPDV will enable loading even though errors are detected. This will allow any good data on the file to be loaded. The data not read can then be loaded (i.e. recovered) manually.

This type of loading should be done cautiously since a bad address may cause overwriting of program or data in RAM.

To enable reading regardless of errors:

(1) Change bit 5 of Monitor variable WSPDV ($Ø249) to "1". Do not change the value of any other bits.

(2) Press RESET (not CTRL RESET, since that will reset bit 5 back to "Ø").

(3) Perform the load function. Errors will still be displayed, however, the loading will continue.

SECTION 10

USING THE TELETYPE INTERFACE

The SBC module TTY interface provides an industry standard 20 MA current loop data transmission capability between the AIM 65/40 system and external equipment. This interface is controlled by the programmable ACIA device and a user provided driver program.

Output and input TTY drivers are described in this section. To use the TTY interface, first connect the TTY interface signals to the SBC connector J3 (see Table L-3 and Figure L-3). Table 10-1 describes the interface signals. Be sure that jumpers JACIA-2, -3 AND -4 are installed in the GND position.

10.1 TTY OUTPUT DRIVER

Figure 10-1 lists a TTY output driver. Data may be output to the TTY by typing S in response to the OUT= prompt.

10.2 TTY INPUT DRIVER

Figure 10-2 lists a TTY input driver. Data may be input from the TTY by typing S in response to the IN= prompt.

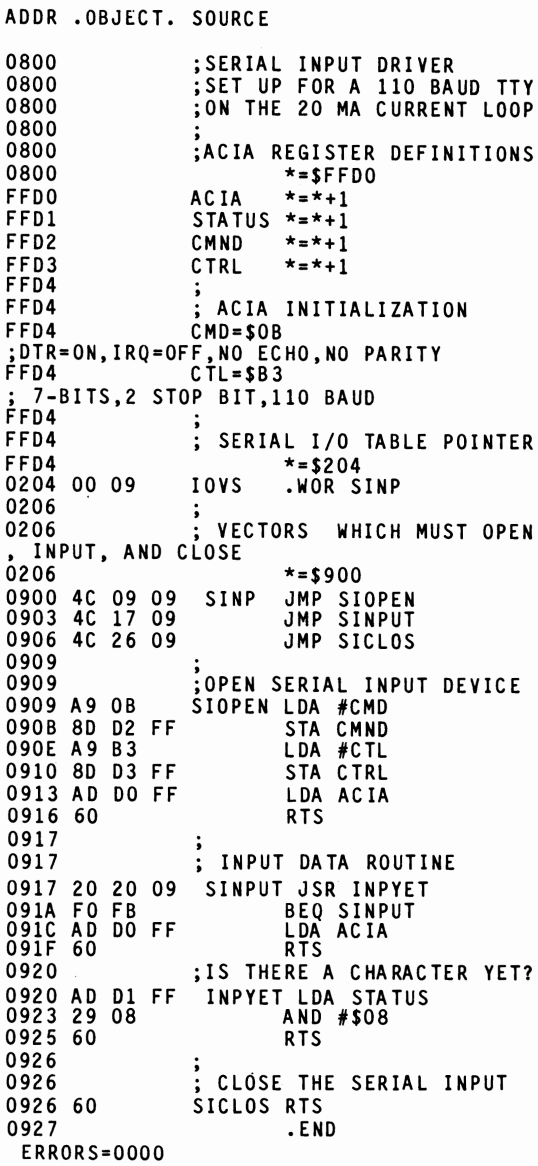
Table 1Ø-1. SBC Connector J3 (TTY) Signal Definitions

Signal Mnemonic	Signal Name and Description
RTS	Request To Send This line transfers a current-mode RTS control signal from the R6551 to an external teletype device. RTS can be programmed to an active 2Ø milliamp condition or a zero current condition under current condition under program control. RTS is used to control the external TTY device.
TD	Transmit Data This line transfers serial, NRZ, current-mode data from the AIM 65/4Ø SBC module to an external teletype unit. During quiscent (no data) states, a mark (2Ø milliamp) condition exists on the TD line. The start bit (LSB) of each word transmitted causes the TD line to go to a space (zero current) followed by a mark or space for each bit of the transmitted word and ending with mark(s) for the stop bit(s). The data rate on the TD line is determined by the programmed baud rate.
RD	Receive Data This line transfers serial, NRZ, current-mode data from the external teletype device into an the AIM 65/40 SBC module. During quiescent states, a marking condition exists on the RD line. When the external device opens the current loop a space condition exists on the RD line. The start bit of each word received from the external device causes the RD line to go to a space, followed by a mark or space for each bit of the incoming word and ending with mark(s) for the stop bit(s). The data rate expected on the RD line is determined by the programmed baud rate.

```
ADDR .OBJECT. SOURCE

0800            ;SERIAL OUTPUT DRIVER
0800            ;SET UP FOR A 110 BAUD TTY
0800            ;ON THE 20 MA CURRENT LOOP
0800            ;
0800            ;ACIA REGISTER DEFINITIONS
0800                   *=$FFD0
FFD0            ACIA   *=*+1
FFD1            STATUS *=*+1
FFD2            CMND   *=*+1
FFD3            CTRL   *=*+1
FFD4            ;
FFD4            ; ACIA INITIALIZATION
FFD4            CMD=$0B
; DTR=ON,IRQ=OFF,NO ECHO,NO PARITY
FFD4            CTL=$B3
; 7-BITS,2 STOP BIT,110 BAUD
FFD4            ;
FFD4            ; SERIAL I/O TABLE POINTER
FFD4                   *=$206
0206 00 08      IOVS   .WOR SOUT
0208            ;
0208            ; VECTORS WHICH MUST OPEN,
 OUTPUT, AND CLOSE
0208                   *=$800
0800 4C 09 08    SOUT  JMP SOOPEN
0803 4C 17 08          JMP SOUTPT
0806 4C 28 08          JMP SOCLOS
0809            ;
0809            ;
0809            ; OPEN THE SERIAL OUTPUT
DEVICE
0809 A9 0B      SOOPEN LDA #CMD
080B 8D D2 FF          STA CMND
080E A9 B3             LDA #CTL
0810 8D D3 FF          STA CTRL
0813 AD D0 FF          LDA ACIA
0816 60                RTS
0817            ;
0817            ; OUTPUT DATA ROUTINE
0817 48         SOUTPT PHA
0818 20 22 08    OUT1  JSR OUTYET
081B F0 FB             BEQ OUT1
081D 68                PLA
081E 8D D0 FF          STA ACIA
0821 60                RTS
0822            ; TRANSMIT REGISTER EMPTY?
0822 AD D1 FF    OUTYET LDA STATUS
0825 29 10             AND #$10
0827 60                RTS
0828            ;
0828            ; CLOSE THE SERIAL OUTPUT
DEVICE
0828 60         SOCLOS RTS
0829                   .END
 ERRORS=0000
```

Figure 1Ø-1. TTY Output Driver



```
ADDR .OBJECT. SOURCE

0800            ;SERIAL INPUT DRIVER
0800            ;SET UP FOR A 110 BAUD TTY
0800            ;ON THE 20 MA CURRENT LOOP
0800            ;
0800            ;ACIA REGISTER DEFINITIONS
0800                   *=$FFD0
FFD0            ACIA   *=*+1
FFD1            STATUS *=*+1
FFD2            CMND   *=*+1
FFD3            CTRL   *=*+1
FFD4            ;
FFD4            ; ACIA INITIALIZATION
FFD4            CMD=$0B
;DTR=ON,IRQ=OFF,NO ECHO,NO PARITY
FFD4            CTL=$B3
; 7-BITS,2 STOP BIT,110 BAUD
FFD4            ;
FFD4            ; SERIAL I/O TABLE POINTER
FFD4                   *=$204
0204 00 09      IOVS   .WOR SINP
0206            ;
0206            ; VECTORS  WHICH MUST OPEN
, INPUT, AND CLOSE
0206                   *=$900
0900 4C 09 09    SINP  JMP SIOPEN
0903 4C 17 09          JMP SINPUT
0906 4C 26 09          JMP SICLOS
0909            ;
0909            ;OPEN SERIAL INPUT DEVICE
0909 A9 0B      SIOPEN LDA #CMD
090B 8D D2 FF          STA CMND
090E A9 B3             LDA #CTL
0910 8D D3 FF          STA CTRL
0913 AD D0 FF          LDA ACIA
0916 60                RTS
0917            ;
0917            ; INPUT DATA ROUTINE
0917 20 20 09    SINPUT JSR INPYET
091A F0 FB             BEQ SINPUT
091C AD D0 FF          LDA ACIA
091F 60                RTS
0920            ;IS THERE A CHARACTER YET?
0920 AD D1 FF    INPYET LDA STATUS
0923 29 08             AND #$08
0925 60                RTS
0926            ;
0926            ; CLOSE THE SERIAL INPUT
0926 60         SICLOS RTS
0927                   .END
 ERRORS=0000
```

Figure 1Ø-2. TTY Input Driver

SECTION 11

AIM 65/40 SBC MODULE DESCRIPTION

This section describes the SBC module hardware operation. The SBC module is divided into the following functional areas for descriptive purposes:

- Power Conversion and Distribution
- Central Processing and Control
  - R65Ø2 CPU
  - System Bus
  - System Clock
  - Reset Conditioning
- On-Board Device Decoding
- RAM
- PROM/ROM
- Interrupt Request Prioritizer
- System VIA
  - Program Step Control
  - Control Signals
  - Audio Recorder Interface
  - Printer and Display Interface
- Keyboard VIA
  - Keyboard Interface
  - Audio Speaker
- User VIA
  - Parallel I/O Interface
- User ACIA
  - RS-232C Interface
  - 2Ø MA TTY Interface
- Expansion Bus Interface

Figure 11-1 illustrates the AIM 65/4Ø SBC module block diagram. Figure 11-2 shows the AIM 65/4Ø SBC module component layout. Tables 11-1 and 11-2 list the physical characteristics and the power requirements.

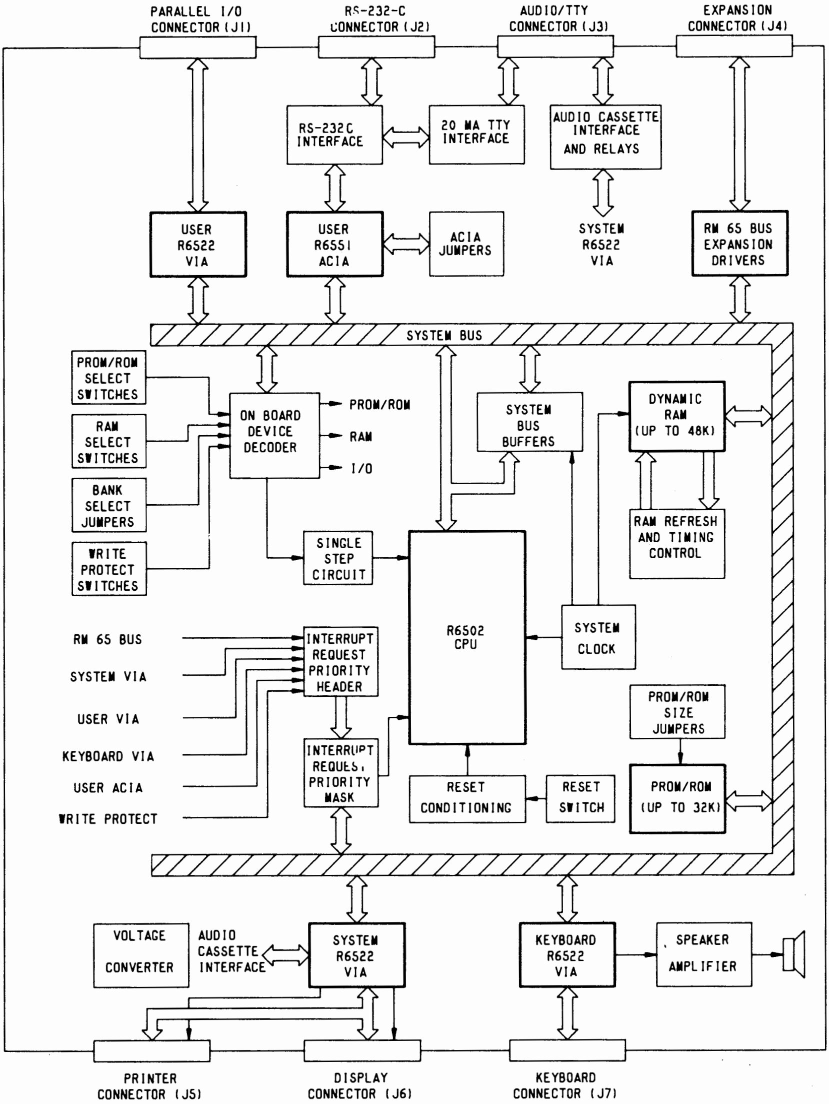


Figure 11-1. AIM 65/40 SBC Module Block Diagram

Table 11-1. SBC Module Physical Characteristics

Parameter	Value
Physical	
Width	11.85 in. (3Ø1 mm)
Length	12.5 in. (318 mm)
Height	Ø.75 in. (19 mm)
Weight	1.Ø lb.
Environment	
Operating Temperature	Ø°C to 7Ø°C
Storage Temperature	-25°C to +85°C
Relative Humidity	Ø% to 85% (without condensation)
Power Connector	4 Post Terminal Block
Interface Connector	
J1 (Parallel I/O)	4Ø-pin edge connector (Ø.1ØØ in. centers). Pre-drilled holes for installation of 4Ø-pin 3M #3432-1ØØ2, or equivalent, mass terminated connector.
J2 (RS-232C)	26-pin edge connector (Ø.1ØØ in. centers). Pre-drilled holes for installation of 25-pin AMP #2Ø6584-1, or equivalent, mass terminated connector.
J3 (Audio/TTY)	2Ø-pin edge connector (Ø.1ØØ in. centers). Mates to Viking 3VH1Ø/1JN5, or equivalent. Pre-drilled holes for installation of 2Ø-pin 3M #3492-1ØØ2, or equivalent, mass terminated connector.
J4 (Expansion)	72-pin edge connector (.1ØØ in. centers). Pre-drilled holes are provided to allow installation on expansion connector. Installation of a 64-pin DIN 41612 Euro-connector (receptacle) or 72-pin TI #H42-51-11-36, or equivalent, receptacle allows one RM 65 module to be directly installed. Installation of a 65-pin DIN XXXXX Euro-connector (jack) allows connection to a 64-conductor mass terminated cable from an RM 65 to AIM 65/4Ø Buffer Module. This allows extension to a 4-, 8- or 16-slot RM 65 motherboard.

Table 11-1. SBC Module Physical Characteristics (Continued)

Parameter	Value
J7 (Keyboard)	40-pin 3M #4595-1002, or equivalent. Mates with 3M #3417-6040, or equivalent, ribbon cable connector.
J5 (Printer) and J6 (Keyboard)	40-pin 3M #4595-2002, or equivalent. Mates with 3M #3417-6040, or equivalent, ribbon cable connector.

Table 11-2. SBC Module Power Requirements

Voltage	Typ.	Max.	Unit
+ 5V ± 5% Regulated	2.5	3.5	A

NOTE

Power Requirements are specified for 8 PROM/ROM devices (32K bytes) 0.6A (typical) and 1.2A (maximum), and for 16 RAM devices (32K bytes) with total 0.9A (typical) and 1.7A (maximum) total, installed.

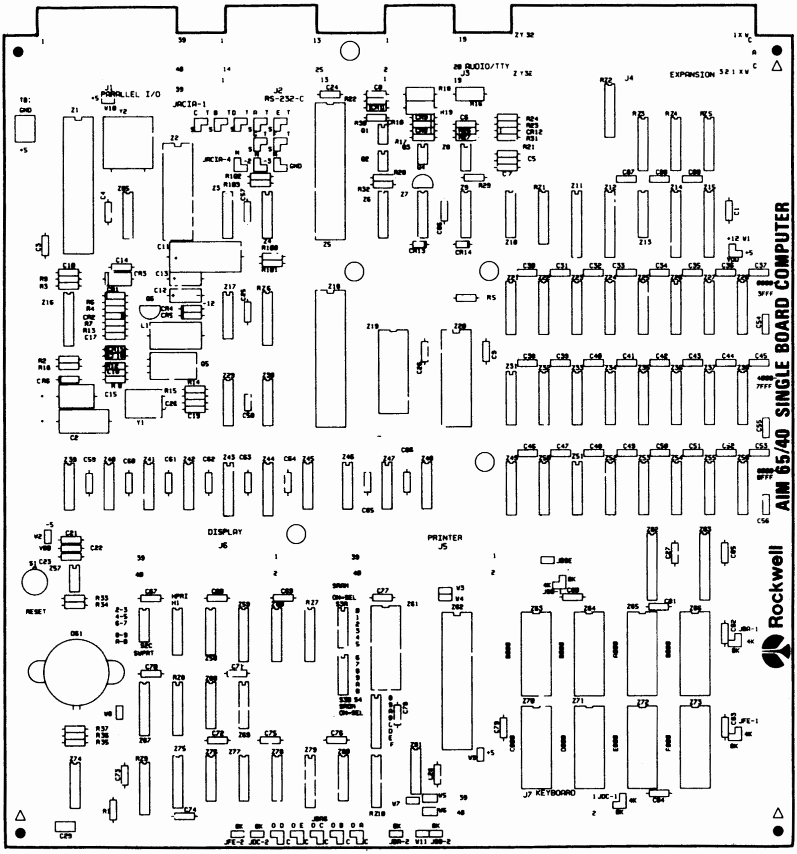


Figure 11-2. AIM 65/4Ø SBC Module Component Layout

11.1 POWER CONVERSION AND DISTRIBUTION

The AIM 65/4Ø SBC module requires a single +5V DC power source for operation, which is connected to terminal strip TB1. This +5V is the logic supply, used by all the TTL and MOS devices. Bypassing capacitors, typically Ø.1 uF, minimize the noise on the power supply lines.

While most devices require only the +5V source, the multi-voltage dynamic RAMS, the RS-232C buffers, and the TTY interface require additional voltages. For these voltages, there is an on-board switching power converter. This Regulating Pulse Width Modulator (Z16), and the associated regulator and filter circuitry generates +12V and -12V, which are routed directly to the RS-232C and TTY Interface Circuitry. The -5V is derived from the -12V, and this, as well as the +12V, are routed to the multi-voltage dynamic RAM devices through wire jumpers W1 (+12V) and W2 (-5V). For single voltage dynamic RAMs, jumper W1 is disconnected from +12V and re-connected to +5V, while W2 is removed (see Section 2.6).

The +5V is also directly routed to the Printer, Display, and Expansion connectors, and through wire jumper W1Ø to the Application connector. These power pins allow interfacing equipment, such as an RM 65 module on the Expansion connector, to connect directly without additional power cables. The maximum current through any connector pin must not exceed 2ØØ mA.

11.2 CENTRAL PROCESSING AND CONTROL

11.2.1 R65Ø2 Microprocessor

The R65Ø2 8-bit Microprocessor (Z18) is the central processing unit (CPU), performing the program execution of the AIM 65/4Ø SBC module. The operating characteristics of the R65Ø2 CPU are discribed in Sections 1 and 2 of the R65ØØ Hardware Manual. The details of its 56 instructions and 13 addressing modes are explained in the R65ØØ Programming Manual. The CPU operates at 1MHz as clocked by FØ from the clock circuit.

11.2.2 System Bus

The R65Ø2 CPU communicates with other on-board AIM 65/4Ø SBC devices over three separate buses --- the Address Bus, the Data Bus and the Control Bus.

The Address Bus consists of 16 address lines (AØ - A15) which allow the CPU to directly access 65,536 (65K) bytes of memory or I/O. Address lines AØ - All are driven directly by the CPU while A12 - A15 are driven onto the Address Bus through non-inverting buffers (Z3Ø). The Address Bus is routed to on-board RAM, PROM/ROM, I/O and decoding logic, and also to the Expansion Bus interface.

The Data Bus (DØ-D7) carries 8-bit data bidirectionally between the R65Ø2 CPU and the on-board RAM, PROM/ROM, and I/O, as well as the Expansion Bus interface.

The Control Bus includes all the basic clock and control signals to and from the R65Ø2 CPU. Reset ($\overline{\text{RES}}$), Interrupt Request ($\overline{\text{IRQ}}$), Non-Maskable Interrupt ($\overline{\text{NMI}}$), Ready (RDY) and Set Overflow (S.O.) are input control signals to the CPU. Read/Write (R/$\overline{\text{W}}$, R/$\overline{\text{W}}$) Phase 1 (Ø1) and Phase 2 (Ø2, $\overline{\text{Ø2}}$) clocks, and SYNC are output signals from the CPU to control data transfers and timing.

The Control Bus consists of many signals which are not basic CPU signals, but are synchronized to or derived from the CPU, and are used on-board or are brought out to the Expansion Bus interface. Among these are the various timing signals required by the dynamic RAMs (FØ, $\overline{\text{FØ}}$, 2FØ, $\overline{\text{2FØ}}$, 4FØ, T1, T5, and $\overline{\text{TRASOFF}}$). Control signals generated by the System VIA include the Bank Selection (BSE, $\overline{\text{BSE}}$), the Step Control ($\overline{\text{RUN}}$/STEP), and the DMA Terminate ($\overline{\text{BDMT}}$) signals. Selection signals for on board device decoding include the I/O selection ($\overline{\text{IOCS}}$, DISROM), ROM selection ($\overline{\text{CSRO8}}$ - $\overline{\text{CSROF}}$) and RAM selection ($\overline{\text{RAMCS}}$, RAMCS). There are also the various interrupt sources from the on-board peripherals and from the Expansion Bus interface ($\overline{\text{BIRQ}}$).

11.2.3 System Clock

The system clock is a crystal controlled oscillator, based around a 16 MHz crystal (Y1) and two TTL inverters (Z29). This 16 MHz signal is buffered to TTL levels by another inverter, and becomes the clock input for a four-stage binary counter (Z43). The counter, with its associated output logic (Z4Ø, Z41, Z42), generates all the timing signals needed by the CPU and the dynamic RAM circuitry. The 1 MHz FØ is the Phase Ø reference for the CPU. The dynamic RAM circuitry requires many timing signals: the 1 MHz FØ and inverse $\overline{\text{FØ}}$; the 2 MHz 2FØ and inverse $\overline{\text{2FØ}}$; the 4 MHz 4FØ; and the composite signals T1 T5, and $\overline{\text{TRASOFF}}$.

11.2.4 Reset and Power-On Conditioning

The Reset Conditioning circuit includes an NE555 Timer (Z57), and associated discrete components, which are configured in the one-shot mode with a 1Ø mS time period. The on-board RESET switch (S1) is connected to the trigger input of the timer, and is in parallel with $\overline{\text{RESET SW}}$ on the Keyboard connector (J7) through wire jumper W6.

A Reset is initiated automatically at SBC power turn-on and whenever a low level is applied to the trigger input of the timer - either by depressing the RESET switch or by grounding $\overline{\text{RESET SW}}$. The output of the Timer through an open collector inverter (Z74) becomes the $\overline{\text{RES}}$ for the CPU and on-board devices, and BRES/ for the Expansion connector (J4).

11.3 ON-BOARD DEVICE DECODING

The On-Board Device Decoding circuitry generates the device select signals for all on-board RAM, PROM/ROM, and I/O. Memory select switches allow the on-board PROM/ROM and RAM to be selected/deselected in 4K byte blocks.

The 74159 4-line to 16-line decoder (Z61) decodes address lines A12 - A15 into 16 mutually exclusive outputs, with each output indicating a 4K-byte block within the 65K byte memory map. The PROM/ROM Bank Select jumpers (JBAS-A to JBAS-E) control the gating of the decoder outputs A through E with the bank select gates (Z78, Z8Ø). These five blocks can be independently assigned common to both banks or ($\overline{\text{IOCS}}$) dedicated to Bank Ø (that is, BSE is low). The decoder output F is always common to both banks, but an I/O select gate (Z8Ø) disables the F block when on-board I/O is addressed. This F block select, bank select conditioned outputs A through E, and decoder outputs Ø though 9 are used for on-board memory selection.

The on-board ROM is selected/deselected in 4K byte blocks by the ROM select switches (SROM:S4-8 to SA-F) within the address range of $8ØØØ to $FFFF. These block selects are gated with Ø2 and R/$\overline{\text{W}}$ (Z4Ø, Z76, Z77) to create chip selects ($\overline{\text{CSRO8}}$ to $\overline{\text{CSROF}}$) only for selected ROM during the valid portion of a read cycle. When one of these select lines is low, the corresponding ROM device is active.

The on-board RAM is selected/deselected in 4K byte blocks by the RAM select switches (SRAM:S3-Ø to S3-B). Jumper JBSE allows all RAM (except for block Ø which is always common to both banks) to be selected common to both banks or dedicated to Bank Ø. The RAM selection logic (Z17, Z4Ø, Z6Ø) forces the RAM select (RAMCS) high when any selected RAM block is addressed in the proper bank. The $\overline{\text{RAMCS}}$ and the inverted RAMCS signals are used by the RAM circuitry for device selection.

The 74LS138 3-line to 8-line decoder (Z44) decodes address lines A5 to A7 into eight outputs, each indicating a 32-bit block within the 65K memory map. This decoder is only enabled when A8 is high, A9 to All are high (Z45) and $\overline{\text{IOCS}}$ is low, which corresponds to the upper page (256 bytes) of the upper 4K byte block. Three of these decoded outputs are used for I/O device selection (Y4-Y6), creating 96 bytes for on-board I/O from $FF8Ø to $FFDF. Because this I/O is within the address range of the F block ROM, a signal is generated (Z45) when the I/O is active to disable the ROM (DISROM).

The Priority Select logic (Z46, Z69) uses the decoder output Y4 to generate a write strobe (PRSEL) for the Interrupt Request Priority Latch ($FF80-$FF9F). The decoder output Y5 and A4 select between the User VIA ($FFA0 - $FFAF) and the System VIA ($FFB0 - $FFBF), while Y5 and A4 select between the Keyboard VIA ($FFC0 - $FFCF) and the ACIA ($FFD0 - $FFDF).

The Bus Driver Select logic (Z46, Z75) senses when any selected on-board ROM, RAM or I/O is addressed or when a DMA is occurring ($\overline{\text{DMA}}$), in order to inhibit the Data bus on the Expansion connector (BUS DRIVER SEL).

11.4 RAM

For on-board read and write storage, the AIM 65/40 SBC module uses dynamic Random Access Memory (RAM) devices. These RAMs are 16,384 x 1 bit (16Kx1) devices that are installed into 16-pin sockets, with 24 RAM sockets on the SBC. The AIM 65/40 SBC requires a minimum of 16K bytes of RAM memory, expandable in 16K byte blocks to the maximum of 48K.

The AIM 65/40 SBC can operate with either tri-voltage or single voltage dynamic RAM devices. The tri-voltage RAMs require +5V, +12V and -5V while the single voltage RAMs require only +5V for operation. Your AIM 65/40 may be configured to operate with either type. Section 2.6 describes where to install the RAMs and how to position the on-board wire jumpers (W1,W2) to operate with either the tri-voltage or single voltage RAMs. Table 2-11 lists the pin assignments for the two types. The installed RAM should be either of the two listed part numbers, or equivalent.

The Memory Controller device (Z19), Address Multiplexer (Z20), and the Memory Controller logic (Z39, Z45, Z47, Z48), use the clocks derived from the System Timing to sequence the signals to the RAM devices. A chip select is generated for the Memory Controller device (Z19) while F0 is high whenever a selected 4K RAM block is addressed (RAMCS) and no write inhibit occurs from the write protection circuitry ($\overline{\text{IRQWP}}$). During the normal Read or Write cycles, the memory controller generates the column address strobe ($\overline{\text{CAS}}$), the row address strobe for the addressed RAM bank ($\overline{\text{RASØ}}$, $\overline{\text{RAS1}}$, $\overline{\text{RAS2}}$), and the select signal (ROWEN) for the address multiplexer (Z2Ø). $\overline{\text{TRASOFF}}$ is synchronized with T1 (Z47) to disable the row address strobes at the proper time. During a refresh cycle, the memory controller generates the select signal to place the Refresh Counter on the address multiplexer (REFEN) and creates row address strobes for all RAM devices.

The address multiplexer takes the fourteen LSB address lines (AØ-A13) and a seven bit refresh count and multiplexes them onto the seven RAM address lines ($\overline{\text{OØ}}$-$\overline{\text{O6}}$). The ROWEN and REFEN signals control whether the row address (AØ-A6), the column address (A7-A13) or the refresh count are passed to the RAM devices.

The refresh counter is a modulo 128 counter within the Address Multiplexer device (Z2Ø). The refresh clock is derived from $\overline{\text{TRASOFF}}$ through a divide by 12 counter (Z39), thus providing a refresh strobe every six microseconds, which increments the refresh counter. This provides a full refresh of all 128 rows of the dynamic RAM devices approximately every Ø.7 milliseconds. The refresh cycles occur within the Phase 1 time period so they are transparent to the CPU.

The data transceiver (Z82) transfers data between the dynamic RAM devices (DIØ-DI7) and the System Data Bus, depending on device selection (RAMCS) and data direction (R/$\overline{\text{W}}$) signals. Data being read from the RAMs (DOØ-DO7) is latched into the Data Latch (Z83) when $\overline{\text{CAS}}$ is low and held while $\overline{\text{CAS}}$ is high.

The Write Protect Control logic (Z3Ø, Z46, Z67) uses the Write Protect Switches (SWPRT:S2-1 to S2-5) to inhibit writing to the RAM in the 8K byte blocks $2ØØØ-$3FFF, $4ØØØ-$5FFF, $8ØØØ-$9FFF and $AØØØ-$BFFF, respectively. When a Write Protect switch is on, any write within that associated 8K bytes generates an interrupt request ($\overline{\text{IRQWP}}$), and inhibits writing into on-board RAM. The lower 8K bytes ($ØØØØ-$1FFF) cannot be write protected.

11.5 PROM/ROM

Permanent program or data may be installed on the AIM 65/40 SBC module in either Read Only Memory (ROM) or Programmable Read Only Memory (PROM) devices. There are eight PROM/ROM sockets on the AIM 65/40 SBC module in which 2K-, 4K- or 8K-byte PROM, ROM or equivalent devices which meet the pin assignment requirements listed in Table 2-10 may be installed. Section 2.5 describes where to install the PROM/ROMs and how to position the on-board size jumpers and select switches.

If 2K-byte or 4K-byte PROM/ROM devices, such as the R2332, are installed, jumpers J98-1, JBA-1, JDC-1 and JFE-1 must be positioned to connect pin 21 (S2) of sockets Z64, Z66, Z73 and Z71, respectively, to +5V. Jumpers J98-2, JBA-2, JDC-2 and JFE-2 must also be removed to isolate chip select lines CSRO8, CSROA, CSROC and CSROE from chip select lines CSRO9, CSROB, CSROD and CSROF, respectively.

If 8K-byte PROM/ROM devices are installed, such as the R2364A, jumpers J98-1, JBA-1, JDC-1, JFE-1 must be positioned to connect pin 21 of sockets Z64, Z66, Z73 and Z71, respectively, to address line A12. Jumpers J98-2, JBA-2, JDC-2 and JFE-2 must also be installed to connect chip select lines CSRO8, CSROA, CSROC and CSROE to chip select lines CSRO9, CSROB, CSROD and CSROF, respectively, making the ROM chip selects cover an 8K byte address range. Sockets Z64, Z66, Z71 and Z73 will be populated, while the adjacent sockets, i.e. Z63, Z65, Z70 and Z72 must be left empty.

A mixture of 4K-byte (or compatible 2K-byte) and 8-byte PROM/ROM devices may be installed but they must be installed in socket pairs with both jumpers set accordingly for each pair.

11.6 INTERRUPT REQUEST PRIORITY

Another powerful feature of the AIM 65/40 SBC module is the interrupt request capability. There are six primary sources for the SBC interrupt requests. These are the Write Protect $\overline{\text{IRQWP}}$, User ACIA $\overline{\text{IRQA}}$, Keyboard VIA $\overline{\text{IRQK}}$, User VIA $\overline{\text{IRQU}}$, System

VIA $\overline{\text{IRQS}}$, and the Expansion Bus $\overline{\text{BIRQ}}$. The Interrupt Request prioritizer allows interrupts to be masked below any priority level. Only interrupt requests above the masked level will be passed to the CPU. The interrupt request mask level is assigned by writing the desired mask at address $FF8Ø.

The interrupt request priority header (HPRI:H1) is a removable header that allows the interrupt sources to be assigned any masking priority. The factory configured AIM 65/4Ø SBC is assigned the masking priorities shown in Table 2-1Ø. The actual interrupt prioritization is described in Section 2.5.

When any interrupt request occurs, it must propagate through the masking logic (Z58, Z68, Z69), to reach the CPU $\overline{\text{IRQ}}$ line. The highest priority interrupt, $\overline{\text{IRQ7}}$, is connected directly to the CPU IRQ, and cannot be masked by the hardware. Each lower priority interrupt request, $\overline{\text{IRQ6}}$ to $\overline{\text{IRQ1}}$, can be masked by a bit, Q5 to QØ, of the interrupt request mask latch (Z59). Because of the propogation through the ladder, a masked bit will inhibit all lower priority interrupt requests. Thus if mask bit Q2 is set, any interrupt requests from $\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$, or $\overline{\text{IRQ1}}$ will reach the CPU, but those above this level, $\overline{\text{IRQ4}}$-$\overline{\text{IRQ7}}$, will be transferred.

11.7 SYSTEM VIA

The R6522 System VIA (Z5) controls the on-board Run/Step instruction execution mode, the Bank Select Enable signals, the DMA Terminate signal, the Audio Recorder Interface, and the Display and Printer interfaces. The System VIA is assigned the addresses $FFBØ-$FFBF, common to both banks. $\overline{\text{IRQS}}$ is an interrupt source for the Interrupt Request Prioritizer.

11.7.1 Program Step Control

A major purpose of the AIM 65/4Ø system is to develop and debug programs. To facilitate this development, the Program Step hardware (Z3Ø, Z79) is provided. When enabled by $\overline{\text{RUN}}$/STEP being high, this circuitry generates a non-maskable interrupt

($\overline{\text{NMI}}$) for every instruction executed below the step address limit, which is the \$AØØØ with wire jumper W11 removed, or \$9ØØØ with W11 installed. This allows each instruction of programs under development in the lower memory to be stepped (i.e. trapped through the $\overline{\text{NMI}}$ vector) while higher memory programs, which includes peripherals and the debug monitor, execute at normal speed. Non-Maskable Interrupt firmware in the AIM 65/4Ø Monitor supports single step execution of CPU instructions.

The Program Step logic is controlled by the $\overline{\text{RUN}}$/STEP signal, System VIA port B bit 3 (PB3). When $\overline{\text{RUN}}$/STEP is false, the SBC is in the run mode, with Program Step logic disabled. When $\overline{\text{RUN}}$/STEP is true, the SBC is in the step mode, with an NMI interrupt occuring for each instruction executed below the step address limit.

11.7.2 Control Signals

The Bank Select Enable signal (BSE) and its logical inverse ($\overline{\text{BSE}}$) are controlled by the System VIA, port B bit 2 (PB2). The Bank Select Enable signals are used by the on-board device decoders to determine which 65K byte bank the CPU is addressing, with a low BSE selecting bank Ø and a high BSE selecting bank 1. The SBC is always operating in only one bank at a time, although some memory and peripherals (such as the System VIA) are addressed in either bank, i.e. they are common to both banks. The Bank Select Enable is also available at the Expansion Interface as the RM 65 Buffered Bank Address (BADR/).

The DMA Terminate signal ($\overline{\text{BDMT}}$) is controlled by the System VIA, port B bit 6 (PB6). This signal is available at the Expansion Interface as the RM 65 Buffered DMA Terminate (BDMT/) signal. Upon VIA reset or under software control, $\overline{\text{BDMT}}$ is set true to terminate any DMA transfers in progress.

The Write Protect Interrupt Request ($\overline{\text{IRQWP}}$), which is generated by an attempted write into protected memory, is an interrupt source for the interrupt request prioritizer as well as to the System VIA port A control line 1 (CA1). This means that a write protect interrupt request is polled by reading the System VIA Interrupt Flag Register (IFR).

11.7.3 Audio Recorder Interface

The AIM 65/40 supports mass storage for program and data on magnetic audio tape. The output data from the System VIA, port B bit 7 (PB7) is buffered (Z10), filtered, and brought out to the Audio/TTY connector (J3) as MICROPHONE. The tape input from EARPHONE is limited (Z8) and passed to the System VIA, port B control line 1 (CB1). The tape filtering and limiting is only signal conditioning - the generation and recovery of data must be done by the CPU.

There are two reed relays (Z7, Z9) provided to control two tape recorders. The relay contacts are available on the Audio/TTY connector (CONTROL 1, CONTROL 1 RTN, CONTROL 2, CONTROL 2 RTN). The System VIA port B bits 4 and 5 (PB4 and PB5) are buffered (Z10) to control tape relays 1 and 2, respectively.

11.7.4 Printer and Display Interface

The remaining System VIA port and control lines are routed to the Printer Connector (J5) and the Display Connector (J6), which are designed to support these peripherals. Wherever applicable, the pin assignments for these connectors are compatible with the Parallel I/O connector (J1). Refer to the pin assignments in Appendix L.

The System VIA port A lines (PA0-PA7) go to both the display and the printer connectors, typically as Display/Printer Data. In addition to $\overline{\text{RESET}}$ from the control bus, both connectors also receive $\overline{\text{PAPER FEED}}$ from the Keyboard connector (J7).

There are two signals that are unique to each of these connectors. The printer $\overline{\text{STROBE}}$ and $\overline{\text{ACK}}$ signals from the printer connector (J5) are routed to system VIA port B bit 1 (PB1) and port A control line 1 (CA1), respectively. The display $\overline{\text{STROBE}}$ and $\overline{\text{ACK}}$ lines are connected from the display connector (J6) to SYSTEM VIA port B bit Ø (PBØ) and port B control line 2 (CB2), respectively. Typically, $\overline{\text{STROBE}}$ indicates that data is available to the peripheral and $\overline{\text{ACK}}$ acknowledges that it has been received by the peripheral.

11.8 KEYBOARD VIA

The R6522 VIA (Z62) interfaces with the Keyboard connector and the on-board audio speaker circuitry. The Keyboard VIA is assigned the addresses $FFCØ - $FFCF and is common to both banks. $\overline{\text{IRQK}}$ is an interrupt request source to the Interrupt Request Prioritizer.

11.8.1 Keyboard Interface

The Keyboard connector (J7) gives access to most of the peripheral lines of the Keyboard VIA. Wherever applicable, the Keyboard connector pin assignments are compatible with the Parallel I/O connector (J1). This connector is intended to interface to an AIM 65/4Ø Keyboard, and the signal names reflect this.

The Keyboard VIA port A lines (MRTØ-MRT7), port B lines (MSBØ-MSB7), and the CA2 control line (MSB8) will typically form a matrix of nine strobes (MSB's) by eight returns (MRT's), with a switch at every intersection. This switch matrix is scanned by driving one strobe line low and sensing the return lines. If any return line is low, a key is down, with the "Ø" bits indicating the return line - thus showing the key position within the matrix. This process is repeated for each of the nine strobe lines to yield any or all of the keys down in the keyboard matrix.

The eight return lines (MRTØ-MRT7) are also the inputs of the key-down gate (Z81). If any of the eight inputs are low, the output is high, indicating a key-down. This output drives the Keyboard VIA port A control line 1 (CA1). Typically, all strobe lines can be set low and then any pressed key will generate an interrupt through CA1. To service the interrupt, the keyboard must be scanned, yielding the key or keys down. Wire jumper W4 can be removed if the keyboard interrupt is not required.

Three additional signals, typically connecting to switches outside of the keyboard matrix with one pole grounded, are $\overline{\text{RESET SW}}$, $\overline{\text{ATTN SW}}$, and $\overline{\text{PAPER FEED}}$. The $\overline{\text{RESET SW}}$ goes to the input of the reset conditioning circuitry, in parallel with the on-board reset switch (S1). Wire jumper W6 can be removed to disconnect this signal. The $\overline{\text{ATTN SW}}$ is conditioned by debouncing circuitry (Z74) to generate an $\overline{\text{NMI}}$ interrupt. With jumpers W8 removed and W7 installed, the $\overline{\text{NMI}}$ will not be generated, but the signal will be sensed on CB1. $\overline{\text{PAPER FEED}}$ is connected directly to the Printer (J5) and the Display (J6) connectors. Wire jumper W3 can be removed to disconnect this signal.

11.8.2 Audio Speaker

A piezo-electric speaker (DS1) and amplifier (Z74, Z80) provides a programmable audio indicator. This speaker can alert the operator of invalid keyboard inputs, announce the completion of an operation, or warn of an improper condition.

The Audio Speaker is controlled by the Keyboard VIA port B control line 2 (CB2). To provide sound, this control line must be toggled high and low at the frequency of the desired sound. Wire jumper W5 can be removed to disable this control line if the audio speaker is not required.

11.9 USER VIA

The R6522 User VIA (Z1) interfaces with the Parallel I/O connector (J1) to provide user dedicated input and output. The VIA has two 8-bit bidirectional input/output ports, four I/O control lines, two fully programmable 16-bit timer/counters and an 8-bit shift register. There is also control of interrupt generation from seven independent I/O conditions. For a full operating description of the VIA, refer to Section 7. The User VIA is assigned the addresses $FFAØ-$FFAF, common to both banks. $\overline{\text{IRQU}}$ is an interrupt request source for the Interrupt Request Prioritizer.

11.1Ø USER ACIA

The R6551 User ACIA (Z2) is used for asynchronous serial communication either through the RS-232C interface or the 2Ø mA TTY interface.

The ACIA is programmable through its various registers. The control register allows the baud rate, the word length and the number of stop bits to be programmed. A command register sets the ACIA interrupt and parity modes, while the status register reflects interrupt and data transfer conditions. The baud rates are derived from a 1.8432 MHz crystal (Y2). For a full operating description of the ACIA, refer to Section 8. The User ACIA is assigned the addresses $FFDØ-$FFD3, common to both banks. $\overline{\text{IRQA}}$ is an interrupt request source for the Interrupt Request Prioritizer.

11.1Ø.1 RS-232C Interface

The RS-232C interface contains the line drivers (Z3) and line receivers (Z4) to invert and buffer the signals between the RS-232C interface levels (+12V to -12V) on the RS-232C connector (J2) and the TTL levels of the ACIA (ØV to +5V). The JACIA-1 jumpers configure the interface to operate as a data set or a data terminal. The other jumpers (JACIA-2, -3, -4) either route or bypass control signals from the interfacing equipment.

11.1Ø.2 2Ø MA TTY Interface

The TTY Interface contains current mode drivers and receivers for a 2Ø MA current loop on the Audio/TTY connector (J3).

The receive data current loop (RD, RETURN) is coupled by an opto-isolator (Q3) to TTL levels, inverted (Z41) and combined with the RS-232C receive data (Z42) for input to the ACIA on RD.

The TTY transmit data uses the TD output of the ACIA, which is inverted and buffered (Z85, Z6) to drive an opto-isolator (Q2). The transmit data current loop (TD, RETURN) is controlled by the output of Q2. When the TTY loop is not being used, as sensed by no load in the transmit data current loop, the TTY receive data is forced to a one (Q4), allowing the RS-232C receive data to pass freely into the ACIA.

The TTY request to send (RTS) uses the $\overline{\text{RTS}}$ output of the ACIA, which is inverted and buffered (Z85, Z6) to drive an opto-isolator (Q1). The request to send current loop (RTS, RETURN) is controlled by the output of Q1.

11.11 EXPANSION BUS

The AIM 65/4Ø SBC module allows expansion to off-board resources via the RM 65 bus. This allows expansion with additional memory (RAM, ROM), controllers (Floppy Disk, CRT, IEEE-488 etc.), I/O (GPIO&Timer, ACIA, etc.) or custom circuitry on prototyping modules. The RM 65 Bus signal description is given in Table 11-3. The Expansion connector (J4) can drive one RM 65 module directly, or several modules by using an AIM 65/4Ø to RM 65 buffer module.

Data transceivers (Z11) invert and transfer eight bits of parallel data (BDØ/-BD7/) between the SBC module and the RM 65 bus. The direction of the transceivers is controlled by the read/write (R/$\overline{\text{W}}$) signal from the CPU. The transceivers are inhibited (tri-stated) by BUS DRIVER SEL when any selected on-board memory is addressed or the bus float signal (BFLT/) is active.

Address buffers (Z14, Z15) invert and transfer 16 parallel address bits (BAØ/-BA15/) from the SBC module onto the RM 65 bus. These buffers are inhibited when BFLT/ is active.

All clock and control signals between the SBC module and the RM 65 bus are buffered. The read/write (BR/$\overline{\text{W}}$, BR/$\overline{\text{W}}$/), phase 2 clock (BØ2, BØ2/), sync (BSYNC) and bank address (BADR/ -renamed on-board $\overline{\text{BSE}}$) signals have non-inverting buffers (Z12) that are inhibited when BFLT/ is active. The non-maskable (BNMI/), interrupt request (BIRQ/), ready (BRDY) and set overflow (BSO) have open-collector buffers (Z13). The bus float signal (BFLT/) is buffered to become $\overline{\text{DMA}}$. The reset (BRES/), DMA terminate (BDMT/) and phase 1 clock (BØ1) are the only signals from the SBC module that are not inhibited by a bus float.

Table 11-3. SBC Connector J4 (Expansion) Signal Definitions

Mnemonic	Signal Name and Signal Description
	NOTE: All signals interfaced to and from the SBC module are driven at TTL voltage levels.
+5V	+5V dc supplied to the RM 65 Bus from the SBC module.
GND	Ground System ground.
BDØ/-BD7/	Buffered Data Bits Ø-7 Eight bi-directional inverted data lines transfer 8-bit data bytes between the Data Transceivers in the SBC module and the Bus. The Data Transceivers are disabled (tri-state) when BFLT/ is active.
BAØ/-BA15/	Buffered Address Bits Ø-15 Sixteen address lines transfer an inverted 16-bit parallel address from the Address Buffers in the SBC module onto the Bus. The Address Buffers are disabled (tri-state) when BFLT/ is active.
BADR/	Buffered Bank Address The BADR/ signal is driven by the SBC module onto the Bus. The level of BADR/ is controlled by the System VIA (BSE). A high BADR/ signal (BSE=Ø) addresses the lower 65K (Bank Ø) memory bank; a low BADR/ signal (BSE=1) addresses the upper 65K (Bank 1) memory bank. BADR/ is disabled when BFLT/ is active.

Table 11-3. SBC Connector J4 (Expansion) Signal Definitions (Continued)

Mnemonic	Signal Name and Signal Description
BØ1	Buffered Phase 1 Clock The BØ1 signal is the system clock generated by the SBC module for the Bus.
BØ2	Buffered Phase 2 Clock The BØ2 signal is generated by the SBC module to synchronize data transfers on the bus. The address and read/write lines are set-up in the negative portion of BØ2. The data lines are set-up in the positive portion of BØ2. BØ2 is disabled when BFLT/ is active.
BØ2/	Buffered Phase 2 Clock "NOT" The BØ2/ signal is generated by the SBC module to synchronize data transfers on the Bus (the logical inverse of BØ2). The address and read/write lines are set-up in the positive portion of BØ2/. The data lines are set-up in the negative portion BØ2/. BØ2/ is disabled when BFLT/ is active.
BR/$\overline{\mathrm{W}}$	Buffered Read/Write The BR/$\overline{\mathrm{W}}$ signal is generated by the SBC module to control the direction of data transfer on the Bus. A high BR/$\overline{\mathrm{W}}$ (read operation) enables the SBC module to receive data from the bus. A low BR/$\overline{\mathrm{W}}$ (write operation) enables the SBC module to transmit data onto the bus. BR/$\overline{\mathrm{W}}$ is disabled (tri-state) when BFLT/ is active.

Table 11-3. SBC Connector J4 (Expansion) Signal Definitions (Continued)

Mnemonic	Signal Name and Signal Description
BR/$\overline{W}$/	Buffered Read/Write "Not" The BR/$\overline{W}$/ signal is generated by the SBC module to control the direction of data transfer on the Bus (the logical inverse of BR/$\overline{W}$). A low BR/$\overline{W}$/ indicates a read operation. A high BR/$\overline{W}$/ indicates a write operation. BR/$\overline{W}$/ is disabled (tri-state) when BFLT/ is active.
BSYNC	Buffered Sync The BSYNC signal is driven by the SBC module onto the Bus. The BSYNC signal goes high during the positive portion of a Ø1 clock signal when the CPU is performing an op code fetch and stays high for the remainder of that cycle. The BSYNC signal is disabled (tri-state) when BFLT/ is active.
BSO	Buffered Set Overflow The BSO signal is received from the Bus by the SBC module. A negative transition on this line sets the overflow flag in the R65Ø2 CPU.
BRDY	Buffered Ready The BRDY signal is received from the Bus by the the SBC module. When the R65Ø2 CPU receives a low BRDY, the CPU will stop execution in the next read cycle. Execution will resume when BRDY returns high.

Table 11-3. SBC Connector J4 (Expansion) Signal Definitions (Continued)

Mnemonic	Signal Name and Signal Description
BFLT/	<u>Buffered Bus Float</u> The BFLT/ signal is received from the Bus by the SBC module. This line is brought low (active) to disable SBC control of the Bus for DMA transfers.
BDMT/	<u>Buffered DMA Terminate</u> The BDMT/ signal is driven from the SBC module onto the Bus. The level of BDMT/ is controlled by the System VIA ($\overline{\mathrm{BDMT}}$). A low (active) BDMT/ signal terminates the DMA operation.
BIRQ/	<u>Buffered Interrupt Request</u> The BIRQ/ signal is received by the SBC module from the Bus. BIRQ/ is forced low by any module to request service. This line corresponds to the $\overline{\mathrm{BIRQ}}$ signal of the Interrupt Request Prioritizer.
BNMI/	<u>Buffered Non-Maskable Interrupt</u> The BNMI/ signal is received by the SBC module from the Bus. BNMI/ is forced low by any module to request service. This interrupt corresponds to the $\overline{\mathrm{NMI}}$ signal of the R65Ø2 CPU and cannot be masked.
BRES/	<u>Buffered Reset</u> The BRES/ signal is generated by the SBC module for the Bus. BRES/ is pulsed low for 1Ø milliseconds at power-on. BRES/ is held low while the RESET switch is depressed and remains low for 1Ø milliseconds after release. BRES/ provides a hardware reset to the modules on the bus.

SECTION 12

AIM 65/40 GRAPHICS PRINTER DESCRIPTION

The AIM 65/40 Graphics Printer (A65/40-0600) is an intelligent 40-column thermal printer that connects with the AIM 65/40 SBC module over a Centronics type parallel interface. The assembly contains an R6504 CPU, one R2332 4K-byte ROM, one R6532 RAM I/O Timer (RIOT), printer drivers and a PU1840/4 Olivetti thermal printer. A microcomputer based controller relieves the host computer (in this case, the AIM 65/40 SBC module) from the task of monitoring and controlling the thermal heads and the printer motor timing. Figure 12-1 shows the AIM 65/40 Graphics Printer. Tables 12-1 and 12-2 list the printer assembly physical characteristics and power requirements.

The graphics printer provides a permanent record of user commands, data and programs as well as AIM 65/40 system status, prompts and messages. Printing up to four lines per second, the 7 x 8 dot matrix printer provides rapid, quiet and reliable operation.

The printer operates in response to command and data characters received from the SBC module. Both commands and data are encoded in 8-bit ASCII format (see Appendix F), however some of the commands deviate from standard control commands. Two types of commands -- control and escape -- control the operation of the printer. These commands basically tell the printer what mode to operate in, how to handle the data in the print buffer, and when to print the contents of the print buffer.

There are two print modes, normal and graphics. In the normal mode, the complete 96 standard ASCII characters and 160 semi-graphic and special characters can be printed.

In the graphics mode, any character, symbol, or graphic that can be composed within a 280 x n dot format can be printed, where 280 is the number of horizontal dots in a dot row across the paper and n is the number of vertical dot rows.

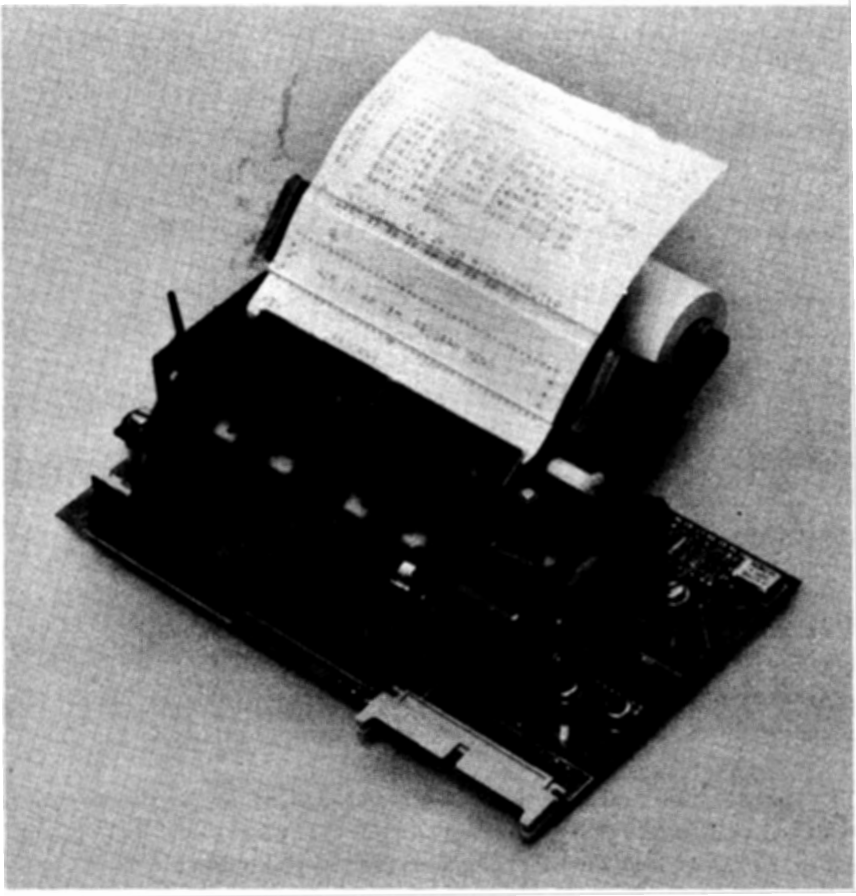


Figure 12-1. AIM 65/40 Graphics Printer

Table 12-1. Printer Physical Characteristics

Characteristic	Value
Physical	
Width	7.6 in. (193 mm)
Length	6.3 in. (16Ø mm)
Height	3.Ø in. (77 mm)
Weight	1.Ø lb. (Ø.45 Kg)
Environment	
Operating Temperature	Ø°C to 5Ø°C
Storage Temperature	Ø°C to +7Ø°C
Relative Humidity	Ø% to 85% (without condensation)
Interface Connection	4Ø-pin 3M #3495-1ØØ2 or equivalent receptacle. Mates with 3M #3417-6Ø4Ø, or equivalent, ribbon cable connector.
Power Connector	Three-post terminal block

NOTE

Dimensions do not include paper holder and module stand-offs.

Table 12-2. Printer Power Requirements

Voltage	Typ.	Max	Peak	Unit
+5V ±5% Regulated	Ø.3	Ø.4	Ø.4	A
+24V (21.4 - 27.6) Unregulated	2.5	4.Ø	6.3*	A

NOTE

* +24V peak current specified as worst case with printer duty cycle of 75%. For most cases, a +24V 4A power supply is sufficient.

The printer stores received data characters into an 8Ø-byte printer buffer. (The buffer is treated as two separate 4Ø-byte buffers during graphics operation, see Section 12.1.4) Internal character and cursor positioning allows the printer to operate in parallel with an AIM 65/4Ø 4Ø-character display.

A self-test function checks the controller RAM then prints the ROM checksum, the entire character set, and characters to check horizontal dot generation and vertical dot alignment.

The complete printer assembly mounts on top of the AIM 65/4Ø SBC module by means of five 1 1/4-inch stand-offs. However, it can also be removed and installed up to six feet away from the SBC module for remote operation with interface connection through a user provided cable.

12.1 PRINTER ASSEMBLY OPERATION

The Graphics Printer operates in the AIM 65/4Ø system in parallel with the AIM 65/4Ø 4Ø-character Display. Data sent to the printer may also be sent to the display. The primary difference is that the data in the print buffer is usually not printed until a carriage return is issued or until the 4Ø-character line is full, whereas the display shows each character as it is received or deleted along with a cursor.

The commands are the same for the printer and display wherever possible with minimum differences to account for display versus printer output media. It is easiest to think of the printer in terms of 4Ø-character window within an 8Ø-character print buffer. The contents of the window can be printed at anytime to print what is visible on the 4Ø-character display. In addition, a cursor position (visible on the display but invisible inside the printer) tells the printer where the next received character is to be placed in the print buffer.

Note that some of the commands processed by the printer appear to do nothing. These commands allow the printer to stay in sync with unique display commands.

12.1.1 Control Commands

The control commands (see Table 12-3) are individual ASCII character codes ($ØØ - $1F) which are issued from the SBC module under operator or program control. The commands are normally issued under program control by I/O ROM subroutines (see Section 6.5) which are called by a system or application program. The commands may also be issued manually from the AIM 65/4Ø or terminal keyboard when the Monitor is in the Direct Peripheral Control mode (CTRL Z, see Section 4.2.11) by holding the CTRL key down while typing a valid command key. The control commands definitions are:

CTRL A - Clear Buffer

Clears the 8Ø-byte print buffer to all spaces ($2Ø) and performs a carriage return.

CTRL B - Clear to End of Buffer

Causes all positions to the right of (and including) the cursor to be cleared with spaces ($2Ø) to the end of the line.

CTRL C - Clear Buffer

Same as CTRL A.

CTRL D - Clear to End of Buffer

Same as CTRL B.

CTRL E - Clear Buffer

Same as CTRL A.

Table 12-3. Printer Control Commands

ASCII Code	Control Character	Description
00	CTRL @	*
01	CTRL A	Clear Buffer
02	CTRL B	Clear to End of Buffer
03	CTRL C	Clear Buffer
04	CTRL D	Clear to End of Buffer
05	CTRL E	Clear Buffer
06	CTRL F	Clear to End of Buffer
07	CTRL G	*
08	CTRL H	Backspace (Left Arrow)
09	CTRL I	Forespace (Right Arrow)
0A	CTRL J	Line Feed (Down Arrow)
0B	CTRL K	Line Feed (Down Arrow)
0C	CTRL L	Form Feed
0D	CTRL M	Carriage Return (Home On Line)
0E	CTRL N	Carriage Return (Home On Line)
0F	CTRL O	Carriage Return (Home On Line)
10	CTRL P	Pass Through Next Character
11	CTRL Q	*
12	CTRL R	*
13	CTRL S	Toggle Insert Character Mode
14	CTRL T	Delete One Character
15	CTRL U	*
16	CTRL V	*
17	CTRL W	Cursor On
18	CTRL X	Cursor Off
19	CTRL Y	Cold Reset
1A	CTRL Z	Warm Reset
1B	CTRL [	Escape Command (ESC)
1C	CTRL \	Print Without Clear
1D	CTRL]	Print Display Image (Window)
1E	CTRL ^	Paper Feed
1F	CTRL _	*

NOTE:

(*) Characters with no indicated function are acknowledged, but do not otherwise affect the printer operation.

CTRL F - Clear to End of Buffer

Same as CTRL B.

CTRL H - Backspace (Left Arrow)

Causes the cursor to move one position to the left in the print buffer. If the cursor is at the left side of the window, but not at the first position of the line, the window will scroll from left to right one position.

CTRL I - Forespace (Right Arrow)

Causes the cursor to move one position to the right in the buffer.

CTRL J - Line Feed (Down Arrow)

Prints the entire 80 character print buffer on two 40-character lines. If 40 or less characters are in the buffer, only one line is printed. If the buffer is empty, one blank line is printed. The buffer is cleared following the print.

CTRL K - Line Feed (Down Arrow)

Same as CTRL J.

CTRL L - Form Feed

Causes the printer to skip to the top of the next page as defined by the current environment (see ESC E G and ESC E P commands Section 12.1.2).

CTRL M - Carriage Return (Home on Line)

Positions the cursor at position one in the print buffer.

CTRL N - Carriage Return (Home on Line)

Same as CTRL M.

CTRL O - Carriage Return (Home On Line)

Same as CTRL M.

CTRL P - Pass Through Next Character

Causes the next character received to be stored directly into the print buffer, ignoring any special meaning of control codes or the bit 7 attribute. This allows any of the 32 additional characters to be printed 66 (encoded $ØØ-$1F).

CTRL S - Toggle Insert Character Mode On/Off

When the <u>insert</u> character mode is on, all characters received are inserted into the buffer just before the cursor position. This causes the characters to the right of (and including) the cursor to move one position to the right. If the buffer becomes full (8Ø characters), no more characters wil be inserted (they will be discarded). All control functions (carriage return, delete, etc.) will function normally while in insert mode.

Note that this is different from the <u>replace</u> character mode (the normal mode of operation), where the character under the cursor is replaced by the received character then the cursor advanced one position.

CTRL T - Delete One Character

Causes the character over which the cursor is positioned to be deleted and all characters to the right to move left one position to fill in the vacated space. No action occurs if there are no characters under and to the right of the cursor.

CTRL W - Cursor On

Causes a cursor indicating the position of the next character entry to be enabled. This is the default mode.

CTRL X - Cursor Off

Causes no cursor to be enabled. Used when no user input is required.

CTRL Y - Cold Reset

Clears the print buffer and modes. Performs carriage return. Initializes normal characters, line gap, dot print time and print lines/page, environment.

CTRL Z - Warm Reset

Clears the print buffer, sets the window to positions 1-4Ø and sets the cursor to position one.

CTRL [- Escape Command (ESC)

Informs the printer to accept the next character as an escape command (see Section 12.1.2).

CTRL \ - Print Without Clear

Prints the contents of the 8Ø-byte print buffer without clearing the buffer after printing.

CTRL] - Print Display Image

Prints the 4Ø-character display image, i.e. the window.

CTRL ^ - Paper Feed

Advances the printer one line. If the next line is beyond the page length, (see ESC A P command) the paper is also advanced the number of lines specified by the page gap (see ESC A G command).

12.1.2 Escape Commands

The escape (ESC) commands are two or more sequential ASCII codes. The first code is the ESC character ($1B) and the second code is the actual command. Additional characters specify command parameter values are sometimes. Each command is initiated and terminated automatically upon receipt of the last required character with the exception of the ESC E commands.

The set environment commands (ESC E) must be terminated by a carriage return ($ØD) before they take effect. After the required number of characters is processed for each command, the printer looks for another ESC E command character (e.g., A, G, P, or T) or the carriage return. This allows several ESC E commands to be set up before initiating any of them.

An escape command can be entered from the AIM 65/4Ø or a terminal keyboard in the Monitor Direct Peripheral Control mode (CTRL Z, see Section 4.2.11) by first typing the ESC key and then the command key and parameter values (if appropriate).

Table 12-4 summarizes the printer escape commands.

Table 12-4. Printer Escape Commands

Character Sequence	Hex Codes		
Command & Parameters	Command	Parameters	Function

ESC = (Line) (Pos)	1B 3D	yy zz	Set Cursor Position
			yy = Don't Care
			zz = Cursor Position
			= $2Ø - $6F

ESC E A (Code)	1B 45 41	yy	Set Bit 7 Attributes yy = $ØØ = $Ø2 (Bits Ø,2-7 = Don't Care)
ESC E G (Line)	1B 45 47	yy	Set Page Gap yy = Line Count = $Ø1 - $FF
ESC E P (Line)	1B 45 5Ø	yy	Set Page Length yy = Line Count = $Ø1 - $FF
ESC E T (Time)	1B 45 54	yy	Set Dot Time yy = Dot On-Time = $2Ø - $3F
ESC G	1B 47		Enter Graphics Mode
ESC S	1B 53		Transmit Character Definition
ESC T	1B 54		Perform Display Self-Test
ESC W (Pos)	1B 57	yy	Set Window Position yy = $2Ø - $69

The escape commands are defined as follows:

ESC = (Line) (Position) - Set Cursor Position

Moves the cursor to any position in the 8Ø character print buffer. The ESC = sequence is followed by two characters. The first character is ignored and may be any value. The second character is the cursor position and may range from $2Ø (position 1) - $6F (position 8Ø). If the cursor is within the first 4Ø positions in the buffer, the window is set to the first 4Ø positions.

ESC E A (Code) CR - Set Bit 7 Attributes

If bit 7 of the individual print character code is a "Ø" (see Figure 12-2), the character is printed as either an alphanumeric character ($2Ø to $7F) or as a special semi-graphic character ($8Ø to $FF) depending on what bit Ø of the code word sent with the ESC E A sequence. The command interpretation is:

ESC E A Code	Print Character Code	
Bit 1	Bit 7 = "1"	Bit 7 = "Ø"
Ø	Normal	Normal
1	Semi-Graphic	Normal

All other bits in the ESC E A code word are ignored.

ESC E G (Lines) - Set Page Gap

Sets the gap between printed pages in number of character lines. The value may range from Ø to 255 ($FF). The default value set by the printer upon RESET is Ø.

ESC E P (Lines) - Set Page Length

Sets the printed page length in number of character lines. The value may range from 1 to 63 ($3F). The default value set by the printer upon RESET is 25 ($19).

ESC E T (Value) - Set Dot Print Time

Sets the length of time that a dot is energized while printing. The value may range from 1 ($01) to 63 ($3F). The default value set by the printer upon RESET is 58 ($3A).

ESC G - Enter Graphics Mode

Causes the printer to interpret subsequent bytes as dot patterns to be directly printed (see Section 12.1.4).

ESC S (Char) (8 Strobes) - Transmit Character Definition

Upon receipt of each strobe following the ESC S Char sequence, the printer transmits a byte containing a row dot pattern of the received character back to the SBC module. The top row is transmitted first with bit 7 containing the left-most column data value. Bit 0 is always zero.

$\overline{\text{ACK}}$ is set low for 10 us then returned high after each byte is sent to the SBC module. The SBC module must toggle the $\overline{\text{STROBE}}$ low then high to acknowledge receipt before the next character is sent.

ESC T - Perform Display Self Test

Initiates printer self test (see Section 12.1.5).

ESC W (Position) - Set Window Position

Sets the starting position of the 40 character display window within the 80-character buffer. This allows the print buffer to stay in sync with the 40-character display. The ESC W sequence is followed by the value of the starting position. The value may range from $20 (position 1) to $49 (position 40).

12.1.3 Normal/Semi-graphics Character Mode

256 standard and special characters may be displayed in the normal mode of operation (see Figure 12-2). These characters are grouped into three categories:

- o 96 Standard characters ($2Ø - $7F)
- o 128 Semi-graphic characters ($8Ø - $FF)
- o 32 Control characters ($ØØ - $1F)

a. Standard Character Set (Normal Mode)

The standard 96 alphabetic, numeric and special ASCII characters are encoded from $2Ø through $7F. These characters are printed when the print normal characters mode has been selected with the ESC E A $ØØ command.

b. Extended Character Set (Semi-graphics Mode)

128 special characters are provided in the semi-graphics mode and are encoded from $8Ø through $FF. These characters are printed when bit 7 of the character code is set to a "1" and the print semi-graphic characters mode has been selected with the ESC E A $Ø2 command control (see Section 12.1.2).

Semi-graphic characters are usually generated under program control rather than by the operator since only standard characters (ASCII codes $2Ø-$7E) may be entered from the keyboard. A user defined program can be written to set bit 7 to "1" to convert from normal to semi-graphic characters. The following procedure can also be used to set the bit 7 to "1" and to command the display in to semi-graphics mode. This is handy to see how the semi-graphic characters are displayed.

(1) Enter a string of normal characters corresponding to the desired semi-graphic characters (i.e. bit 7 = "Ø" rather than "1", see Appendix E) into the Editor Text Buffer.

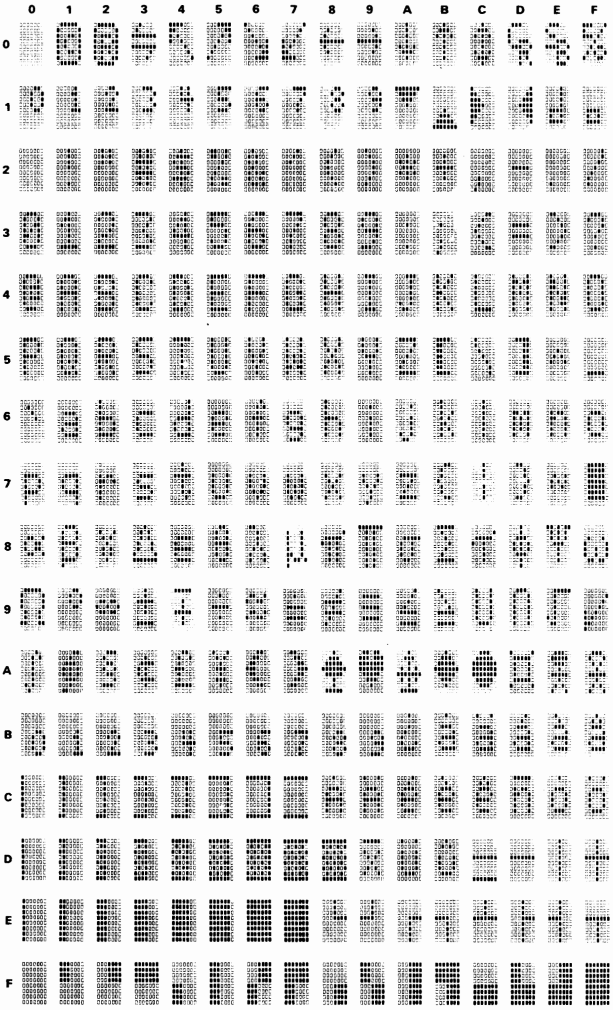

0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Figure 12-2. Printer Normal/Semi-graphics Character Set

(2) Change bit 7 from "Ø" to "1" for each of the character codes using the Monitor M and "/" commands.

(3) Enter the Direct Peripheral Control mode with the Monitor CTRL Z command.

(4) Command the semi-graphics mode using the ESC E A 2 command sequence.

(5) Press ATTN to return to the Monitor (not RESET since that will cause the display to reinitialize).

(6) Go to the top of the Editor with the T command and list the characters as desired.

c. Control Character Set

32 additional special characters are encoded from $ØØ - $1F. These character codes, which are normally control commands, are interpreted as special characters when preceded by a CTRL P ($1Ø). Each character printed must be received as a two byte sequence, i.e. CTRL P then the special character code.

12.1.4 Graphics Mode

In the graphics mode, the printer interprets each received byte as dot patterns rather than commands or normal/semi-graphic characters. There are 28Ø dots per horizontal row, and any desired number of lines that can be printed.

To operate in the graphics mode, the printer must receive a CTRL G ($Ø7) command followed by exactly 4Ø bytes for each dot row to be printed. The contents of each byte specifies which of the seven dots per thermal head are to be energized. Bit 7 in the received byte controls dot 7 and bit 1 controls dot 1 (bit Ø is ignored). Figure 12-3 illustrates graphic mode printing action.

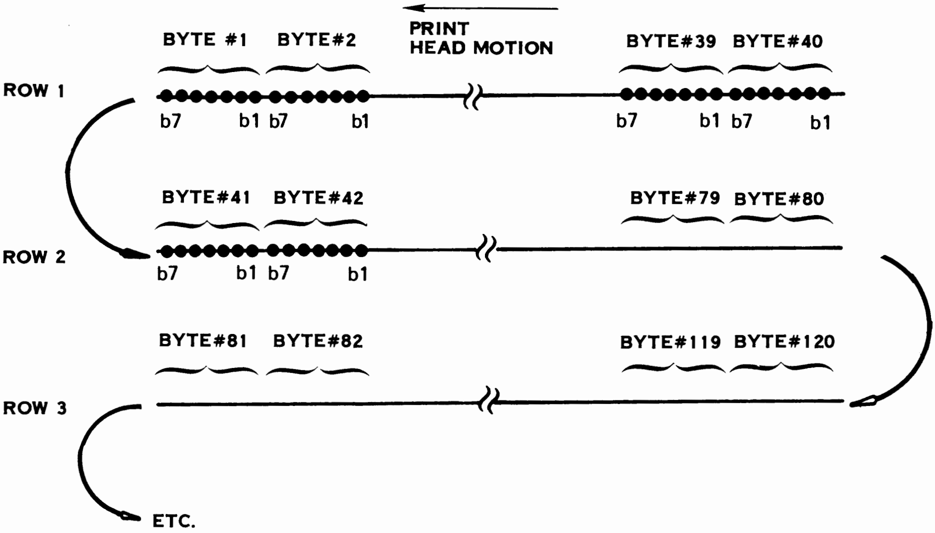


Figure 12-3. Printer Graphics Mode Dot Pattern Positioning

When in the graphics mode, it is necessary to keep the print head moving to prevent a gap of up to two dots from occurring due to coasting if the printer motor is stopped. The print buffer is double-buffered (i.e., two separate 4Ø-byte buffers) to allow receipt of the next row of dot patterns while the current row is being printed. Data transfer is permitted for approximately 13 milliseconds out of each 2Ø millisecond dot row print cycle. If the 4Ø new bytes are not received during printing of the current row, the printer will automatically exit the graphics mode. Upon termination of the graphics mode, the printer performs a warm reset in order to receive new commands and data.

The easiest way to transmit graphics data to the printer is to first store the digitized picture in AIM 65/4Ø SBC RAM. The data should be formatted and ready to send directly to the printer. A buffer area of 4Ø-bytes (per dot row) by the number of desired dot rows is needed to store the digitized picture.

For example, a 16ØØ byte picture buffer is required to hold a picture consisting of 4Ø rows, while a 11,2ØØ byte buffer is needed to store a square picture of 28Ø dots wide by 28Ø dot rows long. The graphics driver then needs only to initialize the graphics printer mode and to transmit the data sequentially starting with the first byte and ending with the last byte. The most critical elements of the driver is to transmit the data within the timing requirements of the printer and to detect the acknowledges from the printer.

The example program in Figure 12-4 includes three user functions - a bit pattern fill routine, and a diagonal pattern generator routine and a picture output driver.

The bit pattern fill routine, executed by pressing the F1 key, loads a fixed bit pattern into the picture buffer. This function can be used to test the graphics driver function. The bit pattern can be altered by changing the constant PATNR. This loader should be replaced by an application program to load the actual data.

Similiarly, the diagonal pattern generator fills the buffer area with a shifting one bit pattern. Pressing the F2 Key will execute this program.

The picture driver program is executed by pressing the F3 Key. The program returns to the Monitor upon completion.

The location and the size of the buffer area can be changed by altering constants BEGIN and TOP. You may want to experiment with this example to become familiar with transmitting a graphics picture to the printer.

12.1.5 Printer Self Test

The printer self test performs a functional test of the controller, drivers, print mechanism and thermal heads. It also prints out characters for a visual check of the print head alignment.

```
ADDR .OBJECT. SOURCE

0800          ; EXAMPLE OF GRAPHICS
0800          ;Z PAGE VARIABLES
0800                 *=$14
0014          BASE   *=*+2
0016          END    *=*+2
0018 80       PATRN  .BYT $80
0019          ; GRAPHICS BUFFER
0019          BEGIN=$3A00
0019          TOP=$3F00
0019          CHAR   =0
0019          ; AIM65/40 PRINTER
0019          PTRSTB =$FD
0019          UNSTRB=$03

0019          ; AIM 65/40 I/O ROM
0019          OPNPTR=$F8C8
0019          PUTPTR=$F8F1
0019          CLOPTR=$F8B3
0019          WAITP=$F935
0019          ; AIM 65/40 MONITOR
0019          COMIN1=$A314
0019                 *=$250
0250 00 0A    F1     .WOR FILL
0252 30 0A    F2     .WOR LINES
0254 77 0A    F3     .WOR PAINT
0256          ; SYSTEM VIA
0256                 *=$FFB0
FFB0          PB     *=*+1
FFB1          PA     *=*+1
FFB2          DDRB   *=*+1
FFB3          DDRA   *=*+1
FFB4                 *=*+9
FFBD          IFR    *=*+1
FFBE                 *=$A00
0A00          ;
0A00          ; GRAPHICS PROGRAM
0A00          ;
0A00          ; FILL BUFFER WITH CHAR
0A00          ; ENTER WITH THE F1 KEY

0A00          ;SET BUFFER LIMITS
0A00 A9 00    FILL   LDA #<BEGIN
0A02 85 14           STA BASE
0A04 A9 3A           LDA #>BEGIN
0A06 85 15           STA BASE+1
0A08 A9 00           LDA #<TOP
0A0A 85 16           STA END
0A0C A9 3F           LDA #>TOP
0A0E 85 17           STA END+1
0A10 A0 00           LDY #0
```

Figure 12-4. Example Graphics Driver Program

```
0A12              ;LOOP TILL BUFFER FILLED
0A12 A9 00        CLOOP   LDA #CHAR
; THE FILL CHARACTER
0A14 91 14                STA (BASE),Y
0A16 E6 14                INC BASE
0A18 D0 02                BNE *+4
0A1A E6 15                INC BASE+1
0A1C A5 15                LDA BASE+1
0A1E C5 17                CMP END+1
0A20 90 F0                BCC CLOOP
0A22 F0 03                BEQ MORE
0A24 4C 14 A3      DONE   JMP COMIN1
0A27 A5 14        MORE    LDA BASE
0A29 C5 16                CMP END
0A2B 90 E5                BCC CLOOP
0A2D 4C 14 A3             JMP COMIN1
0A30              ;
0A30              ;
0A30              ; FILLS THE BUFFER WITH
0A30              ; DIAGONAL LINES
0A30              ; ENTER WITH THE F2 KEY

0A30              ;SET UP BUFFER LIMITS
0A30 A9 00        LINES   LDA #<BEGIN
0A32 85 14                STA BASE
0A34 A9 3A                LDA #>BEGIN
0A36 85 15                STA BASE+1
0A38 A9 00                LDA #<TOP
0A3A 85 16                STA END
0A3C A9 3F                LDA #>TOP
0A3E 85 17                STA END+1
0A40 A0 00                LDY #0
0A42 A2 28                LDX #40
0A44              ; LOOP TILL FULL BUFFER
0A44 A5 18        BLOOP   LDA PATRN
;THE SEED PATTERN
0A46 91 14                STA (BASE),Y
0A48              ;CHANGE PATTERN AT 40
0A48 CA           FORTY   DEX
0A49 D0 05                BNE ONWARD
0A4B A2 28                LDX #40
0A4D 20 6A 0A             JSR CARCHG
0A50 E6 14        ONWARD  INC BASE
0A52 D0 02                BNE *+4
0A54 E6 15                INC BASE+1
0A56 A5 15                LDA BASE+1
0A58 C5 17                CMP END+1
0A5A 90 E8                BCC BLOOP
0A5C F0 03                BEQ SMORE
0A5E 4C 14 A3      THRU   JMP COMIN1
0A61 A5 14        SMORE   LDA BASE
0A63 C5 16                CMP END
0A65 90 DD                BCC BLOOP
0A67 4C 14 A3             JMP COMIN1
```

Figure 12-4. Example Graphics Driver Program (Continued)

```
0A6A             ;
0A6A             ;CHANGE PATTERN CHARACTER
0A6A 46 18       CARCHG LSR PATRN
0A6C A5 18              LDA PATRN
0A6E C9 01              CMP #1
0A70 D0 04              BNE KEEP
0A72 A9 80       AGAIN  LDA #$80
; THE PATTERN
0A74 85 18              STA PATRN
0A76 60          KEEP   RTS
0A77             ;
0A77             ; PRINT OUT BUFFER AREA
0A77             ; ENTER WITH THE F3 KEY

0A77             ;SET THE BUFFER LIMITS
0A77 A9 00       PAINT  LDA #<BEGIN
0A79 85 14              STA BASE
0A7B A9 3A              LDA #>BEGIN
0A7D 85 15              STA BASE+1
0A7F 58                 CLI
0A80 20 C8 F8           JSR OPNPTR
0A83 A9 1B              LDA #$1B
0A85 20 F1 F8           JSR PUTPTR
0A88 20 35 F9           JSR WAITP
0A8B A0 47              LDY #$47
0A8D 78                 SEI
0A8E 8C B1 FF     GR1   STY PA
0A91 A9 C1              LDA #$C1
0A93 8D BD FF           STA IFR
0A96 AD B0 FF           LDA PB
0A99 29 FD              AND #PTRSTB
0A9B 8D B0 FF           STA PB
0A9E 09 03              ORA #UNSTRB
0AA0 8D B0 FF           STA PB
0AA3 A0 00              LDY #0
0AA5 B1 14              LDA (BASE),Y
0AA7 A8                 TAY
0AA8 E6 14              INC BASE
0AAA D0 02              BNE NOOVER
0AAC E6 15              INC BASE+1
0AAE A5 15       NOOVER LDA BASE+1
0AB0 C5 16              CMP END
0AB2 A5 15              LDA BASE+1
0AB4 E5 17              SBC END+1
0AB6 B0 08              BCS DUMP2
0AB8 AD BD FF     GR2   LDA IFR
0ABB 4A                 LSR A
0ABC 90 FA              BCC GR2
0ABE B0 CE              BCS GR1
```

Figure 12-4. Example Graphics Driver Program (Continued)

```
0AC0                    ; ENTER ON LAST LINE
0AC0                    DUMP2
0AC0 20 E6 0A                  JSR WAIP
0AC3 8C B1 FF                  STY PA
0AC6 20 D6 0A                  JSR STROBP
0AC9 20 E6 0A                  JSR WAIP
0ACC AD B1 FF                  LDA PA
0ACF 58                        CLI
0AD0 20 B3 F8                  JSR CLOPTR
0AD3 4C 14 A3                  JMP COMIN1
0AD6                    ;STROBE OUT CHAR FAST
0AD6 48                 STROBP PHA
0AD7 AD B0 FF                  LDA PB
0ADA 29 FD                     AND #PTRSTB
0ADC 8D B0 FF                  STA PB
0ADF 09 03                     ORA #UNSTRB
0AE1 8D B0 FF                  STA PB
0AE4 68                        PLA
0AE5 60                        RTS
0AE6                    ; WAIT FOR HANDSHAKE
0AE6 48                 WAIP   PHA
0AE7 AD BD FF            WAITP1 LDA IFR
0AEA 4A                        LSR A
0AEB 90 FA                     BCC WAITP1
0AED 68                        PLA
0AEE 60                        RTS
0AEF                           .END
```

Figure 12-4. Example Graphics Driver Program (Continued)

The printer self test may be initiated by sending the printer an ESC T command (see Section 12.1.2) or by use of the self test jumper on the printer module.

a. <u>Using the ESC T Command</u>

To use the ESC T command, the printer must be connected to the SBC module, the self test jumper installed in the N position and power applied.

(1) Issue the ESC T command under program or keyboard control. The test will be immediately performed.

(2) Upon test completion, the printer will return to the normal character and command input mode.

(3) Repeat the test as many times as required by re-sending the ESC T command.

b. <u>Using the Self Test Jumper (Printer Disconnected)</u>

The printer self test can be run independently of the interfacing system as follows:

(1) Turn printer power off.

(2) Install the self test jumper in the S (self test) position.

(3) Turn printer power on. The test will be immediately performed.

(4) The test will be repeated automatically every 6Ø minutes.

(5) Turn printer power off.

(6) Return the self test jumper to the N (Normal) position.

c. Using the Self Test Jumper (Printer Connected)

The printer self test can be run using the self test jumper while connected to the SBC module and operating as follows:

(1) Install the self test jumper in the S position.

(2) Press RESET. The self test will be immediately performed.

(3) The test will repeat automatically every 60 minutes.

(4) Return the self test jumper to the N position.

(5) Press RESET. The printer will return to normal character and command processing.

d. Printer Self Test Operation

The test operates as follows:

(1) The 128 bytes of RAM are tested. The results of the Test are then printed as:

```
        RAM OK   (test passed)
   or   RAM FAIL (test failed)
```

(2) The ROM checksum is computed and printed for visual evaluation, e.g.:

```
   ROM=8544
```

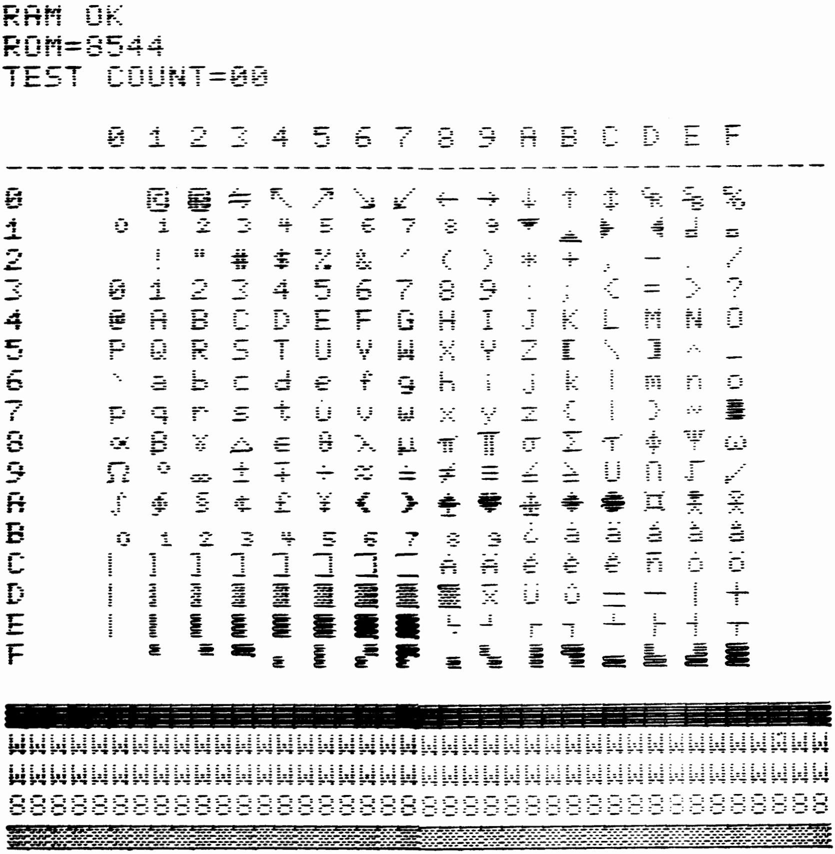
(3) The test count is printed e.g.:

```
   TEST COUNT=00  (first pass)
```

e. The entire normal and semi-graphics character set is printed in table form with corresponding ASCII codes. The LSB (bit 0-3) of the code is the horizontal heading and the MSB (bits 4-7) of the code is the vertical heading.

f. Five rows of semi-graphic and normal characters are printed for a visual check of print head alignment and individual thermal head operation. A solid line of semi-graphics bar characters turns on all dots on. Two rows of W's and one row of 8's allow vertical dot alignment check. A row of shaded bar characters alternates rows of energized dots between solid top and bottom dot lines.

A complete self test is shown in Figure 12-5.



```
RAM OK
ROM=8544
TEST COUNT=00

     0 1 2 3 4 5 6 7 8 9 A B C D E F
-----------------------------------------
2      ! " # $ % & ' ( ) * + , - . /
3    0 1 2 3 4 5 6 7 8 9 : ; < = > ?
4    @ A B C D E F G H I J K L M N O
5    P Q R S T U V W X Y Z [ \ ] ^ _
6    ` a b c d e f g h i j k l m n o
7    p q r s t u v w x y z { | } ~

WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
8888888888888888888888888888888888888888
```

Figure 12-5. Typical Printer Self Test Printout

12.2 PRINTER ASSEMBLY INTERFACE DESCRIPTION

The printer assembly connects to the AIM 65/40 through connector J4 and a 4-inch 40-conductor ribbon cable. Table 12-5 lists the printer assembly interface signals and pin assignments. The connector J4 pin locations are shown in Figure 12-5 while Table 12-6 defines the interface signals.

The printer assembly receives data from the SBC module in a handshake manner to ensure proper data transmission between the two modules. Figure 12-6 shows the interface waveforms and Table 12-7 specifies the interface timing.

The $\overline{\text{STROBE}}$ is normally high from the SBC module indicating no data transfer. $\overline{\text{STROBE}}$ is set low after a character has been placed on the data lines by the SBC module and has stabilized. This generates an $\overline{\text{IRQ}}$ interrupt in the display controller. $\overline{\text{STROBE}}$ is held low for at least 50 ns then is set back high. The printer assembly reads the data within 50 us of $\overline{\text{STROBE}}$ going low.

The acknowledge ($\overline{\text{ACK}}$) line from the display assembly is normally high. Upon processing completion of the new character or command, $\overline{\text{ACK}}$ is pulsed low for about 7 us. This processing time prior to $\overline{\text{ACK}}$ low is typically 25 - 300 us. Note that a new character may be placed on the data lines as early as 25 us after the $\overline{\text{STROBE}}$ goes high, however a new $\overline{\text{STROBE}}$ should not be sent until the previous character has been acknowledged.

12.3 PRINTER ASSEMBLY FUNCTIONAL DESCRIPTION

The printer assembly consists of four major functions:

- o Controller
- o Printer Drivers
- o Printer Mechanism
- o Power Supply

Table 12-5. Printer Connector J4 Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	+5V	+5 Vdc	
2	NC		
3	NC		
5	NC		
7	NC		
9	NC		
11	NC		
13	BUSY	Busy	O
15	$\overline{\text{PAPER FEED}}$	Paper Feed	O
17	$\overline{\text{RES}}$	Reset	O
19	$\overline{\text{STROBE}}$	Strobe	O
21	Data 7	Data Line 7	I/O
23	Data 6	Data Line 6	I/O
25	Data 5	Data Line 5	I/O
27	Data 4	Data Line 4	I/O
29	Data 3	Data Line 3	I/O
31	Data 2	Data Line 2	I/O
33	Data 1	Data Line 1	I/O
35	Data Ø	Data Line Ø	I/O
37	NC		
39	$\overline{\text{ACK}}$	Acknowledge	I
4Ø	+5V	+5 Vdc	

NOTE

1. Even numbered pins 4-34 are connector to GND.
2. On-board +5V power can be disconnected from pins 1 and 4Ø by removing the +5V jumper.

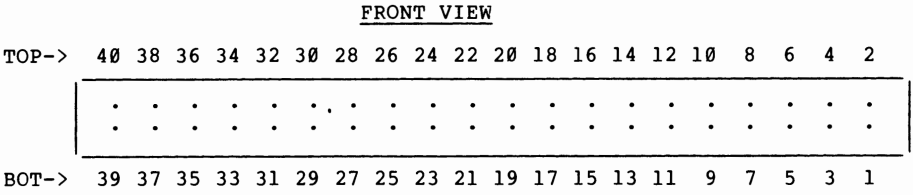

FRONT VIEW
TOP-> 4Ø 38 36 34 32 3Ø 28 26 24 22 2Ø 18 16 14 12 1Ø 8 6 4 2
BOT-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1

Figure 12-6. Printer Connector J4 Pin Locations

Table 12-6. Printer Connector J4 Signal Definition

Signal Mnemonic	Signal Name and Description
+5V	+5Vdc Supplies printer module logic power when the +5V jumper is installed on the module.
GND	Power and Signal Ground Signal ground and power return (if the +5V jumper is installed) to the interfacing equipment.
D0-D7	Data Lines Bidirectional data lines between the interfacing equipment and the printer assembly. Normally inputs to the printer, but used as outputs in the Transmit Character Definition (ESC S) command.
$\overline{\text{STROBE}}$	Data Strobe Received by the printer from the interfacing equipment. Normally $\overline{\text{STROBE}}$ is pulsed low by the SBC to indicate that valid data is available on D0-D7. When the printer is in the Transmit Character Definition mode (ESC S), $\overline{\text{STROBE}}$ is pulsed low to acknowledge that the interfacing equipment has received the data on D0-D7 and is ready to accept new data.
$\overline{\text{ACK}}$	Data Acknowledge Generated by the printer module for the interfacing equipment. Normally $\overline{\text{ACK}}$ is pulsed low to acknowledge that the printer has accepted the data on D0-D7 and is ready to accept new data. When the printer is in the Transmit Character Definition mode (ESC S), $\overline{\text{ACK}}$ is pulsed low by the printer to indicate that valid data is available.
BUSY	Printer Busy Generated by the printer module for the interfacing equipment. BUSY is low when the printer is ready to receive data. When $\overline{\text{STROBE}}$ is received, BUSY is set high until an $\overline{\text{ACK}}$ is sent, indicating the printer is not ready to receive new data. BUSY is not always required by the interfacing equipment.

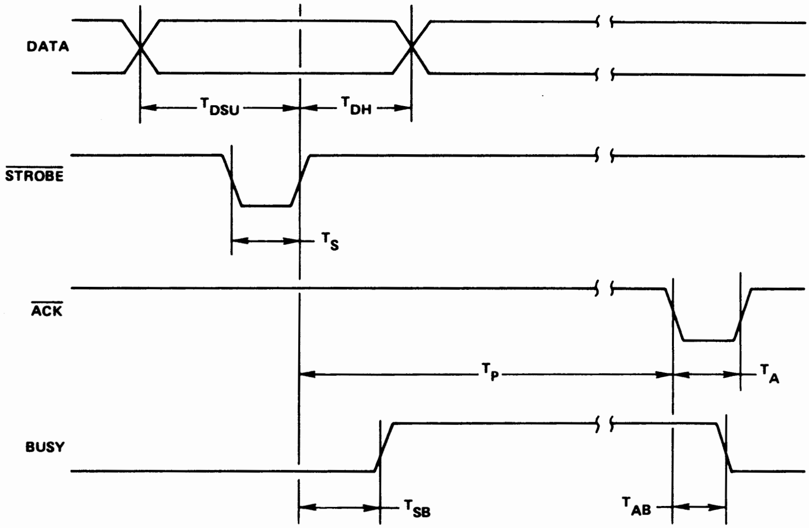


Figure 12-7. Printer Interface Waveforms

Table 12-7. Printer Interface Timing

Parameter	Symbol	Min	Typ	Max	Units
Data Set-Up	T_{DSU}	0	-	-	uS
Data Hold	T_{DH}	25	-	-	uS
Strobe Pulse Width	T_S	50	-	-	nS
Processing Time (non-printing)	T_P	0.13	0.15	12	mS
Processing Time (printing)	T_P	0.25	-	0.5	S
Acknowledge Width	T_A	-	5	-	uS
Strobe-to-Busy	T_{SB}	-	14	25	nS
Acknowledge-to-Busy	T_{AB}	-	20	40	nS

NOTE

tr, tf = 10 to 30 ns.

The block diagram in Figure 12-8 illustrates the interfaces between the functions.

12.3.1 Controller

The R6504 CPU (Z1) is a dedicated microprocessor, with responsibility for communicating with the host processor, setting up the thermal print elements, and controlling the printer motor. The R6504 has an 8K byte address range, into which all RAM, ROM, and I/O are mapped as shown in Figure 12-9.

Timing is supplied by a 4 MHZ crystal controlled oscillator (Y1, Z9) which is divided down by four (Z10) for a symmetrical 1 MHZ clock reference. The RESET circuitry consists of an NE555 Timer (Z8) and associated discrete components, which are configured in the one-shot mode with a 10 millisecond time period. A RESET is initiated automatically at power turn-on and whenever a low-level level is applied to $\overline{\text{RES}}$ on the Printer Interface connector (J4), which is first buffered (Z9, Z11).

The Printer program instructions and character set dot patterns are stored in 4K-byte ROM (Z2) R32L4. This can be replaced with a compatible PROM/ROM device for custom applications.

The R6532 RIOT (Z3) supplies all RAM and most of the I/O required for the printer. The 128 bytes RAM is used for program variables and the processor stack. Sixteen port lines, described in Table 12-8, service the Printer Interface, activate the motor, sense the motor position, and load the Thermal Element shift registers.

The Printer Interface connector (J4) provides access to the Printer Module control signals. The printer control commands and characters are transferred on the lines D0 to D7 lines. $\overline{\text{ACK}}$ and $\overline{\text{STROBE}}$ are used to "handshake" data across the interface with BUSY also available. Figure 12-7 shows the data transfer and protocol and Table 12-7 defines the timing. $\overline{\text{RES}}$ and PAPER FEED are also received from the interfacing equipment.

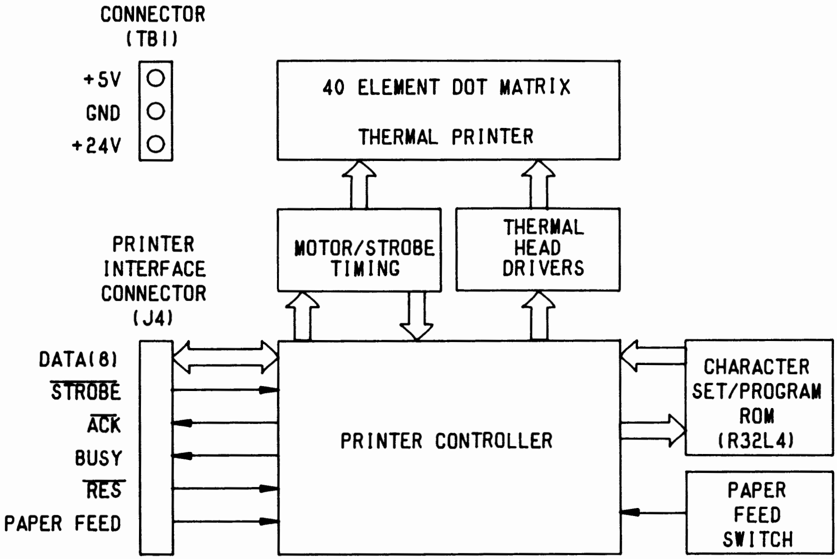


Figure 12-8. Printer Assembly Block Diagram

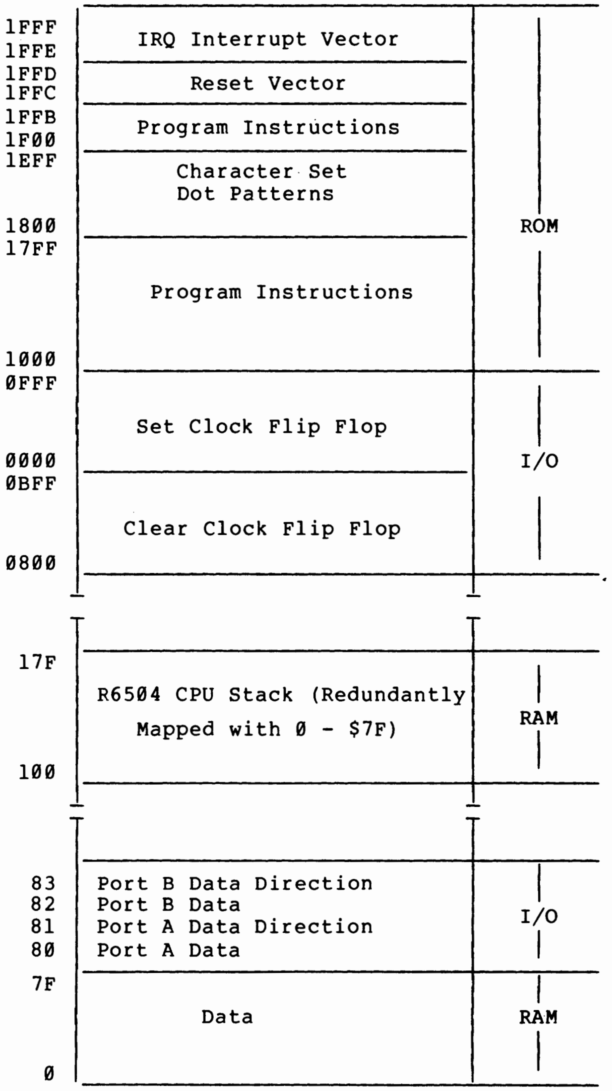


Figure 12-9. Printer Controller Memory Map

Table 12-8. Printer Controller I/O Port Assignment

I/O Port Bits	Function
PAØ - PA7	8 bits of parallel data to/from printer.
PBØ	Serial data to printer driver Z7 (TE31-TE4Ø)
PB1	Output enable to printer drivers Z4-Z7
PB2	Serial data to printer driver Z6 (TE21-T3Ø)
PB3	Motor control for printer
PB4	Serial data to printer driver Z5 (TE11-TE2Ø)
PB5	Acknowledge ($\overline{\text{ACK}}$) to host computer
PB6	Serial data to printer driver Z4 (TE1-TE1Ø)
PB7	Position strobes from printer

The $\overline{\text{STROBE}}$ signal resets the Busy flip-flop (Z13), whose Q output generates an interrupt request ($\overline{\text{IRQ}}$) signal to the R65Ø4 CPU while the inverse output $\overline{\text{Q}}$ is BUSY on the Printer Interface connector. The BUSY flip-flop is set when an $\overline{\text{ACK}}$ is generated by the controller.

The clock for the Thermal Element shift registers is controlled by the three most significant address lines (A1Ø-A12). The Shift Register clock flip-flop (Z12) has A1Ø for a data input. The coincidence of A11, $\overline{\text{A12}}$, and Ø2 creates a clock for the Shift Register clock flip-flop. Thus the Shift Register clock is generated by sequential memory access within specific address ranges.

The paper feed switch (S1) on the printer activates the PAPER FEED signal when the switch is held down. When the PAPER FEED is active or the controller is beginning to print, a motor on is generated (Z14). This enables the Motor Speed regulator (Z11, Q2), which starts the motor. This also controls the Motor/Strobe Timing generation.

When printing is required, the controller starts the motor. After the motor has been turned on, the controller waits for the dot strobe signal from the motor to switch to a low level. The controller then energizes the necessary thermal elements within 2Ø microseconds in order to print odd dots in the correct positions in the required matrices. At the same time, a counter is incremented to indicate that one of the dots per element is printed. The controller then waits for the dot strobe signal to switch high before printing the necessary even dots. The counter is incremented again. This sequence of events is repeated until all of the dots for the character row have been printed.

12.3.2 Printer Drivers

The Thermal Element shift registers (Z4-Z7) convert four serial signals from the RIOT (PBØ, PB2, PB4, PB6) into parallel outputs (1Ø per driver) to drive the 4Ø thermal elements of the printer (TE1 to TE4Ø). Dots are generated by enabling (PB1) these drivers for approximately 1.7 ms while the head is moving and disabling for Ø.6 ms between dots. During this off time new serial data is shifted into the four drivers.

The Motor/Strobe Timing provides feedback to the controller about printhead position. START, SP1, and SP2 come from the motor to indicate the beginning of a line (START), odd dot positions (SP1), and even dot positions (SP2). These signals are combined (Z11, Z12) to create a dot strobe signal which is sensed by the controller.

12.3.4 Printer Mechanism

The printer mechanism (Olivetti PU 184Ø) is connected to the printer through three connectors, two for the thermal head signals (J1, J2) and one for the motor signals (J3). Tables 12-9 through 12-11 list the signals and pin assignments.

Table 12-9. Printer Connector J1 Pin Assignments

Pin	Signal	Pin	Signal
1	Thermal Element Column 20	12	Thermal Element Column 10
2	Thermal Element Column 19	13	Thermal Element Column 9
3	Thermal Element Column 18	14	Thermal Element Column 8
4	Thermal Element Column 17	15	Thermal Element Column 7
5	Thermal Element Column 16	16	Thermal Element Column 6
6	Thermal Element Column 15	17	Thermal Element Column 5
7	Thermal Element Column 14	18	Thermal Element Column 4
8	Thermal Element Column 13	19	Thermal Element Column 3
9	Thermal Element Column 12	20	Thermal Element Column 2
10	Thermal Element Column 11	21	Thermal Element Column 1
11	Thermal Element Common(VTH)		

Table 12-10. Printer Connector J2 Pin Assignments

Pin	Signal	Pin	Signal
1	Thermal Element Column 40	12	Thermal Element Column 30
2	Thermal Element Column 39	13	Thermal Element Column 29
3	Thermal Element Column 38	14	Thermal Element Column 28
4	Thermal Element Column 37	15	Thermal Element Column 27
5	Thermal Element Column 36	16	Thermal Element Column 26
6	Thermal Element Column 35	17	Thermal Element Column 25
7	Thermal Element Column 34	18	Thermal Element Column 24
8	Thermal Element Column 33	19	Thermal Element Column 23
9	Thermal Element Column 32	20	Thermal Element Column 22
10	Thermal Element Column 31	21	Thermal Element Column 21
11	Thermal Element Common(VTH)		

Table 12-11. Printer Connector J3 Pin Assignments

Pin	Signal
1	Motor +
2	Motor -
3	Common
4	Start
5	SP1
6	SP2

The printer mechanism contains two thermal heads, each one with 20 thermal dots for a total of 40 thermal elements. Each thermal element moves in front of the paper back and forth for the length of one character (7 dots). The D.C. motor moves the print heads as well as actuating the paper feed roller.

12.3.4 Power Supply

The Printer requires both +5Vdc and +24Vdc to operate. The +5 Volts and +24 Volts are supplied from an external source to terminal strip TB1. The +5V is also available from the Printer Interface connector (J4) when the +5 Volt jumper wire (W1) is installed. +5 volts is for MOS, LS, and TTL logic operation. +24 volts operates the thermal heads and printer motor and +8V to power the printer drivers is derived from the +24V (Q6). Test point G provides access to signal ground.

The Motor Speed regulator (Q3, Q4, and associated components) sets the motor voltage (Test Point M) which controls the speed. Potentiometer R12 allows this voltage to be adjusted. An increase in this voltage will speed up the motor.

The Dot Intensity regulator (Q5, Q6 and associated components) sets the thermal element voltage (Test Point T), which affects the dot intensity. Potentiometer R17 allows this voltage to be adjusted. An increase in this voltage will darken the printed dots.

SECTION 13

AIM 65/40 40-CHARACTER DISPLAY DESCRIPTION

The AIM 65/40 40-Character Display (A65/40-0400) is an alphanumeric display that interfaces to the AIM 65/40 SBC module over a parallel interface. The assembly contains a Futaba 40-SY-01Z 40-character, 16-segment fluorescent display, one R6504 CPU, one R2316 2K-byte ROM, one R6532 RAM, I/O and Timer (RIOT) and the fluorescent display drivers. The R6504 CPU and resident firmware unburden the host computer (in this case the AIM 65/40 SBC module) from the task of refreshing the display, scrolling characters and other control functions. Figure 13-1 shows the AIM 65/40 40-Column Display assembly. The physical characteristics and power requirements are specified in Tables 13-1 and 13-2, respectively.

The 40-Character display provides a crisp wide display of user commands, data and programs as well as AIM 65/40 System status prompts and messages. Character scrolling and blinking functions expand its use for longer messages and operator alerts. Internal screen oriented editing functions off-load character handling from the SBC module until editing is complete then transmits the data back to the SBC module for line update.

The display operates in response to command and data characters received from the SBC module. Both commands and data are encoded in 8-bit ASCII format (see Appendix F), however some of the commands deviate from standard control commands. Two types of commands -- control and escape -- control the operation of the display. These commands basically tell the display what mode to operate in, how to handle and edit the display buffer and when to display the data.

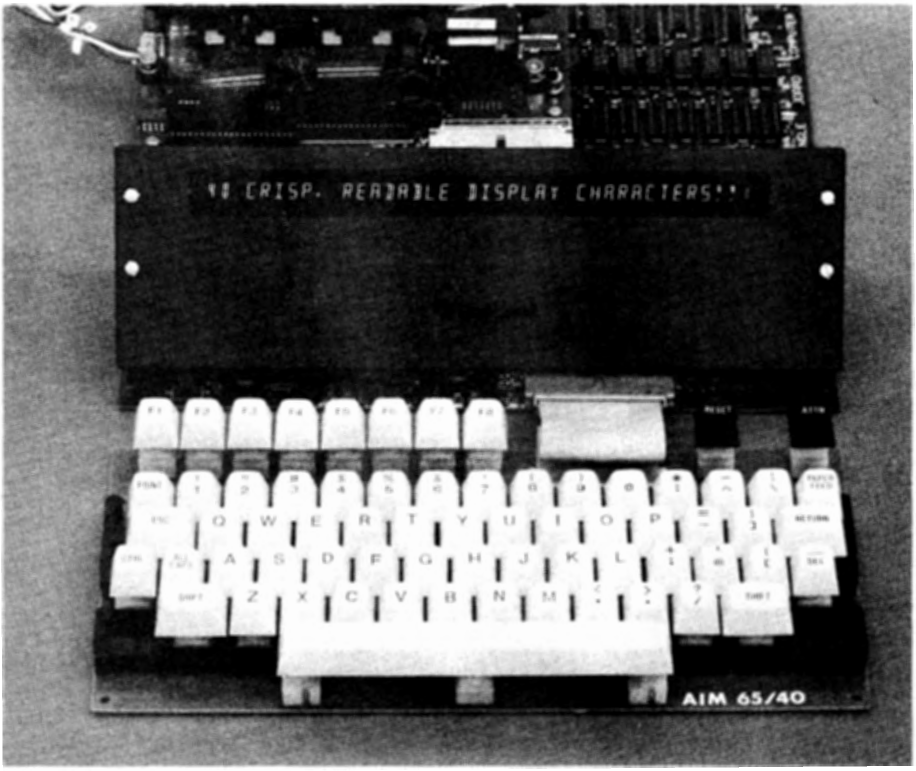


Figure 13-1. AIM 65/40 40-Character Display

Table 13-1. Display Physical Characteristics

Characteristic	Value
Physical	
Width	11.85 in. (3Ø1 mm)
Length	3.6 in. (91 mm)
Height	1.5 in. (38 mm)
Weight	8 oz.
Environment	
Operating Temperature	Ø°C to 7Ø°C
Storage Temperature	-55°C to +8Ø°C
Relative Humidity	Ø% to 95% (without condensation)
Digit Dimensions	
Height	.24 in. (6 mm)
Width	.12 in. (6 mm)
Interface Connection	4Ø-pin 3M #3495-1ØØ2 or equivalent receptacle. Mates with 3M #3417-6Ø4Ø, or equivalent, ribbon cable connector.
Power Connector	Two-post terminal block

NOTE

Dimensions do not include mounting brackets.

Table 13-2. Display Power Requirements

Voltage	Typ.	Max	Unit
+5V ±5% Regulated	.8	1.Ø	A

There are two display modes, normal and graphics. In the normal mode, the complete 96 standard ASCII characters (lower case is indicated by a period associated with the character) as well as 96 special semi-graphic and special characters can be displayed.

In the graphics mode, any character that can be composed with the 16 individual segments per digit can be displayed.

A display self test function tests the controller RAM, displays the controller ROM checksum, displays the complete character set, and illuminates all segments of each digit.

The complete display assembly mounts on the front of the AIM 65/4Ø SBC module by means of two angle brackets. it can also be removed and installed up to six feet away from the SBC module for remote operation with interface connection through a user provided cable.

13.1 DISPLAY ASSEMBLY OPERATION

The 4Ø-Character Display operates in the AIM 65/4Ø system in parallel with the AIM 65/4Ø Graphics Printer. Data sent to the display may also be sent to the printer. The primary difference is that the display shows each character as it is received or deleted whereas the printer normally prints the data only upon receiving a carriage return or when the 4Ø-character line is full.

The commands are the same between the display and printer wherever possible with minimum differences to account for display versus printer output media. It is easiets to think of the display in terms of a 4Ø-character display window within an 8Ø-character display buffer. The contents of the window are always displayed. As data fills the display buffer beyond the first 4Ø characters, the window can be moved right or left to cause the data to appear to be scrolled in and out of the display. A cursor position is always maintained which shows where the next character is to be received or deleted.

Note that some commands processed by the display may appear to do nothing. These commands allow the display to stay in sync with printer unique commands.

13.1.1 Control Commands

The control commands (see Table 13-3) are individual ASCII character codes ($ØØ - $1F) which are issued from the SBC module under operator or program control. The commands are normally issued under program control by I/O ROM subroutines (see Section 6.5) which are called by a system or application program. The commands may also be issued manually from the AIM 65/4Ø or terminal keyboard when the Monitor is in the Direct Peripheral Control mode (CTRL Z, see Section 4.2.11) by holding the CTRL key down while typing a valid command key. The control commands definitions are:

CTRL A - Clear Line

Causes all positions on the line display to be cleared to spaces ($2Ø) and performs a carriage return.

CTRL B - Clear to End of Line

Causes all positions to the right of (and including the cursor) to be cleared with spaces ($2Ø) to the end of the line.

CTRL C - Clear Line

Same as CTRL A.

CTRL D - Clear to End of Line

Same as CTRL B.

Table 13-3. Display Control Commands

ASCII Code	Control Character	Description
00	CTRL @	*
01	CTRL A	Clear Line
02	CTRL B	Clear to End of Line
03	CTRL C	Clear Line
04	CTRL D	Clear to End of Line
05	CTRL E	Clear Line
06	CTRL F	Clear to End of Line
07	CTRL G	*
08	CTRL H	Backspace (Left Arrow)
09	CTRL I	Forespace (Right Arrow)
0A	CTRL J	Warm Reset
0B	CTRL K	Warm Reset
0C	CTRL L	Warm Reset
0D	CTRL M	Carriage Return (Home On Line)
0E	CTRL N	Carriage Return (Home On Line)
0F	CTRL O	Carriage Return (Home On Line)
10	CTRL P	Pass Through Next Character
11	CTRL Q	*
12	CTRL R	*
13	CTRL S	Toggle Insert Character Mode
14	CTRL T	Delete One Character
15	CTRL U	*
16	CTRL V	*
17	CTRL W	Turn Cursor On
18	CTRL X	Turn Cursor Off
19	CTRL Y	Warm Reset
1A	CTRL Z	Cold Reset
1B	CTRL [	Escape Command (ESC)
1C	CTRL \	*
1D	CTRL]	*
1E	CTRL ^	*
1F	CTRL _	*

NOTE:

(*) Characters with no indicated function are acknowledged, but do not otherwise affect the display.

CTRL E - Clear Line

Same as CTRL A.

CTRL F - Clear to End of Line

Same as CTRL B.

CTRL H - Backspace (Left Arrow)

Causes the cursor to move one position to the left on the display. If the cursor is at the left side of the window, but not at the first position of the line, the window will scroll from left to right one position.

CTRL I - Forespace (Right Arrow)

Causes the cursor to move one position to the right on the display.

CTRL J - Warm Reset

Clears the display buffer and modes (i.e., insert mode and graphics mode).

CTRL K - Warm Reset

Same as CTRL J.

CTRL L - Warm Reset

Same as CTRL J.

CTRL M - Carriage Return (Home on Line)

Positions the cursor at the left margin position one and characters 1 through 40 in the display buffer to be displayed in positions 1 through 40.

CTRL N - Carriage Return (Home on Line)

Same as CTRL M.

CTRL O - Carriage Return (Home On Line)

Same as CTRL M.

CTRL P - Pass Through Next Character

Causes the next character received to be stored directly into the display buffer, ignoring any special meaning of control codes or bit 7 attribute. This allows 32 additional characters to be displayed (encoded $ØØ-$1F).

CTRL S - Toggle Insert Character Mode On/Off

When the insert character mode is on, all characters received are inserted into the line just before the cursor position. This causes the characters to the right of (and including) the cursor to move one position to the right. If the buffer becomes full (8Ø characters), no more characters will be inserted (they will be discarded). All control functions (carriage return, delete, etc.) will function normally while in insert mode.

Note that this is different from the replace character mode (the normal mode of operation), where the character under the cursor is replaced by the received character then the cursor advanced one position.

CTRL T - Delete One Character

Causes the character over which the cursor is positioned to be deleted and all characters to the right to move left one position to fill in the vacated space. No action occurs if there are no characters under and to the right of the cursor.

CTRL W - Turn Cursor On

Causes a cursor indicating the position of the next character entry to be displayed. This is the default mode.

CTRL X - Turn Cursor Off

Causes no cursor to be displayed. Used when no user input is required.

CTRL Y - Cold Reset

Clears the display buffer and modes. Performs carriage return. Initializes blink rates, auto scroll rate, cursor character and display environment.

CTRL Z - Warm Reset

Same as CTRL J.

CTRL [- Escape Command (ESC)

Informs the display to accept the next character as an escape command (see Section 12.1.2).

13.1.2 Escape Commands

The escape (ESC) commands are two or more sequential ASCII codes. The first code is the ESC character (ASCII $1B) and the second code is the actual command. Additional characters specify command parameter values as required. Each command is initiated automatically upon receipt of the last character with the exception of the ESC E commands. The set environment commands (ESC E) must be terminated by a carriage return before they take effect. After the characters are processed for each command, the printer looks for another ESC E command character (i.e., A, B, C, R or S) or the carriage return. This allows several ESC E commands to be set up before initiating them.

An escape command can be entered from the AIM 65/40 or a terminal keyboard in the Monitor Direct Peripheral Control mode (CTRL Z, see Section 4.2.11) by first typing and releasing the ESC key then typing the command key and parameter values (if appropriate). Table 13-4 summarizes the display escape commands. The escape commands are defined as follows:

ESC = (Line) (Position) - Set Cursor Position

Moves the cursor to any position in the 80 character display buffer. The ESC = sequence is followed by two characters. The first character is ignored and may be any value. The second character is the cursor position and may range from $20 (position 1) - $6F (position 80).

ESC A - Turn Auto-Scroll On

Causes the display to be scrolled to the right at the rate specified by the ESC E R command

ESC E A (Code) CR - Set Bit 7 Attributes

If bit 7 of the individual display character code is a "0" (see Figure 13-2), the character is steadily displayed as a normal alphanumeric character, i.e. character code $20 - $7F.

Table 13-4. Display Escape Commands

Character Sequence	Hex Codes		
Command & Parameters	Command	Parameters	Function

ESC = (Line) (Pos)	1B 3D	yy zz	Set Cursor Position
			yy = Don't Care
			zz = Cursor Position
			= $2Ø - $6F

ESC A	1B 41		Turn Auto-Scroll On
ESC E A (Code)	1B 45 41	yy	Set Bit 7 Attributes yy = $ØØ = $Ø1 = $Ø2 = $Ø3 (Bits 2-7 =Don't Care)
ESC E B (On) (Off)	1B 45 42	yy zz	Set Character Blink Rate yy = On-Time =$Ø - $FF zz = Off-Time =$Ø - $FF
ESC E C (On) (Off)	1B 45 43	yy zz	Set Cursor Blink Rate yy = On-Time = $2Ø - $7F zz = Off-Time = $2Ø - $7F
ESC E R (Rate)	1B 45 52	yy	Set Auto-scroll Rate yy = $4Ø - $7F
ESC E S (Char)	1B 45 53	yy	Set Cursor Character yy = character code = $41 - $7F
ESC G	1B 47		Enter Graphics Mode
ESC I	1B 49		Toggle Display Inhibit
ESC T	1B 54		Perform Display Self-Test
ESC W (Pos)	1B 57	yy	Set Window Position yy = $2Ø - $47
ESC X (Char)	1B 58	yy	Transmit Display Line yy = L, P or B

If bit 7 of the individual display character code is a "1" (see Figure 13-2), the character is displayed in accordance with the code byte sent with the ESC E A sequence. The code interpretation is:

Bit No.		Display Character Code	
1	Ø	Bit 7 = "1"	Bit 7 = "Ø"
Ø	Ø	Steady Normal	Steady Normal
Ø	1	Blinking Normal	Steady Normal
1	Ø	Steady Semi-Graphic	Steady Normal
1	1	Blinking Semi-Graphic	Steady normal

Bit Ø of the ESC E A code byte controls the blinking function of the character. The character blink rate is controlled by the ESC E C command sequence. Bit 1 controls the normal/semi-graphic character. Normal character codes are interpreted from ASCII code $2Ø - $7F as shown in Figure 13-2. Semi-graphic characters are interpreted from ASCII code $8Ø - $BF as also shown in Figure 13-2. Note that bits 2-7 of the ESC E A code byte are ignored.

ESC E B (On) (Off) CR - Set Character Blink Rate

Set the rate at which characters are blinked. The characters to be blinked must have bit 7 of the character code set to a "1". Note that blinking normal and blinking semi-graphic characters cannot be simultaneously displayed since bit 7 also specifies normal or semi-graphic character. The ESC E B sequence is followed by two characters. The first character is the character on-time. This value may range from $Ø (shortest) to $FF (longest) while the reset value is $26. The second character is the character off-time and may also vary from $Ø to $FF The reset value is $27.

ESC E C (On) (Off) CR - Set Cursor Blink Rate

Sets the rate at which the cursor is blinked. The ESC E C sequence is followed by two characters. The first character is the cursor on-time and may vary from $Ø (shortest) to $FF (longest) while reset value is $26. The second character is the cursor off-time and may also range from $Ø (shortest) to $FF (longest) while the reset value is $27.

ESC E R (Rate) CR - Set Auto-scroll Rate

Sets the rate at which the displayed character automatically scrolls to the left in response to the ESC A command. The rate may vary from $4Ø (fastest) to $7F (slowest). The reset value is $4F.

ESC E S (Char) CR - Set Cursor Character

Sets the character which is displayed as the cursor. The cursor may be any ASCII character ($21 - $7E). The reset value is $7F (DEL character).

ESC G - Enter Graphics Mode

Causes the display to illuminate individual display elements directly from subsequently received characters. (see Section 2.2.4). Line feed (ASCII ØA) resets the mode.

ESC I - Toggle Display Inhibit (On/Off)

Toggles the display inhibit on/off. When display inhibit is toggled on, control and data characters will continue to be processed however, data characters will not be displayed. When display inhibit is toggled off, characters will be displayed in a normal fashion.

ESC T - Perform Display Self Test

Initiates display self test (see Section 13.1.5).

ESC X (L, P or B) - Transmit Display Line

Transmits the contents of the line buffer, the cursor position and the start of the window to the SBC module. The ESC X sequence must be followed by any character, which is ignored. The display then transmits all characters in the buffer to the SBC module. Each character is preceded by a CTRL P ($1Ø) character. The display sends a NULL ($ØØ) to indicate end of the characters in the buffer. A CTRL Z ($1A) and CTRL A ($Ø1) is then sent followed by the cursor position ($ØØ - $4F) and the start of the window ($ØØ - $4F), respectively.

ESC W (Position) - Set Window Position

Sets the starting position of the display window. The ESC W sequence is followed by the value of the starting position. The value may range from $2Ø (position 1) to $6F (position 4Ø).

13.1.3 Normal/Semi-graphics Character Mode

192 standard and special characters may be displayed in the normal mode of operation (see Figure 13-2). These characters are grouped into three categories:

- o 96 Standard characters ($2Ø - $7F)
- o 64 Semi-graphic characters ($8Ø - $BF)
- o 32 Control characters ($ØØ - $1F)

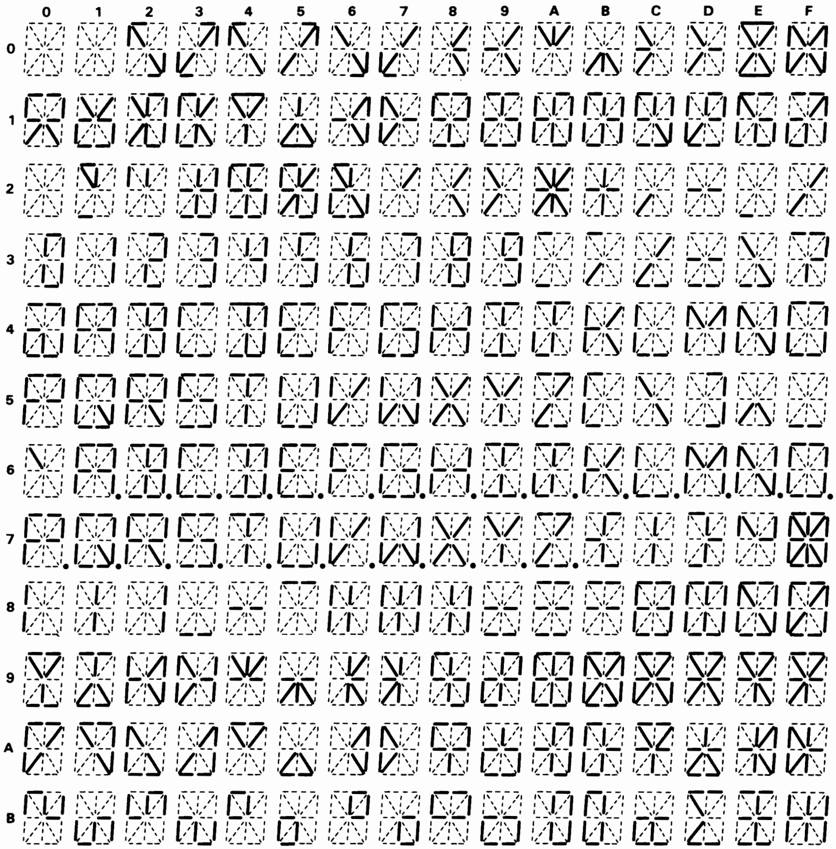


Figure 13-2. Display Normal/Semi-graphics Character Set

a. Standard Character Set (Normal Mode)

The standard 96 alphabetic, numeric and special ASCII characters are encoded from $2Ø through $7F. The lower case alphabetics are represented as upper case with the associated character decimal point illuminated. These characters are displayed when the normal display mode has been selected with the ESC E A command (see Section 13.1.2).

b. Extended Character Set (Semi-graphics Mode)

64 special characters are provided in the semi-graphics mode and are encoded from $8Ø through $BF. These characters are displayed when bit 7 of the character code is set to a "1" and either the steady or blinking semi-graphic character display mode has been selected with the ESC E A command.

Semi-graphic characters are usually generated under program control rather than by the operator since only standard characters (ASCII codes $2Ø-$7E) may be entered from the keyboard. A user defined program can be written to set bit 7 to "1" to convert from normal to semi-graphic characters. The following procedure can also be used to set the bit 7 to "1" and to command the display in to semi-graphics mode. This is handy to see how the semi-graphic characters are displayed.

(1) Enter a string of normal characters corresponding to the desired semi-graphic characters (i.e. bit 7 = "Ø" rather than "1", see Appendix E) into the Editor Text Buffer.

(2) Change bit 7 from "Ø" to "1" for each of the character codes using the Monitor M and "/" commands.

(3) Enter the Direct Peripheral Control mode with the Monitor CTRL Z command.

(4) Command the semi-graphics mode using the ESC E A 2 command sequence.

(5) Press ATTN to return to the Monitor (not RESET since that will cause the printer to reinitialize).

(6) Go to the top of the Editor with the T command and list the characters as desired.

c. Control Character Set

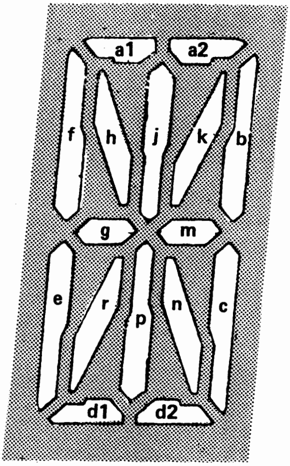
32 additional special characters are available by preceding each data character with CTRL P. The characters are encoded $ØØ - $1F (see Figure 13-2) and are transmitted to the display as a two character sequence, the first character being CTRL P ($1Ø).

13.1.4 Graphics Mode

The graphics mode is selected by the ESC G sequence (see Section 13.1.2). When operating in the graphics mode, the display does not use the character generator. It drives any one or any combination of the 16-segment digit display drivers directly to create the desired graphical representations. Only 4Ø characters can be driven, and there is no scrolling.

Each of the character fonts are individually addressable. Two consecutive bytes are required to drive one digit (see Figure 13-3). The first byte drives segments a1 through f and the second byte drives segments g through r.

The characters $ØØ - $ØF are accepted as control commands (e.g., carriage return, line feed, etc.) during the graphics mode. To use these codes as graphics commands to illuminate digit segments, each of these codes must be preceded by the CTRL P ($15) character similar to sending control character in the normal/semi-graphics mode (see Section 13.1.3).



CHARACTER SEGMENTS ILLUMINATED

	7	6	5	4	3	2	1	0
BYTE 1	f	e	d1	d2	c	b	a2	a1
BYTE 2	r	p	n	m	k	j	h	g

Figure 13-3. Display Graphics Mode Character Segment Addressing

13.1.5 Display Self Test

The display self test performs a functional test of the controller, segment drivers, character drivers. It also prints out characters for a visual check of the character segments.

The display self test may be initiated by sending the display an ESC T command (see Section 13.1.2) or by use of the self test jumper on the display module.

a. Using the ESC T Command

To use the ESC T command, the display must be connected to the SBC module, the self test jumper installed in the N position and power applied.

(1) Issue the ESC T command under program or keyboard control. The test will be immediately performed.

(2) Upon test completion, the display will return to the normal character and command input mode.

(3) Repeat the test as many times as required by re-sending the ESC T command.

b. Using the Self Test Jumper (Display Disconnected)

The display self test can be run independently of the interfacing system as follows:

(1) Turn display power off.

(2) Install the self test jumper in the S (self test) position.

(3) Turn display power on. The test will be immediately performed.

(4) Turn display power off.

(5) Return the self test jumper to the N (Normal) position.

c. Using the Self Test Jumper (Display Connected)

The display self test can be run using the self test jumper while connected to the SBC module and operating as follows:

(1) Install the self test jumper in the S position.

(2) Press RESET. The self test will be immediately performed.

(3) Return the self test jumper to the N position.

(5) Press RESET. The display will return to normal character and command processing.

d. Display Self Test Operation

The test operates as follows:

(1) The 128 bytes of RAM are tested. The results of the Test are then displayed as:

```
    RAM OK   (test passed)
or  RAM FAIL (test failed)
```

(2) The ROM checksum is computed and displayed for visual evaluation, e.g.:

```
    ROM=D57D
```

e. The cursor character is displayed in each display position from left to right. The periods associated with each character are all displayed. Press any key to stop the display at any time. Press a key again to resume display sequencing.

f. The entire normal and semi-graphics character set is displayed between cursor characters.

13.2 DISPLAY ASSEMBLY INTERFACE DESCRIPTION

The display assembly connects to the AIM 65/40 SBC module through connector J1 and a 4-inch 40-conductor ribbon cable. Table 13-5 lists the display assembly interface signals and pin assignments. The connector J1 pin locations are shown in Figure 13-4 while Table 13-6 defines the interface signals.

The display assembly receives data from the SBC module in a handshake manner to ensure proper data transmission between the two modules. Figure 13-4 shows the interface waveforms and Table 13-7 specifies the interface timing.

Table 13-5. Display Connector J1 Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	+5V	+5 Vdc	
2	NC		
3	NC		
5	NC		
7	NC		
9	NC		
11	NC		
13	BUSY	Busy	O
15	NC		O
17	$\overline{\text{RES}}$	Reset	O
19	$\overline{\text{STROBE}}$	Strobe	O
21	Data 7	Data Line 7	I/O
23	Data 6	Data Line 6	I/O
25	Data 5	Data Line 5	I/O
27	Data 4	Data Line 4	I/O
29	Data 3	Data Line 3	I/O
31	Data 2	Data Line 2	I/O
33	Data 1	Data Line 1	I/O
35	Data 0	Data Line 0	I/O
37	NC		
39	$\overline{\text{ACK}}$	Acknowledge	I
40	+5V	+5 Vdc	

NOTE

1. Even numbered pins 4-34 are connector to GND.
2. On-board +5V power can be disconnected from pins 1 and 40 by removing the +5V jumper.

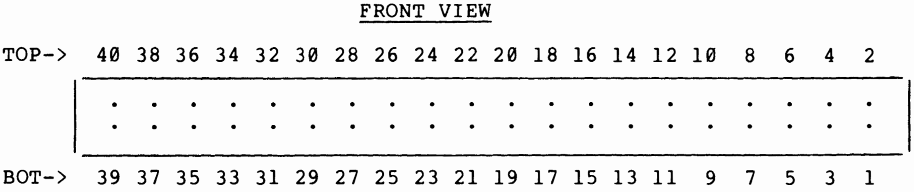

FRONT VIEW
TOP-> 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2
BOT-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1

Figure 13-4. Display Connector J1 Pin Locations

Table 13-6. Display Connector J1 Signal Definitions

Signal Mnemonic	Signal Name and Description
+5V	+5Vdc Supplies display module logic power when the +5V jumper is installed on the module.
GND	Power and Signal Ground Signal ground and power return (if the +5V jumper is installed) to the interfacing equipment.
DØ-D7	Data Lines Bidirectional data lines between the interfacing equipment and the display assembly. Normally inputs to the display, but used as outputs in the Transmit Display Line (ESC X) command.
$\overline{\text{STROBE}}$	Data Strobe Received by the display from the interfacing equipment. Normally $\overline{\text{STROBE}}$ is pulsed low by the SBC to indicate that valid data is available on DØ-D7. When the display is in the Transmit Display Line mode (ESC X), $\overline{\text{STROBE}}$ is pulsed low to acknowledge that the interfacing equipment has received the data on DØ-D7 and is ready to accept new data.
$\overline{\text{ACK}}$	Data Acknowledge Generated by the display for the interfacing equipment. Normally $\overline{\text{ACK}}$ is pulsed low to acknowledge that the display has accepted the data on DØ-D7 and is ready to accept new data. When the display is in the Transmit Display Line mode (ESC X), $\overline{\text{ACK}}$ is pulsed low by the display to indicate that valid data is available.
BUSY	Printer Busy Generated by the display module for the interfacing equipment. BUSY is low when the display is ready to receive data. When $\overline{\text{STROBE}}$ is received, BUSY is set high, remaining high until an $\overline{\text{ACK}}$ is sent, indicating the display is ready to receive new data. BUSY is not always required by the interfacing equipment.

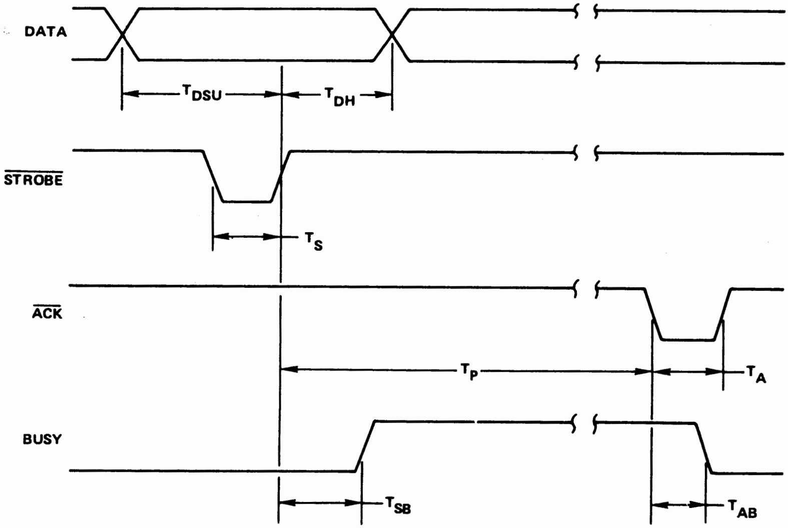


Figure 13-5. Display Interface Waveforms

Table 13-7. Display Interface Timing

Parameter	Symbol	Min	Typ	Max	Units
Data Set-Up	T_{DSU}	Ø	-	-	uS
Data Hold	T_{DH}	25	-	-	uS
Strobe Pulse Width	T_S	5Ø	-	-	nS
Processing Time	T_P	-	5ØØ	-	uS
Acknowledge Width	T_A	-	5	-	uS
Strobe-to-Busy	T_{SB}	-	14	25	nS
Acknowledge-to-Busy	T_{AB}	-	2Ø	4Ø	nS

NOTE

tr, tf = 1Ø to 3Ø ns.

The $\overline{\text{STROBE}}$ is normally high from the SBC module indicating no data transfer. $\overline{\text{STROBE}}$ is pulsed low before or after a character has been placed on the data lines by the SBC module. After the data lines have stabilized, $\overline{\text{STROBE}}$ is pulsed back high to indicate data available. This generates an interrupt in the display controller. The display assembly reads the data within 50 us of $\overline{\text{STROBE}}$ going high.

The acknowledge ($\overline{\text{ACK}}$) line from the display assembly is normally high. Upon receipt and processing of the new data, $\overline{\text{ACK}}$ is pulsed low for about 7 us. This processing time prior to $\overline{\text{ACK}}$ low is typically 25 - 300 us.

Note that a new character may be placed on the data lines as early as 25 us after the $\overline{\text{STROBE}}$ goes high, however a new $\overline{\text{STROBE}}$ should not be sent until the prior character has been acknowledged.

13.3 DISPLAY ASSEMBLY FUNCTIONAL DESCRIPTION

The Display Assembly consists of four major functions:

- o Controller
- o Character and Segment Drivers
- o Fluorescent Display
- o DC/DC Power Converter

The block diagram in Figure 13-6 illustrates the interfaces between the functions.

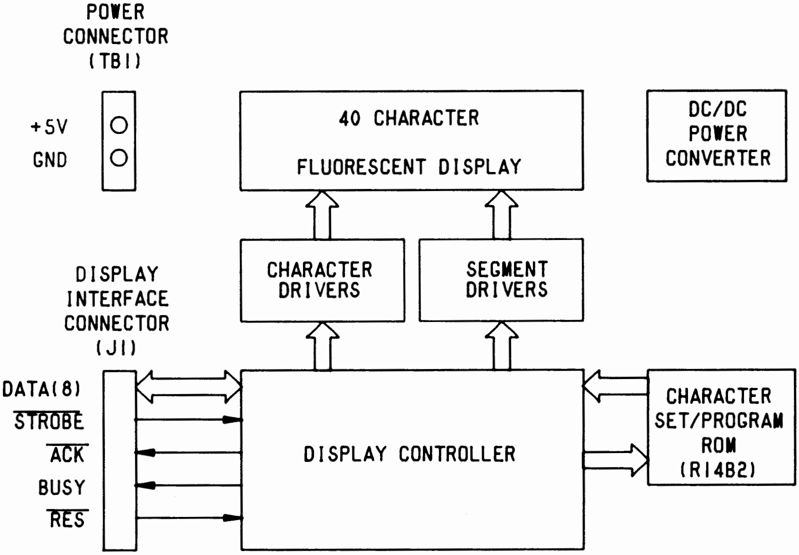


Figure 13-6. Display Assembly Block Diagram

13.3.1 Controller

The R6504 CPU (Z11) is a dedicated microprocessor, with responsibility for communicating with the host processor, setting up the segment drivers, and strobing the character drivers. The R6504 has an 8K byte address range, into which all RAM, ROM, and I/O are mapped as shown in Figure 13-7.

Timing is supplied by a TTL oscillator (Z14, C5) which is divided down by four (Z7) for a symmetrical clock reference, typically 150 Khz. The RESET circuitry consists of an NE555 Timer (Z13) and associated discrete components, which are configured in the one-shot mode with a 10 millisecond time period. A RESET is initiated automatically at power turn-on and whenever a low-level is applied to $\overline{\text{RES}}$ on the Display Interface connector (J1).

The Display Program and Character Set are stored in a 2K byte ROM (Z12) R14B2. This can be replaced with a compatible PROM/ROM device for custom applications.

The R6532 RIOT (Z10) supplies all RAM and much of the I/O required for the display. The 128 bytes RAM is used for program variables and the processor stack. Sixteen port lines service the Display Interface, load the character and segment driver shift registers, and blank the character and segment drives.

The Display Interface connector (J1) provides access to the Display module control signals. The display control commands and characters are transferred on the lines D0 to D7. $\overline{\text{ACK}}$ and $\overline{\text{STROBE}}$ are used to "handshake" data across the interface, with BUSY also available with wire jumper W2 installed. Figure 13-5 shows the data transfer and protocol and Table 13-7 defines the timing. $\overline{\text{RES}}$ and $\overline{\text{PAPERFEED}}$ are also received from the interfacing equipment.

The $\overline{\text{STROBE}}$ signal resets the Busy flip-flop (Z8), whose Q output generates and interrupt request ($\overline{\text{IRQ}}$) signal to the R65Ø4 CPU while the inverse output $\overline{\text{Q}}$ is Busy on the Printer Interface connector. The Busy flip-flop is set when an $\overline{\text{ACK}}$ is generated by the controller.

The clock for the Character Driver shift registersis controlled by the three most significant address lines (A1Ø-A12). The Shift Register clock flip-flop (Z8) has A1Ø for a data input. The coincident of A11, and Ø2 (Z9) creates a clock for the Shift Register clock flip-flop. Thus the Shift Register clock is generated by sequential memory access within specific address ranges.

13.3.2 Display Drivers

The Segment Drivers (Z3, Z4) consist of two 1Ø-bit shift registers which convert two serial signals from the RIOT (PBØ, PB4) into parallel outputs (9 used per driver). These high voltage outputs (about 5ØV) drive the segment amodes on the display tube.

The Character Drivers (Z1, Z2, Z5, Z6) consist of four 1Ø-bit sift registers which are cascaded to form a single 4Ø-bit shift register. This shift register is loaded serially by the RIOT (PB1), with the 4Ø parallel high voltage outputs driving the character grids on the display tube.

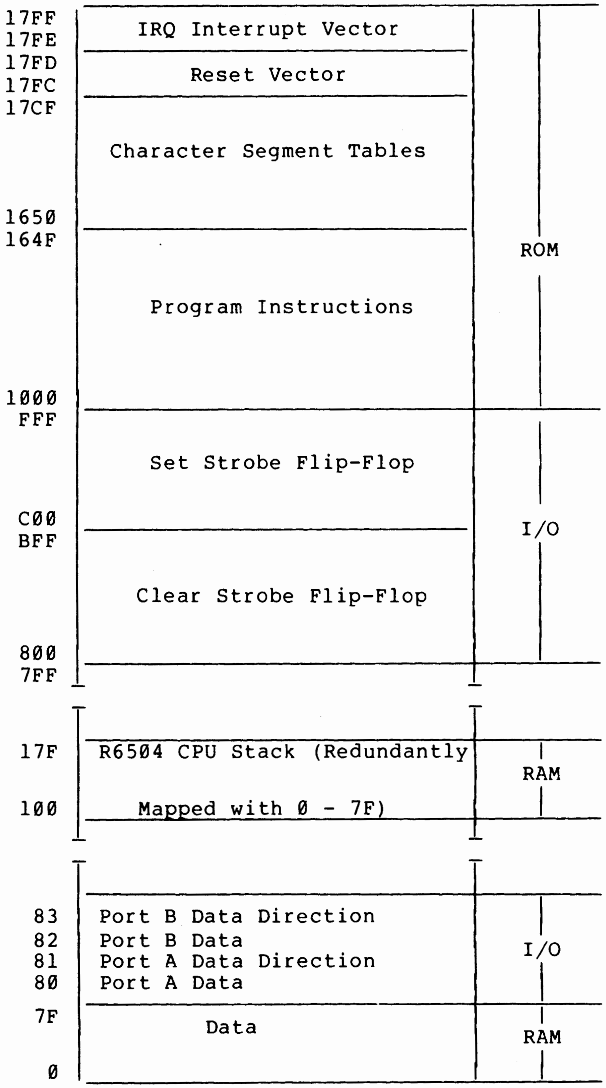
The character shift register (4Ø-bit) input data line is set to a logic one at the beginning of each refresh cycle. All following input data is logic zero. This single one-bit is thus shifted through the register, resulting in each character is being selected in succession. The data is strobed into the segment registers between characters while the display is being blanked.

13.3.3 Display

The display device is a Futcha 4Ø-SY-Ø1Z alphanumeric fluorescent display tube. The display tube consists of 4Ø characters each of which has 16-segments a decimal point, and a comma. The vacuum fluorescent display is enclosed in a glass envelope. The display segments produce a blue-green color when a grid voltage (character pins) and an anode voltage (segment pins) both occur.

13.3.4 Power Converter

The Display module requires a single +5 volt power source. The +5 volts is supplied from an external source to terminal block TB1, or from the Display Interface connector (J1) when jumper wire W1 is installed. The +5 volts is for the MOS, LS, and TTL logic, as well as the DC-DC power converter. The DC-DC power converter (PS1) is a non-adjustable, encapsulated module which converts the +5 volts to 7 Vac for the flourescent display tube filaments and +48 VDC for the character and segment drivers.



Address	Contents	
17FF 17FE	IRQ Interrupt Vector	ROM
17FD 17FC	Reset Vector	
17CF 1650	Character Segment Tables	
164F 1000	Program Instructions	
FFF C00	Set Strobe Flip-Flop	I/O
BFF 800	Clear Strobe Flip-Flop	
7FF		
17F 100	R6504 CPU Stack (Redundantly Mapped with Ø - 7F)	RAM
83	Port B Data Direction	I/O
82	Port B Data	
81	Port A Data Direction	
8Ø	Port A Data	
7F Ø	Data	RAM

Figure 13-7. Display Controller Memory Map

Table 13-9. Display Controller I/O Port Assignment

I/O Port Bits	Function
PAØ - PA7	8 bits of parallel data to/from display
PBØ	Serial data to segment drivers
PB1	Serial data to character drivers
PB2	Clock for character drivers (positive edge trigger)
PB3	Blanking to segment and character drivers (logic 1 = blank)
PB4	Serial data to segment drivers
PB5	Acknowledge ($\overline{\text{ACK}}$) to host computer
PB6	Unused
PB7	Unused

SECTION 14

AIM 65/4Ø STANDARD KEYBOARD DESCRIPTION

The AIM 65/4Ø Standard Keyboard (A65/4Ø-Ø2ØØ) is a terminal-style alphanumeric keyboard that interfaces to the AIM 65/4Ø SBC module through a parallel interface. There are 55 main keys including a locking ALL CAPS key, which allow entry of 128 ASCII printable and control characters. A separate horizontal row of eight function keys support user-defined functions (as well as screen oriented functions in the AIM 65/4Ø EDITOR). Distinctly marked RESET and ATTN keys, physically separated from each other and the other keys, allow easy keyboard control of initialization and user-defined interrupt processing. Figure 14-1 shows the AIM 65/4Ø Keyboard. Table 14-1 lists the physical and electrical characteristics.

14.1 KEYBOARD ASSEMBLY INTERFACE DESCRIPTION

The Keyboard assembly connects to the AIM 65/4Ø SBC module through an interfacing 4Ø-conductor ribbon cable which connects to Keyboard connector J1. The interface signals at connector J1 are listed in Table 14-2, with the pin locations shown in Figure 14-2.

14.2 KEYBOARD ASSEMBLY FUNCTIONAL DESCRIPTION

The keyboard has a complement of 64 keys connected within a 9 row by 8 column switch matrix (see Figure 14-3). A block diagram of the keyboard is shown in Figure 14-3. A schematic of the keyboard is shown in Figure 14-4.

One side of each key switch is connected to row (strobe) line while the other side is connected to column (return) line. Depression of a key closes the switch which completes the circuit between the input strobe and the output return lines.

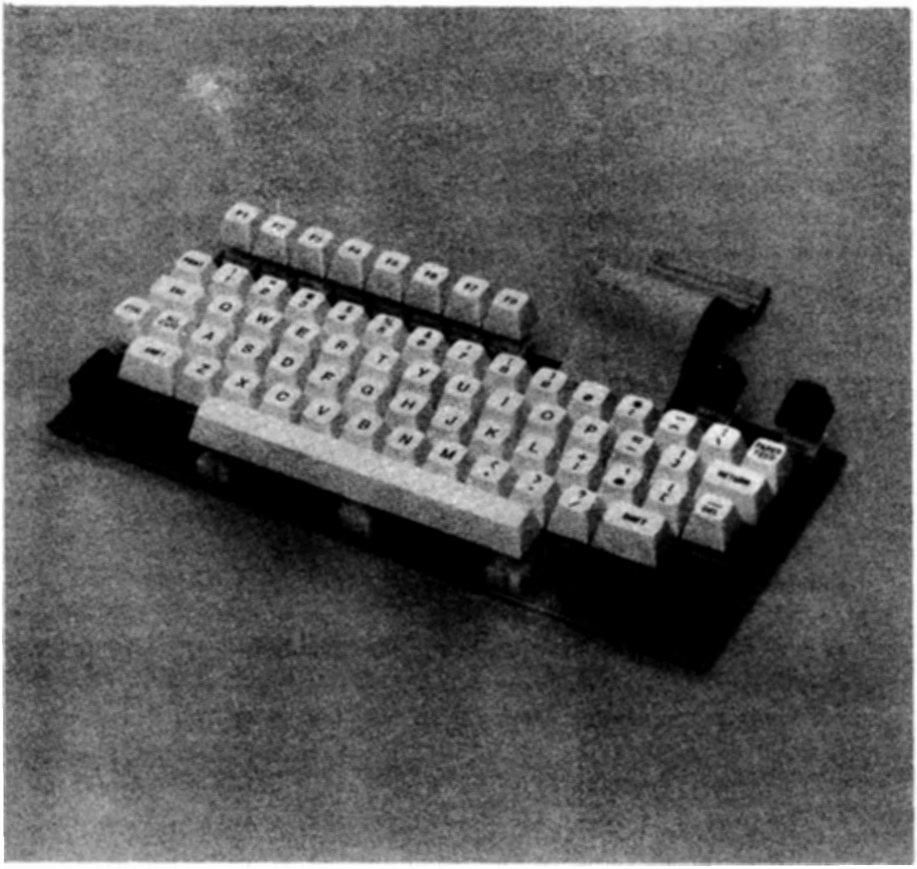


Figure 14-1. AIM 65/40 Keyboard

Table 14-1. Keyboard Physical Characteristics

Characteristic	Value
Physical	
Length	11.85 in. (3Ø1 mm)
Width	5.25 in. (133 mm)
Height	1.25 in. (32 mm)
Environment	
Operating Temperature	Ø°C to 5Ø°C
Storage Temperature	-4Ø°C to 55°C
Relative Humidity	Ø% to 95% (without condensation)
Interface Connection	4Ø-pin 3M #3495-1ØØ2, or equivalent, receptacle. Mates with 3M #3417-6Ø4Ø, or equivalent ribbon cable connector.
Key Cap Colors	

RESET and ATTN	Body = Black (T45ØØ)
	Legend = Fog (T25ØØ)

All others	Body = Fog (T25ØØ)
	Legend = Black (T45ØØ)
	(Colors per Borg Warner color chart "spectrum 2ØØ")

Table 14-2. Keyboard Connector J1 Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output (Typical)
4Ø	RESET	Reset Switch	O
39		NC	
38	ATTN	Attention Switch	O
36	MSB7	Matrix Strobe 7	I
34	MSB6	Matrix Strobe 6	I
32	MSB5	Matrix Strobe 5	I
3Ø	MSB4	Matrix Strobe 4	I
28	MSB3	Matrix Strobe 3	I
26	MSB2	Matrix Strobe 2	I
24	MSB1	Matrix Strobe 1	I
22	MSBØ	Matrix Strobe Ø	I
2Ø	MRT7	Matrix Return 7	O
18	MRT6	Matrix Return 6	O
16	MRT5	Matrix Return 5	O
14	MRT4	Matrix Return 4	O
12	MRT3	Matrix Return 3	O
1Ø	MRT2	Matrix Return 2	O
8	MRT1	Matrix Return 1	O
7	RESET RTN	RESET Switch Return	I
6	MRTØ	Matrix Return Ø	O
5	ATTN RTN	ATTN Switch Return	I
4	MSB8	Matrix Return 8	O
3	PAPER FEED RTN	Paper Feed Switch Return	I
2	PAPER FEED	Paper Feed Switch	O
1		NC	

NOTES

1. Odd numbered pins 9-37 are connected together.
2. The pin number assignments are reversed from the AIM 65/4Ø SBC module connector to provide a one to one signal routing through a 4Ø-conductor ribbon cable.

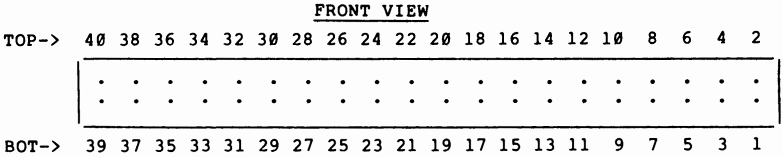

FRONT VIEW
TOP-> 4Ø 38 36 34 32 3Ø 28 26 24 22 2Ø 18 16 14 12 1Ø 8 6 4 2
BOT-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1

Figure 14-2. Keyboard Connector J1 Pin Locations

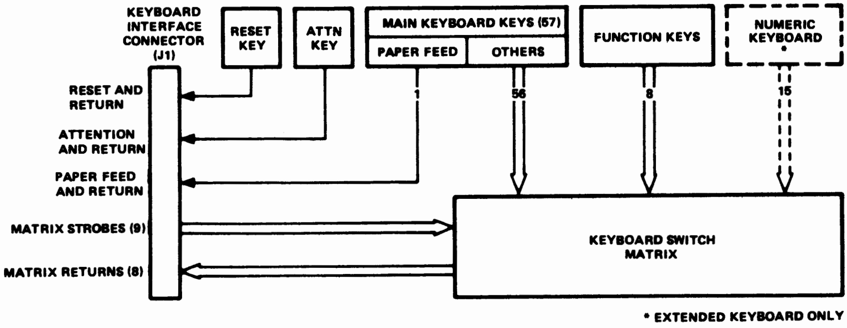
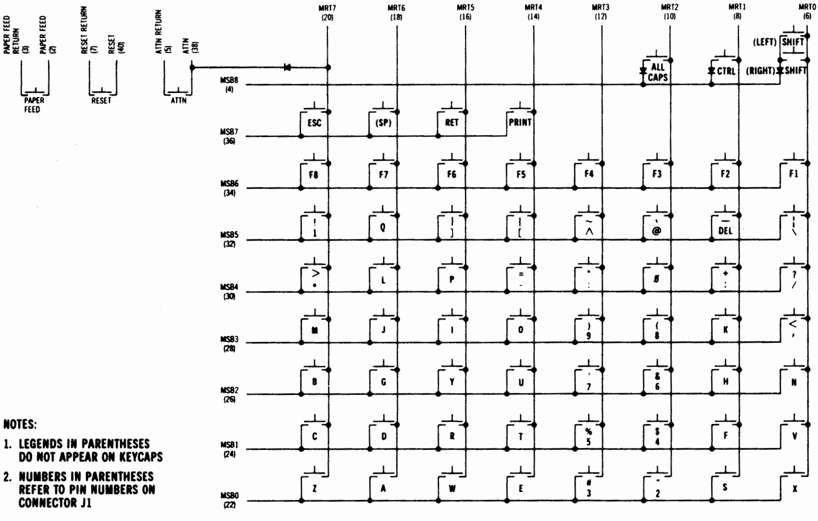


Figure 14-3. Standard Keyboard Block Diagram



NOTES:

1. LEGENDS IN PARENTHESES DO NOT APPEAR ON KEYCAPS
2. NUMBERS IN PARENTHESES REFER TO PIN NUMBERS ON CONNECTOR J1

Figure 14-4. Keyboard Schematic

Three additional switches (RESET, ATTN, PAPER FEED) are outside the matrix.

The interfacing software can determine which switch in the matrix is closed by scanning the keyboard. A row is scanned by issuing an output on only one of the nine strobe lines (MSBØ-MSB8) and sampling the eight return lines (MRTØ-MRT7). A low bit in this return line word indicates the column of a depressed key. This is repeated for each of the nine strobe lines to scan the entire keyboard. Software can then determine which key was pressed by correlating the sampled return line data with key positions.

The RESET, PAPER FEED and ATTN keys, when pressed, close a circuit to the respective return line. In addition, depression of ATTN can be detected through a matrix return line (MRT7).

The actual keyboard input software can be fairly complex when you consider such factors as upper/lower case, control characters, function keys, key debounce, multiple key depression, and keyboard driven interrupt processing. Refer to the I/O ROM assembly listing for a detailed example of keyboard input processing.

SECTION 15

AIM 65/40 MONITOR/EDITOR FIRMWARE DESCRIPTION

The AIM 65/40 Monitor/Editor is designed to operate with the AIM 65/40 SBC, printer and display modules. It is structured to be compatible with the I/O ROM and may be considered to be an application program by the I/O ROM. It uses the I/O ROM constants and variables described in Section 6.1, initializes through the auto-start interface as detailed in Section 6.2, establishes NMI and IRQ interrupt linkage to I/O ROM interrupt handlers as explained in Section 6.3 and 6.4, and uses I/O ROM subroutines described in Section 6.5.

This section describes the Monitor/Editor design structure and interface with the I/O ROM interrupt processing as well as user oriented constants, variables and subroutines.

15.1 MEMORY MAP

The Monitor and Editor program is located in $A000-$BFFF as shown in Figure 2-5. Figure 15-1 shows a memory map of the Monitor/Editor. Figure 6-2 locates the Monitor constants and variable along with the I/O parameters. Refer to the Monitor/Editor assembly listing (document no. 2965092) for the detail assembly language instruction. Tables 15-1 and 15-2 detail the monitor constants and variables respectively.

15.1.1 Monitor Constants

These constants (see Table 15-2) are initialized to the stated values upon cold RESET and may be altered under operator or program control. These constant's may be further defined as follows:

BFFB BC76	Mnemonic Entry
BC75 B979	Disassembler
B978 B7E2	Editor Subroutines
B7E1 B26F	Editor Command Functions
B26D B11D	Text Editor Entry and Command Dispatcher
B11C ADFB	Monitor Subroutines
ADFA A445	Monitor Command Functions
A444 A3ØB	Command Dispatcher
A3ØA A273	Memory I/O Routines
A274 A22C	Break Instruction
A22B A16E	Attention and Single Step
A16D AØ3D	Monitor Entry
AØ3C AØ73	Messages
AØ72 AØØØ	Auto-Start Subroutine and Constants

Figure 15-1. Monitor/Editor Memory Map

Table 15-1. Monitor Constants

Address	Label	No. Bytes	Cold Reset Value	Parameter
Ø25Ø	F1KEY	2	A322	Function Key F1 Vector
Ø252	F2KEY	2	A322	Function Key F2 Vector
Ø254	F3KEY	2	A322	Function Key F3 Vector
Ø256	F4KEY	2	A322	Function Key F4 Vector
Ø258	F5KEY	2	A322	Function Key F5 Vector
Ø25A	F6KEY	2	A322	Function Key F6 Vector
Ø25C	F7KEY	2	A322	Function Key F7 Vector
Ø25E	F8KEY	2	A322	Function Key F8 Vector
Ø26Ø	TEXT	2	2ØØØ	Start of Edit Buffer
Ø262	END	2	3FFF	End of Edit Buffer
Ø264	STSAVE	2	18ØØ	Start of Symbol Table
Ø266	KTBLSZ	2	1FFF	End of Symbol Table
Ø268	NDSYM	2	ØØØØ	Current Number of Symbols
Ø26A	ARECSZ	1	1E	Size of ASCII Object Record

Table 15-2. Monitor Variables

Address	Label	No. Bytes	Cold Reset Value	Parameter
Ø26B	UCMDIV	2	A31E	Monitor Command Vector
Ø26D	EDCIV	2	B1F7	Editor Command Vector
Ø26F	HISTP	1	ØØ	Y Reg for Soft Trace
Ø27Ø	KEYCMD	2	ØØØØ	Stack Pointers
Ø272	DISASW	1	ØØ	Disassembly Control Flag

F1KEY - through F8KEY
Vectors pointing to the start address of a user defined subroutine. Initialized to point to the Monitor invalid command handling (COMERR).

TEXT - First address of the Editor Text Buffer. Initialized to $2ØØØ. Can be changed under operator control by entering an address in response to the FROM= prompt in the Monitor Enter Editor (E command) or Recover Text Buffer (C command) functions.

END - Last address of the Editor Text Buffer. Initialized to $3FFF. Can be changed under operator control by entering an address in response to the TO= prompt in the Monitor Enter Editor (E command) or Recover Editor (C command) functions.

STSAVE - First address of the symbol table. Initialized to $18ØØ. Usually changed under operator control in optional software, e.g. assembler, to handle more or less symbols or to relocate in memory. Can also be changed using the Monitor. M and "/" commands.

KTBLSZ - Last address of the symbol table. Initialized to $3FFF. Usually handled in a similar manner as STSAVE.

NOSYM - Number of symbols in the symbol table. Incremented by one for each symbol entered with the ";" command or loaded from optional software.

ARECSZ - Number of characters in an ASCII object record dumped to audio tape. Initialized to 3Ø ($1E). Can be changed using the M and "/" commands.

15.1.2 Monitor Variables

These variables (see Table 15-2) change value during Monitor operation and are generally not accessed or altered by the user.

15.2 PROGRAM STRUCTURE

Monitor linkage, constants, variables and interrupt linkage are initialized during Monitor reset processing performed during Auto-Start. Return is to the I/O ROM to allow optional software and application programs to be initialized.

An NMI Interrupt handler processes the ATTN key and single step instruction execution to disassemble and display one instruction.

An IRQ Interrupt handler processes the BRK instruction and also disassembles and displays one instruction through the NMI handling.

The command processing gets a key from the keyboard, jumps indirect through variable UCMDIV to Monitor decoding to allow user access to the key. The Monitor then jumps indirect to the command function processing. The function terminates with an RTS to return to the command processor, which then repeats the process.

A flowchart of the Monitor Processing is shown in Figure 15-2.

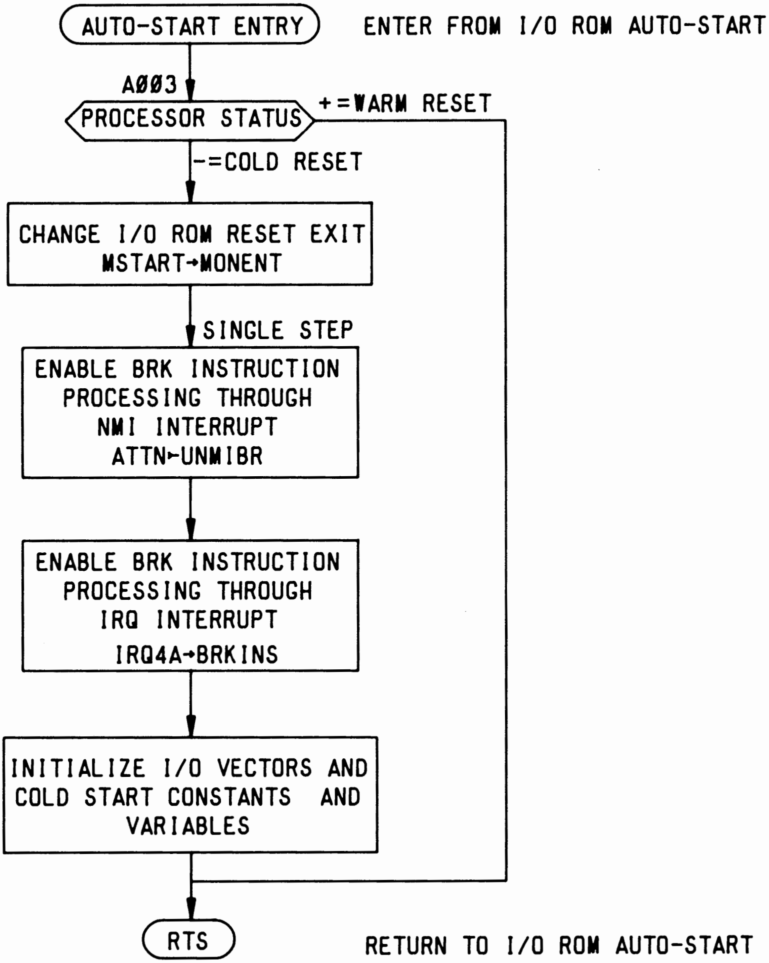


Figure 15-2. Monitor Processing Flowchart

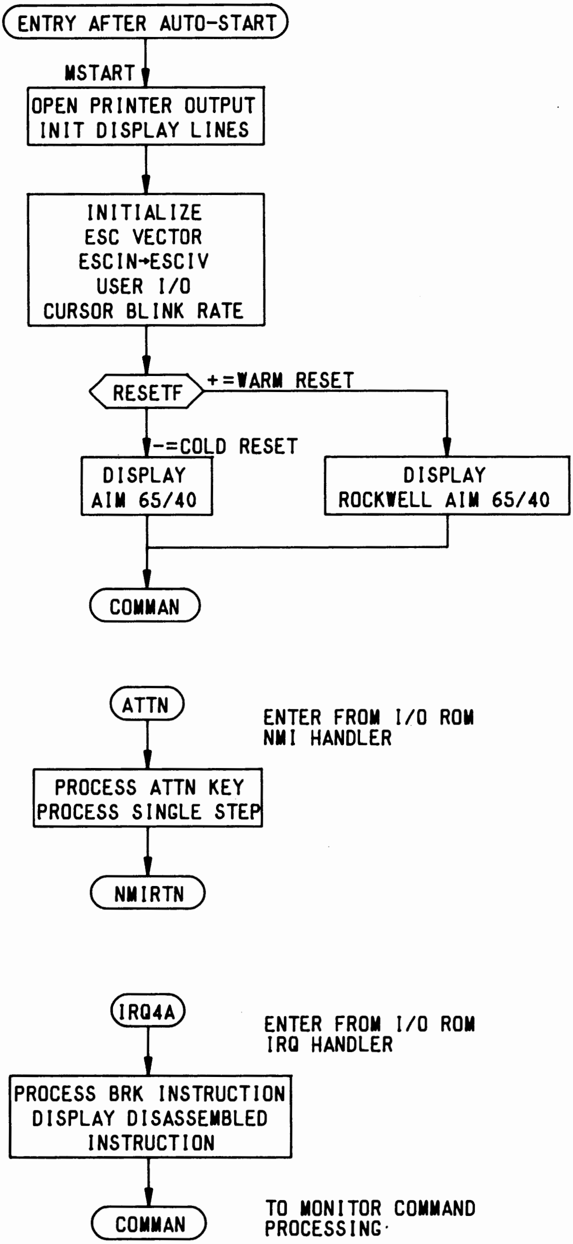


Figure 15-2. Monitor Processing Flowchart (Continued)

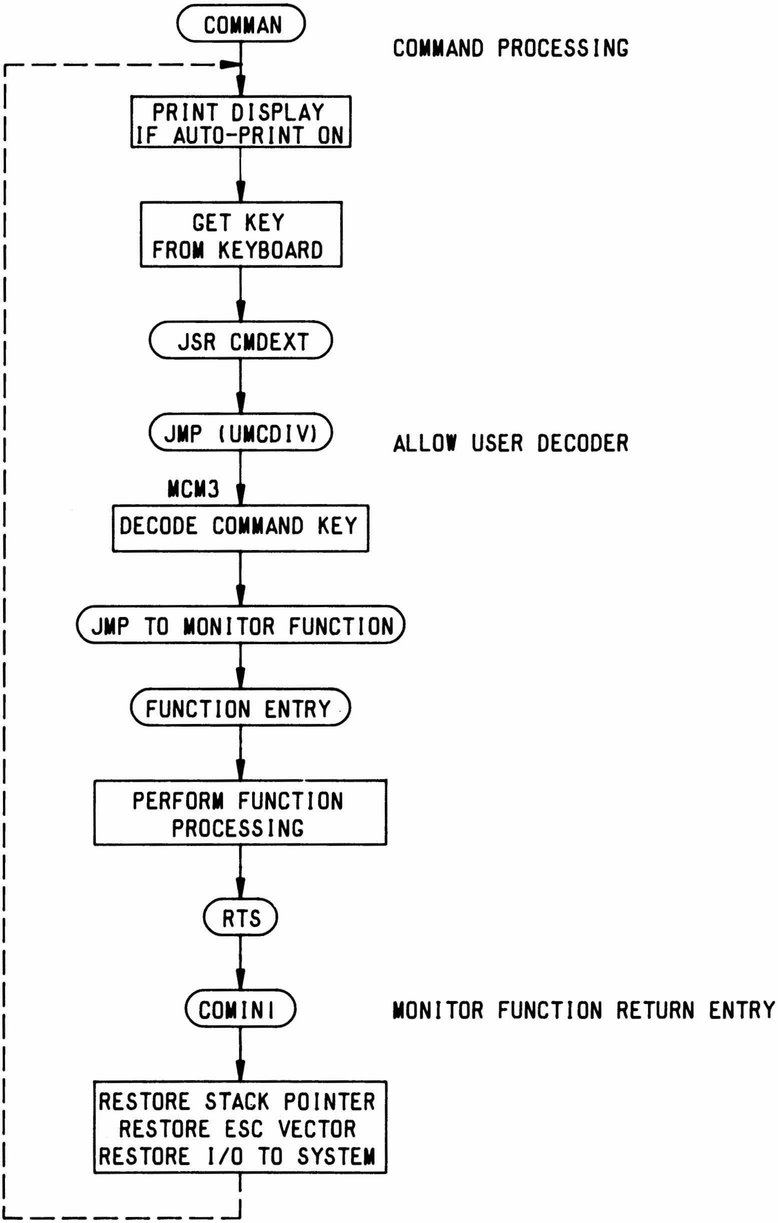


Figure 15-2. Monitor Processing Flowchart (Continued)

15.3 Monitor Subroutines

The Monitor/Editor ROMS contain I/O related subroutines that build upon or use the I/O ROM subroutines as well as utility subroutines. These subroutines are useful in applications where the Monitor/Editor ROMs remain installed. The following paragraphs describe the basic operation of these subroutines. Consult the Monitor/Editor ROMs assembly listing (document no. 2965ON92) for the detail subroutine operation. Since these subroutines are designed primarily for debug and edit use, there may be some modes of operation that are apparent only by a detail examination of the assembly code.

a. Setting Up the Active Input/Output Device

WHEREI AE9B I A,Y

Sends "IN=" to the display/printer and sets the active input device code word (INDEV) and the Y-Register corresponding to the entered letter:

Entered Char	Device	INDEV	Y-Reg
RETURN	Keyboard	$00	$00
SPACE	Keyboard	$00	$00
S	Serial	$01	$01
T	Audio Tape	$02	$02
M	Memory	$03	$03
F	Floppy Disk	$04	$04
U	User 1	$05	$05
V	User 2	$06	$06

WHEREIS AE95 I A,Y

Sends "OFFSET=0000" to the display/printer, enters an offset address, and sets the active input device code word (INDEV) and the Y-Register corresponding to the entered letter. Uses WHEREI.

WHEREØ   AEB9   I   A,Y

Sends "OUT=" to the display/printer and sets up the active output device code word (OUTDEV) and Y-Register corresponding to the entered letter:

Char	Device	OUTDEV	Y-Reg
RETURN	Display	$ØØ	$ØØ
SPACE	Display	ØØ	$ØØ
S	Serial	Ø1	$Ø4
T	Audio Tape	Ø2	$Ø8
M	Memory	Ø3	$ØC
F	Floppy disk	Ø4	$1Ø
U	User 1	Ø5	$14
V	User 2	Ø6	$18
P	Printer	Ø7	$1C
X	Null	Ø8	$2Ø

b. Input from the Keyboard

TAGKIA   BØA9   I   A

Checks the keystack and returns immediately with C=Ø if no key is available, otherwise returns with the input character in the A-Register and C=1.

c. Input from the Active Input Device

ADDIN   AF27   I   A,X,Y

Sends an equal sign ($3D) to the display/printer the inputs an address or symbol for an address (the symbol value must be previously entered into the symbol table) from the active input device. Converts the address from ASCII to binary and stores it in ADDR and ADDR+1. Returns with ADDR in X, ADDR+1 in Y and the input terminator in A.

NUMIN AF95 I A,X,Y

Sends a slash ($2F) to the display/printer and inputs a four digit decimal number from the active input device. Converts the number from ASCII to binary and stores it in COUNT and COUNT+1. Returns with COUNT in X, COUNT+1 in Y and the input terminator in A.

FROM AE39 I A,X,Y

Sends "FROM=XXXX" to the display/printer (XXXX=old address), enters a new address in hexadecimal or as a previously defined symbol from the keyboard, and returns with the new address in the X-Register (LSP) and the Y-Register (MSP). Enter with the old address in the X-Register (LSP) and the Y-Register (MSP)

FROMX AE35 I A,X,Y

Sends "FROM=0000" to the display/printer and uses FROM to enter a new address. The old address defaults to 0.

TO AE4B I A,X,Y

Sends "TO=XXXX" to the display/printer (XXXX = old address), enters a new address in hexadecimal or as a previously defined symbol from the keyboard and returns with the new address in the X-Register (LSP) and the Y-Register (MSP). Enter with the old address in the X-Register (LSP) and the Y-Register (MSP).

TOX AE47 I A,X,Y

Sends "TO=0000" to the display/printer and uses TO to enter a new address. The old address defaults to 0.

RDBYTE   A87Ø   I   A

If the active input device is audio tape, jumps to INPUT in the I/O ROM. For other devices, gets two input hexadecimal numbers from the active input device, converts them to binary, and returns with the numbers in the A-Register (first number in the MSP).

RDBYTO   A95F   I   A

Gets a character from the active input device, converts it to binary, packs it in BYTE and sends it to the display/printer.

INPUTU   A399   I   A

Gets a character from the active input device and returns with it in A with lower case alphabetics ($61 - $7A) forced to upper case ($41 - $51).

RDNIBL   A85B   I   A

Gets a character from the active input device and returns with it in the A-Register if it is a NUL ($Ø), SPACE ($2Ø) or hexadecimal number. If it is a hex number, it also converts it from ASCII to binary and packs it into the LSP (bit Ø-3) of BYTE. The beeper is sounded if the character is any other and the subroutine waits for another input character. Calls INPUTU, A2HEX and PACK.

RCHEK   BØ8Ø   I   A

Returns if no key is on the keystack. If a key is available processes it as follows:

SPACE	Waits for next key
Ø-9	Sets VSPEED
ESC	Jumps through ESCIV vector

VRCHEK BØ7D I A

Delays VSPEED times 8 ms then performs RCHEK. Returns with input characters in A.

SETOFF ACA4 I A,X,Y

Sends "OFFSET=ØØØØ" to the display/printer, inputs an address and stores it in OFFSET (LSP) and OFFSET+1 (MSP). Returns with the LSP in the X-Register and the MSP in the Y-Register.

d. <u>Output to the Display/Printer</u>

QM AE52 O A

Sounds the beeper and sends a question mark character ($3F) to the display/printer.

EQUAL AE59 O A

Outputs an equal sign character ($3D) to the display/printer.

WRITAD AE24 O A,X

Converts the address in ADDR and ADDR+1 to ASCII and sends it to the display/printer.

WRADXY AE21 O A,X

Loads the address in the X and Y Registers into ADDR and ADDR+1, converts it to ASCII and sends it to the display/printer.

WRADBK AE2D O A,X

Converts the address in ADDR and ADDR+1 to ASCII, sends it to the display/printer and backspaces the cursor four positions.

e. Output to the Active Output Device

CRLF   AE5E   O   A

Outputs a carriage return character ($ØD) and, if the active output device is display (SPACE or RETURN), serial (S), V (user 1), printer (P) or null (X), a line feed ($ØA) followed by the number of nulls ($ØØ) contained in the NULL variable, to the active output device.

CRCLOS   AE84   O   A

Calls CRLF if the active output device is the display (OUTDEV=Ø), outputs an end-of-file ($1A) to the active output device, then calls CLOSEO to close the active output device and return output to the display.

CLOSEQ   AE8D   O   A

Outputs an end-of-file ($1A) to the active output device then calls CLOSEO to close the active output device and return output to the display.

HOMEA   AE7F   O   A

Outputs a carriage return ($ØD) to the active output device.

f. Utility Functions

A2DEC   BØ44   U   A

Converts a decimal number (Ø-9) in the A-Register from ASCII to binary and returns with it in the A-Register with C=Ø. Returns with C=1 without converting it if the character is not Ø-9.

A2HEX B03C U A

Converts a hexadecimal number (Ø-F) in the A-Register from ASCII to binary and returns with it in the A-Register with C=Ø. Returns with C=1 without converting it if the character is not Ø=F.

DELAY BØBC U A, X, Y

Delays from 8Ø to 65,535 microseconds. Call with the delay value in the X (MSP) and Y (LSP) registers.

VISUAL BØF4 U X, Y

If the active output device is not interactive, delays VSPEED times 8 ms.

PACK A844 U

Shifts the LSP of BYTE into the MSP then adds the MSP of the A-Register to the LSP of BYTE.

SECTION 16

TROUBLESHOOTING AND ADJUSTMENTS

16.1 TROUBLESHOOTING

Your AIM 65/40 system has been functionally tested for correct operation at the factory before packaging for shipment. Should the system appear not to operate correctly, consult the troubleshooting procedure in Table 16-1. If the problem cannot be corrected, refer to the service instructions on the warranty card.

16.2 AIM 65/40 PRINTER ADJUSTMENTS

The printer has been adjusted at the factory, and no further adjustment should be required during normal operation. There are four adjustments on the printer that may be required, however, after extended operation.

16.2.1 Release Level Print Adjustment

With the head release lever in the PRINT position, wing "A" of the level should not touch the Thermal Head group. There must be visible clearance at "B" so that the Thermal Head group may rest on the platen (see Figure 16-1a).

16.2.2 Release Level Release Adjustment

When the head release lever is in the RELEASE position, the Thermal Head group must be held away from the platen. Minimum clearance is 0.8mm, as shown. To obtain both these conditions, form wings "A" as necessary (see Figure 16-1b).

16.2.3 Motor Gear Mesh Adjustment

Motor gear mesh is adjusted by loosening the top and bottom motor mounting screws, and repositioning the motor as necessary. Mesh between the motor and the large transmission gear must be as deep as possible without binding. When this condition is obtained, tighten the motor mounting screws (see Figure 16-1c).

16.2.4 Vertical Dot Alignment Adjustment

To adjust vertical dot alignments, print a series of eights and ones: 81818181... or perform a printer self test which prints a series of Ws and eights.

Loosen the strobe cap mounting nut slightly. Rotate the strobe cap until all vertical dots are in line. Tighten the strobe cap mounting nut (see Figure 16-1d).

Table 16-1. Troubleshooting Procedure

Symptom	Possible Cause	Corrective Action
1. No display	1a. Application program is hung up.	1a. Press RESET or CTRL RESET
	1b. I/O constants or variables have been altered	1b. Press CTRL RESET
	1c. +5V absent or out of tolerance	1c. Ensure proper +5V to SBC and display modules (see 2.1.3).
	1d. Display interface cable is disconnected	1d. Ensure display interface cable is connected at both ends (see 2.1.2)
	1e. I/O and Monitor ROMs are not installed and selected	1e. Ensure I/O and Monitor ROMs are installed and selected (see 2.1.4)
	1f. Application program Auto-Start processing is incorrect	1f. Remove application PROM/ROM from $8ØØØ-$EFFF and turn on system (see 6.2)
	1g. Socketed components are loose or installed improperly	1g. Ensure socketed components are securely installed in the proper position
	1h. PROM/ROM jumpers are installed improperly	1h. Ensure PROM/ROM jumpers are installed correctly (see 2.1.4)

Table 16-1. Troubleshooting Procedure (Continued)

Symptom	Possible Cause	Corrective Action
2. No response to keyboard	2a. Keyboard interface cable disconnected	2a. Ensure Keyboard interface cable is connected at both ends
	2b. Application program hung up	2b. Press RESET or CTRL RESET
3. No printout	3a. Auto-Print Off	3a. Press CTRL P
	3b. Printer Release level is in Release position	3b. Move Release level to Print position
	3c. +24V is absent or out of tolerance	3c. Ensure proper +24V (see 2.1.3)
	3d. +5V is absent or out of tolerance	3d. Ensure proper +5V (see 2.1.3)
	3e. Printer interface cable is disconnected	3e. Ensure Printer interface cable is connected at both ends
4. Printer is not printing one or more columns	4. See 3	4. See 3
5. Printer printing too fast or too dark	5a. Potentiometer R17 out of adjustment	5a. Adjust R17 counterclockwise to darken printout, or clockwise to lighten printout
6. Printer printing too fast or too slow	6a. Potentiometer R12 out of adjustment	6a. Adjust R12 clockwise for slower operation, or counterclockwise for faster

Table 16-1. Troubleshooting Procedure (Continued)

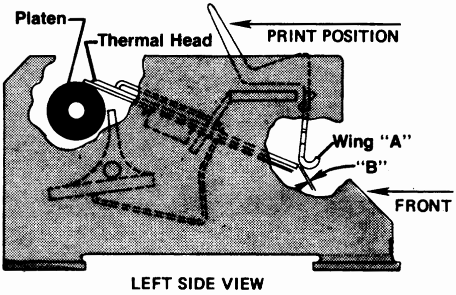
Symptom	Possible Cause	Corrective Action
7. Printer vertical dots are misaligned	7a. Printer speed is too fast	7a. Adjust R12 clockwise for slower operation
	7b. Print vertical dots are out of adjustment	7b. See Printer Vertical Dot adjustment (Section 16.2.4)
8. Printer is not printing evenly or consistently	8a. Loose +24V power or GND connection	8a. Ensure proper connections on power supply and and TB1
	8b. Foreign material between printer elements and paper	8b. Release Printer Paper Release bar and ensure nothing is between the print element and the paper
	8c. Printer Thermal Head is not resting on the platen	8c. See Printer Release level adjustment (Section 16.2.2)
9. Printer motor runs slow or is stopped when energized.	9a. Motor gear mesh is too tight	9a. See Printer Gear Mesh adjustment (Section 16.2.3)
10. Printer motor runs but Thermal head does not move	10a. Motor gear mesh is too loose	10a. See Printer Gear Mesh adjustment (Section 16.2.3)
	10b. Printer Release lever is in Release position	10b. Move lever to Print position

Table 16-1. Troubleshooting Procedure (Continued)

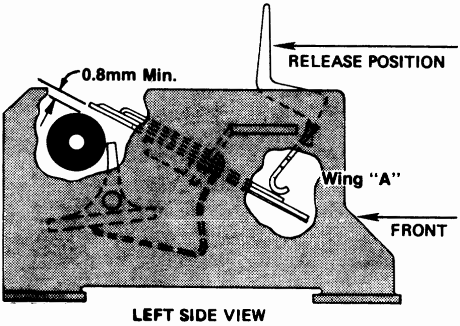
Symptom	Possible Cause	Corrective Action
11. Audio Tape Recorder Motor does not operate	11a. Inoperative Recorder	11a. Disconnect all AIM 65/40 lines from recorder and verify proper recorder operation
	11b. Incorrect recorder control line installation	11b. Verify recorder installation per Section 9.1
	11c. Incomplete recorder control line connection	11c. Remove control line from recorder. Put recorder in Play Mode and verify tape movement. With at least one audio line (IN or OUT) attached and proper tape control line ON, connect tape control line to recorder and verify continued recorder motor operation. Wiggle tape control line plug in recorder REM jack to ensure proper plug connection.

Table 16-1. Troubleshooting Procedure (Continued)

Symptom	Possible Cause	Corrective Action
	11d. Wrong tape control line pair used.	11d. Try the other tape control line pair.
12. Audio Tape does not read properly.	12a. Inoperative Recorder	12a. See 11a.
	12b. Incorrect recorder installation.	12b. Verify recorder interface connection and checkout (See 9.1)
	12c. Incorrect volume and tone adjustments on recorder.	12c. Adjust recorder volume and tone controls (see 9.1)

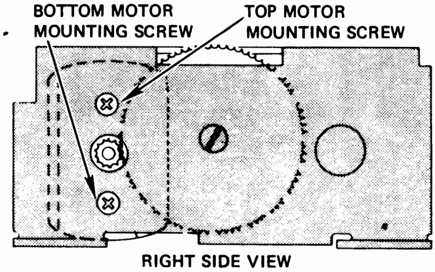


a. Release Lever Print Adjustment

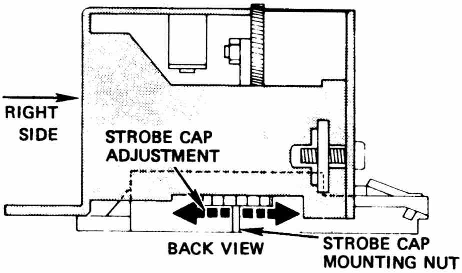


b. Release Lever Release Adjustment

Figure 16-1. AIM 65/40 Printer Adjustments



c. Motor Gear Adjustment



d. Vertical Dot Alignment

Figure 16-1. AIM 65/40 Printer Adjustments (Continued)

APPENDIX A

AIM 65/40 COMMAND DEFINITIONS

[ADDRESS] Hexadecimal address in the range $0000 to $FFFF composed of 1 to 4 hexadecimal digits terminated by either RETURN or SPACE. Backspace (DEL) is allowed and only valid hexadecimal digits will be accepted. If a symbol table is present, any symbol may be entered by preceding the symbol with a semicolon (;).

[BYTE] Hexadecimal value in the range $00 to $FF composed of two hexadecimal digits. Two digits must always be entered and the entry of the second digit terminates the entry. The entry of a SPACE results in no modification.

[DECIMAL NUMBER] A decimal number in the range 0 to 9999 composed of 1 to 4 decimal digits terminated by a RETURN or SPACE. If a RETURN is entered without any digits, the value entered is 1. The entry of a "." the value is infinite.

[DISPLAY]

H = Hexadecimal
A = ASCII
B = Binary
Z = Alphanumeric

[FILENAME] A string of 1 to 5 characters. The string is terminated by either a RETURN or SPACE. Errors in the string may be corrected by use of the DELETE key before termination of the string.

[INPUT DEVICE] A single letter specifying the input device to be used. The possible entries are:

SPACE or RETURN	System terminal (keyboard)
M	Memory
F	Floppy disk (user defined)
U	User defined
V	User defined
T	Audio tape
S	Serial (user defined)

[MNEMONIC OPCODE] A three-letter R6500 mnemonic abbreviation.

[OUTPUT DEVICE] A single letter specifying the output device to be used. The possible entries are:

SPACE or RETURN	System terminal (display/printer)
P	AIM 65/40 printer
F	Floppy disk (user defined)
U	User defined
V	User defined
X	Dummy output device
T	Audio tape
S	Serial (user defined)

[VERIFICATION] The single character Y in response to the query "ARE YOU SURE?" enables the invoked operation to continue. Any entry other than Y terminates the operation.

[WORD] A two-byte hexadecimal entry in the range of $0000 to $FFFF terminated by RETURN.

APPENDIX B

AIM 65/4Ø DEBUG MONITOR COMMAND SUMMARY

This appendix gives a summary of the Debug Monitor Commands. All items to be input by the user are indicated within braces, parenthesis or brackets. The first set of brace, { }, enclosing the character is an input command. If an item is enclosed in parentheses, ( ), only one of the items is to be input. If an item is enclosed in the following brackets, [ ], the options for that item are defined in Appendix A.

MONITOR CONTROL COMMANDS

CTRL RESET	Enter and Initialize Monitor (Cold Reset)
RESET	Enter Monitor (Warm Reset)
ATTN	Non-Maskable Interrupt
ESC	Escape to Monitor Command Level

E	Initialize Text Buffer and Enter Editor
	{E}
	EDIT FROM=[ADDRESS] TO=[ADDRESS]
	IN=[INPUT DEVICE]

C	Recover Text Buffer and Enter Editor
	{C}
	EDIT FROM=[ADDRESS] TO=[ADDRESS]
	IN=[INPUT DEVICE]

T	Re-enter Text Editor {T}
+	Repeat Last Command
&	Execute Command String {&} FROM=[ADDRESS]
O	Toggle Memory Bank {O} MEMORY BANK Ø/1
CTRL Z	Direct Peripheral Control { }

CTRLZ	CTRL Z - SBC Module RAM Self Test { } ARE YOU SURE? [Y/N] TO=[ADDRESS] XXXX (no. of tests completed)
CTRL C	Clear Display and Home Cursor
CTRL N	Home Cursor
@	ENTER OUTPUT RATE @ = [0-9]
F1-F8	Enter Function 1 - Function 8

DISPLAY/ALTER REGISTERS

R	Display Register Contents {R}*=HHHH P=BBBBBBBB A=HH X=HH Y=HH S=HH
A	Display/Alter Accumulator {A}=[BYTE]
X	Display/Alter X Register {X}=[BYTE]
Y	Display/Alter Y Register {Y}=[BYTE]
P	Display/Alter Processor Status {P}=[BYTE]
S	Display/Alter Stack Pointer {S}=[BYTE]
*	Display/Alter Program Counter {*}=[ADDRESS]

DISPLAY/ALTER MEMORY

M	Display Selected Memory Contents {M}[ADDRESS] HH HH HH HH HH HH HH HH AAAAAAAA
SPACE	Display Higher Memory Contents { }HHHH HH HH HH HH HH HH HH HH AAAAAAAA
-	Display Lower Memory Contents {-}HHHH HH HH HH HH HH HH HH HH AAAAAAAA
/	Alter Current Memory Contents {/}HHHH [BYTE] [BYTE] [BYTE] [BYTE] [BYTE] [BYTE] [BYTE] [BYTE] AAAAAAAA

ENTER/DISASSEMBLE INSTRUCTIONS

```
 I            Enter Mnemonic Instruction
              {I}
                   *=[ADDRESS]
                   [MNEMONIC] [HEX OPERAND]

 K            Disassemble Memory
              {K}*=[ADDRESS]/[DECIMAL NUMBER]
              OUT=[OUTPUT DEVICE]

 ;            Enter Symbol Value
              {;}[SYMBOL]=[WORD]
```

EXECUTION/TRACE

```
 G            Execution of Program
              {G} [ADDRESS] [RETURN]   Run Mode
              {G} [ADDRESS] [SPACE]/[S][Decimal Number] Step
              Mode

 Z            Toggle Instruction Trace Mode
              {Z} ON/OFF

 J            Display Register Heading
              {J}
                P   A   X   Y   S

 H            Display Jump and Branch History
              {H}
              XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
              XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

 V            Toggle Symbol Table On/Off
```

BREAKPOINT MANIPULATION

```
 #            Clear Breakpoints
              {#} OFF

 4            Toggle Breakpoint Enable On/Off
              {4}OFF/ON

 B            Set Breakpoint
              {B}BRK/(Ø,1,2,3,4,5,6,7)=[ADDRESS]

 ?            Display Breakpoints
              {?}Ø     1      2      3      4      5      6      7
              [ADDR] [ADDR] [ADDR] [ADDR] [ADDR] [ADDR] [ADDR] [ADDR]
```

LOAD/DUMP MEMORY

L	Load Memory {L}OFFSET=[HEX NUMBER] IN=[INPUT DEVICE]
D	Dump Memory {D}FROM=[ADDRESS] TO=[ADDRESS] OFFSET=[HEX NUMBER] MORE?(Y,N) TYPE=(A,B) OUT=[OUTPUT DEVICE]
F	Verify Memory (Binary Format Only) {F} OFFSET=[HEX NUMBER] IN=[INPUT DEVICE] OUT=[OUTPUT DEVICE]

PERIPHERAL CONTROL COMMAND

CTRL P	Toggle Auto-Print On/Off {CTRL P} AUTO-PRINT ON/OFF
PRINT	Print Display Contents
1	Toggle Recorder 1 Control On/Off {1} ON/OFF
2	Toggle Recorder 2 Control On/Off {2} ON/OFF
3	Verify Tape Checksum {3} UNIT = [INPUT DEVICE] FILE = [FILE NAME]

APPENDIX C

AIM 65/40 TEXT EDITOR COMMAND SUMMARY

This appendix gives a summary of the Text Editor Commands. All inputs to be input by the user are in brackets and parenthesis. If an item is enclosed in parentheses, ( ), only one of the items is to be input. If an item is enclosed in brackets, the options for that item are defined in Appendix A.

The flashing "equal sign" indicates that the system is at Edit Editor command level, therefore Editor commands may be entered. Screen oriented commands are valid after entering the E or F6 Commands. The screen edit <u>replace</u> mode is indicated by a flashing * cursor while the screen edit <u>insert</u> mode is indicated by a flashing —< cursor. A RETURN, F7 or F8 effects the change and exits the screen edit mode.

EDITOR ENTRY COMMANDS (FROM MONITOR)

E	Enter Text Editor {E} EDIT FROM=[ADDRESS] TO=[ADDRESS] IN=[INPUT DEVICE]
C	Recover Text Buffer and Re-enter Editor {E} EDIT PROM=[ADDRESS] TO=[ADDRESS]
T	Re-enter Text Editor {T}

EDITOR CONTROL COMMANDS

S	Enter Screen Edit Mode
ESC	(ESC) Escape to Editor Command Level
Q	Quit Editor and Enter Debug Monitor ={Q}
+	Repeat Last Command
CTRL C	Clear Display and Home Cursor
CTRL N	Home Cursor

LINE ORIENTED COMMANDS

```
R             Read Multiple Lines
              ={R} IN=[INPUT DEVICE]
              *[STRING]

I             Insert One Line
              ={I}
              *[STRING]

O             Overlay Current Line
              ={O}
              *[STRING]

K             Delete (Kill) Multiple Lines
              ={K}/[DECIMAL NUMBER]
              Lines to be killed preceeded by "/".
              ARE YOU SURE? (Y,N)
              Note:  If only 1 line is to be deleted,
                     verification is not required.

U             Go Up Multiple Lines
              ={U}/[DECIMAL NUMBER]

D             Go Down Multiple Lines
              ={D}/[DECIMAL NUMBER]

T             Go to Top Line
              ={T}

B             Go to Bottom Line
              ={B}

L             List Multiple Lines
              ={L}/[DECIMAL NUMBER] [RETURN/SPACE]
              OUT=[OUTPUT DEVICE]

G             Go to Line Number
              ={G}/[DECIMAL NUMBER]

SPACE         Display Current Line
              ={ }

?             Display Current and Last Line Addresses
              ={?}HHHH HHHH
```

STRING ORIENTED COMMANDS

```
F             Find Character String
              ={F}*[STRING(2Ø character)]

C             Change One Character String to Another Character
              String
              ={C} OLD=[STRING(2Ø character)] NEW=
              [STRING (to 8Ø characters)]/[DECIMAL NUMBER]
              [RETURN/SPACE]
              Note:  RETURN = selective change
                     SPACE = automatic change
```

SCREEN ORIENTED COMMANDS

F1 (or CTRL Q) Home Cursor on Line

F2 (or CTRL R) Clear Line to Right

F3 (or CTRL S) Toggle Insert Mode On/Off. Inserts any character to the left of the blinking left arrow when turned on. Replaces any character under the cursor when turned off.

F4 (or CTRL T) Delete Character Under Cursor.

NOTE

Note that the DEL key deletes the character to the left of the cursor.

F5 (or CTRL U) Move Cursor Left (left arrow).

F6 (or CTRL V) Move Cursor Right (right arrow) and enter Screen Edit Mode.

F7 (or CTRL W) Move Line/Cursor Down (down arrow).

F8 (or CTRL X) Move Line/Cursor Up (up arrow).

CTRL A Add a Line.

CTRL B Break a Line.

CTRL C Delete a Line.

APPENDIX D

I/O AND MONITOR MESSAGES

This appendix explains the messages that are output by the AIM 65/40 I/O ROM and Debug Monitor/Text Editor.

D.1 I/O ROM MESSAGES

PRINTER DOWN

A printout has been attempted but the printer failed to respond.

I/O ERROR

A function key was pressed which has an unattached I/O.

WRITE PROTECT AT

Data was attempted to be written into a write protected area (via write protect switches)

(ESC)

The user has aborted from the present command and returned to the command entry level.

CHECKSUM ERROR

The block checksum read from the input file is does not match the checksum computed from reading the input block characters (see Appendix I).

SYNC ERROR

The bit frequency read from the audio tape is different from that expected, resulting in a loss of bit synchronization (see Appendix I).

BLOCK ERROR

The block number read from the audio tape input block does not match the expected block number (see Appendix I).

D.2 MONITOR/EDITOR ROM MESSAGES

MEM FAIL

The memory has failed to store the requested data.

END

The line pointer is at the end (bottom) of the text buffer.

DONE

Dump from the AIM 65/4Ø to an external device (D, K or L command has been completed)

MEMORY BANK

The AIM 65/4Ø is in Bank Ø or Bank 1 as commanded by the Memory Bank command (O).

LOAD ERROR

The checksum read from the input data does not match the checksum computed from the input data (see Appendix H).

TOP

The line pointer is at top of the text buffer.

APPENDIX E

ASCII CHARACTER SET

HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII
ØØ	Ø	NUL	2Ø	32	SP	4Ø	64	@	6Ø	96	`
Ø1	1	SOH	21	33	!	41	65	A	61	97	a
Ø2	2	STX	22	34	"	42	66	B	62	98	b
Ø3	3	ETX	23	35	#	43	67	C	63	99	c
Ø4	4	EOT	24	36	$	44	68	D	64	1ØØ	d
Ø5	5	ENQ	25	37	%	45	69	E	65	1Ø1	e
Ø6	6	ACK	26	38	&	46	7Ø	F	66	1Ø2	f
Ø7	7	BEL	27	39	'	47	71	G	67	1Ø3	g
Ø8	8	BS	28	4Ø	(	48	72	H	68	1Ø4	h
Ø9	9	HT	29	41	)	49	73	I	69	1Ø5	i
ØA	1Ø	LF	2A	42	*	4A	74	J	6A	1Ø6	j
ØB	11	VT	2B	43	+	4B	75	K	6B	1Ø7	k
ØC	12	FF	2C	44	,	4C	76	L	6C	1Ø8	l
ØD	13	CR	2D	45	-	4D	77	M	6D	1Ø9	m
ØE	14	SO	2E	46	.	4E	78	N	6E	11Ø	n
ØF	15	SI	2F	47	/	4F	79	O	6F	111	o
1Ø	16	DLE	3Ø	48	Ø	5Ø	8Ø	P	7Ø	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	5Ø	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	2Ø	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	12Ø	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	9Ø	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	6Ø	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	3Ø	RS	3E	62	>	5E	94	↑	7E	126	~
1F	31	VS	3F	63	?	5F	95	←	7F	127	DEL

NUL	- Null	DLE	- Data Link Escape
SOH	- Start of Heading	DC	- Device Control
STX	- Start of Text	NAK	- Negative Acknowledge
ETX	- End of Text	SYN	- Synchronous Idle
EOT	- End of Transmission	ETB	- End of Transmission Block
ENQ	- Enquiry	CAN	- Cancel
ACK	- Acknowledge	EM	- End of Medium
BEL	- Bell	SUB	- Substitute
BS	- Backspace	FSC	- Escape
HT	- Horizontal Tabulation	FS	- File Separator
LF	- Line Feed	GS	- Group Separator
VT	- Vertical Tabulation	RS	- Record Separator
FF	- Form Feed	US	- Unit Separator
CR	- Carriage Return	SP	- Space (Blank)
SO	- Shift Out	DEL	- Delete
SI	- Shift In		

ASCII CHARACTER SET (7-BIT CODE)

LSD	MSD	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	–	=	M	]	m	}
E	1110	SO	RS	•	>	N	↑	n	~
F	1111	SI	US	/	?	O	←	o	DEL

NUL	— Null	DLE	— Data Link Escape
SOH	— Start of Heading	DC	— Device Control
STX	— Start of Text	NAK	— Negative Acknowledge
ETX	— End of Text	SYN	— Synchronous Idle
EOT	— End of Transmission	ETB	— End of Transmission Block
ENQ	— Enquiry	CAN	— Cancel
ACK	— Acknowledge	EM	— End of Medium
BEL	— Bell	SUB	— Substitute
BS	— Backspace	ESC	— Escape
HT	— Horizontal Tabulation	FS	— File Separator
LF	— Line Feed	GS	— Group Separator
VT	— Vertical Tabulation	RS	— Record Separator
FF	— Form Feed	US	— Unit Separator
CR	— Carriage Return	SP	— Space (Blank)
SO	— Shift Out	DEL	— Delete
SI	— Shift In		

APPENDIX F

R65ØØ INSTRUCTION SET

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			(IND. X)			(IND), Y			Z PAGE, X			ABS, X			ABS, Y			RELATIVE			INDIRECT			Z PAGE, Y			PROCESSOR STATUS CODES	
MNEMONIC	OPERATION	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	OP	n	#	7 6 5 4 3 2 1 0 N V • B D I Z C	MNEMONIC
ADC	A + M + C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										N V • • • • Z C	ADC
AND	A ∧ M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										N • • • • • Z •	AND
ASL	C ← [7 … 0] ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													N • • • • • Z C	ASL
BCC	BRANCH ON C = 0 (2)																															90	2	2							• • • • • • • •	BCC
BCS	BRANCH ON C = 1 (2)																															B0	2	2							• • • • • • • •	BCS
BEQ	BRANCH ON Z = 1 (2)																															F0	2	2							• • • • • • • •	BEQ
BIT	A ∧ M				2C	4	3	24	3	2																															M_7 M_6 • • • • Z •	BIT
BMI	BRANCH ON N = 1 (2)																															30	2	2							• • • • • • • •	BMI
BNE	BRANCH ON Z = 0 (2)																															D0	2	2							• • • • • • • •	BNE
BPL	BRANCH ON N = 0 (2)																															10	2	2							• • • • • • • •	BPL
BRK	BREAK													00	7	1																									• • • 1 • 1 • •	BRK
BVC	BRANCH ON V = 0 (2)																															50	2	2							• • • • • • • •	BVC
BVS	BRANCH ON V = 1 (2)																															70	2	2							• • • • • • • •	BVS
CLC	0 → C													18	2	1																									• • • • • • • 0	CLC
CLD	0 → D													D8	2	1																									• • • • 0 • • •	CLD
CLI	0 → I													58	2	1																									• • • • • 0 • •	CLI
CLV	0 → V													B8	2	1																									• 0 • • • • • •	CLV
CMP	A − M	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										N • • • • • Z C	CMP
CPX	X − M	E0	2	2	EC	4	3	E4	3	2																															N • • • • • Z C	CPX
CPY	Y − M	C0	2	2	CC	4	3	C4	3	2																															N • • • • • Z C	CPY
DEC	M − 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													N • • • • • Z •	DEC
DEX	X − 1 → X													CA	2	1																									N • • • • • Z •	DEX
DEY	Y − 1 → Y													88	2	1																									N • • • • • Z •	DEY
EOR	A V M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										N • • • • • Z •	EOR
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													N • • • • • Z •	INC
INX	X + 1 → X													E8	2	1																									N • • • • • Z •	INX
INY	Y + 1 → Y													C8	2	1																									N • • • • • Z •	INY
JMP	JUMP TO NEW LOC				4C	3	3																												6C	5	3				• • • • • • • •	JMP
JSR	JUMP SUB				20	6	3																																		• • • • • • • •	JSR
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										N • • • • • Z •	LDA
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	N • • • • • Z •	LDX
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													N • • • • • Z •	LDY
LSR	0 → [7 … 0] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 • • • • • Z C	LSR
NOP	NO OPERATION													EA	2	1																									• • • • • • • •	NOP
ORA	A V M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										N • • • • • Z •	ORA
PHA	A → Ms S − 1 → S													48	3	1																									• • • • • • • •	PHA
PHP	P → Ms S − 1 → S													08	3	1																									• • • • • • • •	PHP
PLA	S + 1 → S Ms → A													68	4	1																									N • • • • • Z •	PLA
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)	PLP
ROL	[7 … 0] ← [C]				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													N • • • • • Z C	ROL
ROR	[C] → [7 … 0]				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													N • • • • • Z C	ROR
RTI	RTRN INT													40	6	1																									(RESTORED)	RTI
RTS	RTRN SUB													60	6	1																									• • • • • • • •	RTS
SBC	A − M − $\overline{C}$ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										N V • • • • Z (3)	SBC
SEC	1 → C													38	2	1																									• • • • • • • 1	SEC
SED	1 → D													F8	2	1																									• • • • 1 • • •	SED
SEI	1 → I													78	2	1																									• • • • • 1 • •	SEI
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										• • • • • • • •	STA
STX	X → M				8E	4	3	86	3	2																												96	4	2	• • • • • • • •	STX
STY	Y → M				8C	4	3	84	3	2													94	4	2																• • • • • • • •	STY
TAX	A → X													AA	2	1																									N • • • • • Z •	TAX
TAY	A → Y													A8	2	1																									N • • • • • Z •	TAY
TSX	S → X													BA	2	1																									N • • • • • Z •	TSX
TXA	X → A													8A	2	1																									N • • • • • Z •	TXA
TXS	X → S													9A	2	1																									• • • • • • • •	TXS
TYA	Y → A													98	2	1																									N • • • • • Z •	TYA

(1) ADD 1 to N IF PAGE BOUNDARY IS CROSSED

(2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE

(3) CARRY NOT = BORROW

(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X	INDEX X	+	ADD	M_7	MEMORY BIT 7
Y	INDEX Y	−	SUBTRACT	M_6	MEMORY BIT 6
A	ACCUMULATOR	∧	AND	n	NO. CYCLES
M	MEMORY PER EFFECTIVE ADDRESS	V	OR	#	NO. BYTES
Ms	MEMORY PER STACK POINTER	V	EXCLUSIVE OR		

APPENDIX G

BINARY, DECIMAL AND HEXADECIMAL TABLES

HEXADECIMAL AND DECIMAL CONVERSION

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
7654		3210		7654		3210		7654		3210	
Byte				Byte				Byte			

POWERS OF 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

APPENDIX H

INPUT/OUTPUT DATA FORMATS

Two types of data are input and output by the Monitor/Editor in conjunction with the I/O ROM: object code and text. Object code is handled in either of two formats -- binary or ASCII, while text is handled only in ASCII. This appendix describes these three data formats.

When outputting the data to peripheral devices, additional peripheral file formatting may be required. The block formatting for audio tape is described in Appendix I.

H.1 OBJECT CODE

Object code refers to instructions or data level as stored in memory. Object code is most often used to describe the processor instructions generated in machine executable form by an assembler or a compiler. These instructions are usually hexadecimal numbers (Ø-9, A-F) which are decoded into individual commands by the processor upon execution, while the data may be in either hexadecimal or ASCII (see Appendix E) form.

Object code is generated in either of two formats by the Monitor Dump function (see Section 4.8.2): binary and ASCII. This data is combined with any additional file formatting required for a specific peripheral type then output to that peripheral. Data recorded in these formats by the Monitor, by optional software (e.g. AIM 65/4Ø assembler) or any other system is compatible with the Monitor Load function (see Section 4.8.1). The Load function strips off peripheral overhead file format characters, determines if the object code is in binary or ASCII data format and processes it accordingly.

H.1.1 Object Code Binary Format

The object code binary format contains the data exactly as it appears in memory. In addition, minimal overhead, i.e. starting address and number of bytes and check bits, maximizes data content included in a file. This format is the most compact therefore it requires less mass storage space and minimizes recording and reading time. (However, data in this format can not be directly displayed or printed by standard peripheral devices.) This format is used mainly for high density mass storage of program and/or data.

The object code binary format is output when B is entered in response to the TYPE= prompt in the Monitor dump command (see Section 4.8.2).

The data record format is:

$$\mathrm{8D}N_3N_2N_1N_0A_3A_2A_1A_0\overset{1}{D_1D_0}\overset{2}{D_1D_0}\ldots\ldots\ldots\ldots\overset{n}{D_1D_0}C_3C_2C_1C_0\mathrm{1A}$$

where:

8D = Object code binary format identifier.

$N_3N_2N_1N_0$ = Number of data bytes in the record ($0001-$FFFF).

$A_3A_2A_1A_0$ = Address of the first data byte in the recorder.

D_1D_0 = One byte of data (two hexadecimal digits).

$C_3C_2C_1C_0$ = Checksum.

1A = End of File marker.

Note: All data is represented in hexadecimal format.

H.1.2 Object Code ASCII Format

Data is converted from internal binary format to output it in ASCII format. ASCII format is compatible with most displays and printers and is used primarily when it is desired to visually observe the object code.

Data recorded in ASCII requires twice as much space to store the data since one 8-bit byte holds only one encoded hexadecimal number whereas one byte will hold two hexadecimal numbers in binary format (i.e. bits Ø - 3 holds the first number and bits 4 - 7 holds the second number).

The object code ASCII format is output when A is entered in response to the TYPE= prompt in the Monitor dump command (see Section 4.8.2).

The object data consists of multiple object data records, each containing a starting address and up to 24 bytes of information.

All files contain at least two object code records: the first record and the last record. The last record uniquely identifies the end of the file data.

a. The format of all object data records in ASCII (except the last record) is:

$$;N_1N_\emptyset A_3A_2A_1A_\emptyset \overset{1}{D_1D_\emptyset}\overset{2}{D_1D_\emptyset}\cdots\overset{n}{D_1D_\emptyset}X_3X_2X_1X_\emptyset CR1A$$

where:

; = Start of the record ($3B).

N_1N_2 = Number of data bytes, from 1 to 24 ($18) in the record.

$A_3A_2A_1A_\emptyset$ = Address of the first data byte in the record.

$D_1D_\emptyset$ = One data character.

$X_3X_2X_1X_{Ø}$ = Record checksum. This is the sum of all the characters in the object code record except the ; character and the record checksum. The checksum is truncated to four hexadecimal digits, i.e., carry is ignored.

CR = Carriage return ($ØD), which indicates end of record.

1A = End of File marker.

Note: All characters are formatted in ASCII when directed to all peripherals except audio tape. When output to audio tape, the N_x, A_x, D_x and X_x characters are first converted to binary format thus recording two hexadecimal numbers per byte. The ";", "CR" and "1A" remain one-byte each in ASCII.

b. The format of the last object code record in ASCII format is:

;ØØ$C_3C_2C_1C_{Ø}X_3X_2X_1X_{Ø}$CR1A

where:

ØØ = Last data record identifier.

$C_3C_2C_1C_{Ø}$ = Number of data records including the last record.

1A = End of File marker.

Note: All characters are formatted in ASCII when directed to all peripherals except audio tape. When output to audio tape, the C_x and X_x characters are first converted to binary format thus recording two hexadecimal numbers per byte. The ";", "CR" and "1A" remain one-byte each in ASCII.

H.2 TEXT FORMAT

Text refers to alphanumeric and special characters as encoded in ASCII format (see Appendix E).

The Editor Text Buffer (see Section 5.1.1) stores text in ASCII. The Editor List function (see Section 5.4.9) combines text from the Text Buffer with peripheral file formal characters and outputs it to the proper port. The Editor Read function (see Section 5.4.1) inputs data in this format into the Text Buffer.

Program source code input into an assembler or compiler is usually text. This allows program instructions to be coded in letters, numbers and special characters.

Text entered into the Text Buffer by the Editor consists of lines of printable characters ($20 - $7F) terminated by a carriage return ($ØD). Each line may contain up to 79 printable characters. The format may be shown graphically as:

$C_{x1}C_{x2}C_{x3}$ ---------- 1A

where:

C_{xy} = Character y in line x, in ASCII.

$ØD_x$ = Line x terminating carriage return, in ASCII.

1A = End of File marker.

APPENDIX I

AIM 65/4Ø AUDIO TAPE FORMAT

The AIM 65/4Ø audio cassette tape format is designed to provide fast, reliable recording and reading of both object code and source code using a low cost audio cassette recorder.

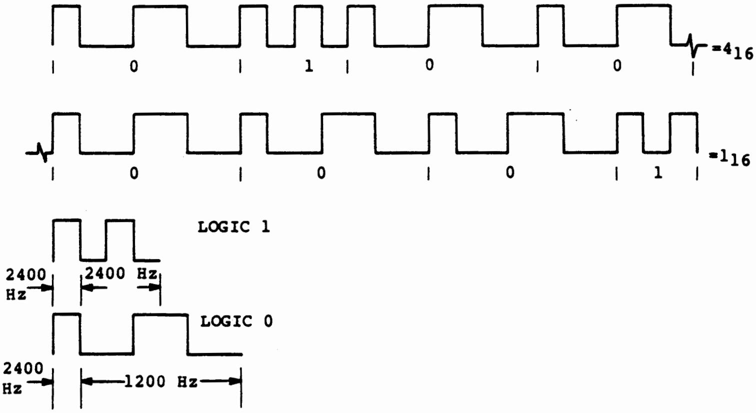
Data is output in the audio tape file format when T is entered pin response to the OUT=prompt, e.g. OUT=T.

Source code and object code (ASCII format only) are compatible with the AIM 65 Microcomputer.

I.1 BIT LOGIC STATE DEFINITION

Each transmitted bit begins with a positive one-half cycle of 24ØØ Hz tone. The following three half-cycles determine the logic state of the bit. Three half-cycles of 24ØØ Hz equal a logic "1". Three half-cycles of 12ØØ Hz tone equals a logic "Ø".

Figure I-1 shows eight bits of data (one byte). If this data is in binary format, the 41 equals two hexadecimal numbers, 4 and 1. If the data is in ASCII format, the character A is represented.



Note: The bit frequency is controlled by the I/O ROM constant TAPSPD (see Section 6.1.2).

Figure I-1. Audio Tape Bit Format

I.2 BLOCK FORMAT DESCRIPTION

The data is recorded in blocked format. One block contains 8Ø bytes of block numbers and data, and from 8Ø to 595 bytes of synchronization, end of synchronization and block checksum information. The block format is:

|<-------- 8Ø Bytes --------->|

SYN	ID	BLK	DATA FIELD	BLK CHKSUM
1616- - - 1616	XX	HH		HHHH
(16Ø)	(1)	(1)	(79)	(2)

a. Synchronization (SYN)

Each block begins with a series of Synchronous Idle (SYN) characters ($16). During read operations, the SYN pattern allows the AIM 65/4Ø to sense the start of the block and synchronize to the incoming serial data stream.

The first block contains 512 SYN characters (about five second's worth). The large number of SYN characters allows easy positioning of the tape prior to a read operation and ensures reading of the first block.

The SYN characters are generated between blocks to create an interblock gap. The number of interblock SYN characters is determined by the I/O ROM constant IRGSYN (see Section 6.1.2). The cold reset value of IRGSYN is 8Ø ($5Ø) which generates 16Ø SYN characters. Note that the second block is preceded by 1Ø6 SYN characters or the number of SYN characters specified by the IRGSYN value, whichever is larger.

The default value of IRGSYN provides sufficient time for operating a recorder using remote control or for performing some processing between reading blocks when remote control is not used.

If remote control is used, the minimum number of SYN characters required by your recorder may be less, e.g. 2Ø rather than 16Ø. You may reduce the gap size by storing a smaller number into IRGSYN before recording. Be sure, however, to provide sufficient SYN characters to allow for recorder degradation and if you want to read your file on another recorder with unknown remote control performance.

b. Data Type Identifier (ID)

A one byte character identifies the data type:

# (ASCII $23) = ASCII data

% (ASCII $25) = Binary data

c. Block Count (BLK)

The block count (BLK) specifies the block number. This number starts with ØØ on the first block and increments by one for each block recorded, in hexadecimal, to $FF. If more than $FF blocks are recorded, the number restarts at ØØ.

d. Data Field

The data field includes the file name, data type, identifiers, the actual object code or text data and any extra data terminating characters. There are three possible ways each block may be formatted: first block, mid-block and last block. Refer to Figure I-2 for an illustration of object code and text data files. The possible subfields in the data field are:

(1) File Name

The file name (FILE NAME) consists of five or more ASCII characters that uniquely identify the file. If the data field contains object code, a carriage return ($ØD) terminates the file name. The number of characters in the file name is controlled by I/O ROM variable TNAMSZ (see Section 6.1.2).

(2) Actual Data

Object code as described in Section H.1 or text as described in Section H.2 is included here.

|<------------ 8Ø Bytes -------------->|

SYN	ID	BLK	FILE NAME	CR	OBJECT CODE	BLK CHKSUM
1616---1616	XX	ØØ	XXXXX	ØD	XX---------------XX	HHHH
(16Ø)*	(1)	(1)	(5)**	(1)	(74)	(2)

FIRST BLOCK

SYN	ID	BLK	OBJECT CODE	BLK CHKSUM
1616---1616	XX	HH	XX-------------------------------XX	HHHH
(16Ø)*	(1)	(1)	(79)	(2)

MID BLOCK

SYN	ID	BLK	OBJECT BLOCK	ZERO FILL	BLK CHKSUM
1616---1616	XX	HH	XX-----------XX	ØØ----------------ØØ	HHHH
(16Ø)*	(1)	(1)	(N)	(79-N)	(2)

LAST BLOCK

a. Object Code Data

|<------------ 8Ø Bytes -------------->|

SYN	#	BLK	FILE NAME	TEXT DATA	BLK CHKSUM
1616---1616	23	ØØ	XXXXX	XX--------------XX	HHHH
(16Ø)*	(1)	(1)	(5)**	(74)	(2)

FIRST BLOCK

SYN	#	BLK	TEXT DATA	BLK CHKSUM
1616---1616	23	HH	XX-------------------------------XX	HHHH
(16Ø)*	(1)	(1)	(79)	(2)

MID BLOCK

SYN	#	BLK	TEXT DATA	CR	ZERO FILL	BLK CHKSUM
1616---1616	23	HH	XX--------XX	ØD	ØØ---------------ØØ	HHHH
(16Ø)*	(1)	(1)	(N)	(1)	(77-N)	(2)

LAST BLOCK

b. Text Data

* Cold reset value shown for IRGSYN (see Section 6.1.2).

** Cold reset value shown for TNAMSZ (see Section 6.1.2).

Figure I-2. Audio Tape File Formats

(3) Text Terminating Carriage Return

If the data is text, a text terminating carriage return ($ØD) is included.

(4) Zero Fill

The remaining data bytes are filled with hexadecimal zeros.

e. Block Checksum (BLK CHKSUM)

The block checksum (BLK CHKSUM) is the hexadeimal sum of the 8Ø data characters, truncated (i.e., carry is ignored) to four hexadecimal digits, and stored in two bytes.

APPENDIX J
AIM 65/40 SYSTEM I/O ADDRESSES

Address	Label	Bytes	Reset	Parameter
FF80	PRIRTY	1	00	IRQ Priority Mask
FFA0	UORB	1	FF	Port B Data Register
FFA1	UORA	1	FF	Port A Data Register
FFA2	UDRB	1	FF	Port B Data Direction Register
FFA3	UDRA	1	00	Port A Data Direction Register
FFA4	UT1CL	1	-	Timer 1 Latch/Counter Low
FFA5	UT1CH	1	-	Timer 1 Latch/Counter High
FFA6	UT1LL	1	-	Timer 1 Latch Low
FFA7	UT1LH	1	-	Timer 1 Latch High
FFA8	UT2CL	1	-	Timer 2 Latch/Counter Low
FFA9	UT2CH	1	-	Timer 2 Counter High
FFAA	USR	1	FF	Shift Register (SR)
FFAB	UACR	1	00	Auxiliary Control Register (ACR)
FFAC	UPCR	1	00	Peripheral Control Register (PCR)
FFAD	UIFR	1	00	Interrupt Flag Register (IFR)
FFAE	UIER	1	80	Interrupt Enable Register (IER)
FFAF	UORAX	1	FF	Port A Data Register (w/o Handshake)
FFB0	SORB	1	FF	Port B Data Register
FFB1	SORA	1	FF	Port A Data Register
FFB2	SDRB	1	FF	Port B Data Direction Register
FFB3	SDRA	1	00	Port A Data Direction Register
FFB4	ST1CL	1	-	Timer 1 Latch/Counter Low
FFB5	ST1CH	1	-	Timer 1 Latch/Counter High
FFB6	ST1LL	1	-	Timer 1 Latch Low
FFB7	ST1LH	1	-	Timer 1 Latch High
FFB8	ST2CL	1	-	Timer 2 Latch/Counter Low
FFB9	ST2CH	1	-	Timer 2 Counter High
FFBA	SSR	1	FF	Shift Register (SR)
FFBB	SACR	1	00	Auxiliary Control Register (ACR)
FFBC	SPCR	1	00	Peripheral Control Register (PCR)
FFBD	SIFR	1	00	Interrupt Flag Register (IFR)
FFBE	SIER	1	80	Interrupt Enable Register (IER)
FFBF	SORAX	1	FF	Port A Data Register (w/o Handshake)
FFC0	KBORB	1	FF	Port B Data Register
FFC1	KBORA	1	FF	Port A Data Register
FFC2	KBDRB	1	FF	Port B Data Direction Register
FFC3	KBDRA	1	00	Port A Data Direction Register
FFC4	KBT1CL	1	-	Timer 1 Latch/Counter Low
FFC5	KBT1CH	1	-	Timer 1 Latch/Counter High
FFC6	KBT1LL	1	-	Timer 1 Latch Low
FFC7	KBT1LH	1	-	Timer 1 Latch High
FFC8	KBT2CL	1	-	Timer 2 Latch/Counter Low
FFC9	KBT2CH	1	-	Timer 2 Counter High
FFCA	KBSR	1	FF	Shift Register (SR)
FFCB	KBACR	1	00	Auxiliary Control Register (ACR)
FFCC	KBPCR	1	00	Peripheral Control Register (PCR)
FFCD	KBIFR	1	00	Interrupt Flag Register (IFR)
FFCE	KBIER	1	80	Interrupt Enable Register (IER)
FFCF	KBORAX	1	FF	Port A Data Register (w/o Handshake)
FFD0	ACIADR	1	00	Data Register
FFD1	ACIASR	1	10	Status Register
FFD2	ACIACM	1	0B	Command Register
FFD3	ACIACN	1	1E	Control Register

APPENDIX K

I/O ROM AND MONITOR SUBROUTINES

K.1 ABBREVIATIONS

D/P	=	Display/Printer	K/B	=	Keyboard
AOD	=	Active Output Device	K/S	=	Key Stack
AID	=	Active Input Device	PTR	=	Printer
MSB	=	Most Significant Byte	DPY	=	Display
LSB	=	Least Significant Byte			
MSP	=	Most Significant Portion			
LSP	=	Least Significant Portion			

K.2 I/O ROM SUBROUTINES

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
ABLK	F37E	O	A	Sends 1 blank to the AOD
ABX	F384	O	A,X	Sends n blanks (in X) to the AOD
ABX3	F382	O	A,X	Sends 3 blanks to AOD
ANYKEY	F622	I		Tests the K/S for a new key and returns with the result in Z. Z = 1, key available. Z = 0, key not available
A2STAK	F593	I	Y	If the K/S is not full, the contents of A is put onto the K/S and the C is cleared. If the K/S is full, C is set without putting A on the K/S
BACK	F498	O	X	Sends n backspaces (in X) to the D/P
BACKSP	F492	O		Sends 1 backspace to the D/P
BANK0	FFE8	U	A	Enables bank zero
BANK1	FFF1	U	A	Enables bank one
BEEP	F467	O		Beeps the number of cycles in BEEPCY at the BEEPON and BEEPOF tone
BLANK	F37A	O	A	Sends 1 blank to the D/P
BLANK2	F377	O	A	Sends 2 blanks to the D/P
BLANK3	F374	O	A	Sends 3 blanks to the D/P
BLANK4	F371	O	A	Sends 4 blanks to the D/P
CDISIN	F83C	I	Y	Closes display input

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
CLOPTR	F8B3	O	A	Closes printer output (call after PUTPTR)
CLOSEI	F323	I	A,Y	Closes the AID per INDEV
CLOSEO	F31C	O	A,Y	Closes the AOD per OUTDEV
CLOSTO	FCØE	O	A	Writes the final tape block and restores the AOD to the D/P
CLR2RT	F396	O	A	Clears the D/P to the right of the cursor
COIF	F324	I/O	A,Y	Closes the AID per Y (see INDEV)
COLD	F11D	U	A,X,Y	Performs a cold reset
CRLOW	F38F	O	A	Sends a CR to the D/P
CURCNG	F3E2	O		Sends cursor command (in A) to D/P
CUROFF	F3D4	O		Blanks the cursor on the D/P
CURON	F3CB	O		Displays the cursor on the D/P
CUR2ST	F3EØ	O	A	Sends asterisk cursor command to D/P
CUR2X	F3DC	O	A	Sends block cursor command to D/P
DISPLY	F361	O		Sends the character in A to the DPY
DMDKEY	F55E	I	A	Strobes the K/B for a key and returns the ASCII value in A
GDISIN	F83Ø	I	A	Gets character from DPY when ACK is detected. Returns with character in A after sending Strobe
GETAPE	FA63	I	A	Gets the next character from the audio tape and returns it in A
GETDT	F84Ø	I	A	Gets a character from the DPY through VECTOR
GETIOV	F33D	I/O	A	Sets VECTOR to point to JMP table
GETKEY	F5AF	I	A	Gets a key from the K/S and returns it in A
GETSER	F963	I	A	Gets a byte from the serial RS-232/TTY port and returns it in A

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function	
GETSTK	F63C	I	A	Gets a key from the K/S and returns it in A	
GETXY	F41C	U		The contents of the X and Y are pulled from the top two values on the stack, respectively	
HOME	F38B	O	A	Sends a CR to the D/P	
H2ASCI	F3F2	U	U	Converts the LSP of A to ASCII	
INALL	F233	I	A	Gets one character from the AID and returns with it in A	
INCVEC	F44A	U		Increments VECTOR by 1 and, if VECTOR becomes Ø, also increment VECTOR+1	
INLOW	F451	I	A	Sets the AID to the K/B	
INPUT	F248	I	A	Gets one character from the AID and returns with it in A unless it is one of the following:	
				Key	Action
				ESC	Displays (ESC), then jumps through ESCIV
				PRINT	Prints the display line then waits for the next input character
				CTRL C	Clears the line, homes the cursor and waits for the next input character
				CTRL N	Homes the cursor and waits for the next input character
				CTRL P	Toggles Auto-Print and waits for the next input character
KEYDWN	F58D	I	A	Examines the K/B for a key and returns with Z: Z = 1, key is depressed. Z = Ø, key is not depressed	
LDAY	F4A6	B	A	Fetches a byte from either RAM bank Ø or 1 as determined by the Y AND A. A contains the offset from the variable S1 of the address vector to which the Y is added	
LEFT	F44Ø	O	A	Shifts A four bits to the left	

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
LFLOW	F392	O	A	Sends a LF to the D/P
LLD	F457	I/O	A	Sets the AID to the K/B and the AOD to the D/P
NOUT	F3AC	O	A	Converts the hex number in the LSP of A to ASCII and sends it to the AOD
NOUTLO	F3C6	O	A	Converts the hex number in the LSP of A and sends it to the D/P
NUMA	F3A4	O	A	Converts 2 hex numbers in A to ASCII (MSP first) sends them to the AOD
NUMABL	F3B2	O	A	Outputs a blank then converts 2 hex numbers in A to ASCII (MSP first) and sends them to the D/P
NUMALO	F3BE	O	A	Converts 2 hex numbers in A to ASCII (MSP first) and sends them to the D/P
ODISIN	F817	I	A	Opens display input. Outputs Transmit Display line and to DPY. Set display port to inputs
OPENI	F21D	I	A,Y	Opens the AID per INDEV
OPENIO	F317	I/O	A	Opens the AID or AOD per Y
OPENO	F312	O	A,Y	Opens the AOD per OUTDEV
OPENTI	F983	I	A,X,Y	Opens audio tape input. Asks for unit no. and file name. Searches for input file.
OPENTO	FBCC	O	A,X,Y	Opens audio tape output. Asks for unit no. and file name. Initializes output routine
OPNPTR	F8C8	O	A	Opens printer output
OUTALL	F32B	O		Converts the character in A to ASCII and sends it to the AOD
OUTLOW	F45B	O	A	Sets the AOD to the D/P
OUTPUT	F352	O		Converts the character in A to ASCII and sends it to the D/P
PNTKEY	F84A	O		Sends the displayed line to the PTR

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
PRINT	F35A	O		Sends the contents of A to the PTR by calling PUTPTR through vector IOVP
PSLS	F34B	O	A	Sends a slash to the D/P
PUTAPE	FBEE	O		Sends the contents of A to the audio tape output buffer
PUTDIS	F7D3	O		Sends the contents of A to the DPY
PUTPTR	F8F1	O		Sends the contents of A to the PTR (call OPNPTR before and CLOPTR after)
PUTSER	F972	O		Sends the contents of A to the RS-232C/TTY serial port
RDRUB	F2A8	I	A,Y	Calls INPUT and then tests for the DEL or CTRL H which will cause Y to be decremented (if non-zero) and the cursor to move one space to the left. If a character other than DEL or CTRL H in input, the character is sent to the D/P. Returns with the input character in A. Call with Ø to $7F in Y
READ	F22C	I	A	Inputs one character from the K/B via the K/S and returns with the ASCII value in A. If the K/S is empty, READ waits until a character is entered from the K/B
REDOUT	F29B	I	A	Inputs one character from the AID and outputs it to the AOD if is is a non-CTRL character. Returns with the character in A
RIGHT	F445	U	A	Shifts A four bits to the right
RSET	F12Ø	U	A,X,Y	If the CTRL key is not pressed, a warm reset is performed. If the CTRL key is pressed, a cold reset is performed when the CTRL key is released
SADDR	F4B7	B		Stores and verifies the contents of A into the RAM bank determined by vector ADDR and the offset from ADDR contained in the Y
SAVXY	F3FD	U		Pushes contents of X and Y onto the stack as the next-to-top and top values, respectively
SEMI	F329	O	A	Sends a semi-colon to the AOD

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
SETBNK	FFF3	B	A	Enables the memory bank in A (A = Ø, bank Ø; A = $Ø4, bank 1)
TOG	F2C8	O	A,X,Y	Toggles the Auto-Print state
TSERI	F96C	I		Tests the RS-232C/TTY input port for ready to receive Z: Z = 1, ready to input
TSERO	F97D	O	A	Tests the RS-232C/TTY output port for ready to transmit: Z = 1, ready to output
WAITD	F8ØC	I		Returns when ACK from the DPY is detected (if DPY is active)
WAITP	F935	I		Returns when ACK from the PTR is detected (if PTR is active). Displays PRINTER DOWN if PTR does not respond
WRAX	F3BA	O	A	Converts the contents of A then X (4 hex numbers) to ASCII (A first) and sends them to the D/P
WRAXA	F3AØ	O	A	Converts the contents of A and then X (4 hex numbers) to ASCII and sends them to the AOD
ZROBNK	FFEØ	B	A	Saves the present bank status and enables bank zero

K.3 MONITOR SUBROUTINES

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
ADDIN	AF27	I	A,X,Y	Sends "=" to D/P, inputs an address in hex or as a symbol and stores it in ADDR and ADDR+1. Returns with ADDR in X, ADDR+1 in Y and the terminator in A
A2DEC	BØ44	U	A	Converts dec number (Ø-9) in A from ASCII to binary and returns with it in A and C=Ø
A2HEX	BØ3C	U	A	Converts hex number (Ø-F) in A from ASCII to binary and returns with it in A and C=Ø
CLOSEQ	AE8D	O		Send EOF to AOD, then closes AOD
CRCLOS	AE84	O	A	Returns if AOD is D/P. Otherwise sends CR (and LF and nulls if OUTDEV is SP, P, S, V OR X) and EOF, then closes AOD
CRLF	AE5E	O		Sends a CR to the AOD. If OUTDEV = T, M, F or U, it returns. If OUTDEV = other (SP, P, S, V or X), also sends LF and the no. of nulls in NULL to AOD
CRNUL	AE67	O	A	Sends LF and the no. of nulls in NULL to the AOD
DELAY	BØBC	U	A,X,Y	Delays from 8Ø to 65,535 microseconds. Call with the delay value in X (MSB) and Y (LSB)
EQUAL	AE59	O	A	Sends "=" to the D/P
FROM	AE39	I	A,X,Y	Outputs "FROM=XXXX" to the D/P and enters an address from the K/B. Enter with old address in X (LSP) and Y (MSP)
FROMX	AE35	O	A,X,Y	Outputs "FROM=ØØØØ" to the D/P and enters an address from the K/B
HOMEA	AE7F	O	A	Sends a CR to the AOD
INPUTU	A399	I	A	Gets a character from the AID, forces letters to upper case and returns with it in A

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
NUMIN	AF95	I	A,X,Y	Sends "/" to the D/P, inputs a 4 digit decimal number, converts it to binary, stores the number in COUNT and COUNT+1. Returns with COUNT in X, COUNT+1 in Y and the input terminator in A
PACK	A844	U		Packs the LSP of A into BYTE. The LSP of BYTE is first shifted to the MSP
RCHEK	BØ8Ø	I	A	Returns if no key is on the K/S, otherwise processes key: SPACE Waits for next key Ø-9 Set VSPEED ESC Jumps through ESCIV Returns with input character in the A-Register (except for SPACE and ESC)
RDBYTE	A87Ø	I	A	If the AID is not audio tape, inputs two hex numbers from the AID, converts them to binary and returns with them in A (first number is the MSP of A).
RDBYTO	A95F	I	A	Gets a character from the AID, converts it to binary and packs it into BYTE, and sends it to the D/P.
RDNIBL	A85B	I	A	Gets a character from the AID and returns with it in A if it is a NULL, SP or hex number. Converts the number to binary and packs it into the LSP of BYTE.
QM	AE52	O	A	Sounds beeper and sends "?" to the D/P
SETOFF	ACA4	I	A,X,Y	Sends "OFFSET=ØØØØ" to the D/P and stores the entered address from the K/B in OFFSET (LSP) and OFFSET+1 (MSP). Returns with LSP in X and MSP in Y
TAGKIA	BØA9	I	A	Checks the K/S and returns with C=Ø is no key is available, otherwise returns with the key in A and C=1
TO	AE4B	I	A,X,Y	Sends "TO=XXXX" to the D/P and inputs an address from the K/B. Enter with old address in X (LSP) and Y (MSP)

Sub. Name	Entry Addr.	Type	Reg. Alt.	Function
TOX	AE47	I	A,X,Y	Outputs "TØ=ØØØØ" to the D/P and enters an address from the K/B
VISUAL	BØF4	U	X,Y	If the AOD is not interactive, delays VSPEED times 8 ms.
VRCHEK	BØ7D	I	A	Delays VSPEED times 8 ms then performs RCHEK. Returns input character in A.
WHEREI	AE9B	I	A,Y	Outputs "IN=" to the D/P and sets the AID with INDEV
WHEREO	AEB9	I	A,Y	Outputs "OUT=" to the D/P and sets the AOD with OUTDEV
WHERIS	AE95	I	A,Y	Outputs "OFFSET=ØØØØ" to D/P, saves entered value, outputs "IN=" to the D/P and sets the AID with INDEV
WRADBK	AE2D	O	A,X	Converts the address in ADDR and ADDR+1 to ASCII, sends it to the D/P and backspaces the cursor 4 positions
WRADXY	AE21	O	A,X	Loads X and Y with the address in ADDR and ADDR+1, converts it to ASCII and sends it to the D/P
WRITAD	AE24	O	A,X	Converts the address in ADDR and ADDR+1 to ASCII and sends it to the D/P

APPENDIX L

SBC MODULE CONNECTOR PIN ASSIGNMENTS

This appendix lists the signals assigned to the pins on the AIM 65/40 SBC module connectors. The signals are defined in the following sections which describe the operation and use of the associated functions:

Connector	Section	Table	
Parallel I/O	7.1	7-1	
RS-232C	8.1	8-1	
Audio/TTY	9.1	9-1	(Audio)
	10.1	10-1	(TTY)
Expansion	11.11	11-1	
Printer	12.2	12-5	
Display	13.2	13-5	
Keyboard	14.2	14-2	

Table L-1. SBC Connector J1 (Parallel I/O) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	CB2	Port B, Control No. 2	I/O
2	+5V	+5 Vdc	
3	CB1	Port B, Control No. 1	I/O
5	PB7	Port B, Bit 7	I/O
7	PB6	Port B, Bit 6	I/O
9	PB5	Port B, Bit 5	I/O
11	PB4	Port B, Bit 4	I/O
13	PB3	Port B, Bit 3	I/O
15	PB2	Port B, Bit 2	I/O
17	PB1	Port B, Bit 1	I/O
19	PBØ	Port B, Bit Ø	I/O
21	PA7	Port A, Bit 7	I/O
23	PA6	Port A, Bit 6	I/O
25	PA5	Port A, Bit 5	I/O
27	PA4	Port A, Bit 4	I/O
29	PA3	Port A, Bit 3	I/O
31	PA2	Port A, Bit 2	I/O
33	PA1	Port A, Bit 1	I/O
35	PAØ	Port A, Bit Ø	I/O
37	CA2	Port A, Control No. 2	I/O
39	CA1	Port A, Control No. 1	I
4Ø	+5V	+5 Vdc	

NOTES

1. Even numbered pins 4-38 are connected to GND.
2. +5 Vdc on pins 2 and 4Ø may be disconnected by removing jumper W1Ø.

BACK VIEW

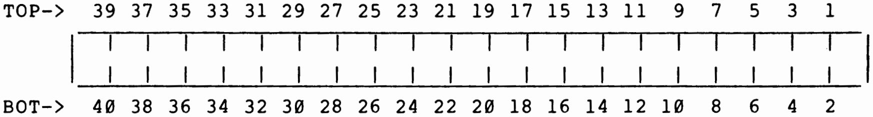

TOP-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1
BOT-> 4Ø 38 36 34 32 3Ø 28 26 24 22 2Ø 18 16 14 12 1Ø 8 6 4 2

Figure L-1. SBC Connector J1 (Parallel I/O) Pin Locations

Table L-2. SBC Connector J2 (RS-232C) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output	
			Data Set Opera.	Data Terminal Operation
1	GND	Chassis Ground		
2	$\overline{TD}$	Transmit Data	I	O
3	$\overline{RD}$	Receive Data	O	I
4	RTS	Request to Send	I	O
5	CTS	Clear to Send	O	I
6	DSR	Data Set Ready	O	I
7	GND	Signal Ground		
8	DCD	Data Carrier Detected	O	I
9-19		Not Used		
20	DTR	Data Terminal Ready	I	O
21-26		Not Used		

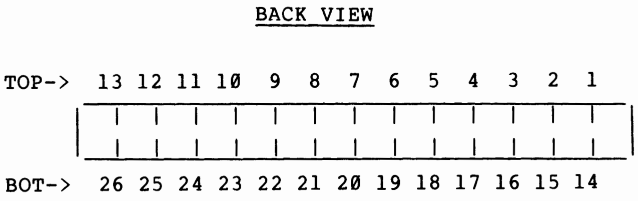

BACK VIEW
TOP-> 13 12 11 10 9 8 7 6 5 4 3 2 1
BOT-> 26 25 24 23 22 21 20 19 18 17 16 15 14

Figure L-2. SBC Connector J2 (RS-232C) Pin Locations

Table L-3. SBC Connector J3 (Audio/TTY) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	TTY RTS	Request to Send	I
3	TTY TD	Transmit Data	O
5	TTY RD	Receive Data	I
7	TTY RTN	-12 Vdc	
9	AUDIO OUT	Audio Output	O
11	AUDIO IN	Audio Input	I
13	CTRL 2 RTN	Control 2 Return	I
15	CTRL 2	Control 2	O
17	CTRL 1 RTN	Control 1 Return	I
19	CTRL 1	Control 1	O
NOTE Even numbered pins 2-2Ø are connected to GND.			

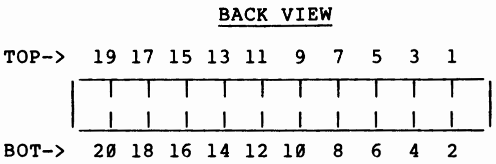

BACK VIEW
TOP-> 19 17 15 13 11 9 7 5 3 1
BOT-> 2Ø 18 16 14 12 1Ø 8 6 4 2

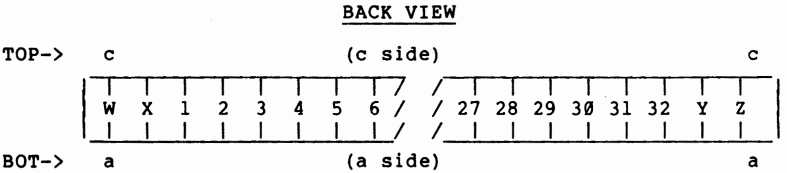
Figure L-3. SBC Connector J3 (Audio/TTY) Pin Locations

Table L-4. SBC Connector J4 (Expansion) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
Wa		Not Connected (See Figure L-4)	
Wc		Not Connected (See Figure L-4)	
Xa	+5V	+5 Vdc (See Figure L-4)	
Xc	+5V	+5 Vdc (See Figure L-4)	
la	GND	Ground	
lc	+5V	+5 Vdc	
2a	BADR/	Buffered Bank Address	I
2c	BA15/	Buffered Address Bit 15	I
3a	GND	Ground	
3c	BA14/	Buffered Address Bit 14	I
4a	BA13/	Buffered Address Bit 13	I
4c	BA12/	Buffered Address Bit 12	I
5a	BA11/	Buffered Address Bit 11	I
5c	GND	Ground	
6a	BA1Ø/	Buffered Address Bit 1Ø	I
6c	BA9/	Buffered Address Bit 9	I
7a	BA8/	Buffered Address Bit 8	I
7c	BA7/	Buffered Address Bit 7	I
8a	GND	Ground	
8c	BA6/	Buffered Address Bit 6	I
9a	BA5/	Buffered Address Bit 5	I
9c	BA4/	Buffered Address Bit 4	I
1Øa	BA3/	Buffered Address Bit 3	I
1Øc	GND	Ground	
11a	BA2/	Buffered Address Bit 2	I
11c	BA1/	Buffered Address Bit 1	I
12a	BAØ/	Buffered Address Bit Ø	I
12c		Not Used	
13a	GND	Ground	
13c		Not Used	
14a		Not Used	
14c	BDRQ1	Buffered DMA Request 1	O
15a		Not Used	
15c	GND	Ground	
16a		Not Used	
16c		Not Used	

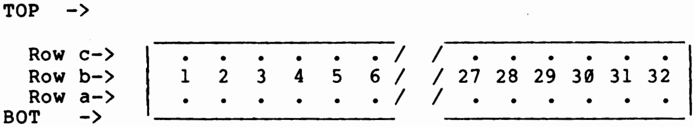
Table L-4. SBC Connector J4 (Expansion) Pin Assignments (Continued)

Pin	Signal Mnemonic	Signal Name	Input/Output
17a		Not Used	
17c		Not Used	
18a	GND	Ground	
18c		Not Used	
19a		Not Used	
19c		Not Used	
2Øa		Not Used	
2Øc	GND	Ground	
21a	BR/$\overline{W}$/	Buffered Read/Write "Not"	I
21c	BDRQ2/	Buffered DMA Request 2	O
22a		Not Used	
22c	BR/$\overline{W}$	Buffered Read/Write	I
23a	GND	Ground	
23c	BACT/	Buffered Bus Active	O
24a	BIRQ/	Buffered Interrupt Request	O
24c		Not Used	
25a	BØ2/	Buffered Phase 2 "Not" Clock	I
25c	GND	Ground	
26a	BØ2	Buffered Phase 2 Clock	I
26c	BRES/	Buffered Reset	I
27a	BD7/	Buffered Data Bit 7	I/O
27c	BD6/	Buffered Data Bit 6	I/O
28a	GND	Ground	
28c	BD5/	Buffered Data Bit 5	I/O
29a	BD4/	Buffered Data Bit 4	I/O
29c	BD3/	Buffered Data Bit 3	I/O
3Øa	BD2/	Buffered Data Bit 2	I/O
3Øc	GND	Ground	
31a	BD1/	Buffered Data Bit 1	I/O
31c	BDØ/	Buffered Data Bit Ø	I/O
32a	+5V	+5 Vdc	
32c	GND	Ground	
Ya	+5V	+5 Vdc (See Note)	
Yc	+5V	+5 Vdc (See Note)	
Za		Not Connected (See Figure L-4)	
Zc		Not Connected (See Figure L-4)	



Note: Pins Wa, Wc, Za and Zc are not connected.

a. AIM 65/4Ø SBC Module Edge Connector



Notes: 1. Row b is connected.
2. Pins Wa, Wc, Xa, Xc, Ya, Yc, Za and Zc are not provided on the Euroconnector.

b. User Provided Euroconnector

Figure L-4. Connector J4 (Expansion) Pin Locations

Table L-5. SBC Connector J5 (Printer) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	+5V	+5 Vdc	
2	NC		
3	NC		
5	NC		
7	NC		
9	NC		
11	NC		
13	NC		
15	$\overline{\text{PAPER FEED}}$	Paper Feed	O
17	$\overline{\text{RES}}$	Reset	O
19	$\overline{\text{STROBE}}$	Strobe	O
21	Data 7	Data Line 7	I/O
23	Data 6	Data Line 6	I/O
25	Data 5	Data Line 5	I/O
27	Data 4	Data Line 4	I/O
29	Data 3	Data Line 3	I/O
31	Data 2	Data Line 2	I/O
33	Data 1	Data Line 1	I/O
35	Data Ø	Data Line Ø	I/O
37	NC		
39	$\overline{\text{ACK}}$	Acknowledge	I
4Ø	+5V	+5 Vdc	

NOTE

Even numbered pins 4-34 are connected to GND.

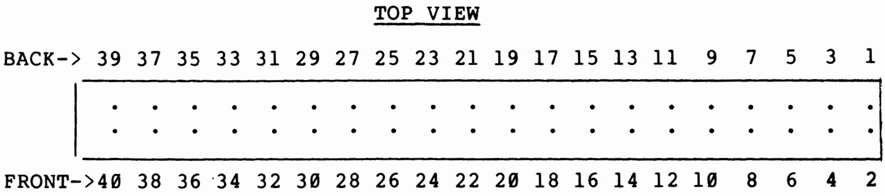

TOP VIEW
BACK-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1
FRONT->4Ø 38 36 34 32 3Ø 28 26 24 22 2Ø 18 16 14 12 1Ø 8 6 4 2

Figure L-5. SBC Connector J6 (Printer) Pin Locations

Table L-6. Connector J6 (Display) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	+5V	+5Vdc	
2	NC		
3	NC		
5	NC		
7	NC		
9	NC		
11	NC		
13	NC		
15	$\overline{\text{PAPER FEED}}$	Paper Feed	O
17	$\overline{\text{RES}}$	Reset	O
19	$\overline{\text{STROBE}}$	Strobe	O
21	Data 7	Data Line 7	I/O
23	Data 6	Data Line 6	I/O
25	Data 5	Data Line 5	I/O
27	Data 4	Data Line 4	I/O
29	Data 3	Data Line 3	I/O
31	Data 2	Data Line 2	I/O
33	Data 1	Data Line 1	I/O
35	Data Ø	Data Line Ø	I/O
37	NC		
39	$\overline{\text{ACK}}$	Acknowledge	I
4Ø	+5V	+5 Vdc	

NOTE

Even numbered pins 4-34 are connector to GND.

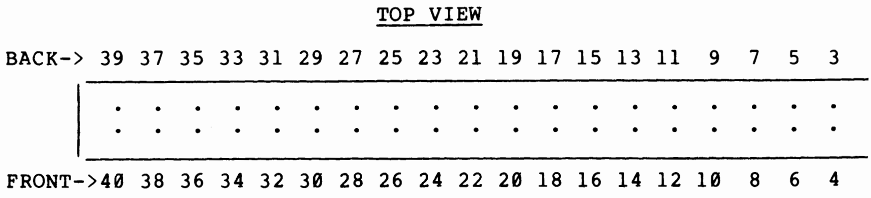

TOP VIEW
BACK-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3
FRONT->4Ø 38 36 34 32 3Ø 28 26 24 22 2Ø 18 16 14 12 1Ø 8 6 4

Figure L-6. SBC Connector J6 (Display) Pin Locations

Table L-7. SBC Connector J7 (Keyboard) Pin Assignments

Pin	Signal Mnemonic	Signal Name	Input/Output
1	$\overline{\text{RES SW}}$	Reset Switch	I
2	+5V	+5 Vdc (2)	
3	$\overline{\text{ATTN SW}}$	Attention Switch	I
5	MSB7	Matrix Strobe 7	O
7	MSB6	Matrix Strobe 6	O
9	MSB5	Matrix Strobe 5	O
11	MSB4	Matrix Strobe 4	O
13	MSB3	Matrix Strobe 3	O
15	MSB2	Matrix Strobe 2	O
17	MSB1	Matrix Strobe 1	O
19	MSB0	Matrix Strobe 0	O
21	MRT7	Matrix Return 7	I
23	MRT6	Matrix Return 6	I
25	MRT5	Matrix Return 5	I
27	MRT4	Matrix Return 4	I
29	MRT3	Matrix Return 3	I
31	MRT2	Matrix Return 2	I
33	MRT1	Matrix Return 1	I
35	MRT0	Matrix Return 0	I
37	MSB8	Matrix Return 8	I
39	$\overline{\text{PAPER FEED}}$	Paper Feed	
40	+5V	+5 Vdc (2)	

NOTES

1. Even numbered pins 4-38 are connected to GND.
2. The +5 Vdc may be disconnected on the SBC module by removing jumper W9.
3. The pin number assignments are reversed from the AIM 65/40 Keyboard connector to provide a one to one signal routing through a 40-conductor ribbon cable.

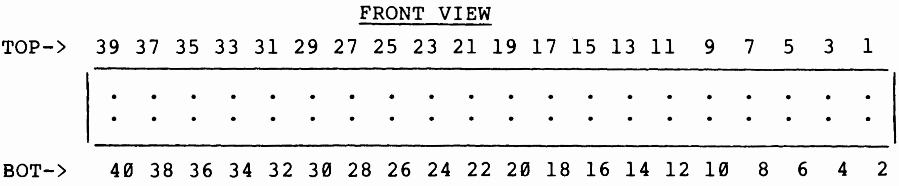

FRONT VIEW
TOP-> 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1
BOT-> 40 38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2

Figure L-7. SBC Connector J6 (Keyboard) Pin Locations

APPENDIX M

DUMB CRT TERMINAL DRIVER PROGRAM

This appendix describes a program to drive a dumb CRT terminal with functions similar to those included in an intelligent AIM 65/4Ø display assembly. This allows the CRT terminal to be used to control AIM 65/4Ø Monitor/Editor functions including the screen editing functions normally resident in the display controller.

This program illustrates use of the AIM 65/4Ø SBC module RS-232C port, I/O ROM subroutines, a function key (F1) to select the CRT operation, and decoding the CTRL X key return control to the AIM 65/4Ø keyboard.

Use the program as follows:

a. Assemble the program using the optional assembler as shown in the included assembly listing. You may want to tailor it to your own needs. The object code, as shown, starts at $38ØØ.

   While a function key is used in this example, you may want to use the auto-start function to select the CRT driver automatically upon RESET (if located at $8ØØØ or above, see Section 6.2).

   Save the source and object on mass media for subsequent update and use.

b. Connect the CRT terminal to SBC module connector J2 (see Table L-2 for the pin assignments).

c. Install the ACIA jumpers on the SBC module to select data set operation as follows (see Sections 2.3.3 and 2.3.4):

Jumper	Position
JACIA-1A through JACIA-1G	S
JACIA-2	GND
JACIA-3	GND
JACIA-4	GND

d. Select 9600 baud on the CRT terminal.

e. Load the driver object code (if not loaded in step a).

f. Press CTRL RESET if you want to perform a cold RESET before continuing.

g. Press the F1 key to switch control from the AIM 65/40 keyboard and display from the AIM 65/40 display to the CRT terminal. Enter subsequent commands from the terminal keyboard.

h. Press the CTRL X to return control from the CRT terminal to the AIM 65/40 keyboard and display to the AIM 65/40 display. Enter subsequent commands from the AIM 65/40 keyboard.

```
0002                     ;
0003 8000                PROG=$8000
0004 0800                VARS=$0800
0005                     ;
0006 0050                MAX    =      80
0007 0018                MAXLIN =      24         ;SCREEN LINE COUNT
0008 FFD0                ACIA   =      $FFD0
0009                     ;
0010                     ; AIM 65/40 ROUTINES
0011                     ;
0012 0200                INVEC  =      $0200
0013 0202                OUTVEC =      $0202
0014 0220                IOVDT  =      $0220
0015 0231                TSTKEY =      $0231      ;GET KB STATUS
0016 027B                ESCKEY =      $27B       ;ESC FLAG
0017                     ;
0018 F06E                IOOK   =      $F06E
0019 F593                A2STAK =      $F593
0020 F5B7                GETDKE =      $F5B7
0021 F963                GETSER =      $F963
0022 F972                PUTSER =      $F972
0023 F96C                TSERI  =      $F96C      ;KB STATUS ROUTINE
0024 FFCE                KBIER  =      $FFCE
0025                     ;
0026                     ; DRIVER RAM VARIABLES
0027                     ;
0028 0000                       *=VARS
0029 0800                BUFFER *=*+MAX+1
0030 0851                CURPOS *=*+1
0031 0852                DLEFLG *=*+1             ;DLE INPUT FLAG
0032 0853                DLEF   *=*+1             ;DLE XMIT FLAG
0033 0854                FLAGS  *=*+1             ;INSERT FLAG
0034 0855                EOL    *=*+1
0035 0856                BASE   *=*+2
0036 0858                SAVEA  *=*+1
0037 0859                SAVEX  *=*+1
0038 085A                SAVEY  *=*+1
0039 085B                SAVES  *=*+1
0040 085C                PEND   *=*+1             ;SEQUENCE PENDING FLAG
0041 085D                OLDCUR *=*+1
0042 085E                NEXT   *=*+1             ;BUFFER DATA POINTER
0043 085F                CFLAG  *=*+1             ;BUFFER NOT EMPTY FLAG
0044 0860                CURSOR *=*+1
0045 0861                TERM   *=*+1             ;XMIT TERM FLAG
0046 0862                LINE   *=*+1             ;CURRENT LINE NO.
0047 0863                TLINE  *=*+1             ;LINE COUNTER
0048
```

```
0050 0864                         *=PROG
0051                       ;
0052                       ; AUTOSTART SEQUENCE -- END WITH RTS
0053                       ;
0054 8000    8F                  .BYT    $8F         ;PROM ID
0055 8001    5A                  .BYT    $5A,$A5     ;AUTOSTART KEY
0056                       ;
0057 8003    AD CE FF      START  LDA     KBIER
0058 8006    29 02               AND     #%00000010
0059 8008    F0 F9               BEQ     START       ;WAIT FOR NO KEY DOWN
0060 800A    AD CE FF            LDA     KBIER
0061 800D    29 FD               AND     #%11111101
0062 800F    8D CE FF            STA     KBIER       ;DISABLE KB IRQ
0063                       ;
0064 8012    A9 08               LDA     #<OTAB      ;INSERT XMIT BACK VECTOR
0065 8014    8D 20 02            STA     IOVDT
0066 8017    A9 83               LDA     #>OTAB
0067 8019    8D 21 02            STA     IOVDT+1
0068                       ;
0069 801C    A9 FF               LDA     #<DTAB      ;SWITCH OUTPUT VECTOR
0070 801E    8D 02 02            STA     OUTVEC
0071 8021    A9 82               LDA     #>DTAB
0072 8023    8D 03 02            STA     OUTVEC+1
0073                       ;
0074 8026    A9 F6               LDA     #<ITAB      ;SWITCH INPUT TO SERIAL PORT
0075 8028    8D 00 02            STA     INVEC
0076 802B    A9 82               LDA     #>ITAB
0077 802D    8D 01 02            STA     INVEC+1
0078                       ;
0079 8030    A9 6C               LDA     #<TSERI     ;SWITCH 'TEST FOR KEY' ROUTINE
0080 8032    8D 31 02            STA     TSTKEY
0081 8035    A9 F9               LDA     #>TSERI
0082 8037    8D 32 02            STA     TSTKEY+1
0083                       ;
0084 803A    A9 0B               LDA     #$0B        ;DTR=L,IRQ=OFF,REM=NORM
0085 803C    8D D2 FF            STA     $FFD2       ;ACIA COMMAND REG
0086 803F    A9 1E               LDA     #$1E        ;8-BITS,1-STOP,9600 BAUD
0087 8041    8D D3 FF            STA     $FFD3       ;ACIA CONTROL REG
0088                       ;
0089 8044    A9 00               LDA     #0          ;INIT 'SEQUENCE PENDING' VARIA
0090 8046    8D 5C 08            STA     PEND
0091 8049    A9 19               LDA     #$19        ;SEND COLD START TO CRT ROUTIN
0092                       ;************************************
```

```
0094                        ; MAIN OUTPUT ROUTINE
0095 804B    8D 58 08       PROCES STA     SAVEA       ;SAVE CHARACTER
0096 804E    8E 59 08              STX     SAVEX
0097 8051    8C 5A 08              STY     SAVEY
0098 8054    BA                    TSX
0099 8055    8E 5B 08              STX     SAVES
0100 8058    AD 5C 08              LDA     PEND        ;SEE IF SEQUENCE PENDING
0101 805B    F0 0E                 BEQ     ONECHR      ;NO SEQUENCE
0102 805D    38                    SEC
0103 805E    E9 01                 SBC     #1
0104 8060    0A                    ASL     A
0105 8061    AA                    TAX
0106 8062    BD 93 83              LDA     SEQTBL,X
0107 8065    BC 94 83              LDY     SEQTBL+1,X
0108 8068    4C 85 80              JMP     BDONE
0109                        ;
0110 806B    AD 58 08       ONECHR LDA     SAVEA
0111 806E    C9 20                 CMP     #$20        ;CONTROL CHAR ?
0112 8070    90 06                 BCC     CNTL        ;YES
0113                        ;
0114                        ; VISIBLE CHARACTER
0115                        ;
0116 8072    20 A2 80       VISIX  JSR     VISI        ;JSR SO OTHERS CAN USE
0117 8075    4C 91 80              JMP     DONE
0118                        ;
0119                        ; PROCESS CONTROL CHARACTERS
0120                        ;
0121 8078    2C 52 08       CNTL   BIT     DLEFLG      ;DLE SEQUENCE ?
0122 807B    30 F5                 BMI     VISIX       ;YES - PUT IN BUFFER
0123 807D    0A                    ASL     A
0124 807E    AA                    TAX
0125 807F    BD 9F 83              LDA     JTBL,X
0126 8082    BC A0 83              LDY     JTBL+1,X
0127 8085    8D 56 08       BDONE  STA     BASE
0128 8088    8C 57 08              STY     BASE+1
0129 808B    AD 58 08              LDA     SAVEA
0130 808E    20 9F 80              JSR     JIND        ;SIMULATE JSR (HHHH)
0131                        ;
0132 8091    AE 5B 08       DONE   LDX     SAVES
0133 8094    9A                    TXS
0134 8095    AD 58 08              LDA     SAVEA
0135 8098    AC 5A 08       SAMEXY LDY     SAVEY
0136 809B    AE 59 08              LDX     SAVEX
0137 809E    60                    RTS                 ; RETURN TO MONITOR
0138                        ;
0139 809F    6C 56 08       JIND   JMP     (BASE)
```

```
0141                          ;
0142                          ; VISIBLE CHARACTER PROCESSING SUBROUTINES
0143                          ;
0144 80A2   A9 00             VISI    LDA     #0
0145 80A4   8D 52 08                  STA     DLEFLG      ;CLEAR DLE FLAG
0146 80A7   AE 51 08                  LDX     CURPOS
0147 80AA   E0 4F                     CPX     #MAX-1
0148 80AC   B0 3C                     BCS     VEXIT
0149 80AE   AD 54 08                  LDA     FLAGS
0150 80B1   10 1D                     BPL     STORE1      ;NOT IN INSERT MODE
0151 80B3   AD 4E 08                  LDA     BUFFER+MAX-2  ;SEE IF LAST CHAR IS A BLA
0152 80B6   C9 20                     CMP     #$20
0153 80B8   D0 30                     BNE     VEXIT
0154 80BA   A0 4E                     LDY     #MAX-2      ;INSERT A CHARACTER
0155 80BC   B9 FF 07          ILOOP   LDA     BUFFER-1,Y
0156 80BF   99 00 08                  STA     BUFFER,Y
0157 80C2   88                        DEY
0158 80C3   CC 51 08                  CPY     CURPOS
0159 80C6   F0 02                     BEQ     INSRT2
0160 80C8   B0 F2                     BCS     ILOOP
0161 80CA   20 CF 81          INSRT2  JSR     SHOWL
0162 80CD   AE 51 08                  LDX     CURPOS
0163                          ;
0164 80D0   AD 58 08          STORE1  LDA     SAVEA       ;STORE A CHARACTER ON BUFFER
0165 80D3   9D 00 08                  STA     BUFFER,X    ;  STORE CHARACTER
0166 80D6   A9 80                     LDA     #$80
0167 80D8   8D 5F 08                  STA     CFLAG       ;SET NOT EMPTY
0168 80DB   AD 58 08                  LDA     SAVEA
0169 80DE   C9 20                     CMP     #$20        ;CONTROL CHAR ?
0170 80E0   10 02                     BPL     STORE3      ;NO
0171 80E2   A9 20                     LDA     #$20        ;YES - GET SPACE
0172 80E4   20 72 F9          STORE3  JSR     SEROUT
0173 80E7   EE 51 08                  INC     CURPOS
0174 80EA   60                VEXIT   RTS
```

```
                   ;
                   ; COLD RESET
                   ;
80EB   A9 00       COLD    LDA    #0
80ED   8D 5C 08            STA    PEND
                   ;
                   ; CLEAR AND HOME SCREEN
                   ;
80F0   A2 17       CLEAN   LDX    #MAXLIN-1
80F2   8E 62 08            STX    LINE
80F5   A9 0A       CLEAN1  LDA    #$0A
80F7   20 72 F9            JSR    SEROUT     ;LINE FEED
80FA   CA                  DEX
80FB   D0 F8               BNE    CLEAN1
                   ;
                   ; HOME ON SCREEN
                   ;
80FD   20 FF 81    HOME    JSR    UP         ;MOVE UP ONE LINE
8100   AD 62 08            LDA    LINE       ;AT TOP YET?
8103   D0 F8               BNE    HOME       ;NO
                   ;
                   ; HOME ON LINE
                   ;
8105   A9 0D       CR      LDA    #$0D       ;CARRIAGE RETURN
8107   20 72 F9            JSR    SEROUT
810A   A9 00               LDA    #0
810C   8D 51 08            STA    CURPOS
810F   8D 52 08            STA    DLEFLG
8112   8D 53 08            STA    DLEF
8115   8D 54 08            STA    FLAGS
8118   60                  RTS
                   ;
                   ; HOME AND CLEAR TOP LINE
                   ;
8119   20 FD 80    HOMCLR  JSR    HOME
811C   20 AB 81            JSR    CEOL6      ;CLEAR BUFFER
811F   4C 45 81            JMP    CLRLIN ;   NOW CLEAR CRT LINE
```

```
                    ;
                    ;  INSERT A LINE
                    ;
8122   20 FF 81     INSLIN JSR    UP
8125   20 45 81            JSR    CLRLIN
8128   4C 05 81            JMP    CR
                    ;
                    ;  DELETE A LINE
                    ;
812B   20 45 81     DELINE JSR    CLRLIN     ;REMOVE LINE FROM SCREEN
                    ;
                    ;  LINE FEED
                    ;
812E   A9 00        LF     LDA    #0
8130   8D 5F 08            STA    CFLAG      ;INDICATE PHYSICAL LINE CLEAR
8133   AD 62 08            LDA    LINE
8136   C9 17               CMP    #MAXLIN-1 ;BOTTOM OF PAGE?
8138   B0 03               BCS    LF1        ;NO
813A   EE 62 08            INC    LINE
813D   20 AB 81     LF1    JSR    CEOL6      ;LF CLEARS BUFFER
8140   A9 0A               LDA    #$0A       ;SPECIAL LINE FEED
8142   20 72 F9            JSR    SEROUT
                    ;
                    ;  CLEAR LINE
                    ;
8145   AE 51 08     CLRLIN LDX    CURPOS
8148   E0 4F        CLRL1  CPX    #MAX-1
814A   90 0C               BCC    CLRL2      ;NOT AT END OF LINE
814C   A2 00               LDX    #0
814E   A9 0D               LDA    #$0D
8150   20 72 F9            JSR    SEROUT     ;MOVE TO START OF LINE
8153   EC 51 08            CPX    CURPOS     ;BACK TO WHERE WE STARTED?
8156   F0 0B               BEQ    CLRL3      ;YES
8158   A9 20        CLRL2  LDA    #$20
815A   20 72 F9            JSR    SEROUT     ;CLEAR CHARACTER
815D   E8                  INX               ;COUNT IT
815E   EC 51 08            CPX    CURPOS     ;BACK TO WHERE WE STARTED?
8161   D0 E5               BNE    CLRL1      ;NO
8163   60           CLRL3  RTS
                    ;
                    ;  BACK SPACE
                    ;
8164   AE 51 08     BS     LDX    CURPOS
8167   CA                  DEX
8168   30 F9               BMI    CLRL3
816A   8E 51 08            STX    CURPOS
816D   A9 08               LDA    #$08
816F   4C 72 F9            JMP    SEROUT
                    ;
                    ;  FORE SPACE
                    ;
8172   AE 51 08     FS     LDX    CURPOS
8175   E0 4F               CPX    #MAX-1
8177   B0 EA               BCS    CLRL3
8179   E8                  INX
```

```
0269 817A    8E 51 08               STX     CURPOS
0270 817D    A9 0C                  LDA     #$0C
0271 817F    4C 72 F9               JMP     SEROUT
0272                         ;
0273                         ;  CLEAR THIS LINE
0274                         ;
0275 8182    20 05 81        CLEAR  JSR     CR          ;CLEAR BUFFER ENTRY POINT
0276                         ;
0277                         ;  CLEAR TO END OF LINE
0278                         ;
0279 8185    2C 5F 08        CEOL   BIT     CFLAG       ;SEE IF PHYSICAL LINE CLEAR
0280 8188    10 21                  BPL     CEOL6       ;YES-END OF LINE
0281 818A    20 C0 82               JSR     ENDOL       ;FIND END OF LINE
0282 818D    E8                     INX
0283 818E    8E 55 08               STX     EOL
0284 8191    A9 20           CEOL3  LDA     #$20
0285 8193    EC 51 08               CPX     CURPOS
0286 8196    F0 06                  BEQ     CEOL4
0287 8198    20 72 F9               JSR     SEROUT      ;CLEAR TERMINAL
0288 819B    CA                     DEX
0289 819C    10 F3                  BPL     CEOL3
0290 819E    A9 08           CEOL4  LDA     #$08        ;BACKSPACE
0291 81A0    EC 55 08               CPX     EOL
0292 81A3    B0 06                  BCS     CEOL6
0293 81A5    20 72 F9               JSR     SEROUT      ;MOVE CURSOR BACK
0294 81A8    E8                     INX
0295 81A9    10 F3                  BPL     CEOL4
0296 81AB    A9 20           CEOL6  LDA     #$20        ;CLEAR BUFFER TO END
0297 81AD    AE 51 08               LDX     CURPOS
0298 81B0    9D 00 08        LOOP   STA     BUFFER,X
0299 81B3    E8                     INX
0300 81B4    E0 50                  CPX     #MAX
0301 81B6    90 F8                  BCC     LOOP
0302 81B8    60                     RTS
0303                         ;
0304                         ;  DELETE A CHARACTER
0305                         ;
0306 81B9    AE 51 08        DELETE LDX     CURPOS      ;DELETE CHARACTER
0307 81BC    E0 4F           DLOOP  CPX     #MAX-1
0308 81BE    B0 0A                  BCS     DOUT
0309 81C0    BD 01 08               LDA     BUFFER+1,X
0310 81C3    9D 00 08               STA     BUFFER,X
0311 81C6    E8                     INX
0312 81C7    4C BC 81               JMP     DLOOP
0313                         ;
0314 81CA    A9 20           DOUT   LDA     #$20
0315 81CC    9D 00 08               STA     BUFFER,X
0316 81CF    20 C0 82        SHOWL  JSR     ENDOL       ;FIND END OF LINE
0317 81D2    E8                     INX                 ; SHOW ONE EXTRA FOR DELETE
0318 81D3    8E 55 08               STX     EOL
0319 81D6    AE 51 08               LDX     CURPOS
0320 81D9    EC 55 08        SHOW3  CPX     EOL         ;SEE IF DONE
0321 81DC    B0 09                  BCS     SHOW4
0322 81DE    BD 00 08               LDA     BUFFER,X    ; GET A CHARACTER
0323 81E1    20 72 F9               JSR     SEROUT
```

```
81E4   E8                    INX
81E5   10 F2                 BPL     SHOW3
                      ;
81E7   8E 5D 08       SHOW4  STX     OLDCUR
81EA   AD 51 08              LDA     CURPOS
81ED   4C 8D 82              JMP     SETCUR
                      ;
                      ; TOGGLE INSERT  MODE
                      ;
81F0   AD 54 08       TINSRT LDA     FLAGS       ;TOGGLE INSERT MODE
81F3   49 80                 EOR     #$80
81F5   8D 54 08              STA     FLAGS
81F8   60                    RTS
                      ;
                      ; DOWN LINK ESCAPE
                      ;
81F9   A9 80          DLE    LDA     #$80
81FB   8D 52 08              STA     DLEFLG      ;SET DLE SEQUENCE
81FE   60                    RTS
                      ;
                      ; MOVE CURSOR UP ONE LINE
                      ;
81FF   CE 62 08       UP     DEC     LINE        ;RECORD LINE POSITION
8202   10 04                 BPL     UPA         ;NOT TO TOP OF SCREEN YET
8204   EE 62 08              INC     LINE
8207   60                    RTS
                      ;
8208   A9 0B          UPA    LDA     #$0B        ;UP A LINE
820A   4C 72 F9              JMP     SEROUT
                      ;
                      ; CLEAR CURRENT LINE TO BOTTOM OF SCREEN
                      ;
820D   AD 62 08       BOTOFF LDA     LINE
8210   8D 63 08              STA     TLINE       ;FOR USE AS A COUNTER
8213   20 45 81              JSR     CLRLIN ;    NOW CLEAR IT
8216   AD 62 08       BOT1   LDA     LINE
8219   C9 17                 CMP     #MAXLIN-1   ;AT BOTTOM?
821B   F0 06                 BEQ     BOT2        ;YES
821D   20 2E 81              JSR     LF          ;CLEAR NEXT LINE
8220   4C 16 82              JMP     BOT1
                      ;
8223   AD 62 08       BOT2   LDA     LINE
8226   CD 63 08              CMP     TLINE       ;BACK TO STARTING LINE?
8229   F0 06                 BEQ     BOT3        ;YES
822B   20 FF 81              JSR     UP          ;STEP UP ONELINE
822E   4C 23 82              JMP     BOT2
                      ;
8231   4C 05 81       BOT3   JMP     CR          ;NOW HOME
                      ;
                      ; GO TO BOTTOM LINE ON SCREEN
                      ;
8234   AD 62 08       BOTTOM LDA     LINE
8237   C9 17                 CMP     #MAXLIN-1   ;AT BOTTOM YET?
8239   F0 0B                 BEQ     BOTT1       ;YES
823B   A9 0A                 LDA     #$0A
```

```
0379 823D    20 72 F9             JSR     SEROUT     ;LINE FEED
0380 8240    EE 62 08             INC     LINE
0381 8243    4C 34 82             JMP     BOTTOM
0382                      ;
0383 8246    4C 05 81     BOTT1   JMP     CR         ;HOME ON THIS LINE
```

```
                      ;
8249  A9 01           ESCAPE LDA     #01          ;ESCAPE SEQUENCE
824B  4C 68 82               JMP     ESC5
                      ;
                      ; CHARACTER AFTER ESCAPE
824E  C9 58           ESC1   CMP     #'X          ;TRANSMIT LINE
8250  F0 24                  BEQ     EATONE
8252  C9 3D                  CMP     #'=          ;TAB
8254  F0 16                  BEQ     ESCEQ
8256  C9 45                  CMP     #'E          ;SEQ, TERM WITH CR
8258  F0 55                  BEQ     ESCE
825A  C9 57                  CMP     #'W
825C  F0 18                  BEQ     EATONE
825E  C9 4C                  CMP     #'L
8260  F0 14                  BEQ     EATONE
8262  C9 55                  CMP     #'U
8264  F0 10                  BEQ     EATONE
8266  A9 00           ESC4   LDA     #0           ;RESET ESCAPE SEQUENCE
8268  8D 5C 08        ESC5   STA     PEND
826B  60                     RTS
                      ;
826C  A9 02           ESCEQ  LDA     #02          ;SET TO GET Y(VERT TAB)
826E  4C 68 82               JMP     ESC5
                      ;
8271  A9 03           ESCEQY LDA     #03          ;THROW AWAY VERT TAB
8273  4C 68 82               JMP     ESC5         ;SET FOR X(HORZ TAB)
                      ;
8276  A9 06           EATONE LDA     #6           ;THROW AWAY CHAR
8278  4C 68 82               JMP     ESC5
                      ;
827B  AE 51 08        ESCEQX LDX     CURPOS       ;HORZ TAB
827E  8E 5D 08               STX     OLDCUR
8281  38                     SEC
8282  E9 20                  SBC     #32
8284  C9 4F                  CMP     #MAX-1
8286  90 02                  BCC     NOTP
8288  A9 4F                  LDA     #MAX-1
828A  8D 51 08        NOTP   STA     CURPOS
828D  38              SETCUR SEC                  ; SEE WHICH WAY TO MOVE
828E  ED 5D 08               SBC     OLDCUR
8291  10 0E                  BPL     FORW
8293  A8                     TAY                  ; GET COUNT INTO Y
8294  A9 08           BACK1  LDA     #$08         ;MOVE BACK
8296  C0 00                  CPY     #0           ;SEE IF DONE
8298  F0 CC                  BEQ     ESC4
829A  20 72 F9               JSR     SEROUT
829D  C8                     INY
829E  4C 94 82               JMP     BACK1
                      ;
82A1  A8              FORW   TAY                  ; MOVE FORWARD
82A2  A9 0C           FORW1  LDA     #$0C
82A4  C0 00                  CPY     #0
82A6  F0 BE                  BEQ     ESC4
82A8  20 72 F9               JSR     SEROUT
82AB  88                     DEY
```

```
0440 82AC    4C A2 82                 JMP     FORW1
0441                          ;
0442 82AF    A9 04            ESCE    LDA     #04
0443 82B1    4C 68 82                 JMP     ESC5
0444                          ;
0445 82B4    A9 05            ESCES   LDA     #05
0446 82B6    4C 68 82                 JMP     ESC5
0447                          ;
0448 82B9    C9 0D            ESCESW  CMP     #$0D
0449 82BB    D0 18                    BNE     RTSINS
0450 82BD    4C 66 82                 JMP     ESC4
0451                          ;
0452                          ;  FIND END OF LINE
0453                          ;
0454 82C0    A2 4F            ENDOL   LDX     #MAX-1
0455 82C2    BD 00 08         EN1     LDA     BUFFER,X
0456 82C5    C9 20                    CMP     #$20
0457 82C7    D0 03                    BNE     EN2
0458 82C9    CA                       DEX
0459 82CA    10 F6                    BPL     EN1
0460 82CC    E8               EN2     INX
0461                          ;  ALLOW FOR A BLANK LINE
0462 82CD    D0 06                    BNE     RTSINS      ;NOT BLANK
0463 82CF    2C 5F 08                 BIT     CFLAG
0464 82D2    10 01                    BPL     RTSINS      ;LINE IS EMPTY
0465 82D4    E8                       INX
0466 82D5    60               RTSINS  RTS
```

CONTROL CHARACTER PROCESSING

```
0468                        ;
0469                        ; SERIAL OUT ROUTINE
0470                        ;
0471 F972                   SEROUT =      PUTSER
0472                        ;
0473                        ; SERIAL IN ROUTINE
0474                        ;
0475 82D6   8E 59 08        DMDKEY STX    SAVEX
0476 82D9   8C 5A 08               STY    SAVEY
0477 82DC   20 6C F9        DMDKE  JSR    TSERI      ;CHAR RDY ?
0478 82DF   F0 FB                  BEQ    DMDKE      ;NO
0479 82E1   20 63 F9               JSR    GETSER     ;GET CHAR
0480 82E4   29 7F                  AND    #$7F       ;REMOVE PARITY BIT
0481 82E6   C9 1B                  CMP    #$1B       ;ESC ?
0482 82E8   D0 03                  BNE    DMD1       ;NO
0483 82EA   8D 7B 02               STA    ESCKEY     ;YES-TELL MONITOR
0484 82ED   20 93 F5        DMD1   JSR    A2STAK     ;PUT ON STACK
0485 82F0   20 98 80               JSR    SAMEXY
0486 82F3   4C B7 F5               JMP    GETDKE     ;DECODE CHAR
0487                        ;
0488 82F6   4C 6E F0        ITAB   JMP    IOOK
0489 82F9   4C D6 82               JMP    DMDKEY     ;INPUT VECTOR TABLE
0490 82FC   4C 6E F0               JMP    IOOK
0491                        ;
0492 82FF   4C 6E F0        DTAB   JMP    IOOK
0493 8302   4C 4B 80               JMP    PROCES     ;OUT VECTOR TABLE
0494 8305   4C 6E F0               JMP    IOOK
```

```
                    ; TRANSMIT BACK TO HOST ROUTINES
                    ;
8308    4C 11 83    OTAB    JMP     OPENT
830B    4C 2B 83            JMP     READT
830E    4C 6A 83            JMP     CLOSET
                    ;
                    ; 'OPEN' ROUTINE
                    ;
8311    8D 58 08    OPENT   STA     SAVEA
8314    8E 59 08            STX     SAVEX
8317    20 C0 82            JSR     ENDOL       ;FIND END OF LINE
831A    8E 55 08            STX     EOL
831D    A2 00               LDX     #0          ;RESET TO BUFFER START
831F    8E 61 08            STX     TERM        ;CL TERMINATION FLAG
8322    8E 53 08            STX     DLEF
8325    AD 58 08            LDA     SAVEA
8328    4C 63 83            JMP     ROUT
                    ;
                    ; READ ONE CHARACTER FROM DISPLAY ROUTINE
                    ;
832B    8E 59 08    READT   STX     SAVEX
832E    AD 61 08            LDA     TERM        ;TERMINATION SEQ ?
8331    D0 38               BNE     READ3       ;YES
8333    AE 5E 08            LDX     NEXT
8336    EC 55 08            CPX     EOL         ;END OF LINE ?
8339    90 09               BCC     READ1       ;NO
833B    A9 0F               LDA     #$0F
833D    8D 61 08            STA     TERM        ;SET TERM FLAG
8340    A9 00       READ6   LDA     #0          ;GET NULL (END OF LINE)
8342    F0 22               BEQ     READ2
8344    BD 00 08    READ1   LDA     BUFFER,X    ;GET CHAR
8347    C9 20               CMP     #$20        ;CONTROL CHAR ?
8349    10 17               BPL     READ7       ;NO
834B    2C 53 08            BIT     DLEF        ;DLE SEQUENCE ?
834E    10 08               BPL     READ8       ;NO - START ONE
8350    A0 00               LDY     #0
8352    8C 53 08            STY     DLEF        ;CLEAR DLE FLAG
8355    4C 62 83            JMP     READ7       ;SEND CONTROL CHAR
                    ;
8358    A9 80       READ8   LDA     #$80
835A    8D 53 08            STA     DLEF        ;SET DLE SEQUENCE
835D    A9 10               LDA     #$10        ;DLE CHAR
835F    4C 66 83            JMP     READ2       ;SEND IT
                    ;
8362    E8          READ7   INX
8363    8E 5E 08    ROUT    STX     NEXT        ;SAVE NEXT CHAR INDEX
8366    AE 59 08    READ2   LDX     SAVEX
8369    A8                  TAY                 ;SET ZERO STATUS
836A    60          CLOSET  RTS
                    ;
                    ; TERMINATION SEQUENCE
                    ;
836B    C9 0F       READ3   CMP     #$0F
836D    D0 08               BNE     READ4
836F    CE 61 08            DEC     TERM
```

```
0551 8372   A9 1A                LDA    #$1A     ;EOT
0552 8374   4C 66 83             JMP    READ2
0553                     ;
0554 8377   C9 0E        READ4   CMP    #$0E
0555 8379   D0 08                BNE    READ5
0556 837B   CE 61 08             DEC    TERM
0557 837E   A9 18                LDA    #$18     ;ROW 24
0558 8380   4C 66 83             JMP    READ2
0559                     ;
0560 8383   C9 0D        READ5   CMP    #$0D
0561 8385   D0 B9                BNE    READ6    ;SEQUENCE DONE
0562 8387   CE 61 08             DEC    TERM
0563 838A   AD 51 08             LDA    CURPOS   ;GET CURSOR INDEX
0564 838D   18                   CLC
0565 838E   69 20                ADC    #$20
0566 8390   4C 66 83             JMP    READ2
```

```
0568                    ; FIRMWARE TABLES
0569                    ;
0570                    ; JUMP TABLE FOR SEQUENCES
0571                    ;
0572 8393   4E 82       SEQTBL .WORD  ESC1
0573 8395   71 82              .WORD  ESCEQY   ;=Y
0574 8397   7B 82              .WORD  ESCEQX   ;=X
0575 8399   B4 82              .WORD  ESCES    ;E
0576 839B   B9 82              .WORD  ESCESW   ;E TILL CR
0577 839D   66 82              .WORD  ESC4     ;EATONE
0578                    ;
0579                    ; JUMP TABLE FOR CONTROL CHARS
0580                    ;
0581 839F   D5 82       JTBL   .WORD  RTSINS   ;CNTL-@
0582 83A1   82 81       C01    .WORD  CLEAR    ;CNTL-A
0583 83A3   85 81       C02    .WORD  CEOL
0584 83A5   F0 80       C03    .WORD  CLEAN
0585 83A7   0D 82       C04    .WORD  BOTOFF
0586 83A9   F0 80       C05    .WORD  CLEAN    ;CNTL-E
0587 83AB   0D 82       C06    .WORD  BOTOFF
0588 83AD   D5 82       C07    .WORD  RTSINS
0589 83AF   64 81       C08    .WORD  BS
0590 83B1   72 81       C09    .WORD  FS
0591 83B3   2E 81       C0A    .WORD  LF       ;CNTL-J
0592 83B5   FF 81       C0B    .WORD  UP
0593 83B7   D5 82       C0C    .WORD  RTSINS
0594 83B9   05 81       C0D    .WORD  CR
0595 83BB   19 81       C0E    .WORD  HOMCLR
0596 83BD   19 81       C0F    .WORD  HOMCLR   ;CNTL-O
0597 83BF   F9 81       C10    .WORD  DLE
0598 83C1   D5 82       C11    .WORD  RTSINS
0599 83C3   D5 82       C12    .WORD  RTSINS
0600 83C5   F0 81       C13    .WORD  TINSRT
0601 83C7   B9 81       C14    .WORD  DELETE   ;CNTL-T
0602 83C9   22 81       C15    .WORD  INSLIN
0603 83CB   2B 81       C16    .WORD  DELINE
0604 83CD   D5 82       C17    .WORD  RTSINS
0605 83CF   D5 82       C18    .WORD  RTSINS
0606 83D1   EB 80       C19    .WORD  COLD     ;CNTL-Y
0607 83D3   EB 80       C1A    .WORD  COLD     ;WARM ?
0608 83D5   49 82       C1B    .WORD  ESCAPE   ;CNTL-[
0609 83D7   2E 81       C1C    .WORD  LF       ;CNTL-\
0610 83D9   34 82       C1D    .WORD  BOTTOM   ;CNTL-]
0611 83DB   FF 81       C1E    .WORD  UP       ;CNTL-^
0612 83DD   D5 82       C1F    .WORD  RTSINS
0613                           .END

ERRORS = 0000 <0000>
```

```
SYMBOL TABLE


ACIA   FFD0    #0008
A2STAK F593    #0019  0484
BACK1  8294    #0427  0432
BASE   0856    #0035  0127  0128  0139
BDONE  8085     0108 #0127
BOTOFF 820D    #0356  0585  0587
BOTTOM 8234    #0375  0381  0610
BOTT1  8246     0377 #0383
BOT1   8216    #0359  0363
BOT2   8223     0361 #0365  0369
BOT3   8231     0367 #0371
BS     8164    #0256  0589
BUFFER 0800    #0029  0151  0155  0156  0165  0298  0309  0310  0315  0322
                0455  0526
CEOL   8185    #0279  0583
CEOL3  8191    #0284  0289
CEOL4  819E     0286 #0290  0295
CEOL6  81AB     0211  0233  0280  0292 #0296
CFLAG  085F    #0043  0167  0228  0279  0463
CLEAN  80F0    #0184  0584  0586
CLEAN1 80F5    #0186  0189
CLEAR  8182    #0275  0582
CLOSET 836A     0500 #0544
CLRLIN 8145     0212  0218  0223 #0239  0358
CLRL1  8148    #0240  0251
CLRL2  8158     0241 #0247
CLRL3  8163     0246 #0252  0258  0267
CNTL   8078     0112 #0121
COLD   80EB    #0179  0606  0607
CR     8105    #0199  0219  0275  0371  0383  0594
CURPOS 0851    #0030  0146  0158  0162  0173  0202  0239  0245  0250  0256
                0259  0265  0269  0285  0297  0306  0319  0328  0415  0422
                0563
CURSOR 0860    #0044
C0A    83B3    #0591
C0B    83B5    #0592
C0C    83B7    #0593
C0D    83B9    #0594
C0E    83BB    #0595
C0F    83BD    #0596
C01    83A1    #0582
C02    83A3    #0583
C03    83A5    #0584
C04    83A7    #0585
C05    83A9    #0586
C06    83AB    #0587
C07    83AD    #0588
C08    83AF    #0589
C09    83B1    #0590
C1A    83D3    #0607
C1B    83D5    #0608
C1C    83D7    #0609
C1D    83D9    #0610
C1E    83DB    #0611
C1F    83DD    #0612
```

```
SYMBOL TABLE


C10    83BF    #0597
C11    83C1    #0598
C12    83C3    #0599
C13    83C5    #0600
C14    83C7    #0601
C15    83C9    #0602
C16    83CB    #0603
C17    83CD    #0604
C18    83CF    #0605
C19    83D1    #0606
DELETE 81B9    #0306  0601
DELINE 812B    #0223  0603
DLE    81F9    #0340  0597
DLEF   0853    #0032  0204  0510  0529  0532  0536
DLEFLG 0852    #0031  0121  0145  0203  0341
DLOOP  81BC    #0307  0312
DMDKE  82DC    #0477  0478
DMDKEY 82D6    #0475  0489
DMD1   82ED     0482 #0484
DONE   8091     0117 #0132
DOUT   81CA     0308 #0314
DTAB   82FF     0069  0071 #0492
EATONE 8276     0391  0397  0399  0401 #0412
ENDOL  82C0     0281  0316 #0454  0506
EN1    82C2    #0455  0459
EN2    82CC     0457 #0460
EOL    0855    #0034  0283  0291  0318  0320  0507  0520
ESCAPE 8249    #0386  0608
ESCE   82AF     0395 #0442
ESCEQ  826C     0393 #0406
ESCEQX 827B    #0415  0574
ESCEQY 8271    #0409  0573
ESCES  82B4    #0445  0575
ESCESW 82B9    #0448  0576
ESCKEY 027B    #0016  0483
ESC1   824E    #0390  0572
ESC4   8266    #0402  0429  0437  0450  0577
ESC5   8268     0387 #0403  0407  0410  0413  0443  0446
FLAGS  0854    #0033  0149  0205  0333  0335
FORW   82A1     0425 #0434
FORW1  82A2    #0435  0440
FS     8172    #0265  0590
GETDKE F5B7    #0020  0486
GETSER F963    #0021  0479
HOMCLR 8119    #0210  0595  0596
HOME   80FD    #0193  0195  0210
ILOOP  80BC    #0155  0160
INSLIN 8122    #0217  0602
INSRT2 80CA     0159 #0161
INVEC  0200    #0012  0075  0077
IOOK   F06E    #0018  0488  0490  0492  0494
IOVDT  0220    #0014  0065  0067
ITAB   82F6     0074  0076 #0488
JIND   809F     0130 #0139
JTBL   839F     0125  0126 #0581
```

```
SYMBOL TABLE


KBIER   FFCE     #0024  0057  0060  0062
LF      812E     #0227  0362  0591  0609
LF1     813D      0231 #0233
LINE    0862     #0046  0185  0194  0229  0232  0346  0348  0356  0359  0365
                  0375  0380
LOOP    81B0     #0298  0301
MAX     0050     #0006  0029  0147  0151  0154  0240  0266  0300  0307  0419
                  0421  0454
MAXLIN  0018     #0007  0184  0230  0360  0376
NEXT    085E     #0042  0519  0541
NOTP    828A      0420 #0422
OLDCUR  085D     #0041  0327  0416  0424
ONECHR  806B      0101 #0110
OPENT   8311      0498 #0504
OTAB    8308      0064  0066 #0498
OUTVEC  0202     #0013  0070  0072
PEND    085C     #0040  0090  0100  0180  0403
PROCES  804B     #0095  0493
PROG    8000     #0003  0050
PUTSER  F972     #0022  0471
READT   832B      0499 #0516
READ1   8344      0521 #0526
READ2   8366      0525  0538 #0542  0552  0558  0566
READ3   836B      0518 #0548
READ4   8377      0549 #0554
READ5   8383      0555 #0560
READ6   8340     #0524  0561
READ7   8362      0528  0533 #0540
READ8   8358      0530 #0535
ROUT    8363      0512 #0541
RTSINS  82D5      0449  0462  0464 #0466  0581  0588  0593  0598  0599  0604
                  0605  0612
SAMEXY  8098     #0135  0485
SAVEA   0858     #0036  0095  0110  0129  0134  0164  0168  0504  0511
SAVES   085B     #0039  0099  0132
SAVEX   0859     #0037  0096  0136  0475  0505  0516  0542
SAVEY   085A     #0038  0097  0135  0476
SEQTBL  8393      0106  0107 #0572
SEROUT  F972      0172  0187  0200  0235  0244  0248  0261  0271  0287  0293
                  0323  0352  0379  0430  0438 #0471
SETCUR  828D      0329 #0423
SHOWL   81CF      0161 #0316
SHOW3   81D9     #0320  0325
SHOW4   81E7      0321 #0327
START   8003     #0057  0059
STORE1  80D0      0150 #0164
STORE3  80E4      0170 #0172
TERM    0861     #0045  0509  0517  0523  0550  0556  0562
TINSRT  81F0     #0333  0600
TLINE   0863     #0047  0357  0366
TSERI   F96C     #0023  0079  0081  0477
TSTKEY  0231     #0015  0080  0082
UP      81FF      0193  0217 #0346  0368  0592  0611
UPA     8208      0347 #0351
VARS    0800     #0004  0028
```

```
SYMBOL TABLE


VEXIT  80EA     0148  0153 #0174
VISI   80A2     0116 #0144
VISIX  8072    #0116  0122
.NARG  ****
```

APPENDIX N

SELECTED BIBLIOGRAPHY

N.1 GENERAL PROGRAMMING AND ITERFACING TECHNIQUES

Chapin, N., Flowcharts, Auerbach, Princeton, N. J., 1971.

Hughes, J. K. and J. I. Michtom, A Structured Approach to Programming, Prentice-Hall, Englewood Cliffs, N. J., 1977.

Kerningham, B. W. and P. J. Plauger, The Elements of Programming Style, McGraw-Hill, New York, 1974.

Leventhal, L.A., Introduction to Microprocessors: Software, Hardware, Programming, Prentice-Hall, Englewood Cliffs, N. Y., 1978, Chapter 6.

Peatman, J., Microcomputer-based Design, McGraw-Hill, New York, 1977.

McGowan, C.L. and J. R. Kelly, Top-Down Structured Programming Techniques, Petrocelli/Charter, New York, 1975.

Wakerly, J.F., "Microprocessor Input/Output Architecture", Computer, February, 1977, pp. 26-33.

Yourdon, E., Techniques of Program Structure and Design, Prentice-Hall, Englewood Cliffs, N. J., 1976.

N.2 R6502 PROGRAMMING AND INTERFACING

Butterfield, J. et. al., The First Book of KIM, Hayden, Rochelle Park, N. J., 1978.

Camp, R. C. et. al., Microprocessor System Engineering, Matrix, Portland, Or., 1979.

DeJong, M., Programming & Interfacing the R6502, Howard W. Sams & Co., Indianapolis In., 1979.

Foster, C. C., Programming a Microcomputer: 6502, Addison-Wesley, Reading, Ma., 1978.

Leventhal, L.A., 6502 Assembly Language Programming, Osborne & Associates, Berkeley, Ca., 1979.

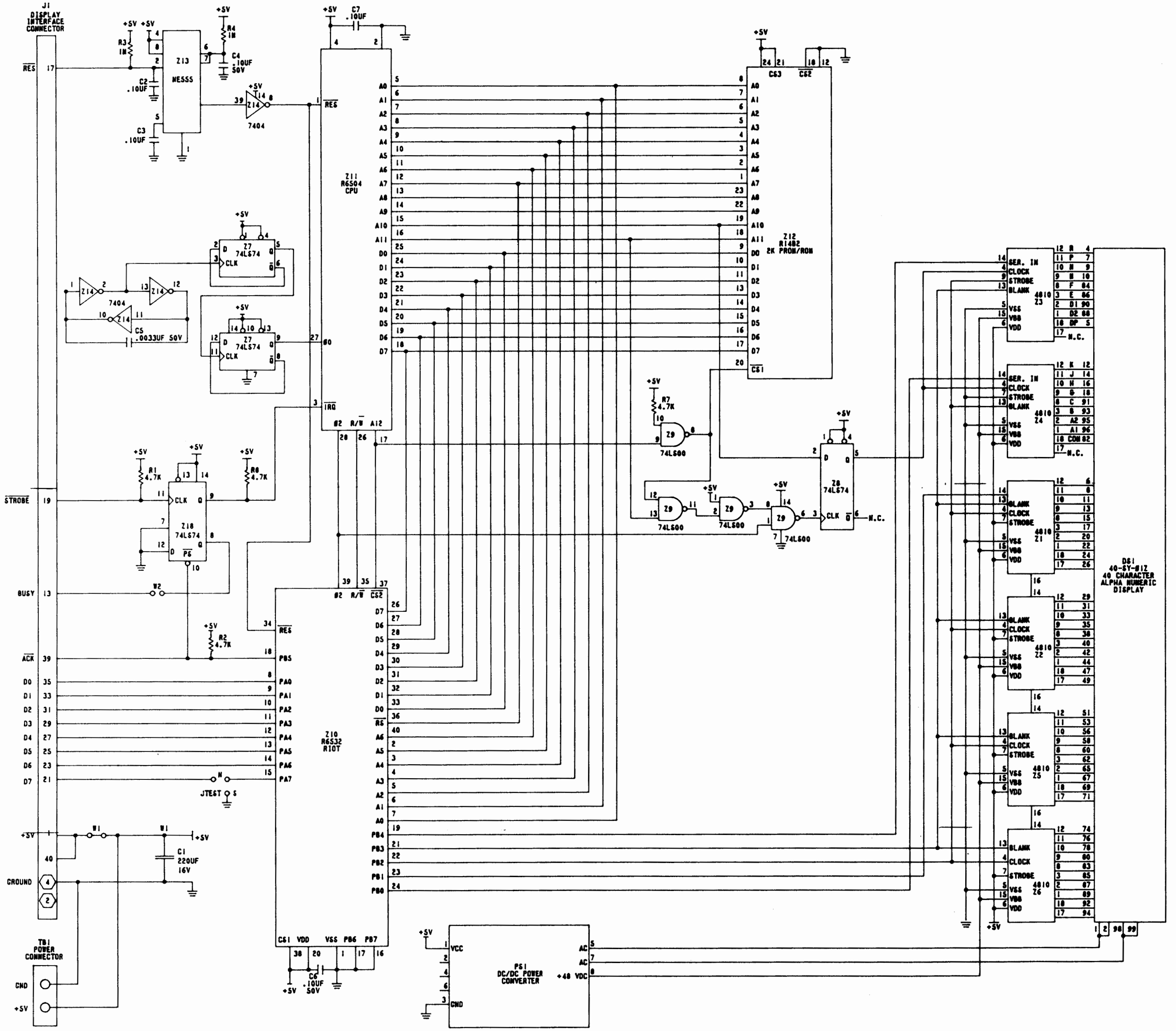
Scanlon, L., 6502 Software Design, Howard W. Sams & Co., Indianapolis, In., 1980.

Notes

Notes

Notes

Notes



1. NOTE: UNLESS OTHERWISE SPECIFIED: CAPACITOR VALUES ARE IN MICROFARADS ±20% AND ARE 50V

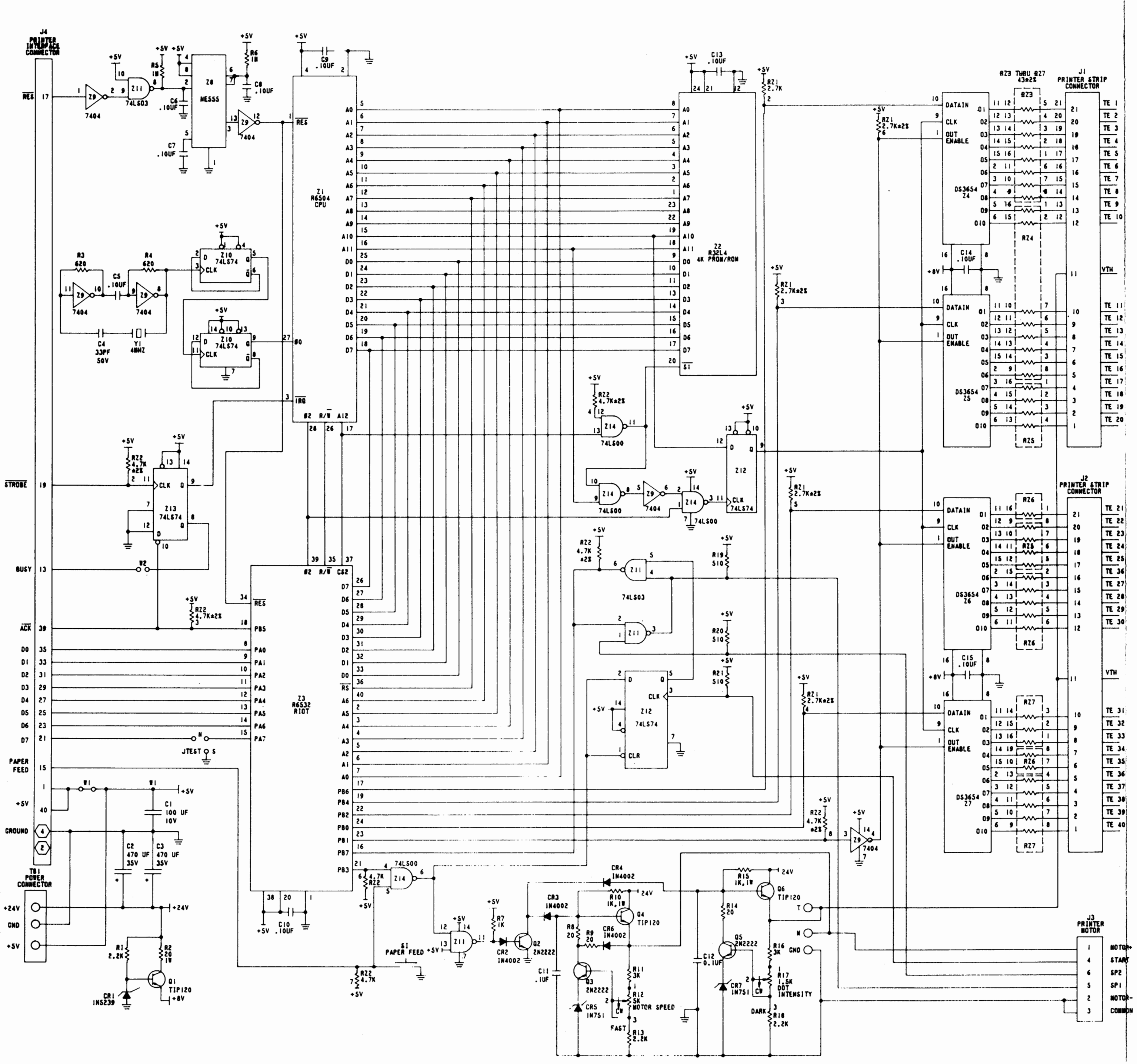
⬡2 PINS 2,3,5,7,9,11,15,16&37 ARE NOT CONNECTED

3. RESISTOR VALUES ARE IN OHMS±5% AND ARE 1/4 W

4. EVEN NUMBERED. 4 THRU 38 ARE COMMON

REF: PA20-X521

SCHEMATIC-AIM65/40 DISPLAY ASSEMBLY .



NOTE: UNLESS OTHERWISE SPECIFIED:

1. CAPACITOR VALUES ARE IN MICROFARADS ±20% AND ARE 50V

⬡2 PINS 2,3,5,7,9,11,15&37 ARE NOT CONNECTED

3. RESISTOR VALUES ARE IN OHMS±5% AND ARE 1/4 W

⬡4 EVEN NO. 4 THRU 38

REF: PA20-X511

SCHEMATIC-AIM65/40 PRINTER ASSEMBLY

CUT ALONG THIS LINE

DOCUMENT REGISTRATION FORM

Please fill out and retum this card to automatically receive all updates to your manual.

DOCUMENT NAME:

DOCUMENT NUMBER: **REVISION:**

(Name)

(Company)

(Street Address)

(City) (State) (Zip)

PLACE
STAMP
HERE

Documentation Manager
501-300
Rockwell International
ELECTRONIC DEVICES DIVISION
4311 Jamboree Road
P.O. Box C
Newport Beach, CA 92660

ELECTRONIC DEVICES DIVISION REGIONAL ROCKWELL SALES OFFICES

HOME OFFICE
Electronic Devices Division
Rockwell International
4311 Jamboree Road
P.O. Box C
Newport Beach, California 92660
Tel: 800-854-8099 (800-854-4030)
In California 800-422-4230
TWX: 910 591-1698

UNITED STATES
Electronic Devices Division
Rockwell International
1842 Reynolds
Irvine, California 92714
(714) 833-4655
TWX: 910 595-2518

Electronic Devices Division
Rockwell International
921 Bowser Road
Richardson, Texas 75080
(214) 996-6500
Telex: 73-307

Electronic Devices Division
Rockwell International
10700 West Higgins Rd., Suite 102
Rosemont, Illinois 60018
(312) 297-8862
TWX: 910 233-0179 (RI MED ROSM)

Electronic Devices Division
Rockwell International
5001B Greentree
Executive Campus, Rt. 73
Marlton, New Jersey 08053
(609) 596-0090
TWX: 710 940-1377

EUROPE
Electronic Devices Division
Rockwell International GmbH
Fraunhoferstrasse 11
D-8033 Munchen-Martinsried
West Germany
(089) 859-9575
Telex: 0521/2650 rimd d

Electronic Devices Division
Rockwell International
Heathrow House, Bath Rd.
Cranford, Hounslow
Middlesex, England
(01) 759-9911 Ext. 33
Telex: 851-25463

Electronic Devices
Rockwell Collins
Via Boccaccio, 23
20123 Milano
Italy
498.74.79

FAR EAST
Electronic Devices Division
Rockwell International Overseas Corp.
Itohpia Hirakawa-cho Bldg.
7-6, 2-chome, Hirakawa-cho
Chiyoda-ku, Tokyo 102, Japan
(03) 265-8806
Telex: J22198

YOUR LOCAL REPRESENTATIVE

7/81

Rockwell
International