RM 65 FAMILY

RM 65 RUN-TIME BASIC

USER'S MANUAL

RM 65 FAMILY

Rockwell International

Document No.
29801N10
Order No. 810
June 1982

RM 65 FAMILY

RM 65 RUN-TIME BASIC

RM 65 FAMILY

USER'S MANUAL

Document No.
29801N10
Order No. 810
June 1982

NOTICE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

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 Order No. 202	R6500 Microcomputer Programming Manual
29650N31 Order No. 201	R6500 Microcomputer Hardware Manual
29650N36 Order No. 209	AIM 65 Microcomputer User's Guide
29650N49 Order No. 221	AIM 65 BASIC User's Manual
29650N55 Order No. 233	AIM 65 8K BASIC Reference Card
29650N76 Order No. 269	AIM 65 PROM Programmer & CO-ED User's Manual
29650N01 Order No. 801	RM 65 General Purpose Input/Output and Timer (GPIO) Module User's Manual
29801N02 Order No. 802	RM 65 Floppy Disk Controller (FDC) Module User's Manual
29801N04 Order No. 804	RM 65 Asynchronous Communications Interface Adapter (ACIA) Module User's Manual
29801N05 Order No. 805	RM 65 8K Static RAM Module User's Manual
29801N06 Order No. 806	RM 65 16K PROM/ROM Module User's Manual
29801N08 Order No. 808	RM 65 32K Dynamic RAM Module User's Manual
29801N09 Order No. 809	RM 65 Single Board Computer (SBC) User's Manual
29801N14 Order No. 814	RM 65 CRT Controller (CRTC) Module User's Manual
29801N15 Order No. 815	RM 65 IEEE-488 Bus Interface Module User's Manual
29801N17 Order No. 817	RM 65 Multi-Function Peripheral Interface (MPI) Module User's Manual

SECTION 2

INSTALLATION

The RM 65 Run-Time BASIC (RM65-Ø122) is provided in a Rockwell 8K-byte R2364 ROM (R29Ø6). 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.

AIM 65 BASIC

Address	Contents
FFFF E000	AIM 65 Debug Monitor/ Text Editor ROMs
DFFF D000	User Available
CFFF B000	AIM 65 BASIC ROMs
AFFF A000	AIM 65 I/O
9FFF 200	User Available
1FF 100	R6502 CPU Stack
FF A0	Audio Tape Buffer
9F D7	User Available
D6 0	BASIC Variables

RM 65 Run-Time BASIC

Address	Contents
FFFF D000	User Available
CFFF B000	RM 65 Run-Time BASIC ROM
AFFF 300	User Available
2FF 200	BASIC Variables
1FF 100	R6502 CPU Stack
FF 6E	User Available
6D 0	BASIC Variables

Figure 2-1. RM 65 and AIM 65 BASIC Memory Maps

RM 65 Module Firmware

FFFF A000	User Available
9FFF 9800	CRTC Module ROM and I/O
97FF 9000	CRTC Module RAM
8FFF 8000	FDC Module ROM and I/O
7FFF 7000	IEEE-488 Module ROM and I/O
6FFF 800	User Available
7FF 700	FDC Module Default Input Buffer
6FF 600	FDC Module Default Output Buffer
5FF 561	
560 4A0	FDC module Variables & Constants
49F 400	
3FF 347	CRTC Module Variables & Constants
346 300	IEEE-488 Module Variables & Constants
2FF 200	
1FF 100	Reserved for R6502 CPU Stack
FF F0	Reserved for System Variables
EF EA	IEEE-488 Module Variables
E9 DF	
DE D7	FDC Module Variables
D6 0	

Figure 2-2. RM 65 Module Firmware Memory Map

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:

Purpose	Address Range
Map Socket Z10 (Application PROM)	\$F000-\$FFFF
Map Socket Z8 (Run-Time BASIC ROM)	\$B000-\$CFFF
Map I/O	\$A000-\$AFFF
Map RAM	\$0000-\$0FFF

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

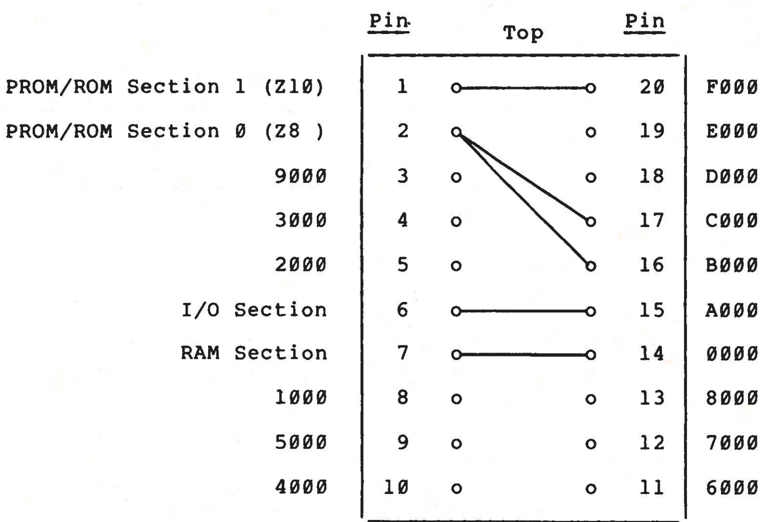

Figure 2-3. Example Base Address Selection Header

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

Table 2-1. RM 65 SBC Module PROM/ROM Selection Jumpers

Section (Socket)	PROM/ROM (see note 1)	Edge Connector Version		Euroconnector Version	
		Jumper	Position	Jumper	Position
Section Ø (Z8)	2K (see note 2)	E2A E2B	OFF ON	E2A E2B	ON OFF
	4K	E2A E2B	OFF ON	E2A E2B	ON OFF
	8K	E2A E2B	ON OFF	E2A E2B	OFF ON
Section 1 (Z1Ø)	2K (see note 3)	E3 E4	B A or B	E3 E4	B A or B
	4K	E3 E4	B A	E3 E4	B A
	8K	E3 E4	A A	E3 E4	A A

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 ($XØØØ-$X7FF) only (pin 18 = All = Ø).

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 ($XØØØ-$X7FF) only (pin 18 = All = Ø).

   E4 = B: The 2K-byte PROM/ROM is mapped into both the lower ($XØØØ-$X7FF) and the upper ($X8ØØ-$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 Z1Ø (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 Ø 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 (R29Ø6) 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 $BØØØ to the upper 4K-bytes of socket Z12:

S1-1	CLOSED
S1-2	CLOSED
S1-3	OPEN
S1-4	CLOSED

(2) Assign \$CØØØ 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 <u>not</u> 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 \$BØØØ-\$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 \$DØØØ 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.

SECTION 3

OPERATION

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), CRTC, 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.

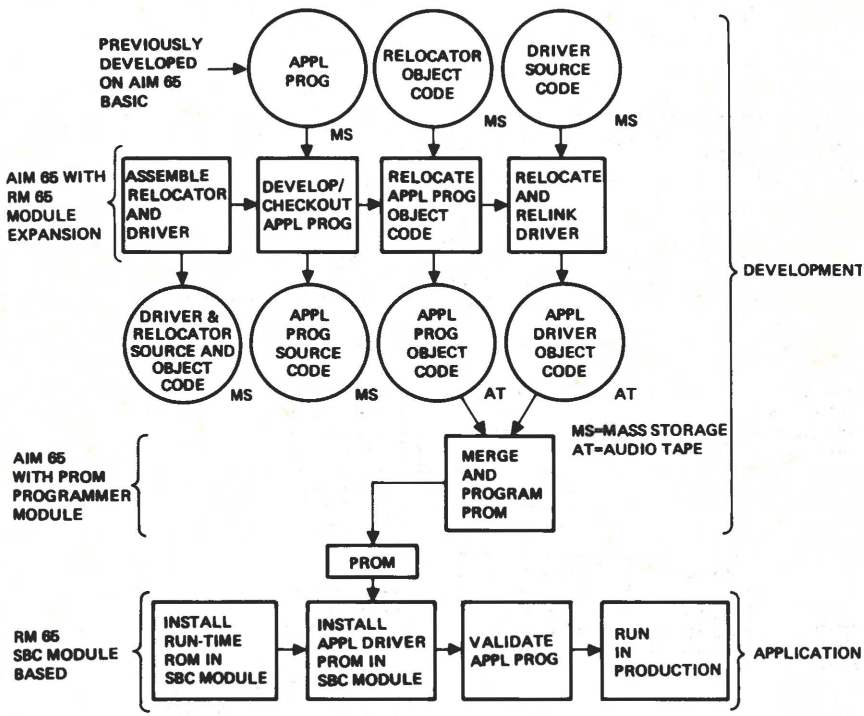

Figure 3-1. Typical Development of a 4K-Byte Application Program

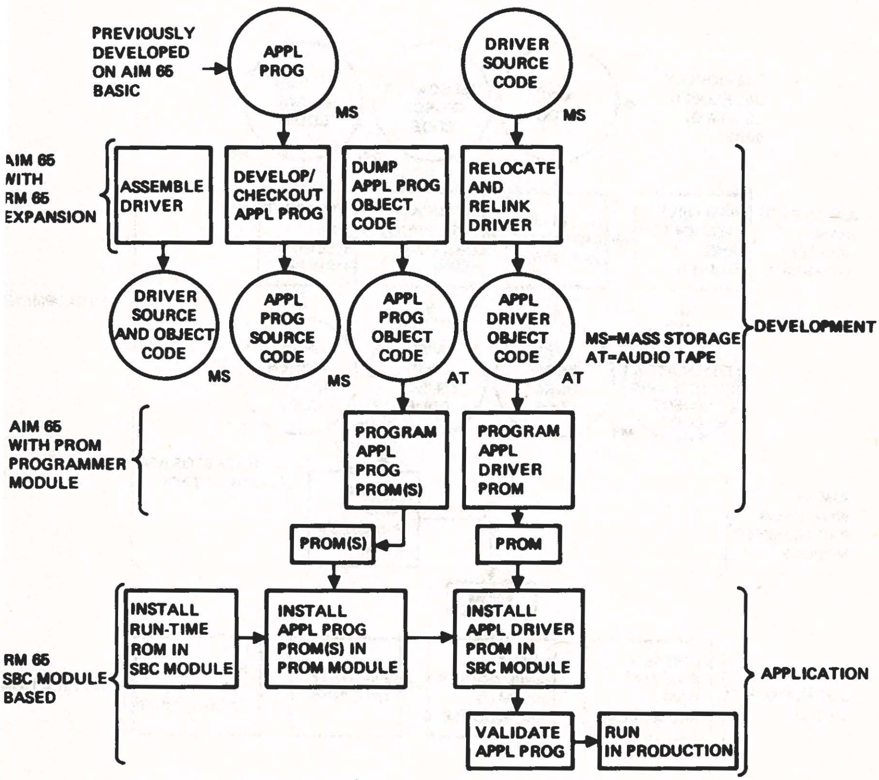

Figure 3-2. Typical Development of a 16K-Byte Application Program

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:

Address	Contents
\$0300 - \$03FF	Driver Object Code
\$0400 - \$1BFF	Application Program and Variables
\$1C00 - \$1FFF	Assembly Symbol Table
\$2000 - \$2FFF	Driver Source Code

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.

c. On the AIM 65 Master Module, install an Assembler (A65-010) ROM into socket Z24 and remove any PROM/ROM drives installed in sockets Z25 and Z26.

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 Microcomputer (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 F1 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 <u>not</u> 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., $Ø3ØØ and $DØØØ, respectively, in this example:

```
<M>ØØØ4 XX XX XX XX
</>ØØØ4 ØØ Ø3 ØØ DØ
```

d. Execute the Relocator program,

```
<*>=ØØØC
<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., $DØØØ-$DFFF.

```
PAGE 0001   BASIC RELOCATOR FOR AIM 65 AND RM 65 R/T BASIC

ADDR OBJECT    SOURCE

                ;
                ; FIRST SET CORRECT VALUES INTO PGMST AND OLDADR ,
                ; THEN EXECUTE THE PROGRAM STARTING AT RELOC .
                ;
                      *=4
                COMIN=$E1A1                ;AIM 65 MONITOR RETURN
                PGMST  *=*+2               ;NEW PROGRAM START ADDRESS
                OLDADR *=*+2               ;OLD PGMST FROM DEVELOPMENT MODE
                NEXT   *=*+2
                OFFSET *=*+2
                ;
                ; RELOCATOR PROGRAM BEGINS HERE ...
                ; FIRST CALCULATE THE OFFSET TO NEW LOCATION
000C A5 04      RELOC  LDA PGMST           ;NEW PROGRAM START ADDR
000E A6 05             LDX PGMST+1
0010 D8                CLD
0011 38                SEC
0012 E5 06             SBC OLDADR
0014 85 0A             STA OFFSET
0016 8A                TXA
0017 E5 07             SBC OLDADR+1
0019 85 0B             STA OFFSET+1
                ; SET UP POINTERS FOR FIRST STATEMENT
001B A2 00             LDX #0
001D A0 01             LDY #1
001F A5 06             LDA OLDADR
0021 85 08             STA NEXT
0023 A5 07             LDA OLDADR+1
0025 85 09             STA NEXT+1
                ; EXECUTE THIS CODE ONCE FOR EACH STATEMENT
0027 A1 08      DONECK LDA (NEXT,X)        ;NEXT LINE ADDR LOW
0029 11 08             ORA (NEXT),Y        ;NEXT LINE ADDR HIGH
002B F0 18             BEQ DONE            ;IF ZEROS ==>
                ; RELOCATE THE CURRENT LINE
002D 18         RELCLN CLC
002E A1 08             LDA (NEXT,X)
0030 48                PHA
0031 65 0A             ADC OFFSET
0033 81 08             STA (NEXT,X)
0035 B1 08             LDA (NEXT),Y        ;LINE NUMBER LOW
0037 48                PHA
0038 65 0B             ADC OFFSET+1
003A 91 08             STA (NEXT),Y
                ; POINT TO THE NEXT PROGRAM LINE
003C 68                PLA
003D 85 09             STA NEXT+1
003F 68                PLA
0040 85 08             STA NEXT
0042 4C 27 00          JMP DONECK          ; ==>
0045 4C A1 E1   DONE   JMP COMIN           ;RETURN TO MONITOR ==>
                        END

 ERRORS=0000
```

Figure 3-3. Relocator Assembly Listing

3.4 PREPARING THE PROM/ROM

The AIM 65 PROM PROGRAMMER & CO-ED module (A65-ØØ6) 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 $\overline{\text{RES}}$ 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).

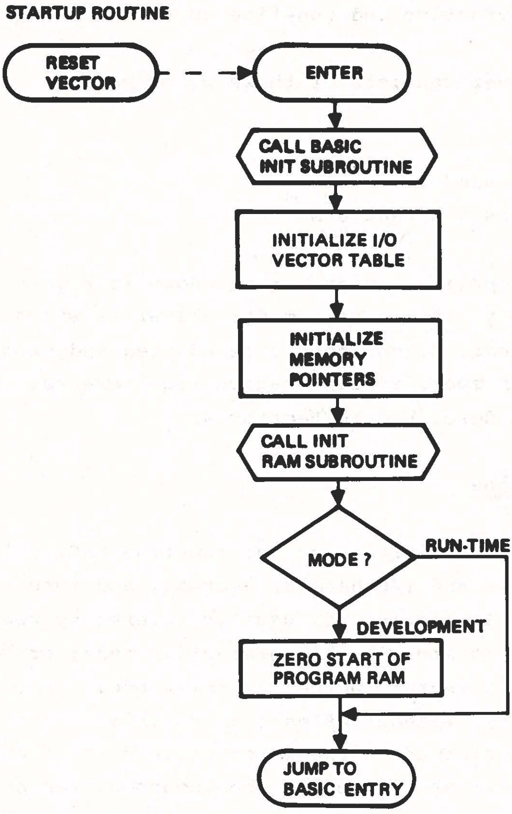

Figure 4-1. Application Driver Flowchart

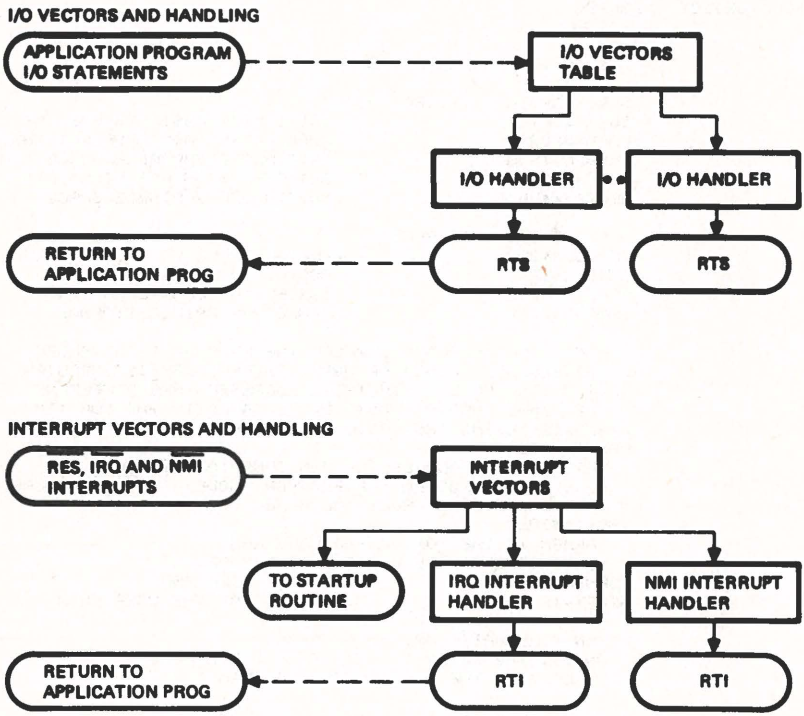

Figure 4-1. Application Driver Flowchart (Cont'd)

```
PAGE 0001    RM 65 RUN-TIME BASIC DRIVERS - STANDARD MODEL

ADDR OBJECT   SOURCE

                 ;
                 ; 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                ;AOD 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.
                 PGMST=$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
                 ; (DEVELOPMENT MODE AND RUN-TIME MODE).
                 TOPMEM=$800                ;TOP OF USER RAM
                 START=$F000                ;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
F000 20 11 CF    COLD   JSR INIT            ;COLD RESET INTO R/T BASIC
                 ; DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM
F003 A2 17              LDX #$17            ;12 VECTORS
F005 BD 3C F0    SETUP  LDA TABLE,X
F008 9D 00 02           STA VECTBL,X
F00B CA                 DEX
F00C 10 F7              BPL SETUP
                 ; CLRLIN CHARACTER IS ISSUED AT THE END OF EVERY LINE
F00E A9 02              LDA #2
F010 8D 42 02           STA CLRLIN
                 ; SET POINTER TO USER PROGRAM PROM
F013 A9 01              LDA #<PGMST
F015 A0 03              LDY #>PGMST
F017 85 22              STA $22
F019 84 23              STY $23
```

Figure 4-2. Model Application Driver Assembly Listing

```
ADDR OBJECT    SOURCE

                ; SET POINTER TO SCRATCH PAD RAM.
                ; DURING PROGRAM DEVELOPMENT, THIS AREA
                ; MUST BE ABOVE THE PROGRAM AREA IN RAM.
F01B A9 03              LDA #<VARST
F01D A0 03              LDY #>VARST
F01F 85 24              STA $24
F021 84 25              STY $25
                ; SET POINTER TO TOP OF MEMORY.
                ; DURING PROGRAM DEVELOPMENT, THIS LIMIT MUST
                ; BE ABOVE THE PROGRAM AND VARIABLE SPACE.
F023 A9 00              LDA #<TOPMEM
F025 A0 08              LDY #>TOPMEM
F027 85 2E              STA $2E
F029 84 2F              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 (NCMWID) IF REQUIRED .
                ; ( AFFECTS PRINT WITH , BUT LINWID OVERRIDES )
                ; DEFAULT POSITION IS THE 30TH CHARACTER ($1E)
                ;
                ; INITIALIZE RAM POINTERS TO A CLEARED STATE
F02B 20 A4 B5           JSR CLEARC
                ;
                ; 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.
F02E A9 00              LDA #0
F030 8D 00 03           STA PGMST-1
F033 8D 01 03           STA PGMST
F036 8D 02 03           STA PGMST+1
F039 4C 99 B0           JMP WARM            ;COME UP IN BASIC
                ; FOR RUN-TIME MODE , USE THIS CODE TO COME UP RUNNING .
                ;    JSR RUNC               ;SETUP R/T BASIC FOR RUN
                ;    JMP NEWSTT             ;AND EXECUTE AWAY ...
                ;    *=*+8                  ;ADJUST THE PROGAM COUNTER
                ;
                ; I/O VECTOR TABLE IS SET UP WITH USER I/O DRIVERS.
                ;
F03C 54 F0      TABLE   .WOR WHERIN         ;OPEN INPUT & SET IN
F03E 56 F0              .WOR WHEROT         ;OPEN OUTPUT & SET OUT
F040 58 F0              .WOR SCRLOW         ;OUTPUT CR TO TERMINAL
F042 5A F0              .WOR CRLF           ;OUTPUT A CR TO THE AOD
F044 5C F0              .WOR OUTCLO         ;CLOSE THE OUTPUT FILE
F046 5E F0              .WOR INCLO          ;CLOSE THE INPUT FILE
F048 60 F0              .WOR INALL          ;INPUT THROUGH THE AID
F04A 62 F0              .WOR OUTALL         ;OUTPUT TO THE AOD
F04C 64 F0              .WOR ANYKEY         ;RETURN Z=1 IF NO KEY
F04E 66 F0              .WOR RESPTR         ;RESTORE PRINTER STATE
F050 68 F0              .WOR FORCEP         ;FORCE PRINT & SAVE STATUS
F052 6A F0              .WOR CHKCTC         ;RETURN KEY DOWN IN A
```

Figure 4-2. Model Application Driver Assembly Listing (Cont'd)

```
ADDR OBJECT    SOURCE

                 ;
                 ; THESE I/O ROUTINES WILL BE DEPENDENT ON THE SYSTEM .
                 ; TYPICALLY EACH NOP WOULD BE REPLACED
                 ; WITH APPLICABLE CODE OR ELIMINATED .
                 ;
F054 EA          WHERIN NOP
F055 60                 RTS
                 ;
F056 EA          WHEROT NOP
F057 60                 RTS
                 ;
F058 EA          SCRLOW NOP
F059 60                 RTS
                 ;
F05A EA          CRLF   NOP
F05B 60                 RTS
                 ;
F05C EA          OUTCLO NOP
F05D 60                 RTS
                 ;
F05E EA          INCLO  NOP
F05F 60                 RTS
                 ;
F060 EA          INALL  NOP
F061 60                 RTS
                 ;
F062 EA          OUTALL NOP
F063 60                 RTS
                 ;
F064 EA          ANYKEY NOP
F065 60                 RTS
                 ;
F066 EA          RESPTR NOP
F067 60                 RTS
                 ;
F068 EA          FORCEP NOP
F069 60                 RTS
                 ;
F06A EA          CHKCTC NOP
F06B 60                 RTS
                 ;
                 ; THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...
                 ; THIS IS THE RUN TIME ADDRESS ( BEGIN ).
F06C 00                 .BYT 0
F06D 00 00       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 R65Ø2 CPU hardware interrupt vectors at $FFFA-$FFFF are included in the AIM 65 Monitor. User alterable vectors (IRQV4 at $A4ØØ, NMIV2 at $A4Ø2, and IRQV2 at $A4Ø4) 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

Vector Location	Vector Name	Used by	Purpose	AIM 65 Subroutine	AIM 65 Addr
$200-$201	WHEREI	LOAD	Determine AID.	WHEREI	$E848
$202-$203	WHEREO	SAVE	Determine AOD.	WHEREO (See Note 1)	$E871
$204-$205	SCRLOW	Command Processing	Output CR & LF to display/printer.	CRCK	$EA24
$206-$207	CRLF	System Output	Output a CR to the AOD.	CRLF	$E9F0
$208-$209	OUTCLO	PRINT PRINT!	Close the AOD.	DU11	$E50A
$20A-$20B	INCLO	INPUT INPUT!	Close to AID.	DU13	$E520
$20C-$20D	INALL	INPUT INPUT! READ	Input a character.	INALL (See Note 1)	$E993
$20E-$20F	OUTALL	PRINT PRINT!	Output a character to the AOD.	OUTALL (See Note 1)	$E9BC
$210-$211	ANYKEY	GET	Check keyboard for key down.	ROONEK (See Note 1)	$ECEF
$212-$213	CLOPTR	INPUT! PRINT!	Close printer input.	(See Note 2)	-
$214-$215	OPNPTR	INPUT! PRINT!	Open printer output.	(See Note 2)	-
$216-$217	CHKCTC	Command input	Input a character from the keyboard.	ROONEK (See Note 1)	$ECEF

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 4-2. I/O Vector Description

Subroutine	Description
WHEREI	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.
WHEREO	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.
SCRLOW	SCRLOW is called to output a CR ($ØD) to the system terminal. It is called only if the value of the OUTFLAG ($Ø243) 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.

Table 4-2. I/O Vector Description (Continued)

Subroutine	Description
CRLF	CRLF is called to output a CR ($ØD) 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.
OUTCLO	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.
INCLO	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.
INALL	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.

Table 4-2. I/O Vector Description (Continued)

Subroutine	Description
OUTALL	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 $Ø2) must be stored in variable CLRLIN ($242). CLRLIN is initially set to $FF. The ASCII value of the output character must be in the A register. All registers must be saved. In an AIM 65 system this vector should point to the AIM 65 Monitor OUTALL subroutine.
ANYKEY	ANYKEY is called by the GET function to sample the system terminal keyboard. The CPU zero flag (Z) is set if a key is not depressed, otherwise the zero flag is reset. No register values must be saved. In an AIM 65 system, this vector should point to a user provided subroutine which sets the ROLLFL flag ($A47F) and calls ROONEK.
CLOPTR	CLOPTR is called by the PRINT! and INPUT! functions. CLOPTR must close the printer output and restore the printer status in PRIFLG ($Ø247) to the value it was before the OPNPTR subroutine was called. The printer status can be saved in SAVFLG ($Ø2DF). The X and Y register values must be saved. In the AIM 65 system, the saved printer status must be stored in PRIFLG ($A411)

Table 4-2. I/O Vector Description (Continued)

Subroutine	Description
OPNPTR	OPNPTR is called only by the PRINT! and INPUT! functions. OPNPTR must save the current printer status in PRIFLG (at $Ø247) into a temporary location, e.g., SAVFLG at $Ø2DF, and open the printer output by storing $8Ø 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 $8Ø stored in PRIFLG (at $A411).
CHKCTC	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.

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.

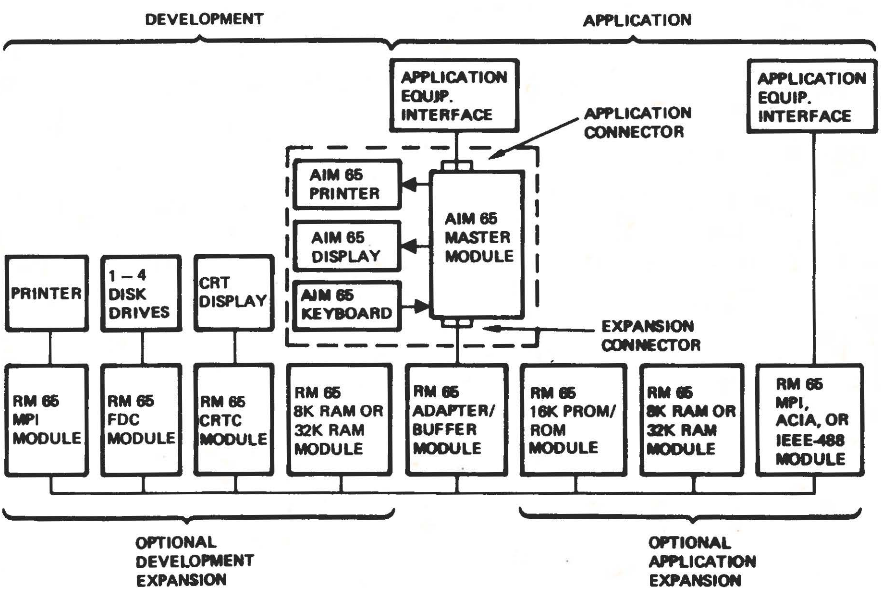

Figure 4-3. Typical AIM 65 Development Configuration

```
ADDR OBJECT   SOURCE

              ;
              ; THIS EXAMPLE IS TO DEVELOP SHORT BASIC PROGRAMS (LESS
              ; THAN 4K BYTE) ON AIM 65 WITHOUT ADDITIONAL RAM .
              ;
              ; AIM 65 MONITOR I/O ROUTINES EQUATES
              WHEREI=$E848
              WHEREO=$E871
              CRCK=$EA24
              CRLF=$E9F0
              DU11=$E50A
              DU13=$E520
              ROONEK=$ECEF
              GETKY=$EC43
              INALL=$E993
              OUTALL=$E9BC
              PSLS=$E7DC
              CUREAD=$FE83
              ;
              ; 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
              CLEARC=$B5A4                ;INITIALIZE VARIABLE SPACE
              ;
              ; R/T BASIC VARIABLES
              VECTBL=$200                 ;VECTOR TABLE OF I/O DRIVERS
              OUTFLG=$243                 ;AOD 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
              ;  <F1> = COLD ENTRY
              ;  <F2> = WARM ENTRY
                        *=$10C
010C 4C 00 03           JMP COLD
010F 4C 99 B0           JMP WARM
```

Figure 4-4. Example AIM 65 Interactive Driver

```
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
                ; DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM
0303 A2 17             LDX ##$17         ;12 VECTORS
0305 BD 3C 03   SETUP  LDA TABLE,X
0308 9D 00 02          STA VECTBL,X
030B CA                DEX
030C 10 F7             BPL SETUP
                ; CLRLIN CHARACTER IS ISSUED AT THE END OF EVERY LINE
030E A9 02             LDA #2
0310 8D 42 02          STA CLRLIN
                ; SET POINTER TO USER PROGRAM PROM
0313 A9 B4             LDA #<PGMST
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 B0          JMP WARM          ;COME UP IN BASIC
                ;
                ; R/T BASIC I/O TABLE STRUCTURE AND I/O ROUTINES
                ;
033C 48 E8      TABLE  .WOR WHEREI       ;WHERIN - OPEN INPUT & SET IN
033E 54 03             .WOR WHROUT       ;WHEROT - OPEN OUTPUT & SET OUT
0340 24 EA             .WOR CRCK         ;SCRLOW - 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 FILE
0346 20 E5             .WOR DU13         ;INCLO - CLOSE THE INPUT FILE
0348 64 03             .WOR INU          ;INALL - INPUT THROUGH THE AID
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 WHEREO          ;OUTPUT DEVICE?
0357 AD 13 A4           LDA OUTDEV
035A C9 0D              CMP #$0D
035C D0 02              BNE STOROT
035E A9 00       STORO0 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 0D              CMP #$0D
0369 F0 06              BEQ DOTERM          ;TERMINAL MUST ALSO ECHO
036B 4C 93 E9           JMP INALL
                 ;
036E 20 DC E7    DEL    JSR PSLS            ;BACK UP DISPLAY
0371 20 83 FE    DOTERM JSR CUREAD
0374 C9 7F              CMP #$7F            ;DELETE
0376 D0 EB              BNE DORTS           ;YES ==>
0378 88                 DEY                 ;BACKUP THE DISPLAY
0379 10 F3              BPL DEL
037B C8                 INY
037C 10 F3              BPL DOTERM          ;ALWAYS ==>
                 ;
037E C9 02       OUTU   CMP #2              ;CLEAR LINE CHARACTER?
0380 F0 E1              BEQ DORTS           ;YES ==> IGNORE IT
0382 4C BC E9           JMP OUTALL
                 ;
0385 20 0A E5    OUTCLS JSR DU11
0388 4C 5E 03           JMP STORO0          ;SET TERMINAL AS OUTPUT ==>
                 ;
038B 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
03A6 8D 7F A4           STA ROLLFL          ;MAKE IT READ KEY AGAIN
03A9 20 EF EC           JSR ROONEK
03AC A9 FF              LDA #$FF
03AE 8D 7F A4           STA ROLLFL          ;MAKE IT READ KEY NEXT TIME
03B1 98                 TYA                 ;SET OR CLEAR Z FLAG
03B2 60                 RTS
                 ;
                 ; THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...
03B3 00                 .BYT 0
03B4 00 00       BEGIN  .DBY 0
                        .END
```

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.

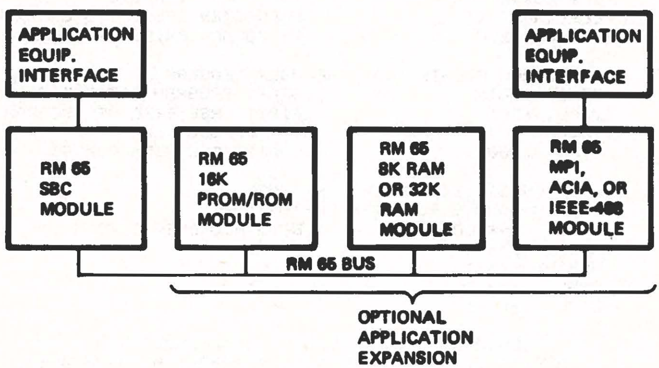

Figure 4-5. Typical RM 65 Run-Time Configuration

```
ADDR OBJECT    SOURCE

                ;
                ; 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
0800            INIT=$CF11                 ;INITIALIZE BASIC PARAMETERS
0800            RUNC=$B593                 ;SET EXECUTION FOR FIRST LINE
0800            NEWSTT=$B6DB               ;SAME AS 'NEW' COMMAND
0800            WARM=$B099                 ;WARM ENTRY POINT FOR R/T BASIC
0800            CLEARC=$B5A4               ;INITIALIZE VARIABLE SPACE
                ;
                ;R/T BASIC VARIABLES
0800            VECTBL=$200                ;VECTOR TABLE OF I/O DRIVERS
0800            OUTFLG=$243                ;ADD FOR R/T BASIC
0800            CLRLIN=$242                ;CHARACTER ISSUED TO CLEAR EACH LINE
0800            SAVFLG=$2DF                ;TEMPORARY PRIFLG STORAGE
                ;
                ; PROGRAM EQUATES FOR THE USER PROGRAM .
0800            PGMST=BEGIN                ;START PROGRAM AFTER I/O DRIVER
0800            VARST=$301                 ;FIRST FREE BYTE OF SBC RAM
0800            TOPMEM=$800                ;TOP OF SBC RAM
0800            START=$D000                ;COMPATIBLE WITH AIM 65
                ;
                ; << RS-232 SERIAL CRT TERMINAL >>
                ; ACIA MODULE REGISTER DEFINITIONS
0800                  *=$7000              ;BASE ADDRESS IS 70XX
7000            ACIA   *=*+1
7001            STATUS *=*+1
7002            CMND   *=*+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
7004                  *=$7110              ;BASE ADDRESS IS 71XX
7110            PORTB  *=*+1
7111            PORTA  *=*+1
7112            DDRB   *=*+1
7113            DDRA   *=*+9
711C            PCR    *=*+1
711D            IFR    *=*+1
711E            MPIDR=$7120                ;MPI DATA DIRECTION REGISTER
                ; MPI MODULE PRINTER CONSTANTS
711E            IDRAB=$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 $FFFA
DFFA 00 D0            .WOR COLD            ;NMI
DFFC 00 D0            .WOR COLD            ;RESET
DFFE 00 D0            .WOR COLD            ;IRQ
```

Figure 4-6. Example RM 65 SBC Run-Time Driver

```
PAGE 0002   RM 65 RUN-TIME BASIC DRIVERS - RUN-TIME MODE

ADDR OBJECT    SOURCE

                  ;
                  ; RUN TIME BASIC DRIVER PROGRAM
                  ; ON ENTRY TO R/T BASIC , USE COLD
                  ;
E000                      *=START           ;ADDRESS OF THE USER PROM
D000 20 11 CF     COLD    JSR INIT          ;COLD RESET INTO R/T BASIC
                  ; DOWNLOAD THE I/O VECTORS FROM TABLE INTO RAM
D003 A2 17                LDX #$17          ;12 VECTORS
D005 BD 41 D0     SETUP   LDA TABLE,X
D008 9D 00 02             STA VECTBL,X
D00B CA                   DEX
D00C 10 F7                BPL SETUP
                  ; INITIALIZE THE ACIA MODULE
D00E A9 0B        OPENAC  LDA #$0B          ;DTR=ON, IRQ=OFF, NO ECHO, NO PARITY
D010 8D 02 70             STA CMND
D013 A9 1E                LDA #$1E          ;8-BITS, 1 STOP BIT, 9600 BAUD
D015 8D 03 70             STA CTRL
D018 AD 00 70             LDA ACIA          ;CLEAR OUT RECEIVER
                  ; CLRLIN CHARACTER IS ISSUED AT THE END OF EVERY LINE
D01B A9 02                LDA #2
D01D 8D 42 02             STA CLRLIN
                  ; SET POINTER TO USER PROGRAM PROM
D020 A9 F0                LDA #<PGMST
D022 A0 D0                LDY #>PGMST
D024 85 22                STA $22
D026 84 23                STY $23
                  ; SET POINTER TO SCRATCH PAD RAM.
D028 A9 01                LDA #<VARST
D02A A0 03                LDY #>VARST
D02C 85 24                STA $24
D02E 84 25                STY $25
                  ; SET POINTER TO TOP OF MEMORY.
D030 A9 00                LDA #<TOPMEM
D032 A0 08                LDY #>TOPMEM
D034 85 2E                STA $2E
D036 84 2F                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 (NCMWID) IF REQUIRED .
                  ; ( AFFECTS PRINT WITH , BUT LINWID OVERRIDES )
                  ; DEFAULT POSITION IS THE 30TH CHARACTER ($1E)
                  ;
                  ; INITIALIZE RAM POINTERS TO A CLEARED STATE
D038 20 A4 B5             JSR CLEARC
                  ; FOR RUN-TIME MODE , USE THIS CODE TO COME UP RUNNING .
D03B 20 93 B5             JSR RUNC          ;SETUP R/T BASIC FOR RUN
D03E 4C DB B6             JMP NEWSTT        ;AND EXECUTE AWAY ...
                  ;
                  ; RUN-TIME BASIC I/O TABLE STRUCTURE
                  ;
D041 EE D0        TABLE   .WOR RETURN       ;OPEN INPUT & SET IN
D043 EE D0                .WOR RETURN       ;OPEN OUTPUT & SET OUT
D045 59 D0                .WOR CRLF         ;OUTPUT CR TO THE TERM
D047 59 D0                .WOR CRLF         ;OUTPUT CR+LF TO THE AOD
```

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

```
ADDR OBJECT   SOURCE

D049 EE D0              .WOR RETURN        ;CLOSE THE OUTPUT FILE
D04B EE D0              .WOR RETURN        ;CLOSE THE INPUT FILE
D04D 63 D0              .WOR INALL         ;INPUT THROUGH THE AID
D04F 8A D0              .WOR OUTALL        ;OUTPUT TO THE AOD
D051 BA D0              .WOR ANYKEY        ;RETURN Z=1 IF NO KEY
D053 C0 D0              .WOR RESPTR        ;RESTORE PRINTER STATE
D055 C6 D0              .WOR FORCEP        ;FORCE PRINT & SAVE STATE
D057 E6 D0              .WOR CHKCTC        ;RETURN KEY DOWN IN A
                ;
                ; I/O ROUTINES MUST BE PROVIDED FOR USER PERIPHERALS
                ;
                ; ISSUE A CR+LF TO THE TERMINAL (NULLS CAN BE ADDED)
D059 A9 0D      CRLF   LDA #$0D
D05B 20 8A D0          JSR OUTALL
D05E A9 0A             LDA #$0A
D060 4C 8A D0          JMP OUTALL          ;USE JSR TO SEND NULLS
                ;
                ; FETCH AN INPUT CHARACTER FROM THE ACIA
D063 20 7D D0   INALL  JSR SINPUT
D066 C9 7F             CMP #$7F
D068 D0 0E             BNE NOTDEL
D06A 88         DELETE DEY
D06B 10 03             BPL OUTBS
D06D C8                INY
D06E 10 F3             BPL INALL           ;ALWAYS ==>
D070 A9 08      OUTBS  LDA #$08
D072 20 8A D0          JSR OUTALL
D075 4C 63 D0          JMP INALL           ;DONE ==>
D078 C9 08      NOTDEL CMP #$08
D07A F0 EE             BEQ DELETE          ;YES ==>
D07C 60                RTS                 ;DONE ==>
                ; RETURN A CHARACTER FROM THE TERMINAL
D07D AD 01 70   SINPUT LDA STATUS
D080 29 08             AND #$08
D082 F0 F9             BEQ SINPUT
D084 AD 00 70   SINP1  LDA ACIA
D087 29 7F             AND #$7F            ;IGNORE MSB
D089 60                RTS
                ;
                ; SEND AN OUTPUT CHARACTER TO THE AOD
D08A 48         OUTALL PHA
D08B AD 43 02          LDA OUTFLG
D08E D0 0C             BNE PRINTR
                ;ON TERMINAL,WAIT TILL ACIA IS READY
D090 AD 01 70   TERMNL LDA STATUS
D093 29 10             AND #$10            ;TRANSMITTER EMPTY?
D095 F0 F9             BEQ TERMNL          ;NO ==>
D097 68                PLA
D098 8D 00 70          STA ACIA
D09B 60                RTS
                ; ON PRINT, PASS THROUGH ONLY PRINTABLE CHARACTERS
D09C 20 A7 D0   PRINTR JSR WAIT            ;WAIT UNTIL PRINTER IS READY
D09F 68                PLA
D0A0 8D 11 71   CHROUT STA PORTA
D0A3 20 AF D0          JSR STROBE          ;SEND OUT THE CHARACTER
D0A6 60                RTS
```

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

```
ADDR OBJECT    SOURCE

                    ; WAIT UNTIL AN ACKNOWLEDGE IS RECEIVED FROM THE PRINTER
D0A7 AD 1D 71   WAIT    LDA IFR              ;ACKNOWLEDGE IS ON CA2
D0AA 4A                 LSR A                ;MOVE CA2 INTO CARRY FLAG
D0AB 4A                 LSR A
D0AC 90 F9              BCC WAIT             ;NOT READY? ==>
D0AE 60                 RTS
                    ; HANDSHAKE OFF THE CHARACTER
D0AF A9 00      STROBE  LDA #0               ;FORCE STROBE LOW
D0B1 8D 10 71           STA PORTB
D0B4 A9 01              LDA #1               ;FORCE STROBE HIGH
D0B6 8D 10 71           STA PORTB
D0B9 60                 RTS
                    ;
                    ; CHECK FOR ANY KEY DEPRESSION
D0BA AD 01 70   ANYKEY  LDA STATUS
D0BD 29 08              AND #$08             ;SET Z=1 FOR KEY DOWN
D0BF 60                 RTS
                    ;
                    ; RESTORE THE TERMINAL AS OUTPUT
D0C0 A9 00      RESPTR  LDA #00
D0C2 8D 43 02           STA OUTFLG
D0C5 60                 RTS
                    ;
                    ; FORCE THE PRINTER AS OUTPUT
D0C6 A9 50      FORCEP  LDA #'P'
D0C8 8D 43 02           STA OUTFLG
                    ; ON OPEN , SET UP THE VIA AND THE DATA PORT BUFFERS
D0CB A9 0D      PROPEN  LDA #$0D             ;ISSUE A CR AT FIRST
D0CD 8D 11 71           STA PORTA
D0D0 A9 C0              LDA #IMPIDR
D0D2 8D 20 71           STA MPIDR
D0D5 A9 FF              LDA #IDRAB           ;PORTS A AND B ARE OUTPUT
D0D7 8D 13 71           STA DDRA
D0DA 8D 12 71           STA DDRB
D0DD A9 04              LDA #IPCR
D0DF 8D 1C 71           STA PCR
D0E2 20 AF D0           JSR STROBE
D0E5 60                 RTS
                    ;
                    ; CHECK TO SEE IF ANYTHING HAS BEEN RECEIVED
D0E6 20 BA D0   CHKCTC  JSR ANYKEY
D0E9 F0 03              BEQ RETURN           ;NO KEY ==>
D0EB 20 7D D0           JSR SINPUT
D0EE 60         RETURN  RTS
                    ;
                    ; THE ACTUAL BASIC PROGRAM WILL BEGIN HERE ...
D0EF 00                 .BYT 0
D0F0 00 00      BEGIN   .DBY 0
D0F2                    .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

Addr (Hex)	Addr (Dec)	No. Bytes	Purpose
Ø	Ø	1	Search Character
Ø1	1	1	Scan-Between-Quotes Flag
Ø2	2	1	Input Buffer Pointer
Ø3	3	1	Default DIM Flag
Ø4	4	1	TYPE: FF=string, ØØ=numeric
Ø5	5	1	TYPE: 8Ø=integer, ØØ=floating pt.
Ø6	6	1	Data Scan Flag; List Quote Flag; Memory Flag
Ø7	7	1	Subscript Flag; FNx flag
Ø8	8	1	Ø=Input; $4Ø=GET; $98=READ
Ø9	9	1	Comparison Evaluation Flag
ØA	1Ø	1	Flag; Suppress Output if Minus
ØB	11	1	Position of Terminal Carriage
ØC	12	1	Width (length of line)
ØD	13	1	Position Beyond Output Fields
ØE	14	1	Temp String Desc. Stack Pointer
ØF	15	1	Last Temp String Pointer
1Ø-18	16-24	9	Stack of Temp String Descriptors
19-1A	25-26	2	Pointer for Number Transfer
1B-1C	27-28	2	Misc. Number Pointer
1D-21	29-33	5	Product Staging Area for Multiply
22-23	34-35	2	Pointer: Start of BASIC Memory
24-25	36-37	2	Pointer: Start of Variables
26-27	38-39	2	Pointer: Start of Arrays
28-29	4Ø-41	2	Pointer: End of Arrays
2A-2B	42-43	2	Pointer: Bottom of Strings
2C-2D	44-45	2	Pointer: Utility String
2E-2F	46-47	2	Pointer: Limit of BASIC
3Ø-31	48-49	2	Current BASIC Line No.
32-33	5Ø-51	2	Previous BASIC Line No.
34-35	52-53	2	Integer Address
36-37	54-55	2	Pointer to Basic Statement
38-39	56-57	2	Current DATA Line No.
3A-3B	58-59	2	Pointer to Current Data
3C-3D	6Ø-61	2	Input Vector

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

Addr (Hex)	Addr (Dec)	No. Bytes	Purpose
3E-3F	62-63	2	Current Variable Name
40-41	64-65	2	Current Variable Memory Address
42-43	66-67	2	Variable Pointer Memory Address
44-45	68-69	2	Utility Pointer and Save
46	70	1	Comparison Symbol Accumulator
47-4C	71-76	6	Misc. Numeric Work Area
4D-4F	77-79	2	Jump Vector for Functions
50-59	80-89	10	Misc. Numeric Work and Storage Area
5A-5F	90-95	6	Accumulator No. #1 (E,M,M,M,M,S)
60	96	1	Degree of Polynomial to Evaluate
61	97	1	Bits to Shift Right
62-67	98-103	5	Accumulator No. 2 (E,M,M,M,M,S)
68	104	1	Sign of Accumulators EOR'd.
69	105	1	Accumulator No. 1 Overflow
6A-6B	106-107	2	Series Pointer
6C-6D	108-109	2	Textual Pointer

Table A-2. RM 65 Run-Time BASIC Page Two Usage

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

NOTE

*Refer to Section 3 for I/O subroutine requirements

Table A-3. AIM 65 BASIC Page Zero Usage

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

Table A-3. AIM 65 BASIC Page Zero Usage (Cont'd)

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

APPENDIX B

RM 65 AND AIM 65 BASIC DIFFERENCES

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).

NOTES

NOTES

ELECTRONIC DEVICES DIVISION REGIONAL ROCKWELL SALES OFFICES

HOME OFFICE
Electronic Devices Division
Rockwell International
4311 Jamboree Road
Newport Beach, California 92660
Tel 800-854-8099, 800-854-8090
In California 800-422-4230
TWX 910 591-1698

UNITED STATES
Electronic Devices Division
Rockwell International
1842 Reynolds
Irvine, California 92714
(714) 632-3710
TWX 910 505-2518

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. 35
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

4/82

Rockwell International

...where science gets down to business