RM 65 FAMILY

RM 65 RUN-TIME BASIC

USER'S MANUAL
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>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTION 1. INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>1-1</td>
</tr>
<tr>
<td>1.2 Run-Time BASIC User's Manual Description</td>
<td>1-2</td>
</tr>
<tr>
<td>1.3 Reference Documents</td>
<td>1-3</td>
</tr>
<tr>
<td>SECTION 2. INSTALLATION</td>
<td></td>
</tr>
<tr>
<td>2.1 Installation in an RM 65 SBC Module</td>
<td>2-4</td>
</tr>
<tr>
<td>2.2 Installation in an RM 65 PROM/ROM Module</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3 Installation in an AIM 65 Microcomputer</td>
<td>2-8</td>
</tr>
<tr>
<td>SECTION 3. OPERATION</td>
<td></td>
</tr>
<tr>
<td>3.1 Development on an AIM 65 Microcomputer</td>
<td>3-5</td>
</tr>
<tr>
<td>3.2 Relocating the Application Driver</td>
<td>3-8</td>
</tr>
<tr>
<td>3.3 Relocating the Application Program</td>
<td>3-9</td>
</tr>
<tr>
<td>3.4 Preparing the PROM/ROM</td>
<td>3-12</td>
</tr>
<tr>
<td>3.4.1 Merged Application Driver and Program</td>
<td>3-12</td>
</tr>
<tr>
<td>3.4.2 Separate Application Driver and Program</td>
<td>3-12</td>
</tr>
<tr>
<td>SECTION 4. APPLICATION DRIVER REQUIREMENT AND EXAMPLES</td>
<td></td>
</tr>
<tr>
<td>4.1 Application Driver Requirements</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.1 Startup Routines</td>
<td>4-1</td>
</tr>
<tr>
<td>4.1.2 I/O Vectors and Handlers</td>
<td>4-7</td>
</tr>
<tr>
<td>4.1.3 Interrupt Vectors and Handlers</td>
<td>4-7</td>
</tr>
<tr>
<td>4.2 Example Application Drivers</td>
<td>4-13</td>
</tr>
<tr>
<td>4.2.1 Interactive Operation On An AIM 65 Microcomputer</td>
<td>4-13</td>
</tr>
<tr>
<td>4.2.2 Run-Time Operation in an RM 65 SBC Module</td>
<td>4-18</td>
</tr>
</tbody>
</table>

APPENDIX A. BASIC VARIABLES
APPENDIX B. RM 65 AND AIM 65 BASIC DIFFERENCES | B-1
LIST OF FIGURES

Figure | Page
--- | ---
2-1 | RM 65 and AIM 65 BASIC Memory Maps
2-2 | RM 65 Module Firmware Memory Map
2-3 | Example Base Address Selection Header
3-1 | Typical Development of a 4K-Byte Application
3-2 | Typical Development of a 16K-Byte Application
3-3 | Relocator Assembly Listing
4-1 | Application Driver Flowchart
4-2 | Model Application Driver Assembly Listing
4-3 | Typical AIM 65 Development Configuration
4-4 | Example AIM 65 Interactive Driver
4-5 | Typical RM 65 Run-Time Configuration
4-6 | Example RM 65 SBC Run-Time Driver

LIST OF TABLES

Table | Page
--- | ---
2-1 | RM 65 SBC Module PROM/ROM Selection Jumpers
4-1 | I/O Vector Summary
4-2 | I/O Vector Description
A-1 | RM 65 Run-Time BASIC Page Zero Usage
A-2 | RM 65 Run-Time BASIC Page Two Usage
A-3 | AIM 65 BASIC Page Zero Usage
SECTION 1

INTRODUCTION

1.1 OVERVIEW

The RM 65 Run-Time BASIC is a ROM-resident BASIC system designed to operate with an R6502 CPU-based RM 65 Single Board Computer (SBC) module. Contained in one 8K-byte ROM, this BASIC run-time package allows an application program written in BASIC, developed on the AIM 65 Microcomputer and located in PROM/ROM, to execute immediately on the RM 65 SBC module upon power turn-on. Vectored I/O and user provided I/O drivers allow complete application flexibility. The application program, up to 8K-bytes in length and programmed in a PROM/ROM, may be installed in one PROM/ROM socket on an RM 65 SBC module while the run-time BASIC ROM is installed in the other PROM/ROM socket. This allows a wide variety of applications requiring a parallel interface, a serial interface, and/or one or two counters/timers to be programmed in BASIC and implemented in a single RM 65 SBC module using its user dedicated R6522 Versatile Interface Adapter (VIA) device as the application interface.

Larger application programs and other interfaces can be installed using the RM 65 PROM/ROM module and other RM 65 peripheral interface and I/O modules. For example, peripheral equipment with an RS-232C interface can be connected to an RM 65 Asynchronous Communications Interface Adapter (ACIA) module (RM65-5451), while peripherals with a parallel interface can be connected to the RM 65 SBC, Multi-Function Peripheral Interface (MPI) or General Purpose Input/Output and Timer (GPIO) modules, (RM65-5223 and RM65-5222, respectively).
In the AIM 65 Microcomputer based system with RM 65 module expansion, floppy disk drives and CRT display can be connected for development or production use. Either 5 1/4- or 8-inch floppy disk drives can be controlled using the RM 65 Floppy Disk Controller (FDC) module (RM65-5101), while CRT displays up to 25 lines by 80 columns can be driven using the RM 65 CRT Controller (CRTC) module (RM65-5102).

1.2 RUN-TIME BASIC USER'S MANUAL DESCRIPTION

This manual describes the installation and operation of the RM 65 Run-Time BASIC. The BASIC language description is not included in this manual, however, the language reference in the AIM 65 BASIC User's Manual is fully applicable to RM 65 Run-Time BASIC. Any differences in the operation and use between RM 65 Run-Time BASIC and AIM 65 BASIC are described in this manual.

Section 1, Introduction, scopes the RM 65 Run-Time BASIC, describes this manual, and lists reference documents.

Section 2, Installation, tells how to install the Run-Time BASIC ROM in an RM 65 SBC or PROM/ROM module.

Section 3, Operation, describes how to operate the RM 65 Run-Time BASIC on an RM 65 SBC module in the run-time mode or on an AIM 65 Microcomputer, in either the run-time or development mode.

Section 4, Application Driver Requirements and Examples, defines the requirements for the user-provided startup routine and I/O drivers and also describes some example hardware configurations and software drivers.
1.3 REFERENCE DOCUMENTS

Rockwell

29650N30  R6500 Microcomputer Programming Manual
Order No. 202

29650N31  R6500 Microcomputer Hardware Manual
Order No. 201

29650N36  AIM 65 Microcomputer User's Guide
Order No. 209

29650N49  AIM 65 BASIC User's Manual
Order No. 221

29650N55  AIM 65 8K BASIC Reference Card
Order No. 233

Order No. 269

29650N01  RM 65 General Purpose Input/Output
Order No. 801 and Timer (GPIO) Module User's Manual

29801N02  RM 65 Floppy Disk Controller (FDC)
Order No. 802 Module User's Manual

29801N04  RM 65 Asynchronous Communications
Order No. 804 Interface Adapter (ACIA) Module User's Manual

29801N05  RM 65 8K Static RAM Module User's Manual
Order No. 805

29801N06  RM 65 16K PROM/ROM Module User's Manual
Order No. 806

29801N08  RM 65 32K Dynamic RAM Module User's Manual
Order No. 808

29801N09  RM 65 Single Board Computer (SBC)
Order No. 809 User's Manual

29801N14  RM 65 CRT Controller (CRTC) Module
Order No. 814 User's Manual

29801N15  RM 65 IEEE-488 Bus Interface Module
Order No. 815 User's Manual

29801N17  RM 65 Multi-Function Peripheral
Order No. 817 Interface (MPI) Module User's Manual

1-3
SECTION 2

INSTALLATION

The RM 65 Run-Time BASIC (RM65-0122) is provided in a Rockwell 8K-byte R2364 ROM (R2906). After installing the ROM in an RM 65 SBC or PROM/ROM module, the run-time BASIC is ready for use in either the run-time or development mode of operation. A short user-provided program segment must be included in the system prior to use, however, in either mode. This segment must call the BASIC initialization subroutine, load program and variable location vectors, load I/O driver vectors and provide the I/O drivers themselves. These driver requirements are described in Section 3.

Figure 2-1 shows the memory map for the RM 65 Run-Time BASIC, along with the AIM 65 BASIC, for reference. The RM 65 module firmware memory map is also shown for reference in Figure 2-2.

Note that the RM 65 CRTC, FDC and IEEE-488 modules are mapped at their firmware addresses. If a module ROM is not used, the corresponding module I/O can be mapped elsewhere by selecting a different base address on the module.
Figure 2-1. RM 65 and AIM 65 BASIC Memory Maps
<table>
<thead>
<tr>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

**User Available**

<table>
<thead>
<tr>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**FDC Module**

<table>
<thead>
<tr>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
<th>7F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Default Input Buffer**

<table>
<thead>
<tr>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
<th>6F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Default Output Buffer**

<table>
<thead>
<tr>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
<th>5F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**FDC module Variables & Constants**

<table>
<thead>
<tr>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**CRTC Module Variables & Constants**

<table>
<thead>
<tr>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
<th>3F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**IEEE-488 Module Variables & Constants**

<table>
<thead>
<tr>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
<th>2F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Reserved for R6502 CPU Stack**

<table>
<thead>
<tr>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
<th>1F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Reserved for System Variables**

<table>
<thead>
<tr>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
<th>FF</th>
</tr>
</thead>
</table>

**Reserved for IEEE-488 Module Variables**

<table>
<thead>
<tr>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
<th>EF</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Reserved for FDC Module Variables**

<table>
<thead>
<tr>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Reserved for System Variables**

<table>
<thead>
<tr>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

**Reserved for RAM**

<table>
<thead>
<tr>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

Figure 2-2. RM 65 Module Firmware Memory Map

2-3
2.1 INSTALLATION IN AN RM 65 SBC MODULE

The following procedure describes an installation of the RM 65 Run-Time BASIC ROM and a 4K-byte application program PROM in the RM 65 SBC module (RM65-1000). The run-time BASIC ROM is to be installed in one PROM/ROM socket while the application program PROM is to be installed in the other PROM/ROM socket. Consult Section 2 of the RM 65 SBC Module User's Manual for general installation instructions.

CAUTION
The Run-Time BASIC ROM is manufactured using the Metal-Oxide Semiconductor (MOS) process. Since the inadvertent application of high voltages may damage this ROM or other MOS devices, be sure to discharge any static electrical charge accumulated on your body by touching a ground connection (e.g., a grounded equipment chassis) before touching the ROM or module into which it is to be installed. This precaution is especially important if you are working in a carpeted area or in an environment with low relative humidity.

a. Ensure that power is turned OFF to the module in which the ROM is to be installed. Remove the Run-Time BASIC ROM from the shipping container. Inspect the ROM to be sure that the pins are straight and free of foreign material.

b. Install the RM 65 Run-Time BASIC ROM (R2906) in socket Z8.

c. Install the application PROM in socket Z10.
d. Wire a base address selection header and install it in socket Z13 as shown in Figure 2-3 to select the base addresses as follows:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Socket Z10 (Application PROM)</td>
<td>$F000-$FFFF</td>
</tr>
<tr>
<td>Map Socket Z8 (Run-Time BASIC ROM)</td>
<td>$B000-$CFFF</td>
</tr>
<tr>
<td>Map I/O</td>
<td>$A000-$AFFF</td>
</tr>
<tr>
<td>Map RAM</td>
<td>$0000-$0FFF</td>
</tr>
</tbody>
</table>

e. Install jumper E1 in position B and install jumper E6 to assign Bank Address Select (BADR/) to Bank 0.

f. Install jumpers E2-E4 as follows to select the PROM/ROM size (see Table 2-1):

```
Pin | Top | Pin
--- |-----|-----
1   |     | 20  F000
2   |     | 19  E000
9000         | 18  D000
3000         | 17  C000
2000         | 16  B000
I/O Section  | 15  A000
7   |     | 14  0000
1000         | 13  8000
5000         | 12  7000
4000         | 11  6000
```

Figure 2-3. Example Base Address Selection Header
### Table 2-1. RM 65 SBC Module PROM/ROM Selection Jumpers

<table>
<thead>
<tr>
<th>Section (Socket)</th>
<th>PROM/ROM (see note 1)</th>
<th>Edge Connector Version</th>
<th>Euroconnector Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Jumper Position</td>
<td>Jumper Position</td>
</tr>
<tr>
<td>Section 0 (Z8)</td>
<td>2K</td>
<td>E2A OFF</td>
<td>E2A ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2B ON</td>
<td>E2B OFF</td>
</tr>
<tr>
<td></td>
<td>4K</td>
<td>E2A OFF</td>
<td>E2A ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2B ON</td>
<td>E2B OFF</td>
</tr>
<tr>
<td></td>
<td>8K</td>
<td>E2A OFF</td>
<td>E2A ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2B ON</td>
<td>E2B OFF</td>
</tr>
<tr>
<td>Section 1 (Z10)</td>
<td>2K (see note 2)</td>
<td>E3 B</td>
<td>E3 B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E4 A or B</td>
<td>E4 A or B</td>
</tr>
<tr>
<td></td>
<td>4K</td>
<td>E3 B</td>
<td>E3 B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E4 A</td>
<td>E4 A</td>
</tr>
<tr>
<td></td>
<td>8K</td>
<td>E3 A</td>
<td>E3 A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E4 A</td>
<td>E4 A</td>
</tr>
</tbody>
</table>

**NOTES**

1. Typical PROM/ROM devices:
- 2K = TMS2516/i2716 PROM, R2316 ROM, or equivalent.
- 4K = TMS2532 PROM, R2332 ROM, or equivalent.
- 8K = MCM68764 PROM, R2364A ROM, or equivalent.

2. Enabled in lower 2K-byte address space ($X000-$X7FF) only (pin 18 = All = 0).

3. Enabled in either half of the 4K-byte address space depending upon the position of jumper E4:
   - E4 = A: Enabled in lower half of the 4K-byte address space ($X000-$X7FF) only (pin 18 = All = 0).
   - E4 = B: The 2K-byte PROM/ROM is mapped into both the lower ($X000-$X7FF) and the upper ($X800-$XFFF) halves of the 4K-byte address space. This allows the RES, IRQ and NMI vectors located at $X7FA-$X7FF in a 2K-byte PROM to be mapped at $XFFA-$XFFF on the SBC module.
Socket Z8 (8K-byte Run-Time BASIC ROM):

E2A = ON, E2B = OFF (Edge Connector Version)
E2A = OFF, E2A = ON (Euroconnector Version)

Socket Z10 (4K-byte application PROM):

E3 = B  
E4 = A

g. Install jumper E5 to select internal clock reference.

h. Remove jumper E7 to force the DMA terminate (BDMT/) signal to a logic 1 (inactive).

i. Set switches S2-1 through S2-3 to OPEN to assign on-board RAM, I/O and PROM/ROM to both Bank 0 and 1. (Switches S2-4 through S2-6 may be in either position.)

2.2 INSTALLATION IN AN RM 65 PROM/ROM MODULE

The following procedure describes the installation of the RM 65 Run-Time BASIC ROM and an 8K-byte application program PROM/ROM in an RM 65 16K PROM/ROM module (RM65-3216). Refer to Section 2 of the RM 65 PROM/ROM Module User's Manual for general installation instructions.

a. Install the RM 65 Run-Time BASIC ROM (R2906) in socket Z12.

b. Install the application program PROM/ROM in socket Z14.

c. Set switches S1-1 through S1-4 and S2-1 through S2-4 to assign the base address of each of the 4K-byte address spaces in socket Z12 for the RM 65 Run-Time BASIC ROM:

(1) Assign $B000 to the upper 4K-bytes of socket Z12:

S1-1 CLOSED
S1-2 CLOSED
S1-3 OPEN
S1-4 CLOSED
(2) Assign $C000 to the lower 4K-bytes of socket Z12:

S2-1 OPEN
S2-2 OPEN
S2-3 CLOSED
S2-4 CLOSED

d. Set switches S3-1 through S3-4 and S4-1 through S4-4 to assign the base address of the rest of the PROM/ROM module to address ranges not applicable to the specific application, i.e., that do not conflict with either the RM 65 Run-Time BASIC or the application program.

2.3 INSTALLATION IN AN AIM 65 MICROCOMPUTER

The RM 65 Run-Time BASIC ROM may not be installed in the AIM 65 Microcomputer since it is an 8K-byte ROM and the AIM 65 Master Module may accommodate only 4K-byte (or less) PROM/ROM devices.

The application program, however, may be installed in an AIM 65 Master Module if it is programmed in compatible 2K- or 4K-byte PROM/ROMS. In this case, sockets Z25 and Z26 must be unpopulated (since the run-time BASIC at this $B000-$CFFF address range will be installed off-board, e.g., in an RM 65 PROM/ROM Module).

If the application PROM/ROM is installed in socket Z24 (at address $DXXX) and the Monitor ROMs are installed, the program may be started by pressing the N key from the Monitor command level. The startup routine must begin at $D000 in this configuration. Application interrupt handlers can be linked to the Monitor IRQ and NMI interrupt linkage.

If the AIM 65 Monitor ROMs are not used, up to 12K-bytes of application PROM/ROM may be installed in sockets Z22, Z23 and Z24. One application PROM/ROM must be installed in socket Z22 to provide the RES, IRQ and NMI interrupt vectors at $FFFA-$FFFF.
The RM 65 Run-Time BASIC can operate in one of two modes, interactive (sometimes called development) or run-time. In the interactive mode, all BASIC direct and indirect commands available in AIM 65 BASIC (except as defined in Appendix B) may be entered by an operator from a keyboard with BASIC response directed to a display/printer. In the run-time mode, only BASIC indirect commands may be executed since BASIC is initialized to run-time operation upon power turn-on or reset.

In either mode of operation, all I/O operations are application dependent, with I/O processing performed by I/O handlers, either user-provided as part of an application driver or located elsewhere. In both cases, the I/O handlers are pointed to by I/O vectors loaded by a startup routine within the application driver. The user-provided application driver, consisting of the startup routine and I/O handlers must be resident in memory in order to operate RM 65 Run-Time BASIC in either mode of operation.

This section describes how to operate the RM 65 Run-Time BASIC in the interactive mode and how to migrate the application driver (written in assembly language) and/or the application program (written in BASIC) to addresses for execution in the run-time mode in either an RM 65 microcomputer or RM 65 SBC environment.

The application driver and programs are first hosted on the AIM 65 Microcomputer in an interactive mode. This allows an application program, initially developed using the AIM 65 BASIC, to be integrated with the application driver and executed interactively on the AIM 65 Microcomputer for final test. Any corrections to the driver or the application program can easily be made using the AIM 65 Assembler and RM 65 Run-Time BASIC before rehosting them on the RM 65 module.
In fact, after installing RM 65 Run-Time BASIC on the AIM 65 Microcomputer, you may want to develop subsequent application programs in this configuration due to the flexibility of the vectored I/O. The I/O can first be vectored to AIM 65 Monitor I/O subroutines for development then changed to point to run-time drivers to production operation. In addition, development-oriented peripherals, such as floppy disk drives, a CRT display and an 80-column printer can be interfaced to the AIM 65 Microcomputer using RM 65 FDC (with DOS firmware installed), CRTD, and MPI (or GPIO) modules installed in the same card cage as the RM 65 16K PROM/ROM module containing the run-time BASIC. Use of these peripherals greatly improves programmer efficiency thus lowering program development costs.

Figures 3-1 and 3-2 shows the general flow of an application driver and BASIC programs from interactive operation on the AIM 65 Microcomputer to run-time operation on the RM 65 SBC module. Figure 3-1 illustrates migration of a 4K application program to an RM 65 SBC module, while Figure 3-2 shows migration of a larger program to an RM 65 PROM/ROM module.

The described procedure carries an example program from development on an AIM 65 Microcomputer to run-time on an RM 65 SBC module. After using this procedure to become familiar with the methodology, modify the procedure as required for operation in your development and application environment.

Refer to Section 4 for a detailed discussion of the application driver requirements.
Figure 3-1. Typical Development of a 4K-Byte Application Program
Figure 3-2. Typical Development of a 16K-Byte Application Program
3.1 DEVELOPMENT ON AN AIM 65 MICROCOMPUTER

This procedure describes the steps to take to develop an application program on an AIM 65 Microcomputer. The application I/O can be easily tested using the AIM 65 I/O subroutines if the application I/O is similar. The following memory map, corresponding to a 4K-byte application program is used as an example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0300 - $03FF</td>
<td>Driver Object Code</td>
</tr>
<tr>
<td>$0400 - $1BFF</td>
<td>Application Program and Variables</td>
</tr>
<tr>
<td>$1C00 - $1FFF</td>
<td>Assembly Symbol Table</td>
</tr>
<tr>
<td>$2000 - $2FFF</td>
<td>Driver Source Code</td>
</tr>
</tbody>
</table>

It is assumed that additional RAM is available beyond the 4K bytes on-board the AIM 65 Microcomputer for development. RAM can easily be added in the RM 65 card cage using an RM 65 32K Dynamic RAM module (RM65-3132) or an RM 65 8K Static RAM module (RM65-3108). If additional RAM is not available, the upper limit of the application program and variables cannot exceed $0FFF. In this case, the application driver should be assembled separately and object code loaded when needed.

a. Install the RM 65 Run-Time BASIC ROM in an RM 65 16K PROM/ROM module as described in Section 2.2, however, do not install an application PROM/ROM.

b. Install the PROM/ROM module along with any other peripheral and memory modules in a RM 65 card cage and connect the card cage to an AIM 65 Microcomputer.

d. Load the source code for the application driver shown in
the Figure 4-2 assembly listing into the Text Editor, and
return to the Monitor. Locate the Text Buffer from
$2000-$2FFF for this example.

Note that the startup driver will extend from $0300 to the
label BEGIN. The application program will start at BEGIN
while the variables will initially start at BEGIN+2. The
variables must be located above the program during
development (their starting address will increase as the
application program increases in size).

The application driver source code is kept resident in the
Text Buffer throughout this procedure for ease in changing
it during migration to run-time operation. For extended
use in the development mode, the application driver may be
programmed in PROM and installed on the AIM 65 Micro-
computer (e.g., at $DXXX) for immediate operation upon
power turn-on.

e. Assemble the application driver. Locate the symbol table
at $1C00-$1FFF for this example.

f. After verifying the assembled driver is correctly coded,
save the driver source code on mass storage for backup and
future use.

g. Press the Fl key to enter BASIC and to perform a cold
start, i.e., to clear a previously loaded program.

h. Enter/load the application program as required, e.g.:

```
100 FOR N = 1 TO 1000
110 PRINT "TEST", I
120 NEXT I
130 GOTO 100
```
NOTE

The ATAN function is provided in the RM 65 Run-Time BASIC whereas it must be user-provided when using the AIM 65 BASIC (see Appendix B). If the application program was developed on AIM 65 and calls the ATAN function, remove both the altering of the ATAN vector and the ATAN machine code subroutine from the application program before running the program on RM 65 Run-Time BASIC.

i. Execute the application program as required, e.g.:

RUN

NOTE

For continuous operation of the application program in the run-time mode, ensure the following:

1. The application program is designed to remain in execution (e.g., in an endless loop), and there are no END or STOP statements.

2. The application program is fully debugged and there are no external conditions (e.g., input data type, amount, or value) that will cause BASIC to detect an error, stop execution, and attempt to report the error (see Appendix A in the AIM 65 BASIC Language Reference Manual).

3. A $5B code (BREAK command) is not input from a keyboard while running.
j. Press <ESC> to stop execution, i.e., to cause a BREAK, and return to the BASIC command level.

k. Press <ESC> to return to the Monitor command level from the BASIC command level.

l. Press the F2 key to reenter BASIC and to perform a warm start, i.e., to retain the previously loaded program.

m. Enter and execute the application program as required.

n. Press <ESC> to return to the Monitor command level from the BASIC command level.

o. Save the BASIC application program on mass storage for backup or future use.

3.2 RELOCATING THE APPLICATION DRIVER

After the application program has been validated in the interactive mode, the application driver and application program are ready to be relocated to their final run-time locations. The application driver will usually be relocated to the lower part of PROM/ROM addresses. This relocation consists merely of changing the starting address of the object code, for example from $300 to $F000, then reassembling. Other changes to the driver source code must first be made, however, to add interrupt vectors ($FFFA-$FFFF), and to add any application dependent I/O (replacing linkage AIM 65 I/O, if used).

a. Reenter the Text Editor from the AIM 65 Monitor.
b. Change the startup routine origin, add the I/O vectors, and replace linkage to AIM 65 I/O subroutines with run-time I/O handlers (see Figure 4-3).

c. Return to the Monitor and assemble without generating object code (LIST-OUT = <RETURN> and OBJ?Y OUT=X).

d. After verifying that the driver is assembled correctly, reassemble and direct object code to audio tape.

e. Save the run-time application driver source code on mass storage for backup or future use.

3.3 RELOCATING THE APPLICATION PROGRAM

In many cases the application program must be relocated from locations used during development in interactive mode to locations used for run-time operation. For example, a program residing at $400-$12F9 during development can be moved to $F100-$FFF9 for PROM/ROM installation (after merging with the application driver and interrupt vectors ($F000-$F0FF and $FFFA-$FFFF).

For larger programs, (e.g., 16K-bytes) it may be desired to map the application at the same addresses for development (in RAM) as in run-time (in PROM/ROM). This simplifies the migration to PROM/ROM since the application program only has to be programmed into PROM/ROM without relocation. In this case, only the application driver need be relocated, usually to the $FXXX area, since interrupt vectors must be mapped at $FFFA-$FFFF. Note that this mapping may be either separate or redundant, whichever best satisfies the application requirements.

In this example, the application program is relocated to $DXXX so the resultant PROM/ROM can be installed on-board an AIM 65
microcomputer (in socket Z24) or on an RM 65 SBC module (in socket Z8 with the base address header wired to redundantly map the socket to $DXXX and $FXXX).

a. If the Relocator object code is not available on mass storage, assemble the program (see the assembly listing in Figure 3-3) and direct the object code to mass storage. Note that the object code cannot be directed to memory during assembly since the assembler uses zero page (where the Relocator object code is also located).

b. Load the Relocator object code.

c. Enter the old and new starting addresses of the program, i.e., $0300 and $D000, respectively, in this example:

   <M>0004 XX XX XX XX
   </>0004 00 03 00 D0

   d. Execute the Relocator program,

   <*>=000C
   <G>/.

   The program returns to the Monitor command level upon completion.

   NOTE

   The application program cannot be executed after the statement addresses have been changed by the Relocator until the application program is installed at the new addresses, e.g., $D000-$DFFF.
FIRST SET CORRECT VALUES INTO PGMST AND OLDAOR,
THEN EXECUTE THE PROGRAM STARTING AT RELOC.

FIRST CALCULATE THE OFFSET TO NEW LOCATION
RELOC LDA PGMST , NEW PROGRAM START ADDR
LDX PGMST+1

SET UP POINTERS FOR FIRST STATEMENT
LOX #0
LDY #1
LDA OLDAOR
STA NEXT
LDA OLDAOR+1
STA NEXT+1

EXECUTE THIS CODE ONCE FOR EACH STATEMENT
D0NECLN LDA <NEXT>,X ; NEXT LINE ADDR LOW
ORA <NEXT>,Y ; NEXT LINE ADDR HIGH
BEG DONE , IF ZEROS =>

RELOCATE THE CURRENT LINE
RELCNLN CLC
LDA <NEXT>,X>
PHA
ADC OFFSET
STA <NEXT,X>
LDA <NEXT>,Y ;LINE NUMBER LOW
PHA
ADC OFFSET+1
STA <NEXT>,Y

POINT TO THE NEXT PROGRAM LINE
PLA
STA NEXT+1
PLA
STA NEXT
JMP DONECLN ; =>
JMP COMIN ;RETURN TO MONITOR =>
END

ERRORS=0000

Figure 3-3. Relocator Assembly Listing

3-11
3.4 PREPARING THE PROM/ROM

The AIM 65 PROM PROGRAMMER & CO-ED module (A65-006) may be used to program PROMs up to 4K-byte in size, for installation RM 65 SBC and PROM/ROM modules and in the AIM 65 Microcomputer. Refer to the AIM 65 PROM Programmer & CO-ED User's Manual for the detailed operating procedure.

Install the PROM Programmer & CO-ED module on an AIM 65 Microcomputer.

3.4.1 Merged Application Driver and Program

Use this procedure to program a merged application driver and application program; for example, to prepare a single PROM at $FXXX for installation of a 4-byte application program in an RM 65 SBC module.

a. Zero memory in the PROM address area.

b. Load the application program object code from audio cassette.

c. Load the application driver object code from audio cassette.

d. Program the PROM.

3.4.2 Separate Application Driver and Program

Use this procedure to program separate PROMs for the application driver and program; for example, to prepare a 16K-byte application program in four 4K-byte PROMs for installation in an RM 65 16K PROM/ROM module and an application in a 2K-byte PROM for installation in the RM 65 SBC module.
a. Zero memory in the PROM area.

b. Load the application driver or program object code.

c. Program the PROM.
SECTION 4

APPLICATION DRIVER REQUIREMENTS AND EXAMPLES

4.1 APPLICATION DRIVER REQUIREMENTS

This section defines the requirements for the application driver for both interactive and run-time operation.

The application driver consists of three major parts:

- Startup Routine
- I/O Vectors and Handlers
- Interrupt Vectors and Handlers

A flowchart of the application driver is shown in Figure 4-1. An annotated assembly listing of a model driver is shown in Figure 4-2. This model driver should be adapted and expanded as required for your specific application requirements. Two example drivers are described in Section 4.2.

4.1.1 Startup Routine

The startup routine must initialize the run-time BASIC, load the program, variable and I/O handler vectors, and jump to the BASIC entry point. This driver is usually entered by keyboard command through the Monitor in the interactive mode, or vectored to from the RES vector in the run-time mode. Some of the steps may be reordered without affecting operation. Thorough testing should be performed in the interactive mode if any changes are made, however, including the incorporation of application I/O handlers.

Be sure that the variables are located above the program during interactive operation (they can be located anywhere in RAM later for run-time operation).
STARTUP ROUTINE

RESET VECTOR

ENTER

CALL BASIC INIT SUBROUTINE

INITIALIZE I/O VECTOR TABLE

INITIALIZE MEMORY POINTERS

CALL INIT RAM SUBROUTINE

MODE?

RUN-TIME

DEVELOPMENT

ZERO START OF PROGRAM RAM

JUMP TO BASIC ENTRY

Figure 4-1. Application Driver Flowchart
Figure 4-1. Application Driver Flowchart (Cont'd)
THIS IS A MODEL R/T BASIC DRIVER

R/T BASIC ENTRY POINTS
INIT=$CF11  , INITIALIZE BASIC PARAMETERS
WARM=$B099  , WARM ENTRY POINT FOR R/T BASIC
CLEARC=$B5A4  , INITIALIZE VARIABLE SPACE
RUNC=$B5A4  , SET EXECUTION FOR FIRST LINE
NEWSTT=$B6DB  , INITIALIZE VARIABLE SPACE

R/T BASIC VARIABLES
VECTBL=$200  , VECTOR TABLE OF I/O DRIVERS
OUTFLG=$243  , ADD FOR R/T BASIC
CLRLIN=$242  , ISSUED TO CLEAR EACH LINE
SAVFLG=$2DF  , TEMPORARY PRIFLG STORAGE

PGMST IS THE ADDRESS WHERE THE PROGRAM IS DEVELOPED
(DEVELOPMENT MODE) OR WHERE THE PROGRAM IS EXECUTED
(RUN-TIME MODE) IF THESE ADDRESSES ARE DIFFERENT,
THE FINAL PROGRAM MUST BE RELOCATED TO THE RUN-TIME
ADDRESS USING THE BASIC RELOCATER PROGRAM.

VARST=#301
VARST IS THE ADDRESS OF RAM IMMEDIATELY ABOVE THE
DEVELOPING PROGRAM (DEVELOPMENT MODE) OR RAM AVAILABLE
IN FINAL SYSTEM (RUN-TIME MODE)

VARST=#303
TOPMEM IS THE TOP OF VARIABLE RAM
(DEVLOPMENT MODE AND RUN-TIME MODE).

TOPMEM=#800
TOP OF USER RAM
START=#F800
ADDRESS OF THE USER PROM

RUN TIME BASIC DRIVER PROGRAM
ON INITIAL ENTRY TO R/T BASIC, USE COLD.
(COLD IS THE USER DRIVER PROGRAM)
ON RE-ENTRY TO R/T BASIC, USE WARM.
(WARM IS IN THE R/T BASIC ROM)

START=ADDRESS OF THE USER PROM
COLD JSR INIT  , COLD RESET INTO R/T BASIC
COLD RESET INTO R/T BASIC
DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM

Figure 4-2. Model Application Driver Assembly Listing

4-4
SET POINTER TO SCRATCH PAD RAM.

DURING PROGRAM DEVELOPMENT, THIS AREA
MUST BE ABOVE THE PROGRAM AREA IN RAM.

LDA #CVARST
LDY #>VARST
STA #24
STY #25

SET POINTER TO TOP OF MEMORY.
DURING PROGRAM DEVELOPMENT, THIS LIMIT MUST
BE ABOVE THE PROGRAM AND VARIABLE SPACE.

LDA #TOPMEM
LDY #>TOPMEM
STA #2E
STY #2F

CHANGE LENGTH OF LINE (LINWID) IF REQUIRED
AFFECTS LIST AND PRINT WITH,)
DEFAULT WIDTH IS 80 CHARACTERS (#50)
CHANGE POSITION OF LAST PRINT FIELD (NCMID) IF REQUIRED
AFFECTS PRINT WITH, BUT LINWID OVERIDES
DEFAULT POSITION IS THE 30TH CHARACTER (#1E)

INITIALIZE RAM POINTERS TO A CLEARED STATE

FOR DEVELOPMENT MODE, USE THIS CODE TO START
AT THE BOTTOM OF AVAILABLE RAM. THIS IS NOT THE
ADDRESS WHERE THE FINAL PROGRAM WILL RESIDE.

FOR RUN-TIME MODE, USE THIS CODE TO COME UP RUNNING
SETUP R/T BASIC FOR RUN
AND EXECUTE AWAY ...

I/O VECTOR TABLE IS SET UP WITH USER I/O DRIVERS.

<table>
<thead>
<tr>
<th>TABLE</th>
<th>WOR</th>
<th>WHERIN</th>
<th>OPEN INPUT &amp; SET IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEROT</td>
<td>SCRLOW</td>
<td>OUTPUT ER &amp; SET OUT</td>
<td></td>
</tr>
<tr>
<td>CRLF</td>
<td>OUTPUT CR TO TERMINAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTCLO</td>
<td>CLOSE THE OUTPUT FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCLO</td>
<td>CLOSE THE INPUT FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INALL</td>
<td>INPUT THROUGH THE AOD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTALL</td>
<td>OUTPUT TO THE AOD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANKEY</td>
<td>RETURN Z=1 IF NO KEY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESPTR</td>
<td>RESTORE PRINTER STATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORCEP</td>
<td>FORCE PRINT &amp; SAVE STATUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHK-CTC</td>
<td>RETURN KEY DOWN IN A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>OBJECT</td>
<td>SOURCE</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>F054</td>
<td>EA</td>
<td>WHERE IN NOP</td>
<td></td>
</tr>
<tr>
<td>F055</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F056</td>
<td>EA</td>
<td>WHEREOT NOP</td>
<td></td>
</tr>
<tr>
<td>F057</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F058</td>
<td>EA</td>
<td>SCRLow NOP</td>
<td></td>
</tr>
<tr>
<td>F059</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F05A</td>
<td>EA</td>
<td>CRLF NOP</td>
<td></td>
</tr>
<tr>
<td>F05B</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F05C</td>
<td>EA</td>
<td>OUTCLO NOP</td>
<td></td>
</tr>
<tr>
<td>F05D</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F05E</td>
<td>EA</td>
<td>INCLO NOP</td>
<td></td>
</tr>
<tr>
<td>F05F</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F060</td>
<td>EA</td>
<td>INALL NOP</td>
<td></td>
</tr>
<tr>
<td>F061</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F062</td>
<td>EA</td>
<td>OUTALL NOP</td>
<td></td>
</tr>
<tr>
<td>F063</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F064</td>
<td>EA</td>
<td>ANYKEY NOP</td>
<td></td>
</tr>
<tr>
<td>F065</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F066</td>
<td>EA</td>
<td>RESPTR NOP</td>
<td></td>
</tr>
<tr>
<td>F067</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F068</td>
<td>EA</td>
<td>FORCEP NOP</td>
<td></td>
</tr>
<tr>
<td>F069</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>F06A</td>
<td>EA</td>
<td>CHKCTC NOP</td>
<td></td>
</tr>
<tr>
<td>F06B</td>
<td>60</td>
<td>RTS</td>
<td></td>
</tr>
</tbody>
</table>

; THESE I/O ROUTINES WILL BE DEPENDENT ON THE SYSTEM.
; TYPICALLY EACH NOP WOULD BE REPLACED
; WITH APPLICABLE CODE OR ELIMINATED.

; WHEREIN NOP
; WHEREOT NOP
; SCRLow NOP
; CRLF NOP
; OUTCLO NOP
; INCLO NOP
; INALL NOP
; OUTALL NOP
; ANYKEY NOP
; RESPTR NOP
; FORCEP NOP
; CHKCTC NOP

; THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...
; THIS IS THE RUN TIME ADDRESS (BEGIN).

.BY 0
BEGIN .DBY 0
. END

ERRORS=0000

Figure 4-2. Model Application Driver Assembly Listing (Cont'd)
4.1.2 I/O Vectors and Handlers

Since all I/O on the RM 65 Run-Time BASIC is vectored, both vectors and I/O handlers must be included in the application driver. Table 4-1 summarizes the vectors and identifies equivalent AIM 65 subroutines corresponding to the vectors. Table 4-2 describes the detailed I/O subroutine requirements.

Dummy I/O subroutines are shown in the model driver in Figure 4-2. If no I/O is required in the application program, these dummy drivers are not needed since the BASIC initialization subroutine (INIT) loads the I/O vectors to point to RTS instructions internal to RM 65 Run-Time BASIC. If application dependent I/O is needed, replace the NOP instructions with the required instructions.

4.1.3 Interrupt Vectors and Handlers

During interactive operation, the R6502 CPU hardware interrupt vectors at $FFFA-$FFFF are included in the AIM 65 Monitor. User alterable vectors (IRQV4 at $A400$, NMIV2 at $A402$, and IRQV2 at $A404$) provide linkage to the application program interrupt handler during development. Refer to Section 7.8 in the AIM 65 User's Guide for additional information.

For run-time operation, these three vectors must be included in the run-time ROM mapped into $FXXX$ address range. The RES vector should point to the first address of the startup routine while the IRQ and NMI vectors should point to their respective handlers. Interrupt handler linkage is included in the model driver as a guideline.
### Table 4-1. I/O Vector Summary

<table>
<thead>
<tr>
<th>Vector Location</th>
<th>Vector Location</th>
<th>Vector Name</th>
<th>Used by</th>
<th>Purpose</th>
<th>AIM 65 Subroutine</th>
<th>AIM 65 Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>$200-$201</td>
<td>$202-$203</td>
<td>WHERE! LOAD</td>
<td>Determine AID.</td>
<td>WHERE!</td>
<td>$E848</td>
<td>$E848</td>
</tr>
<tr>
<td>$204-$205</td>
<td>SCRLW Command</td>
<td>Processing</td>
<td>Output CR &amp; LF to display/printer.</td>
<td>CRCK</td>
<td>$EA24</td>
<td></td>
</tr>
<tr>
<td>$206-$207</td>
<td>CRLF System</td>
<td>Output</td>
<td>Output a CR to the AOD.</td>
<td>CRLF</td>
<td>$E9F0</td>
<td></td>
</tr>
<tr>
<td>$208-$209</td>
<td>OUTCLO PRINT</td>
<td>Close the AOD.</td>
<td>PRINT!</td>
<td>DUL1</td>
<td>$E50A</td>
<td></td>
</tr>
<tr>
<td>$20A-$20B</td>
<td>INCLO INPUT</td>
<td>Close to AID.</td>
<td>INPUT!</td>
<td>DUL3</td>
<td>$E520</td>
<td></td>
</tr>
<tr>
<td>$20C-$20D</td>
<td>INALL INPUT</td>
<td>Input a character.</td>
<td>INPUT!</td>
<td>INALL</td>
<td>$E993</td>
<td></td>
</tr>
<tr>
<td>$20E-$20F</td>
<td>OUTALL PRINT</td>
<td>Output a character to the AOD.</td>
<td>PRINT!</td>
<td>OUTALL</td>
<td>$E9BC</td>
<td></td>
</tr>
<tr>
<td>$210-$211</td>
<td>ANYKEY GET</td>
<td>Check keyboard for key down.</td>
<td>GET</td>
<td>ROONEK</td>
<td>$ECEF</td>
<td></td>
</tr>
<tr>
<td>$212-$213</td>
<td>CLOPTR INPUT!</td>
<td>Close printer input.</td>
<td>PRINT!</td>
<td>ROONEK</td>
<td>$ECEF</td>
<td></td>
</tr>
<tr>
<td>$214-$215</td>
<td>OPNPTR INPUT!</td>
<td>Open printer output.</td>
<td>PRINT!</td>
<td>ROONEK</td>
<td>$ECEF</td>
<td></td>
</tr>
<tr>
<td>$216-$217</td>
<td>CHKCTC Command</td>
<td>Input a character from the keyboard.</td>
<td>input</td>
<td>ROONEK</td>
<td>$ECEF</td>
<td></td>
</tr>
</tbody>
</table>

#### NOTES

1. Call from user-provided subroutine which performs other processing (see Figure 4-2).
2. Call from user-provided subroutine (see Figure 4-2).
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEREI</td>
<td>WHEREI is called by the LOAD function to determine the active input device (AID). WHEREI must return a character in the A register which identifies the AID. The subroutine called through the INALL vector will then input a character from the AID. No register values must be saved. In an AIM 65 system, this vector should point to the AIM 65 Monitor WHEREI subroutine.</td>
</tr>
<tr>
<td>WHEREO</td>
<td>WHEREO is called by the SAVE function to determine the active output device (AOD). WHEREO must return a character in the A register which identifies the AOD. The subroutine called through the OUTALL vector will then output a characters to the AOD. No register values must be saved. In an AIM 65 system, this vector should point to the AIM 65 Monitor WHEREO subroutine.</td>
</tr>
<tr>
<td>SCRLow</td>
<td>SCRLow is called to output a CR ($0D) to the system terminal. It is called only if the value of the OUTFLAG ($0243) is zero; otherwise, all CR characters are output through vector CRLF. The X and Y register values must be saved and the A register must not return a value of $FF. In an AIM 65 system, this vector should point to the AIM 65 Monitor CRCK subroutine.</td>
</tr>
<tr>
<td>Subroutine</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>CRLF</strong></td>
<td>CRLF is called to output a CR ($0D) to the AOD used by OUTALL. The X and Y register values must be saved and the A register must not return a value of $FF. In an AIM 65 system, this vector should point to the AIM 65 Monitor CRLF subroutine.</td>
</tr>
<tr>
<td><strong>OUTCLO</strong></td>
<td>OUTCLO is called to close the current AOD used by OUTALL and to restore the system terminal as the AOD. No register values need to be saved. In an AIM 65 system, this vector should point to the DU11 subroutine.</td>
</tr>
<tr>
<td><strong>INCLO</strong></td>
<td>INCLO is called to close the current AOD used by OUTALL and to restore the system terminal as the AID. No register values must be saved. In an AIM 65 system, this vector should point to the DU13 subroutine.</td>
</tr>
<tr>
<td><strong>INALL</strong></td>
<td>INALL is called by the input command processing and the INPUT and READ functions. INALL must input a character from the AID. It does not have to echo characters nor process DELETE ($7F) characters. The Y register is the index into the input buffer. The ASCII value of the input characters must be returned in the A register. The X register value must be saved. The Y register must contain the character count minus one. In an AIM 65 system, this vector should point to the AIM 65 Monitor INALL subroutine.</td>
</tr>
</tbody>
</table>
### Table 4-2. I/O Vector Description (Continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OUTALL</strong></td>
<td>OUTALL is called to output a character to the AOD. Run-Time BASIC also outputs a Clear Screen to Right character through OUTALL. The value of this character (normally $02$) must be stored in variable CLRLIN ($242$). CLRLIN is initially set to $FF$. The ASCII value of the output character must be in the A register. All registers must be saved. In an AIM 65 system this vector should point to the AIM 65 Monitor OUTALL subroutine.</td>
</tr>
<tr>
<td><strong>ANYKEY</strong></td>
<td>ANYKEY is called by the GET function to sample the system terminal keyboard. The CPU zero flag (Z) is set if a key is not depressed, otherwise the zero flag is reset. No register values must be saved. In an AIM 65 system, this vector should point to a user provided subroutine which sets the ROLLFL flag ($A47F$) and calls ROONEK.</td>
</tr>
<tr>
<td><strong>CLOPTR</strong></td>
<td>CLOPTR is called by the PRINT! and INPUT! functions. CLOPTR must close the printer output and restore the printer status in PRIFLG ($0247$) to the value it was before the OPNPTR subroutine was called. The printer status can be saved in SAVFLG ($02DF$). The X and Y register values must be saved. In the AIM 65 system, the saved printer status must be stored in PRIFLG ($A411$)</td>
</tr>
<tr>
<td>Subroutine</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>OPNPTR</strong></td>
<td>OPNPTR is called only by the PRINT! and INPUT! functions. OPNPTR must save the current printer status in PRIFLG (at $0247) into a temporary location, e.g., SAVFLG at $02DF, and open the printer output by storing $80 into PRIFLG. Do not save the printer status on the stack. The X and Y register values must be saved. In the AIM 65 system, the printer status in PRIFLG (at $A411) must be saved and $80 stored in PRIFLG (at $A411).</td>
</tr>
<tr>
<td><strong>CHKCTC</strong></td>
<td>CHKCTC is called by the command input function. CHKCTC must check to see if a character is available from the system terminal keyboard and, if it is, load the ASCII value for the key into the A register. The character code will then be checked for a break command, in this case, $1B (ESC). The X and Y register values need not be saved. In the AIM 65 system, the ROONEK subroutine should be called.</td>
</tr>
</tbody>
</table>
4.2 EXAMPLE APPLICATION DRIVERS

4.2.1 Interactive Operation On An AIM 65 Microcomputer

Figure 4-3 shows a typical AIM 65 Microcomputer-based development configuration. An example application driver to support this system is shown in Figure 4-4.
Figure 4-3. Typical AIM 65 Development Configuration
PAGE 0001
RM 65 RUN TIME BASIC DRIVERS - DEVELOPMENT MODE

ADDR OBJECT SOURCE

; THIS EXAMPLE IS TO DEVELOP SHORT BASIC PROGRAMS (LESS THAN 4K BYTES) ON AIM 65 WITHOUT ADDITIONAL RAM.

; AIM 65 MONITOR I/O ROUTINES EQUATES
WHEREI=#E848
WHEREO=#E871
CRLK=#E9F0
DU11=#E50A
DU12=#E520
ROONEK=#ECEF
GETKY=#EC43
INALL=#E993
OUTALL=#E9BC
PSLS=#E7DC
CUREAD=#E833

; AIM 65 VARIABLES USED BY THE I/O ROUTINES
OUTDEV=#A413
INFLG=#A412
PRIFLG=#A411
ROLLFL=#A47F

; R/T BASIC ENTRY POINTS
INIT=#CF11
; INITIALIZE BASIC PARAMETERS
WARM=#B099
; WARM ENTRY POINT FOR R/T BASIC
CLEAPC=#B5A4
; INITIALIZE VARIABLE SPACE

; R/T BASIC VARIABLES
VECTBL=#200
; VECTOR TABLE OF I/O DRIVERS
OUTFLG=#243
; ADD FOR R/T BASIC
CLRLIN=#242
; ISSUED TO CLEAR EACH LINE
SAVFLG=#2DF
; TEMPORARY PRIFLG STORAGE

; PROGRAM EQUATES FOR DEVELOPING THE PROGRAM
PGMST=BEGIN
; USER PROGRAM START ADDRESS
VARST=BEGIN+2
; FIRST BYTE OF SCRATCH PAD RAM
TOPMEM=#1000
; TOP OF USER RAM
START=#300
; ADDRESS TO START FROM

; SET UP MONITOR LINKAGE TO ENTER R/T BASIC FROM KEYBOARD
<CF1> = COLD ENTRY
<CF2> = WARM ENTRY
++#10C

010C 4C 00 03 JMP COLD
010F 4C 99 B0 JMP WARM

Figure 4-4. Example AIM 65 Interactive Driver

4-15
PAGE 0002 RM 65 RUN TIME BASIC DRIVERS - DEVELOPMENT MODE

ADDR OBJECT SOURCE

; ON INITIAL ENTRY TO R/T BASIC , USE COLD.
; ( COLD IS THE USER DRIVER PROGRAM )
; 
; ON RE-ENTRY TO R/T BASIC , USE WARM.
; ( WARM IS IN THE R/T BASIC ROM )

**START ; ADDRESS OF THE PROM
0300 20 11 CF COLD JSR INIT ; COLD RESET INTO R/T BASIC
0303 A2 17 ; DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM
0305 B0 30 03 SETUP LDA TABLE.X
0308 9D 00 02 STA VECTBL.X
030B CA DEX
030C 10 F7 BPL SETUP
030E A9 02 ; CLR LIN CHARACTER IS ISSUED AT THE END OF EVERY LINE
0310 80 42 02 STA CLRLIN

; SET POINTER TO USER PROGRAM PROM
0313 A9 B4 LDA #$17 ; 12 VECTORS
0315 A0 03 LDY #$PGMST
0317 85 22 STA #22
0319 84 23 STY #23

; SET POINTER TO SCRATCH PAD RAM.
031B A9 B6 LDA #$VARST
031D A0 03 LDY #$VARST
031F 85 24 STA #24
0321 84 25 STY #25

; SET POINTER TO TOP OF MEMORY.
0323 A9 00 LDA #$TOPMEM
0325 A0 10 LDY #$TOPMEM
0327 85 2E STA #2E
0329 84 2F STY #2F

; INITIALIZE RAM POINTERS TO A CLEARED STATE
032B 20 A4 B5 JSR CLEARC

; CLEAR PROGRAM AREA
032E A9 00 LDA #0
0330 8D B3 03 STA PGMST-1
0333 8D B4 03 STA PGMST
0336 8D B5 03 STA PGMST+1
0339 4C 99 E0 JMP WARM ; COME UP IN BASIC

; R/T BASIC I/O TABLE STRUCTURE AND I/O ROUTINES

033C 4E E8 TABLE WOR WHEREI ; WHEREIN - OPEN INPUT & SET IN
033E 54 03 WOR WHROUT , WHEREOUT - OPEN OUTPUT & SET OUT
0340 24 E8 WOR CRCK , SCRL - OUTPUT CR TO TERMINAL
0342 F0 E9 WOR CRLF , CRLF - OUTPUT A CR TO THE AOD
0344 85 03 WOR OUTCLS , OUTCLO - CLOSE THE OUTPUT FIFO
0346 20 E5 WOR DULS , INCOL - CLOSE THE INPUT FILE
0348 64 03 WOR INI , INALL - INPUT THROUGH THE A/D
034A 7E 03 WOR OUTU , OUTALL - OUTPUT TO THE AOD
034C A4 03 WOR ROONU , ANYKEY - RETURN Z=1 IF NO KEY
034E 95 03 WOR CLOPTR , RESPTR - RESTORE PRINTER STATE
0350 8B 03 WOR OPNPTR , FORCEP - FORCE PRINT & SAVE STATUS
0352 9C 03 WOR CHKCTC , CHKCTC - RETURN KEY DOWN IN A

---

Figure 4-4. Example AIM 65 Interactive Driver (Cont'd)
PAGE 0003  RM 65 RUN TIME BASIC DRIVERS - DEVELOPMENT MODE

ADDR OBJECT SOURCE

I/O ROUTINES NOT COMPATIBLE WITH AIM 65 MONITOR

0354 20 71 E8 WHROUT JSR WHERE?  ;OUTPUT DEVICE?
0357 AD 13 A4 LDA OUTDEV
035A C9 00 CMP #$00
035D D0 02 BNE STOROT
035E A9 00 STOR00 LDA #0  ;DEVICE 00 TO SUPPRESS EOF
0360 8D 43 02 STOROT STA OUTFLG
0363 60 DORTS RTS

0364 AD 12 A4 INU LDA INFLG
0367 C9 00 CMP #$00
0369 F0 06 BEQ DORTM  ;TERMINAL MUST ALSO ECHO
036B 4C 93 E9 STOR00 LDA #00 TO SUPPRESS EOF

0374 C9 7F CMP #$7F
0376 D0 02 BNE STOROT
0378 00 00 BNE DORTM  ;BACK UP THE DISPLAY
037A 1C 1C JSR PSLS  ;DELETE

037C 12 A4 INU LDA INFLG
037E 02 0A E5 OUTCLS JSR: DORTM

0381 98 11 C9 CB ; SET OR CLEAR

0385 20 0A E5 OUTCLS JSR DU11  ;SET TERMINAL AS OUTPUT ==>
0388 4C 5E 03 JMP STOR00

0388 AD 11 A4 OPNPTR LDA PRIFLG  ;SAVE PRINTER STATUS
038E 8D DF 02 STA SAVFLG
0391 A9 80 LDA #$80  ;FORCE PRINTER ON
0393 D0 03 BNE STRPTR  ;ALWAYS ==>

0395 AD DF 02 CLOPTR LDA SAVFLG  ;RECOVER PRINTER STATUS
0398 8D 11 A4 STRPTR STA PRIFLG
039B 60 RTS

039C 20 EF EC CHKCTC JSR ROONEK  ;KEY DOWN?
039F F0 C2 BEQ DORTS  ;NO ==>
03A1 4C 43 EC JMP GETKY

03A4 A9 FF ROONU LDA #$FF  ;MAKE IT READ KEY AGAIN
03A6 AD 7F A4 STA ROLLFL
03A8 A9 FF LDA #$FF
03AA AD 7F A4 STA ROLLFL  ;MAKE IT READ KEY NEXT TIME
03AB 98 TYA
03B2 60 RTS

THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...

03B3 00 .BYT 0
03B4 00 00 BEGIN .DEY 0

Figure 4-4. Example AIM 65 Interactive Driver (Cont'd)
4.2.2 Run-Time Operation in an RM 65 SBC Module

A typical RM 65 run-time configuration is shown in Figure 4-5. An example run-time driver is shown in Figure 4-6. This example system uses a CRT display/keyboard terminal with an RS-232C serial interface as one application interface and an 80-column printer with a parallel interface as a second application connection.
Figure 4-5. Typical RM 65 Run-Time Configuration

4-19
THIS EXAMPLE IS TO COME UP RUNNING A BASIC PROGRAM.
THE FINAL SYSTEM IS AN SBC, ACIA, AND MPI MODULE.
THE USER PROGRAM RESIDES IN PROM ON THE SBC.

R/T BASIC ENTRY POINTS

0000 INIT=$CF11  ; INITIALIZE BASIC PARAMETERS
0000 RUNC=#E593  ; SET EXECUTION FOR FIRST LINE
0000 NEWST=$E6DB  ; SAME AS 'NEW' COMMAND
0000 WARM=#E093  ; WARM ENTRY POINT FOR R/T BASIC
0000 CLEARC=#E5A4  ; INITIALIZE VARIABLE SPACE

R/T BASIC VARIABLES

0000 VECTBL=#200  ; VECTOR TABLE OF I/O DRIVERS
0000 OUTFLEG=#243  ; AOD FOR R/T BASIC
0000 CLFLEG=#242  ; CHARACTER ISSUED TO CLEAR EACH LINE
0000 SAVFLEG=#2DF  ; TEMPORARY PRIFLAG STORAGE

PROGRAM EQUATES FOR THE USER PROGRAM

0000 PGMST=BEGIN  ; START PROGRAM AFTER I/O DRIVER
0000 VARST#=301  ; FIRST FREE BYTE OF SBC RAM
0000 TOPMEM=#800  ; TOP OF SBC RAM
0000 START=#DFF  ; COMPATIBLE WITH AIM 65

( ( RS-232 SERIAL CRT TERMINAL ) )

ACIA MODULE REGISTER DEFINITIONS

7000 ACIA  *=*++1  ; BASE ADDRESS IS 70XX
7001 STATUS  *=*++1
7002 CYN0  *=*++1
7003 CTRL  *=*++1

( ( CENTRONICS TYPE PRINTER INTERFACE ) )

DATA IS SET UP ON MPI VIA NO. 2 PORT A (BITS 0-6)
DATA STROBE NOT IS ON VIA NO. 2 PORT B (BIT 0)
DATA ACKNOWLEDGE NOT IS SENSED ON VIA NO. 2 CA2
MPI VIA REGISTER DEFINITIONS

7110 PORTB  *=*++1  ; BASE ADDRESS IS 71XX
7111 PORTA  *=*++1
7112 DDRB  *=*++1
7113 DDRA  *=*++9
711C PRC  *=*++1
711D IFR  *=*++1
711E MPIOR=#7120  ; MPI DATA DIRECTION REGISTER

MPI MODULE PRINTER CONSTANTS

711E IDRAB=#FF  ; BOTH DATA PORTS ARE OUTPUT
711E IMPIDR=#C0  ; VIA NO. 2 PORTS A AND B OUTPUTS
711E IPCR=#04  ; POSITIVE EDGE ON ACKNOWLEDGE

SET UP CPU VECTORS TO POINT TO COLD

711E *=#$DFFA  ; ALSO DOUBLE MAPS $FFFFA
DFFA 00 DO  ; WOR COLD
DFFC 00 DO  ; WOR COLD
DFFE 00 DO  ; WOR COLD

Figure 4-6. Example RM 65 SBC Run-Time Driver
RUN TIME BASIC DRIVER PROGRAM
\[\text{ON ENTRY TO R/T BASIC, USE COLD}\]
\[\text{START}\] ADDRESS OF THE USER PROM
\[\text{COLD}\] JSR INIT COLD RESET INTO R/T BASIC
\[\text{DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM}\]
LDX #$17, 12 VECTORS
LDA TABLE,X
STA VECTBL,X
DEX
BPL SETUP
\[\text{INITIALIZE THE ACIA MODULE}\]
OPENAC LDA #$08, DTR=ON, IRQ=OFF, NO ECHO, NO PARITY
STA CHND
LDA #$1E, 8-BITS, 1 STOP BIT, 9600 BAUD
STA CTRL
ACIA #$2
CLRLIN CHARACTER IS ISSUED AT THE END OF EVERY LINE
LDA #2
STA CLRLIN
SET POINTER TO USER PROGRAM PROM
LDA <$PGMST
LDY <$PGMST
STA #22
STA #23
SET POINTER TO SCRATCH PAD RAM.
LDA <$VARST
LDY <$VARST
STA #24
STA #25
SET POINTER TO TOP OF MEMORY.
LDA <$TOPMEM
LDY <$TOPMEM
STA #2E
STA #2F
\[\text{CHANGE LENGTH OF LINE (LINWID) IF REQUIRED}\]
\[\text{DEFAULT WIDTH IS 80 CHARACTERS ($50)}\]
\[\text{CHANGE POSITION OF LAST PRINT FIELD (NCLMWID) IF REQUIRED}\]
\[\text{DEFAULT POSITION IS THE 30TH CHARACTER ($1E)}\]
\[\text{INITIALIZE RAM POINTERS TO A CLEARED STATE}\]
JSR CLEARC
FOR RUN-TIME MODE, USE THIS CODE TO COME UP RUNNING.
JSR RUNC SETUP R/T BASIC FOR RUN
JMP NEWSST AND EXECUTE AWAY...
\[\text{RUN-TIME BASIC I/O TABLE STRUCTURE}\]
TABLE WOR RETURN OPEN INPUT & SET IN
WOR RETURN OPEN OUTPUT & SET OUT
WOR CRLF OUTPUT CR TO THE TERM
WOR CRLF OUTPUT CR+LF TO THE AOD

Figure 4-6. Example RM 65 SBC Run-Time Driver (Cont'd)
Figure 4-6. Example RM 65 SBC Run-Time Driver (Cont'd)
WAIT UNTIL AN ACKNOWLEDGE IS RECEIVED FROM THE PRINTER

LDA IFR ; ACKNOWLEDGE IS ON CA2
LSR A ; MOVE CA2 INTO CARRY FLAG
BCC WAIT ; NOT READY? ==>

HANDSHAKE OFF THE CHARACTER

STROBE LDA #0 ; FORCE STROBE LOW
STA PORTB
LDA #1 ; FORCE STROBE HIGH
STA PORTB
RTS

CHECK FOR ANY KEY DEPRESSION

LDA STATUS
AND #$08 ; SET Z=1 FOR KEY DOWN
RTS

RESTORE THE TERMINAL AS OUTPUT

RESFR LDA #00
STA OUTFLG
RTS

FORCE THE PRINTER AS OUTPUT

LDA #P ; STA OUTFLG
STA OUTFLG
ON OPEN SET UP THE VIA AND THE DATA PORT BUFFERS

PROPEN LDA #$0D ; ISSUE A CR AT FIRST
STA PORTA
LDA #MPIDR
STA MPIDR
LDA #MPIDR
LDA #IPCR
STA DDRA
STA DDRB
STA DDRA
STA DDRB
STA DDRA
STA DDRB
JSR STROBE
RTS

CHECK TO SEE IF ANYTHING HAS BEEN RECEIVED

JSR ANYKEY
BEQ RETURN ; NO KEY ==>
JSR SINPUT
RETURN RTS

THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...

BYT 0
BEGIN 0
END

Figure 4-6. Example RM 65 SBC Run-Time Driver (Cont’d)
APPENDIX A

BASIC VARIABLES

The location of the variables for RM 65 Run-Time BASIC is different from either the AIM 65 BASIC (A65-020) or the AIM 65/40 BASIC (A65/40-7020). Most of these variables, however, with the exception of the I/O vectors at $200-214, are not normally accessed directly by the RM 65 Run-Time application program. The variable locations are listed in this appendix, however, should the application program need to address them explicitly. Application programs developed on AIM 65 or AIM 65/40 BASIC then rehosted on RM 65 BASIC must have the locations of any of these variables changed as appropriate.

Tables A-1 and A-2 list the page zero and page two usage, respectively, by RM 65 BASIC.

Table A-3 lists the page zero usage by AIM 65 BASIC.
Table A-1. RM 65 Run-Time BASIC Page Zero Usage

<table>
<thead>
<tr>
<th>Addr (Hex)</th>
<th>Addr (Dec)</th>
<th>No. Bytes</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Search Character</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>Scan-Between-Quotes Flag</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>1</td>
<td>Input Buffer Pointer</td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>1</td>
<td>Default DIM Flag</td>
</tr>
<tr>
<td>04</td>
<td>4</td>
<td>1</td>
<td>TYPE: FF=string, 00=numeric</td>
</tr>
<tr>
<td>05</td>
<td>5</td>
<td>1</td>
<td>TYPE: 80=integer, 00=float point</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>1</td>
<td>Data Scan Flag; List Quote Flag; Memory Flag</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>1</td>
<td>Subscript Flag; FNx flag</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>1</td>
<td>0=Input; $40=GET; $98=READ</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>1</td>
<td>Comparison Evaluation Flag</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>1</td>
<td>Flag; Suppress Output if Minus</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>1</td>
<td>Position of Terminal Carriage</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>1</td>
<td>Width (length of line)</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>1</td>
<td>Position Beyond Output Fields</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>1</td>
<td>Temp String Desc. Stack Pointer</td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>1</td>
<td>Last Temp String Pointer</td>
</tr>
<tr>
<td>10-18</td>
<td>16-24</td>
<td>9</td>
<td>Stack of Temp String Descriptors</td>
</tr>
<tr>
<td>19-1A</td>
<td>25-26</td>
<td>2</td>
<td>Pointer for Number Transfer</td>
</tr>
<tr>
<td>1B-1C</td>
<td>27-28</td>
<td>2</td>
<td>Misc. Number Pointer</td>
</tr>
<tr>
<td>1D-21</td>
<td>29-33</td>
<td>5</td>
<td>Product Staging Area for Multiply</td>
</tr>
<tr>
<td>22-23</td>
<td>34-35</td>
<td>2</td>
<td>Pointer: Start of BASIC Memory</td>
</tr>
<tr>
<td>24-25</td>
<td>36-37</td>
<td>2</td>
<td>Pointer: Start of Variables</td>
</tr>
<tr>
<td>26-27</td>
<td>38-39</td>
<td>2</td>
<td>Pointer: Start of Arrays</td>
</tr>
<tr>
<td>28-29</td>
<td>40-41</td>
<td>2</td>
<td>Pointer: End of Arrays</td>
</tr>
<tr>
<td>2A-2B</td>
<td>42-43</td>
<td>2</td>
<td>Pointer: Bottom of Strings</td>
</tr>
<tr>
<td>2C-2D</td>
<td>44-45</td>
<td>2</td>
<td>Pointer: Utility String</td>
</tr>
<tr>
<td>2E-2F</td>
<td>46-47</td>
<td>2</td>
<td>Pointer: Limit of BASIC</td>
</tr>
<tr>
<td>30-31</td>
<td>48-49</td>
<td>2</td>
<td>Current BASIC Line No.</td>
</tr>
<tr>
<td>32-33</td>
<td>50-51</td>
<td>2</td>
<td>Previous BASIC Line No.</td>
</tr>
<tr>
<td>34-35</td>
<td>52-53</td>
<td>2</td>
<td>Integer Address</td>
</tr>
<tr>
<td>36-37</td>
<td>54-55</td>
<td>2</td>
<td>Pointer to Basic Statement</td>
</tr>
<tr>
<td>38-39</td>
<td>56-57</td>
<td>2</td>
<td>Current DATA Line No.</td>
</tr>
<tr>
<td>3A-3B</td>
<td>58-59</td>
<td>2</td>
<td>Pointer to Current Data</td>
</tr>
<tr>
<td>3C-3D</td>
<td>60-61</td>
<td>2</td>
<td>Input Vector</td>
</tr>
<tr>
<td>Addr (Hex)</td>
<td>Addr (Dec)</td>
<td>No. Bytes</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>3E-3F</td>
<td>62-63</td>
<td>2</td>
<td>Current Variable Name</td>
</tr>
<tr>
<td>40-41</td>
<td>64-65</td>
<td>2</td>
<td>Current Variable Memory Address</td>
</tr>
<tr>
<td>42-43</td>
<td>66-67</td>
<td>2</td>
<td>Variable Pointer Memory Address</td>
</tr>
<tr>
<td>44-45</td>
<td>68-69</td>
<td>2</td>
<td>Utility Pointer and Save</td>
</tr>
<tr>
<td>46</td>
<td>70</td>
<td>1</td>
<td>Comparison Symbol Accumulator</td>
</tr>
<tr>
<td>47-4C</td>
<td>71-76</td>
<td>6</td>
<td>Misc. Numeric Work Area</td>
</tr>
<tr>
<td>4D-4F</td>
<td>77-79</td>
<td>2</td>
<td>Jump Vector for Functions</td>
</tr>
<tr>
<td>50-59</td>
<td>80-89</td>
<td>10</td>
<td>Misc. Numeric Work and Storage Area</td>
</tr>
<tr>
<td>5A-5F</td>
<td>90-95</td>
<td>6</td>
<td>Accumulator No. #1 (E,M,M,M,M,S)</td>
</tr>
<tr>
<td>60</td>
<td>96</td>
<td>1</td>
<td>Degree of Polynomial to Evaluate</td>
</tr>
<tr>
<td>61</td>
<td>97</td>
<td>1</td>
<td>Bits to Shift Right</td>
</tr>
<tr>
<td>62-67</td>
<td>98-103</td>
<td>5</td>
<td>Accumulator No. 2 (E,M,M,M,M,S)</td>
</tr>
<tr>
<td>68</td>
<td>104</td>
<td>1</td>
<td>Sign of Accumulators EOR'd.</td>
</tr>
<tr>
<td>69</td>
<td>105</td>
<td>1</td>
<td>Accumulator No. 1 Overflow</td>
</tr>
<tr>
<td>6A-6B</td>
<td>106-107</td>
<td>2</td>
<td>Series Pointer</td>
</tr>
<tr>
<td>6C-6D</td>
<td>108-109</td>
<td>2</td>
<td>Textual Pointer</td>
</tr>
</tbody>
</table>

Table A-1. RM 65 Run-Time BASIC Page Zero Usage (Cont'd)
### Table A-2. RM 65 Run-Time BASIC Page Two Usage

<table>
<thead>
<tr>
<th>Addr (Hex)</th>
<th>Addr (Dec)</th>
<th>No. Bytes</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>200-201</td>
<td>512-513</td>
<td>2</td>
<td>WHEREI Vector*</td>
</tr>
<tr>
<td>202-203</td>
<td>514-515</td>
<td>2</td>
<td>WHEREO Vector*</td>
</tr>
<tr>
<td>204-205</td>
<td>516-517</td>
<td>2</td>
<td>SCRLow Vector*</td>
</tr>
<tr>
<td>206-207</td>
<td>518-519</td>
<td>2</td>
<td>CRLF Vector*</td>
</tr>
<tr>
<td>208-209</td>
<td>520-521</td>
<td>2</td>
<td>OUTCLO Vector*</td>
</tr>
<tr>
<td>20A-20B</td>
<td>522-523</td>
<td>2</td>
<td>INCLO Vector*</td>
</tr>
<tr>
<td>20C-20D</td>
<td>524-525</td>
<td>2</td>
<td>INALL Vector*</td>
</tr>
<tr>
<td>20E-20F</td>
<td>526-527</td>
<td>2</td>
<td>OUTALL Vector*</td>
</tr>
<tr>
<td>210-211</td>
<td>528-529</td>
<td>2</td>
<td>ANYKEY Vector*</td>
</tr>
<tr>
<td>212-213</td>
<td>530-531</td>
<td>2</td>
<td>CLOPTR Vector*</td>
</tr>
<tr>
<td>214-215</td>
<td>532-533</td>
<td>2</td>
<td>OPNPTR Vector*</td>
</tr>
<tr>
<td>216-217</td>
<td>534-535</td>
<td>2</td>
<td>CHKCTC Vector*</td>
</tr>
<tr>
<td>218-21A</td>
<td>536-538</td>
<td>3</td>
<td>JMP USR Instruction (Initialized to FCERR)</td>
</tr>
<tr>
<td>21B-239</td>
<td>539-569</td>
<td>31</td>
<td>Character GET Routine</td>
</tr>
<tr>
<td>23A-23E</td>
<td>570-574</td>
<td>5</td>
<td>RND No. Seed</td>
</tr>
<tr>
<td>23F-241</td>
<td>575-577</td>
<td>3</td>
<td>JMP FILE Instruction (Initialized to FCERR)</td>
</tr>
<tr>
<td>242</td>
<td>578</td>
<td>1</td>
<td>CLRLIN</td>
</tr>
<tr>
<td>243</td>
<td>579</td>
<td>1</td>
<td>OUTFLG</td>
</tr>
<tr>
<td>244-245</td>
<td>580-581</td>
<td>2</td>
<td>Exit to Monitor Vector</td>
</tr>
<tr>
<td>246</td>
<td>582</td>
<td>1</td>
<td>Save Y Register</td>
</tr>
<tr>
<td>247</td>
<td>583</td>
<td>1</td>
<td>Printer Flag</td>
</tr>
<tr>
<td>248-24B</td>
<td>584-587</td>
<td>4</td>
<td>Input Buffer Variables</td>
</tr>
<tr>
<td>24C-2CB</td>
<td>588-715</td>
<td>128</td>
<td>Input Buffer</td>
</tr>
<tr>
<td>2CC-2DC</td>
<td>716-732</td>
<td>17</td>
<td>Floating Point Output Buffer</td>
</tr>
</tbody>
</table>

**NOTE**

*Refer to Section 3 for I/O subroutine requirements*
Table A-3. AIM 65 BASIC Page Zero Usage

<table>
<thead>
<tr>
<th>Addr (Hex)</th>
<th>Addr (Dec)</th>
<th>No. Bytes</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-02</td>
<td>0-2</td>
<td>2</td>
<td>New-line Jump</td>
</tr>
<tr>
<td>03-05</td>
<td>3-5</td>
<td>3</td>
<td>USR Jump</td>
</tr>
<tr>
<td>06</td>
<td>6</td>
<td>1</td>
<td>Search Character</td>
</tr>
<tr>
<td>07</td>
<td>7</td>
<td>1</td>
<td>Scan-Between-Quotes flag</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>1</td>
<td>Input Buffer Pointer, No. of Subscripts</td>
</tr>
<tr>
<td>09</td>
<td>9</td>
<td>1</td>
<td>Default DIM Flag</td>
</tr>
<tr>
<td>0A</td>
<td>10</td>
<td>1</td>
<td>Type: FF=string, 00=numeric</td>
</tr>
<tr>
<td>0B</td>
<td>11</td>
<td>1</td>
<td>Type: 80=integer, 00=floating point</td>
</tr>
<tr>
<td>0C</td>
<td>12</td>
<td>1</td>
<td>DATA Scan Flag; LIST Quote Flag; Memory Flag</td>
</tr>
<tr>
<td>0D</td>
<td>13</td>
<td>1</td>
<td>Subscript Flag; FNx Flag</td>
</tr>
<tr>
<td>0E</td>
<td>14</td>
<td>1</td>
<td>0=Input; $40=GET; $98=READ</td>
</tr>
<tr>
<td>0F</td>
<td>15</td>
<td>1</td>
<td>Comparison Evaluation Flag</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>1</td>
<td>flag: Suppress output if minus</td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>1</td>
<td>I/O for prompt suppress</td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>1</td>
<td>Width</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>1</td>
<td>Input Column Limit</td>
</tr>
<tr>
<td>14-15</td>
<td>20-21</td>
<td>2</td>
<td>Integer Address (for GOTO, etc.)</td>
</tr>
<tr>
<td>16-5D</td>
<td>22-93</td>
<td>72</td>
<td>Input Buffer</td>
</tr>
<tr>
<td>5E</td>
<td>94</td>
<td>1</td>
<td>Temp String Descriptor Stack Pointer</td>
</tr>
<tr>
<td>5F-60</td>
<td>95-96</td>
<td>2</td>
<td>Last Temp String Pointer</td>
</tr>
<tr>
<td>61-69</td>
<td>97-105</td>
<td>9</td>
<td>Stack of Descriptors for Temp Strings</td>
</tr>
<tr>
<td>6A-6B</td>
<td>106-107</td>
<td>2</td>
<td>Pointer for Number Transfer</td>
</tr>
<tr>
<td>6C-6D</td>
<td>108-109</td>
<td>2</td>
<td>Misc. Number Pointer</td>
</tr>
<tr>
<td>6E-72</td>
<td>110-114</td>
<td>5</td>
<td>Product Staging Area for Multiply</td>
</tr>
<tr>
<td>73-74</td>
<td>115-116</td>
<td>2</td>
<td>Pointer: Start of BASIC Memory</td>
</tr>
<tr>
<td>75-76</td>
<td>117-118</td>
<td>2</td>
<td>Pointer: Start of Variables</td>
</tr>
<tr>
<td>77-78</td>
<td>119-120</td>
<td>2</td>
<td>Pointer: Start of Arrays</td>
</tr>
<tr>
<td>79-7A</td>
<td>121-122</td>
<td>2</td>
<td>Pointer: End of Arrays</td>
</tr>
<tr>
<td>7B-7C</td>
<td>123-124</td>
<td>2</td>
<td>Pointer: Bottom of Strings</td>
</tr>
<tr>
<td>7D-7E</td>
<td>125-126</td>
<td>2</td>
<td>Pointer: Utility String</td>
</tr>
</tbody>
</table>

A-5
<table>
<thead>
<tr>
<th>Addr (Hex)</th>
<th>Addr (Dec)</th>
<th>No. Bytes</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>7F-80</td>
<td>127-128</td>
<td>2</td>
<td>Pointer: Limit of BASIC Memory</td>
</tr>
<tr>
<td>81-82</td>
<td>129-130</td>
<td>2</td>
<td>Current BASIC Line No.</td>
</tr>
<tr>
<td>83-84</td>
<td>131-132</td>
<td>2</td>
<td>Previous BASIC line No.</td>
</tr>
<tr>
<td>85-86</td>
<td>133-134</td>
<td>2</td>
<td>Pointer to BASIC statement No.</td>
</tr>
<tr>
<td>87-88</td>
<td>135-136</td>
<td>2</td>
<td>Current DATA Line No.</td>
</tr>
<tr>
<td>89-8A</td>
<td>137-138</td>
<td>2</td>
<td>Pointer to current DATA item</td>
</tr>
<tr>
<td>8B-8C</td>
<td>139-140</td>
<td>2</td>
<td>Input Vector</td>
</tr>
<tr>
<td>8D-8E</td>
<td>141-142</td>
<td>2</td>
<td>Current Variable Name</td>
</tr>
<tr>
<td>8F-90</td>
<td>143-144</td>
<td>2</td>
<td>Current Variable Memory Address</td>
</tr>
<tr>
<td>91-92</td>
<td>145-146</td>
<td>2</td>
<td>Variable Pointer for FOR/NEXT</td>
</tr>
<tr>
<td>93-94</td>
<td>147-148</td>
<td>2</td>
<td>Utility Pointer and Save</td>
</tr>
<tr>
<td>95</td>
<td>149</td>
<td>1</td>
<td>Comparison Symbol Accumulator</td>
</tr>
<tr>
<td>96-97</td>
<td>150-151</td>
<td>2</td>
<td>Misc. Numeric Work Area</td>
</tr>
<tr>
<td>98-9B</td>
<td>152-155</td>
<td>2</td>
<td>Work Area; Garbage Yardstick</td>
</tr>
<tr>
<td>9C-9E</td>
<td>156-158</td>
<td>2</td>
<td>Jump Vector for Functions</td>
</tr>
<tr>
<td>9F-A8</td>
<td>159-168</td>
<td>10</td>
<td>Misc Numeric Work and Storage Area</td>
</tr>
<tr>
<td>A9-AE</td>
<td>169-174</td>
<td>6</td>
<td>Accumulator No. 1 (E,M,M,MS)</td>
</tr>
<tr>
<td>AF</td>
<td>175</td>
<td>1</td>
<td>Series Evaluation Constant Pointer</td>
</tr>
<tr>
<td>B0</td>
<td>176</td>
<td>1</td>
<td>Acc. No. 1 high-order (overflow) Word</td>
</tr>
<tr>
<td>B1-B6</td>
<td>177-182</td>
<td>6</td>
<td>Accumulator No. 2 (E,M,M,M,M,S)</td>
</tr>
<tr>
<td>B7</td>
<td>183</td>
<td>1</td>
<td>Sign of Accumulators Eor'd</td>
</tr>
<tr>
<td>B8</td>
<td>184</td>
<td>1</td>
<td>Acc. No.1 low-order (rounding) Word</td>
</tr>
<tr>
<td>B9-BA</td>
<td>185-186</td>
<td>2</td>
<td>Series Pointer</td>
</tr>
<tr>
<td>BB-BD</td>
<td>187-189</td>
<td>3</td>
<td>Error Jump</td>
</tr>
<tr>
<td>BE</td>
<td>190</td>
<td>1</td>
<td>Printer on/off status</td>
</tr>
<tr>
<td>BF-D6</td>
<td>191-214</td>
<td>24</td>
<td>Subroutine: Get Basic char. C6, C7 = BASIC pointer</td>
</tr>
<tr>
<td>D7-DB</td>
<td>215-220</td>
<td>6</td>
<td>RND No. seed</td>
</tr>
</tbody>
</table>
RM 65 Run-Time BASIC includes the code for the ATAN function whereas it must be provided by the application program when using AIM 65 BASIC (see Appendix H in the AIM 65 BASIC Language Reference Manual).