RMB-167I
USER'S GUIDE

Version 1.0

JULY 1996

RIGEL Corporation
P.O. Box 90040
Gainesville, FL 32607
(352) 373-4629
http://www.rigelcorp.com

The abbreviation PC used throughout this guide refers to the IBM Personal Computer or its compatibles. IBM PC is a trademark of International Business Machines, Inc. MS Windows is a trademark of Microsoft, Inc.

WARRANTY

RIGEL CORPORATION- CUSTOMER AGREEMENT

1. **Return Policy**. **This return policy applies only if you purchased the RMB-167i industrial board directly from Rigel Corporation.**
    If you are not satisfied with the items purchased, prior to usage, you may return them to Rigel Corporation within thirty (30) days of your receipt of same and receive a full refund from Rigel Corporation. You will be responsible for shipping costs. Please call (352) 373-4629 prior to shipping. A refund will not be given if the READS package has been opened.

2. **Limited Warranty**. Rigel Corporation warrants, for a period of sixty (60) days from your receipt, that READS disk(s), hardware assembled boards and hardware unassembled components shall be free of substantial errors or defects in material and workmanship which will materially interfere with the proper operation of the items purchased. If you believe such an error or defect exists, please call Rigel Corporation at (352) 373-4629 to see whether such error or defect may be corrected, prior to returning items to Rigel Corporation. Rigel Corporation will repair or replace, at its sole discretion, any defective items, at no cost to you, and the foregoing shall constitute your sole and exclusive remedy in the event of any defects in material or workmanship.

    THE LIMITED WARRANTIES SET FORTH HEREIN ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

    YOU ASSUME ALL RISKS AND LIABILITY FROM OPERATION OF ITEMS PURCHASED AND RIGEL CORPORATION SHALL IN NO EVENT BE LIABLE FOR DAMAGES CAUSED BY USE OR PERFORMANCE, FOR LOSS PROFITS, PERSONAL INJURY OR FOR ANY OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. RIGEL CORPORATION'S LIABILITY SHALL NOT EXCEED THE COST OF REPAIR OR REPLACEMENT OF DEFECTIVE ITEMS.

    IF THE FOREGOING LIMITATIONS ON LIABILITY ARE UNACCEPTABLE TO YOU, YOU SHOULD RETURN ALL ITEMS PURCHASED TO YOUR SUPPLIER.

3. **READS 166** (referred to as simply READS) License. The READS being purchased is hereby licensed to you on a non-exclusive basis for use in only one computer system and shall remain the property of Rigel Corporation for purposes of utilization and resale. You acknowledge you may not duplicate the READS for use in additional computers, nor may you modify, disassemble, translate, sub-license, rent or transfer electronically READS from one computer to another, or make it available through a timesharing service or network of computers. Rigel Corporation maintains all proprietary rights in and to READS for purposes of sale and resale or license and re-license.

    BY BREAKING THE SEAL AND OTHERWISE OPENING THE READS PACKAGE, YOU INDICATE YOUR ACCEPTANCE OF THIS LICENSE AGREEMENT, AS WELL AS ALL OTHER PROVISIONS CONTAINED HEREIN.

4. **Board Kit**. If you are purchasing a board kit, you are assumed to have the skill and knowledge necessary to properly assemble same. Please inspect all components and review accompanying instructions. If instructions are unclear, please return the kit unassembled for a full refund or, if you prefer, Rigel Corporation will assemble the kit for a fee of $30.00. You shall be responsible for shipping costs. The foregoing shall apply only where the kit is unassembled. In the event the kit is partially assembled, a refund will not be available, however, Rigel Corporation can, upon request, complete assembly for a fee based on an hourly rate of $50.00. Although Rigel Corporation will replace any defective parts, it shall not be responsible for malfunctions due to errors in assembly. If you encounter problems with assembly, please call Rigel Corporation at (352) 373-4629 for advice and instruction. In the event a problem cannot be resolved by telephone, Rigel Corporation will perform repair work, upon request, at the foregoing rate of $50.00 per hour.

5. **Governing Law**. This agreement and all rights of the respective parties shall be governed by the laws of the State of Florida.

Table Of Contents

1.   OVERVIEW

The RMB-167i industrial board contains the Siemens SAB C167 16-bit high-performance microcontroller in the metric plastic quad-flat pack package. The board accepts either the SAB C167-LM, the SAB C167SR-LM, or the SAB C167CR-LM. The only change needed on the board for the different processors is the oscillator. The oscillator for the 167-LM is 40 MHz and for the 167SR-LM and 167CR-LM is 5 MHz. Please see the Siemens' documentation for details on system clock speed. The microcontroller is run with a 16-bit nonmultiplexed data bus and an 18-bit nonmultiplexed address bus. The board may be configured several different ways depending on the type of reset and ROM memory options used. The default configuration is the 64K RAM and no ROM mode. In this mode, the monitor program or user program is downloaded to RAM using the SAB C167 bootstrap feature. A set of option headers, decoded by PAL devices make the RMB-167i a flexible hardware platform.

The RMB-167i is compatible with the RMB-166, RMB-166I, RMB-167, and the RMB-165i. The board size, the location and function of all headers are kept the same. The RMB-167i has these features for improved industrial performance.

1. **Industrial strength shielding.** The printed circuit board is a six-layered board with separate VCC and Ground planes for improved shielding designed to operate in noisy industrial applications.
2. **EERPOM/Battery-Backed RAM capability.** Sockets U7 and U8 accept 32K EPROMs (27C256), 32K EEPROMs (28C266), or 32K RAMs (62C256).
3. **Higher memory map resolution.** The PALCE22V10-type device used in U6 has access to A13 to A17 in determining the memory map. The memory map has a resolution of 8 Kilobytes. For example, the first segment may consist of 8K of ROM and 56K of RAM.

1.1   Hardware

- SAB C167 high-performance microcontroller
    - Bootstrap loading feature
    - One serial port
    - One high speed synchronous serial channel
    - Five 16 bit timers
    - Watchdog timer
- Serial port uses a MAX232 driver, terminates at DB-9 connector and a 3-post connector
- 111 bits of general-purpose inputs/outputs
- Accommodates 64K or 256K of SRAM (64K installed)
- Accommodates 64K of EPROM/EEPROM/RAM (not installed)
- Push buttons for RESET# and NMI# (non-maskable interrupt)
- GAL decoded memory map for maximum flexibility
- GAL's can be reprogrammed by user or by RIGEL Corporation
- Microcontroller surface mounted to the board
- Machine screw sockets under all other IC's
- Power on LED
- Power consumption is less than 175mA running at 40MHz
- Flexible and embeddable 4" x 6 1/2" 6 layer industrial board with separate VCC and Ground planes
- Mounting holes in corners

1.2   Software: READS166 Evaluation Package

- Runs in the MS-Windows 3.1 environment.
- Supports the bootstrap loader feature. READS166 downloads a minimal monitor during bootstrapping.
- Source code and description of the bootstrap program.
- Allows downloading and running applications programs
- Demonstration programs

RMON167 - monitor program

- Downloaded after bootstrapping (or may be placed into ROM)
- Supports basic memory and port functions.
- Downloads and runs applications programs.
- Complete source code for user modifications or upgrades.

1.3   Parts List

Your RMB-167i package includes the following:

Hardware

1. RMB-167i board with a 64K of static RAM.

Software

1. RMON167 monitor program with source code.
2. Evaluation version of READS166 for Microsoft Windows.

Documentation

1. User's Guide with circuit diagrams
2. Complete chip documentation from Siemens with application notes
3. Bootstrap file source code.
4. Sample programs.

A regulated 5 volt 500mA (+/- 5%) power source is to be supplied by the user.

2. QUICK START TUTORIAL

2.1 System Requirements for READS166

READS166 runs in MS Windows and is installed in a one-step operation by running the INSTALL.EXE from DOS or the Windows File Manager.

READS166 is designed to work with an IBM PC or compatible 386 or better, running MS-Windows 3.1 or later. Although Windows may be run without a mouse it is much more practical to use one. A mouse is recommended when using READS166.

READS166 uses either COM1 or COM2 to talk to the RMB-167i. These ports are driven using the default interrupt request lines IRQ4 and IRQ3, respectively. Make sure that you do not have other peripherals such as a modem or a serial mouse competing for the same interrupts. They may cause a conflict when running READS166. If you are using a COM port which was used previously for a modem or a serial mouse, the software drivers for these devices should be removed as well. If you remove the mouse or modem without removing the software driver the computer will not recognize that a different device is now using the serial port. The computer will continue to expect mouse or modem commands to and from the serial port. Since READS uses different commands the board will not respond.

2.2 Software Installation

Place the READS166 diskette in your floppy disk drive and run INSTALL. INSTALL may be run from a DOS prompt or from the Windows File Manager. For example if the distribution disk is in drive A:, and you are installing from DOS type:

> ***A:install***

Then enter the drive and directory information as requested.

To install from Windows choose **Run** from the Program Manager's File menu. Type

> ***A:install***

in the Command Line text box. Click on **OK** or press **ENTER** to begin installation.
Then enter the drive and directory information as requested.

2.3 Start up

1. Connect the RMB-167i to a well-regulated 5 Volt supply (+/-5%).
2. Connect the RMB-167i to the PC host via a modem cable.
3. Check to make sure jumpers are the correct position. (JP3 should have a jumper in location 4 and 6.) This is the default configuration for the RMB-167i and the board will be populated this way from the manufacturer.
4. Run the READS166 host driver from MS Windows. You may use the Windows File Manager to launch READS166. You may also start READS166 by double clicking on the READS166 icon.

2.4 Configuring READS and Initiating Host-to-Board Communications

1.   Select the processor type using the **Processor | SAB C167 xR** menu command.
2.   Select the communication port parameters using the **TTY | Setup** menu command. You will need to select the COM port you are using, then the default configuration for the rest of the parameters are as follows: 8 data bits, 9600 baud rate, 1 stop bit and none for the parity bits.
3.   Open the TTY window using the **TTY | Connect** menu command.

2.5 Bootstrapping
1.   Press the reset button on the board and wait 3 seconds.
2.   Select item **TTY | Bootstrap | Bootstrap and download monitor**. The board will now bootstrap.

You can observe as the bytes of the monitor are downloaded to the board. The green LED is turned on during bootstrapping, after the EINIT instruction, but before the monitor is downloaded. When bootstrapping is completed, the READS166 monitor prompt appears in the TTY window.

You may also use the BSO/Tasking Crossview Monitor/Debugger to bootstrap the RMB-167i. You may need to press either the RESET or NMI# button on the RMB-167i.

2.6 Verify that the Monitor is Loaded
Make sure the TTY window is active, clicking the mouse inside the TTY window to activate it if necessary. Then type the letter *"H"* (case insensitive) to verify that the monitor program is responding. The 'H' command displays the available single-letter commands the monitor will recognize.

2.7 Downloading Programs
The example program DEMO05C.ASM repeatedly sends a message to the host in an interrupt driven fashion. The program is already assembled into a HEX file and given in the distribution disk.

Select the **TTY | Download** menu item. Choose the demo program DEMO05C.HEX from the list of files. Press **OK** to download the file. You may view the source code by opening the file as a document using the **File | Open** menu command.

2.8 Running a Program
1.   The program DEMO05C starts at address 4000h as specified by the ORG pseudo operation in its source code. In order to run DEMO05C, select the **TTY | Run** menu item and type *4000* at the address field. Press **OK** to run the program. Alternately after the program is downloaded when the monitor prompt appears you may type *G4000* to run the program.
2.   Some demo programs run in an endless loop. Press the NMI button on the board to terminate the program and return to the monitor. Alternatively, you may press the RESET button. In this case, however, the bootstrapping operation must be repeated, and the monitor program reloaded.

2.9 Using Help
Most of the file operations can be performed by clicking on the icons of the speedbar (toolbar) placed just below the main menu. You can get more information from the

READS166 Help. Simply select **Help | Contents** from the main menu. Once in the Help System, select topic **Toolbar** for information on the icons of the speedbar.

3.   OPERATING NOTES

The RMB-167i needs two connections: to a 5 volt (+/- 5%) well regulated power supply and to the serial port of a host via a modem cable.

3.1   Power

Power is brought to the RMB-167i board by a two-position screw-type terminal block or a DC jack. A well regulated 5V DC source is required. The (+) and (-) terminals are marked on the board. The center connector of the DC jack should be the (-) terminal and the outside the (+) terminal. Note that a diode is placed across the input in reverse. Thus if the power is applied to the RMB-167i board in reverse polarity, the diode will short the power supply attempting to prevent damage to the board. Populated with 64K of CMOS RAM, the RMB-167i draws less than 175 mA.

3.2   Serial Port 0

Serial port 0 is accessed through the RS-232 level converter. The microcontroller supports the transmit and receive signals. A minimal serial port may be constructed with just 3 lines: transmit, receive, and ground, disregarding all hardware handshake signals. Port P1 (HOST) of the RMB-167i is a DB-9 female connector used to connect the board to an IBM compatible PC. A straight-through modem cable may be used. That is a cable connecting pin 2 of the RMB-167i to pin 2 of the host, and similarly pin 3 to pin 3, and pin 5 to pin 5. This cable and a DB9-DB25 adapter is supplied when the board is purchased directly from Rigel Corporation. JP6 is a 3-pin header, which carries the same signals as P1.

3.3   Serial Port 1

Serial port 1 is used as a synchronous serial channel by the SAB-C167i. It has 3 TTL-level signals, MRST (Master Receive Slave Transmit), MTRS (Master Transmit Slave Receive), and SCLK (Serial Clock).

3.4   Push Buttons

3.4.1 Reset (S1)

The reset button is connected to the reset pin of the processor and resets the board. Before bootstrapping press the reset button and wait 3 seconds to allow the processor to initialize. The board is then able to carry out the bootstrap instructions.

3.4.2 NMI (S2)

The NMI button (non-maskable interrupt) is connected to the NMI pin of the processor. When pressed it generates a non-maskable interrupt. RMON places a jump instruction at the NMI vector (address 8). Pressing the NMI, while the RMON is present, invokes the monitor program. This works as long as the monitor program in RAM is not altered. Pressing the NMI button is usually sufficient to interrupt user's program which are downloaded and run under RMON. Application programs placed in ROM may use a similar scheme to initialize the system when the NMI button is pressed.

3.5 LEDS

The RMB-167i has three LEDs. The red LED, when lit, shows power is connected to the board.

The green LED indicator marked RO (Reset Out) is connected to a GAL device. The LED is turned on after system initialization is completed. More specifically, the LED is turned on when the RSTIN# is high and RSTOUT# makes a 0-to-1 transition, which normally follows

an EINIT instruction. <u>The LED RO will be off and remain off until the bootstrap loader successfully completes loading the bootstrap file into RAM.</u>

The yellow LED is an auxiliary LED, whose state is determined by the GAL equations. For example the user may program the yellow LED to indicate the presence of a program in EPROM. In the default configuration the yellow LED is opposite of the green LED.

3.6   Slide Switch (S5)

The slide switch is inactive on the board with the factory GALs installed. The slide switch is intended to be used in an application specific manner. The user may burn GALs to implement the switch. The slide switch is not populated unless the customer requests it.

4.   JUMPER CONFIGURATIONS

4.1   JP3 (Reset Options)

JP3 is a 16 x 2 header. Each pair of posts corresponds to a bit of Port 0. The silkscreen is labeled from 0 to 15 indicating bit or jumper positions. Some of the SAB C167 operating modes are determined during reset by the state of Port 0 bits. (Refer to the Siemens data book for further information.) Inserting a jumper in JP3 connects the corresponding bit of Port 0 to ground via a 15K resistor. This, in turn, sets the operating mode. For example, the Bootstrap mode is invoked when bit 4 of Port 0 is held low at reset. The RMB-167i uses two of these bits for configuring the board.

4.1.1 P4

Bit 4 of Port 0 invokes the bootstrap mode when held low at reset. Inserting a jumper at position 4 (P4) on JP3 grounds bit 4, causing the board to bootstrap at startup or reset. The RMB-167i's default configuration is with this jumper P4, installed. With P4 removed and a user's program with starting address at zero in ROM, the board will run the user's program upon reset.

4.1.2 P6

Bit 6 of Port 0 determines the bus mode of the SAB C167. P6 is inserted to ground bit 6 in the default configuration of the RMB-167i. P6 should remain in place at all times to use the onboard memory of the RMB-167i.

4.2   CFG0 / CFG1 / CFG2

CFG0 / CFG1 / CFG2 configure the memory block size and location. The default memory map of 64K RAM and no ROM is with these jumpers removed. Please see the alternate memory map section (5.4) for more information on these jumpers.

4.3   RAM and ROM Jumpers

These are the two jumpers located next to U8. These jumpers are set according to the type of devices used in sockets U7 and U8. Note that RAMs and EEPROMs have the same pinout. If U7 and U8 hold EPROMs, insert two jumpers, one in each of jumpers, so that the center posts are connected to the end posts marked ROM. If U7 and U8 hold RAM or EEPROM devices, insert the jumpers so that the center posts are connected to the end posts marked RAM. The RMB-167i factory setting is with these jumpers removed.

4.4   Serial Port Jumpers

The SAB C167 has two serial ports. Serial port 0 is accessed through the RS-232 level converter. Serial port 1 is used as a synchronous serial channel by the SAB-C167i.

4.4.1 JP6 (SO)

JP6 is the secondary access for serial port 0. JP6 is a 3-pin header, which carries the same signals as P1.   JP6 is convenient if the board is to be used as an embedded controller. JP6 is also denoted by SO and its 3 lines by G (Ground), T (Transmit), and R (Receive), on the RMB-167i silk-screen. JP6 is intended for embedded uses of the RMB167i when P1 is not populated.

4.4.2 JP7 (SSC)

Header JP7 is the access for serial port 1. Header JP7 carries the 3 TTL-level signals, MRST (Master Receive Slave Transmit), MTRS (Master Transmit Slave Receive), and SCLK (Serial Clock) as well as provides VCC and GND for reference. Header JP7 is also denoted by SSC (Serial Synchronous Channel) on the silk-screen. The individual pins of

the header are denoted by VCC, R, T, C, and GND, which correspond to VCC, MRST, MTSR, SCLK, and GND, respectively.

4.5 JP11 and JP12

These jumpers are not used at present.

5.   MEMORY BLOCK OPTIONS

There are two blocks of memory available on the RMB-167i board. There is a RAM block which may hold 64K or 256K of memory, and there is a ROM block which may hold up to 64K of memory.

5.1   RAM Memory Options

The RAM block is designed to take static RAMs, either 32K 62C256-type, or 128K 681000-type static RAM chips. Alternately 62C256-type battery-backed RAMs may be used in the RAM block. Two chips are needed, one for EVEN and the other for ODD addresses. These chips are placed in 32-pin sockets marked U5 and U6. Place 28-pin 32K RAM devices closer to the 2 X 25 header, away from the processor. Note that the 28-pin RAM devices line up with the ROM sockets.

The SAB C167 may be programmed to insert wait cycles during external memory access. However, in order to run the SAB C167 at its full potential, the RAMs should be rated at 70 nano seconds or faster.

5.2   ROM Memory Options

The ROM block of memory accepts a variety of non-volatile devices. 27C256-type 32K EPROMs, 62C256-type battery-backed RAMs or 28C256-type EEPROMs may reside in this location. Alternately 62C256-type static RAMs may be used. Two chips are needed, one for EVEN and the other for ODD addresses. These chips are placed in 28-pin sockets marked U7 and U8.

The SAB C167 may be programmed to insert wait cycles during external memory access. However, in order to run the SAB C165 at its full potential, the EPROMs should be rated at 85 nano seconds or faster and the battery-backed RAMs should be 70 nano seconds or faster.

5.3   Default Memory Setting

The default memory map assumes 64K of RAM with no EPROM. In this configuration, the board is bootstrapped and run. JP3 should have a jumper in location P4 and P6. This is the default configuration for the RMB-167i and the board will be populated this way from the manufacturer. Programs are downloaded into RAM and then run.

5.4   Alternate Memory Maps

The factory installed GAL supports alternative memory maps with the jumpers CFG0 / CFG1 / CFG2.   CFG0 and CFG1 select RAM and ROM memory blocks, and CFG2 determines memory block sizes. The default configuration of the RMB-167i is with no jumpers installed. This selects 64K of RAM. The following table shows which jumpers to install for the alternate memory maps supported by the factory programmed Gals.

Memory Map with CGF2 jumper removed

Jumper location	Low 48K	High 16K
NONE	RAM	RAM
CFG0	ROM	RAM
CFG1	RAM	ROM
CFG0 and CFG1	ROM	ROM

Memory Map with CGF2 jumper inserted

Jumper location	Low 32K	High 32K
NONE	RAM	RAM
CFG0	ROM	RAM
CFG1	RAM	ROM
CFG0 and CFG1	ROM	ROM

The RAM and ROM blocks of memory can be mapped in a variety of ways with a minimal memory block size of 8K. The EPROMs/EEPROMs may be configured to occupy low memory, high memory, or start at low memory and then relocate to high memory upon initialization by appropriate jumper selections. The GALs may be reprogram by the user, or RIGEL Corporation, to support the different memory maps. Please call Rigel Corporation if more information about user designed memory maps is needed.

6.   THE ROM MEMORY BLOCK

The GAL programs support a variety of alternative memory maps using the jumpers CFG0, CFG1, and CGF2.   See section 5.4 to determine the memory map configuration you wish to use. The following sections describe how to select which chips to use in the ROM memory block, and how to download to, and run programs from, the ROM block.

6.1   Configuring U7 and U8

6.1.1 Chip Selection

The RMB-167i has two sockets marked U7 and U8 which are intended for a variety of non-volatile devices. U7 and U8 are considered to be the ROM block of memory no matter which type of devices are selected. Typically 27C256-type 32K EPROMs, 62C256-type battery-backed RAMs or 28C256-type EEPROMs are used.   Note that RAMs and EEPROMs have the same pinout and while it is possible to populate the sockets with a mix of these two devices it is recommended that you use the same device and speed in both sockets to insure smooth program operation. It is not possible to populate an EPROM in one socket and a RAM or EEPROM in the other since the pinouts of the devices are different. When downloading programs to these devices the odd bytes are loaded into U8 and the even bytes are loaded into U7.

6.1.2 Jumper Selection

The jumpers located next to U8 select which memory devices are to be used in U7 and U8. One end of the jumpers is marked RAM and the other end is marked ROM. Note that since RAMs and EEPROMs have the same pinout the jumper setting is the same for these devices. If U7 and U8 hold RAM or EEPROM devices, insert two jumpers, so that the center posts of both jumpers are connected to the end posts marked RAM.   If U7 and U8 hold EPROMs, insert the two jumpers so that the center posts are connected to the end posts marked ROM.

6.2   Downloading and Running Programs in ROM

The method of loading programs into the ROM block varies depending on the type of devices used. The following are the three options available to the user.

6.2.1 RAMs

Both battery-backed RAMs and static RAMS may be used in U7 and U8. When using RAMs there are two things to remember, any program in a static RAM will be lost if the power to the board is disconnected, and U7 and U8 are the ROM memory block even if populated with RAMs. The procedure for downloading to the two types of RAMS is the same.

Downloading programs
1. With the power disconnected populate U7 and U8 with the RAMs.
2. Insert the two jumpers by U8 in the RAM position.
3. Connect the power to the board.
4. Bootstrap the board as usual.
5. Insert jumpers at CFG0, CFG1, and CGF2 to select the alternate memory map you wish to use. (See section 5.4)
6. Select the **TTY | Download** menu item. Choose your program from the list of files.
7. Press **OK** to download the file.

Running the program with the starting address above zero.

1. In order to run the program select the **TTY | Run** menu item and type the program's starting address at the address field.
2. Select **OK** to run the program.
3. Alternately after the program is downloaded when the monitor prompt appears you may type ***G(starting address)*** to run the program.

Running the program with the starting address at zero.
1. Remove the jumper located in P4 of header JP3. This prevents the processor from entering the bootstrap mode when the reset button is pressed.
2. Press the reset button to run the program.
3. If the power to the board is disconnected the program will run at startup. This will not work with regular 62256K RAMs as the program is lost when the power is turned off. As long as there are no surges or brown out conditions when the power is reconnected to the board the reset button should not have to be pressed when using battery-backed RAMs.

6.2.2 EEPROMs
The RMB-167i accepts 28C256 EEPROMs in U7 and U8. When burning the EEPROMs using the READS166 software you must select the memory map with 48K ROM and 16K RAM, (jumper CFG0 inserted, CFG1 and CFG2 removed). This allows READS166 to burn the EEPROMs and activate the security bit located in EEPROM upper memory. READS166 will not burn the EEPROMs if any other memory map is selected. However, once the EEPROMs are burnt, you may select other memory maps by rearranging the jumpers. There are three different ways you can download (or burn) EEPROMs. The first is to burn the EEPROMs using the READS166 monitor command **TTY | Burn EEPROM.** The second is to use the DOS EEPROM burn utility supplied on the READS166 disc and located on the BBS. The last method is to burn in run time.   This involves writing your program so that it calls a burn EEPROM subroutine in another part of the program. This is especially useful for "in the field" program changes.

Programming the EEPROMs
Burning EEPROMs from the READS166 monitor program
1. With the power disconnected populate U7 and U8 with the EEPROMs.
2. Insert the two jumpers by U8 in the RAM position.
3. Connect the power to the board.
4. Bootstrap the board as usual.
5. Select the memory map with 48K ROM and 16K RAM, jumper CFG0 inserted, CFG1 and CFG2 removed.
6. Select the **TTY | Burn EEPROM** menu item. Choose your program from the list of files.
7. Select **OK** to burn the EEPROM.
8. Once EEPROM is burned you may switch the CFG0, CFG1, and CFG2 jumpers to change the memory map.

Burning EEPROMs using the DOS utility
Set up the board as in steps 1 - 5 above.

At present the README file for the DOS utility is located on the BBS. Please download the README file and follow the directions there.

Burning the EEPROMs in run time

Burning an EEPROM byte takes about 1000 times longer than a memory write cycle. Once a byte is written to the EEPROM, the processor should not address the EEPROM until after the byte is burnt. Thus, the code to burn the EEPROM byte must reside outside the EEPROM.

One approach is to load the burn code into RAM. The demonstration program BURN01.ASM contains a subroutine to burn an EEPROM byte. This subroutine is transferred into RAM and then invoked. The image or copy of the subroutine in RAM burns the EEPROM byte. It returns to the calling program in EEPROM only after the byte is burnt.

Running the program with the starting address above zero.
1. In order to run the program select the **TTY | Run** menu item and type the program's starting address at the address field.
2. Select **OK** to run the program.
3. Alternately after the program is downloaded when the monitor prompt appears you may type **G(starting address)** to run the program.

Running the program with the starting address at zero.
1. Remove the jumper located in P4 of header JP3. This prevents the processor from entering the bootstrap mode when the reset button is pressed.
2. Press the reset button to run the program.
3. Or if the power to the board is disconnected the program will run at startup.   As long as there are no surges or brown out conditions when the power is reconnected to the board the reset button should not have to be pressed when using EEPROMs.

6.2.3 EPROMs

The RMB-167i accepts 27C256-type EPROMs in the ROM memory location. The EPROMs must have the program burned into them using a separate EPROM burner. The RMB-167i will not burn the EPROMs. Using EPROMs in the ROM memory location is the most time consuming, but due to the low cost of the EPROMs it is the most inexpensive method, if you already own an EPROM burner. EPROMs have one other advantage over battery-back RAMs and EEPROMs, they are the most stable method of storing a program in memory. Power disconnects, surges, and brown-outs do not affect their performance.

To develop the EPROM program you may write it and ran out of the RAM block to debug, or you may write and debug it using ROM memory block.   If you use the RAM memory block some of the address locations may need to be changed. If you use EEPROMs or RAM in the ROM location you must remember to change the jumpers above U8 to select the EPROM option when you put the EPROMs on the board. Also any variables in the program must be put into RAM since the EPROM program can not be changed in circuit. Once the program is debugged it may be burned into the EPROMs using a stand alone EPROM burner.

Programming the EPROMs

The EPROMs must have the program burned into them using a separate EPROM burner. The RMB-167i will not burn the EPROMs. When programming the EPROMs, split your code into odd and even bytes. Almost all EPROM burners support this option.

Running the program with the starting address above zero.

1. With the power disconnected populate U7 and U8 with the pre-programmed EPROMs.
2. Insert the jumpers above U8 into the ROM position.
3. Connect the power to the board.
4. Bootstrap the board as usual.
5. Insert jumpers at CFG0, CFG1, and CGF2 to select the alternate memory map you wish to use. (See section 5.4)
6. Select the **TTY | Run** menu item and type the program's starting address at the address field.
7. Press **OK** to run the program.
8. Alternately after the program is downloaded when the monitor prompt appears you may type **G(starting address)** to run the program.

Running the program with the starting address zero.

1. With the power disconnected populate U7 and U8 with the pre-programmed EPROMs.
2. Insert the jumpers above U8 into the ROM position.
3 Remove P4 on header JP3. This prevents the processor from entering the bootstrap mode when the reset button is pressed.
4. Insert jumpers at CFG0, CFG1, and CGF2 to select the alternate memory map you wish to use. (See section 5.4)
5. Connect the power to the board.
6. The program will start from address zero.
7. If the board is powered-up you can restart your program by pressing the reset button.

7. PAL EQUATIONS

A PALCE16V8, and a PALCE22v10 are used. U9 is responsible for the bootstrap loader logic, and the second U10, for the memory decode logic.

7.1 U9 Equations

```
;PALASM Design Description
;----------------------- Declaration Segment ------------
TITLE    RMB-167i Memory Decode Logic -- RAM/EPROM
REVISION 1.0
COMPANY  Rigel Corporation
DATE     05/15/95


CHIP  _r165u9  PALCE16V8


;----------------------- PIN Declarations ---------------
; --- inputs ---
PIN  1         A18                          COMBINATORIAL ;
PIN  2         A19                          COMBINATORIAL ;
PIN  3         A20                          COMBINATORIAL ;
PIN  4         A21                          COMBINATORIAL ;
PIN  5         A22                          COMBINATORIAL ;
PIN  6         A23                          COMBINATORIAL ;
PIN  7         RSTIN_                        COMBINATORIAL ;
PIN  8         RSTOUT_                       COMBINATORIAL ;
PIN  9         CFG0                         COMBINATORIAL ;
PIN 11         CFG1                         COMBINATORIAL ;
; --- outputs ---
PIN  12        HISEG                        COMBINATORIAL ;
PIN  13        RSTCYC                       COMBINATORIAL ;
PIN  14        RSTCYC_                       COMBINATORIAL ;
PIN  15        AUX0                         COMBINATORIAL ;
PIN  16        AUX1                         COMBINATORIAL ;
PIN  17        DUMMY                        COMBINATORIAL ;
PIN  18        LEDRST_                       COMBINATORIAL ;
PIN  19        LEDAUX_                       COMBINATORIAL ;
;----------------------- Boolean Equation Segment ------
EQUATIONS
RSTCYC   =  /(  RSTIN_   * RSTCYC_ )
RSTCYC_  =  /( /RSTOUT_  * RSTCYC  )

HISEG     =  A18 * A19 * A20 * A21 * A22 * A23
AUX0      =  CFG0
AUX1      =  CFG1
DUMMY     =  1
LEDRST_   =  RSTCYC
LEDAUX_   =  RSTCYC_
;------------------------------------------------------------
```

7.2 U10 Equations

```
;PALASM Design Description
; CFG0 and CFG1 select RAM and ROM banks
; CFG2 determines block sizes
;
; Memory Map with CGF2 absent
;
; jumper at          low 48K    high 16K
; ------------       -------    --------
; NONE                 RAM        RAM
; CFG0                 ROM        RAM
; CFG1                 RAM        ROM
; CFG0 and CFG1        ROM        ROM
;
; Memory Map with CGF2 present
;
; jumper at          low 32K    high 32K
; ------------       -------    --------
; NONE                 RAM        RAM
; CFG0                 ROM        RAM
; CFG1                 RAM        ROM
; CFG0 and CFG1        ROM        ROM
;
;-------------------------- Declaration Segment -----------
TITLE    RMB-167i Memory Decode Logic
REVISION 1.1
COMPANY  Rigel Corporation
DATE     05/15/95

CHIP  _r165u10  PALCE22V10

;----------------------- PIN Declarations ---------------
; --- inputs ---
PIN  1          A0                      COMBINATORIAL ;
PIN  2          A13                     COMBINATORIAL ;
PIN  3          A14                     COMBINATORIAL ;
PIN  4          A15                     COMBINATORIAL ;
PIN  5          A16                     COMBINATORIAL ;
PIN  6          A17                     COMBINATORIAL ;
PIN  7          BHE_                    COMBINATORIAL ;
PIN  8          MON                     COMBINATORIAL ;
PIN  9          CFG2                    COMBINATORIAL ;

PIN 10          HISEG                   COMBINATORIAL ;
PIN 11          RESET                   COMBINATORIAL ;
PIN 13          CFG0                    COMBINATORIAL ;
PIN 23          CFG1                    COMBINATORIAL ;

; --- outputs ---
PIN  14         RAMSELL_                COMBINATORIAL ;
PIN  15         RAMSELH_                COMBINATORIAL ;
PIN  16         ROMSELL_                COMBINATORIAL ;
PIN  17         ROMSELH_                COMBINATORIAL ;
PIN  18         MA13                    COMBINATORIAL ;
PIN  19         MA14                    COMBINATORIAL ;
```

```
PIN  20          MA15                      COMBINATORIAL ;
PIN  21          MA16                      COMBINATORIAL ;
PIN  22          MA17                      COMBINATORIAL ;

;---------------------- Boolean Equation Segment ------
EQUATIONS
RAMSELL_ = A0   + (A15+/CFG0) * (/A15+/CFG1) + (/A14 * /CFG2)
RAMSELH_ = BHE_ + (A15+/CFG0) * (/A15+/CFG1) + (/A14 * /CFG2)
ROMSELL_ = A0   + (A15+CFG0)  * (/A15+CFG1)  * (A14 + CFG2)
ROMSELH_ = BHE_ + (A15+CFG0)  * (/A15+CFG1)  * (A14 + CFG2)

MA13     = A13
MA14     = A14
MA15     = A15
MA16     = A16
MA17     = A17
;-------------------------------------------------------
```

8. HEADERS

The RMB-167i board has three headers: the input/output header JP9 and the system header JP8 and the extra input/output header JP10. Ports 2, 3, and 5 are available on JP9. JP8 contains the address, data and control busses. Individual signals of these jumpers are listed below. The tables reflect the physical orientation of the headers and the enumeration of their individual posts. Pin 1 may be identified as the post with the square pad on the printed circuit board.

8.1 JP8 - System Header

Signal			Pins		Signal		
RMB-167 RMB167i	RMB-165 (JP11)	RMB-166 (JP1)			RMB-166 (JP1)	RMB-165 (JP11)	RMB-167 RMB167i
Ground			1	2	VCC (+5V)		
Ground			3	4	VCC (+5V)		
D0			5	6	A0		
D1			7	8	A1		
D2			9	10	A2		
D3			11	12	A3		
D4			13	14	A4		
D5			15	16	A5		
D6			17	18	A6		
D7			19	20	A7		
D8			21	22	A8		
D9			23	24	A9		
D10			25	26	A10		
D11			27	28	A11		
D12			29	30	A12		
D13			31	32	A13		
D14			33	34	A14		
D15			35	36	A15		
RD#			37	38	A16		
ALE			39	40	A17		
RSTIN#			41	42	WR#	WR# or WRL#	
RSTOUT#			43	44	BHE#		
NMI#			45	46	not used		A18
A22		not used	47	48	not used		A19
A21		not used	49	50	not used		A20

8.2 JP9 - Input/Output Header

Signal			Pins		Signal		
RMB-167 RMB167i	**RMB-165 (JP12)**	**RMB-166 (JP2)**			**RMB-166 (JP2)**	**RMB-165 (JP12)**	**RMB-167 RMB167i**
Ground			1	2	VCC (+5V)		
Ground			3	4	VCC (+5V)		
P5.0	not used	P5.0	5	6	P5.1	not used	P5.1
P5.2	not used	P5.2	7	8	P5.3	not used	P5.3
P5.4	not used	P5.4	9	10	P5.5	not used	P5.5
P5.6	not used	P5.6	11	12	P5.7	not used	P5.7
P5.8	not used	P5.8	13	14	P5.9	not used	P5.9
VAGND	not used	VAGND	15	16	S1I	not used	
VAREF	not used	VAREF	17	18	S1O	not used	
P2.0	not used	P2.0	19	20	P3.0		
P2.1	not used	P2.1	21	22	P3.1		
P2.2	not used	P2.2	23	24	P3.2		
P2.3	not used	P2.3	25	26	P3.3		
P2.4	not used	P2.4	27	28	P3.4		
P2.5	not used	P2.5	29	30	P3.5		
P2.6	not used	P2.6	31	32	P3.6		
P2.7	not used	P2.7	33	34	P3.7		
P2.8			35	36	P3.8		
P2.9			37	38	P3.9		
P2.10			39	40	P3.10		
P2.11			41	42	P3.11		
P2.12			43	44	P3.12		
P2.13			45	46	P3.13		
P2.14			47	48	P3.14	not used	
P2.15			49	50	P3.15		

8.3 JP10 - Extra Input/Output Header

Signal		Pins		Signal	
RMB-167i RMB-167	**RMB-165i**			**RMB-165i**	**RMB-167i RMB-167**
Ground		1	2	VCC (+5V)	
Ground		3	4	VCC (+5V)	
P6.0		5	6	P6.1	
P6.2		7	8	P6.3	
P6.4		9	10	P6.5	
P6.6		11	12	P6.7	
A23	not used	13	14	READY#	
VAGND	not used	15	16	not used	
VAREF	not used	17	18	not used	
P5.0	not used	19	20	not used	P7.0
P5.1	not used	21	22	not used	P7.1
P5.2	not used	23	24	not used	P7.2
P5.3	not used	25	26	not used	P7.3
P5.4	not used	27	28	not used	P7.4
P5.5	not used	29	30	not used	P7.5
P5.6	not used	31	32	not used	P7.6
P5.7	not used	33	34	not used	P7.7
P5.8	not used	35	36	not used	P8.0
P5.9	not used	37	38	not used	P8.1
P5.10		39	40	not used	P8.2
P5.11		41	42	not used	P8.3
P5.12		43	44	not used	P8.4
P5.13		45	46	not used	P8.5
P5.14		47	48	not used	P8.6
P5.15		49	50	not used	P8.7

9. BOOTSTRAPPING

The RMB-167i bootstrapping is triggered by grounding P0L.4 at reset. A 32 pin header is used to ground the bits of P0L via a resistor array with a nominal value in the range of 15K to 33K.

Note that the GAL which controls the bootstrap load operation is also responsible for turning on the LED. In its default implementation, the LED is lit once the RSTOUT# signal is activated. For specific applications, the user may alter the operation of the bootstrap logic by changing the GAL equations.

Once the bootstrap loader is invoked the serial port S0 is used to communicate with the C167. The host must first send a 0 byte with 8 data bits, 1 stop bit and no parity bits. The C167 responds with the byte A5h (this changes for the SR and CR versions of the 167) . Then the host expects 32 bytes of code to be downloaded to internal RAM starting at address 0FA40h and run.

Since 32 bytes is not enough to initialize and configure the C165 and then download a user program, a secondary loop is used. This loop is a short piece of code that is placed starting at address 0FA60h, so that when the 32 bytes of primary code are executed, the program continues with the secondary loop. The approach is described in more detail below.

The 32 bytes downloaded are, in hexadecimal,

```
E6 F0 60 FA
9A B7 FE 70
A4 00 B2 FE
7E B7
B4 00 B0 FE
86 F0 BB FC
3D F6
CC 00
CC 00
CC 00
CC 00
```

which correspond to the following short code.

```
        ; origin is 0FA40h
        mov     R0, #0fa60h
   W0:
        jnb     S0RIR, W0
        movb    [R0], S0RBUF
        bclr    S0RIR
        movb    S0TBUF, [R0]
        cmpi1   R0, #0fcbb  ; read 604 bytes
        jmpr    cc_NE, W0
        nop
        nop
        nop
        nop
```

Note that the NOP (no operation) operations are required to fill the 32 bytes, since the bootstrap loader remains active until all 32 bytes are received. When the bootstrap loader receives its last byte and places it in address 0FA5Fh, it makes a jump to 0FA40h and starts executing the code. This is the short loop given above. Note that at this time the internal RAM starting from 0FA60h does not contain any relevant code.

The short loop takes advantage of the serial port S0 which is already initialized. It waits for a user specified number of bytes, 604 bytes in this case, and places these bytes consecutively starting from internal RAM location 0FA60h. When the loop is done (all 604 bytes received) the program continues by executing the NOP operations, and then executing code from 0FA60h on. Thus the 604 bytes loaded by the secondary loop are also interpreted as code.

The user may alter the number of bytes to be loaded by changing the 21st and 22nd bytes (BB and FC) which give the address (the low byte, followed by the high byte) of the last byte to be read by the loop. Note that there is a practical limit to the number of bytes that can be downloaded by this loop: the PEC source and destination pointers as well as the SFRs which occupy addresses FDE0h and above must not be overwritten by data bytes.

Due to the powerful instruction set of the C165, a lot of functionality can be implemented within 604 bytes of code. The 604 bytes contained in the file BTL67.DAT downloads a minimal monitor program. This program contains an initialization routine, subroutines to send and receive characters through the serial port, a subroutine to download code in the Intel Hex format, and a subroutine to jump to any location within the 64K segment. The latter two are invoked by single-letter commands.

The 604 byte-code may be broken down into four sections.

1.   Initialization code to be executed after the 32-byte bootstrap
2.   Code to be written starting at address 0 to be executed after the software reset.
3.   The minimal monitor to be placed starting at address 8000h
4.   The software reset (SRST) instruction to leave the bootstrap mode.

Sections 1 and 4 are somewhat different than sections 2 and 3. The bytes downloaded in sections 1 and 4 are actual instructions which are executed after the 32-byte bootstrap load is completed. Sections 2 and 3 are instructions to poke bytes into memory. More specifically, in section 2, bytes are written to memory locations starting from address 0. In section 3, from address 8000h. The bytes placed into memory locations starting from address 0 is executed after the software reset instruction. This is an initialization program which, upon completion, branches to address 8000h to execute the minimal monitor program.

For example, the initialization code starting at address 0 begins with the two instructions

```
        DISWDT
        EINIT
```

whose machine instructions are (A5 5A A5 A5) and (B5 4A B5 B5),

respectively. The code within the 604-byte download block pokes these bytes starting from address 0. That is, these instructions are placed into memory, one word at a time, as data. The following instructions are used.

```
mov     R1, #0A55Ah          ; begin: DISWDT
mov     0, R1
mov     R1, #0A5A5h
mov     2, R1

mov     R1, #0B54Ah          ;          EINIT
mov     4, R1
mov     R1, #0B5B5h
mov     6, R1
```

This pattern is used throughout sections 2 and 3. First the word is written to register R1. Then the register is copied to memory. The file BTL67.DAT contains the bytes downloaded to the RMB-167i board during bootstrapping. The file BTL.SRC contains the source code.
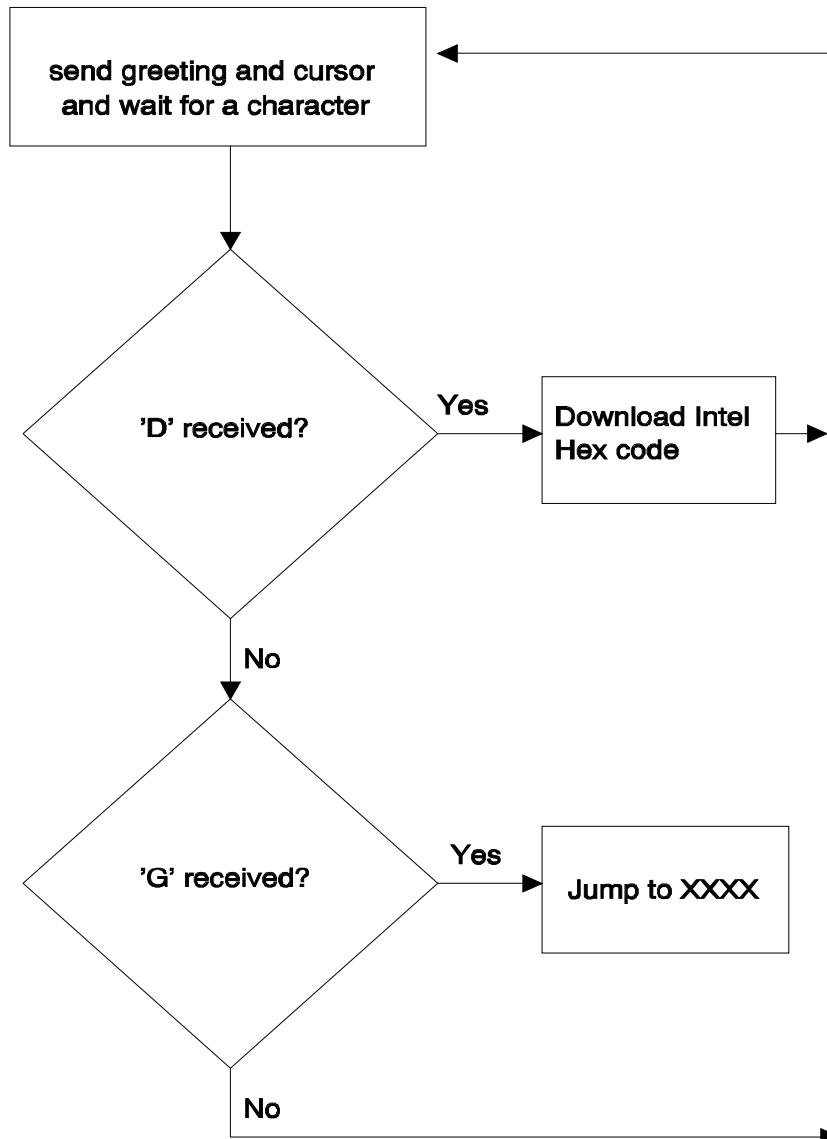
The initialization routines configure the SYSCON register. The internal ROM is disabled and the external bus is activated. Next the CSP and DPP registers are initialized. These steps need to be completed before the EINIT instruction. Note that if the watchdog timer is to be disabled; this too must be done before the EINIT instruction.

10. THE MONITOR PROGRAMS

10.1 Minimal Monitor

The minimal monitor is placed by the bootstrap loader starting at address 8000h. The monitor responds to two single-letter commands 'D' and 'G'. The 'D' command places the monitor in a download mode. Code in the Intel Hex format is expected. Code may be downloaded anywhere in the first 64K segment. The 'G' (Go) command expects 4 hexadecimal characters. These 4 characters specify an address within the first 64K segment. A jump is performed to this address. If a user program is downloaded (using the 'D' command), say at address 0C000h, then the GC000 command branches to the user program. In many cases, the user program is the application program or a monitor program, such as RMON167, and hence, the minimal monitor is no longer required. If, however, the user program wishes to return to the minimal monitor, it should branch to address 8000h. Note that the minimal monitor initializes the stack, so either a call or a jump to address 8000h would work.

The minimal monitor is a loop that executes the following flowchart.

10.2   MON167 Monitor

The monitor program RMON167 allows inspecting and modifying the first 64K segment of RMB-167i memory, configuring the ports, inputting and outputting from the general purpose ports, downloading code in the Intel Hex format, and branching to user code. RMON167 features are invoked by single-letter commands. RMON167 assumes a 40Mhz (5 MHz) system crystal. Serial port 0 is initialized to run at 9600 Baud with 8 bits of data, 1 stop bit and no parity bits.

RMON167 is intended to be downloaded after bootstrapping the RMB-167i board. RMON167 is placed starting at address 0C000h. The first 256 bytes are reserved for monitor variables. The entry point to RMON167 is at address C100h. To set up RMON167, initialize READS166 and the RMB-167i board and invoke the Bootstrap command as explained in the previous section. From the TTY menu, select **Download** to download RMON167.HEX. Branch to and execute RMON167 using the **Run** command under the TTY menu. Specify address C100 since the entry point to RMON167 is at 0C000h. Note that RMON167 places a jump to C100 at the nonmaskable interrupt vector. Thus, RMON167 may subsequently be invoked by pressing the NMI pushbutton on the RMB-167i. RMON167 initializes the stack and resets the interrupts. Thus, even after the NMI button is pressed, RMON167 clears the NMI interrupt by executing a dummy 'return from interrupt' instruction.

Alternatively, RMON167 may be placed in EPROM and invoked upon reset. The source code for RMON167 is given on the distribution disk. RMON167 is not optimized for speed or size, but rather for clarity and pedagogical value. The legal users are encouraged to modify RMON167 and use portions of it in applications programs.

The single-letter commands of RMON167 are explained below.

D   Download HEX file
The D command places RMON167 in a download mode. The monitor expects to receive code in the Intel Hex format through serial port 0. The download mode is terminated when the last line of Intel Hex code is received (when the byte count is 0).

C   Port Configuration
The C command is used to configure the ports, i.e., the port direction registers DPnn. Cn displays the current setting of DPn. Cn=mmmm writes the word mmmm to register DPn.

G   Go
The user code at address xxxx is branched to by the Gxxxx command. Note that the user program may return to RMON167 by a branching instruction to address 0C000h. RMON167 initializes the stack, thus, either a jump or a call instruction may be used to return to RMON167.

H   Help
The H command displays a summary of available monitor commands.

M   Memory

The first 64K segment of the RMB-167i memory may be inspected or modified by the M command. The M command is also useful to poke short programs into memory.

M XXXX displays the current contents of memory address XXXX.

M XXXX=nn inserts the byte nn into memory address XXXX. When this command is used, RMON167 displays the current contents as well as the new contents. The address XXXX is incremented and the current contents of (XXXX+1) are displayed. Consecutive bytes may be written starting at XXXX. The process is terminated if a carriage return or an illegal hexadecimal digit is keyed in.

M XXXX-YYYY displays the block of memory between addresses XXXX and YYYY.

M XXXX-YYYY=nn fills the memory block XXXX to YYYY with byte nn.

P   Port Data

The P command is used to read from or write to the ports. Pn displays the current value of port n. If port n is an input port, then the value read is the current voltage levels applied to the ports. If port n is an output port, Pn returns the current output value to port n. Pn=mmmm sets the current value of output port n to mm.

Note that individual bits of the ports may be programmed as input or output. Thus, the word returned by Pn gives the external voltage levels applied to the input bits and the current values of the output bits.

W   Word Memory

This command is identical to the M command, except that the memory contents are displayed and modified as words (2 bytes). Words start at even address.