

# **RMB-166 USER'S GUIDE**

**Version 1.1  
December 2000**

**RIGEL CORPORATION**  
P.O. BOX 90040, GAINESVILLE, FL 32607  
(352) 373-4629 Fax (353) 373-1786  
[www.rigelcorp.com](http://www.rigelcorp.com), [tech@rigelcorp.com](mailto:tech@rigelcorp.com)



**(C) 2001 by Rigel Corporation.**

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Rigel Corporation.

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.

## **Rigel Corporation's Software License Agreement**

This Software License Agreement ("Agreement") covers all software products copyrighted to Rigel Corporation, including but not limited to: Reads51, rLib51, RbHost, RitaBrowser, FLASH, rChpSim, Reads166, and rFLI.

This Agreement is between an individual user or a single entity and Rigel Corporation. It applies to all Rigel Corporation software products. These Products ("Products") includes computer software and associated electronic media or documentation "online" or otherwise.

Our software, help files, examples, and related text files may be used without fee by students, faculty and staff of academic institutions and by individuals for non-commercial use. For distribution rights and all other users, including corporate use, please contact:

Rigel Corporation, PO Box 90040, Gainesville, FL 32607

or e-mail [tech@rigelcorp.com](mailto:tech@rigelcorp.com)

### **Terms and Conditions of the Agreement**

1. These Products are protected by copyright laws, intellectual property laws, and international treaties. Rigel Corporation owns the title, copyright, and all other intellectual property rights in these Products. We grant you a personal, non-transferable, and non-exclusive license to use the Products. These Products are not transferred to you, given away to you or sold to you.  
  
Non-commercial use: These Products are licensed to you free of charge.  
  
Commercial use: You must contact Rigel Corporation to find out if a licensing fee applies before using these Products.
2. You may install and use an unlimited number of copies of these Products.
3. You may store copies of these Products on a storage device or a network for your own use.
4. You may not reproduce and distribute these Products to other parties by electronic means or over computer or communication networks. You may not transfer these Products to a third party. You may not rent, lease, or lend these Products.
5. You may not modify, disassemble, reverse engineer, or translate these Products.
6. These Products are provided by Rigel Corporation "as is" with all faults.
7. In no event shall Rigel Corporation be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use the Product, even if Rigel Corporation has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitations of consequential or incidental damages, the above limitations may not apply to you.
8. Rigel Corporation makes no claims as to the applicability or suitability of these Products to your particular use, application, or implementation.
9. Rigel Corporation reserves all rights not expressly granted to you in this Agreement.
10. If you do not abide by or violate the terms and conditions of this Agreement, without prejudice to any other rights, Rigel Corporation may cancel this Agreement. If Rigel Corporation cancels this Agreement; you must remove and destroy all copies of these Products.
11. If you acquired this Product in the United States of America, this Agreement is governed by the laws of the Great State of Florida. If this Product was acquired outside the United States of America all pertinent international treaties apply.

## **HARDWARE WARRANTY**

**Limited Warranty.** Rigel Corporation warrants, for a period of sixty (60) days from your receipt, that READS software, RROS, 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. Although Rigel Corporation warranty covers 60 days, Rigel shall not be responsible for malfunctions due to customer errors, this includes but is not limited to, errors in connecting the board to power or external circuitry. This warranty does not apply to products which have been subject to misuse (including static discharge), neglect, accident or modification, or which have been soldered or altered during assembly and are not capable of being tested.

### **DO NOT USE PRODUCTS SOLD BY RIGEL CORPORATION AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS!**

Products sold by Rigel Corporation are not authorized for use as critical components in life support devices or systems. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

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 RIGEL CORPORATION PRIOR TO USE.

**Return Policy.** This policy applies only when product purchased 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. This does not apply to books. Books are non-returnable.

Please call (352) 373-4629 to receive an RMA (Returned Merchandise Authorization) number prior to returning product. You will be responsible for shipping costs.

All returns must be made within 30 days of date of invoice and be accompanied by the original invoice number and a brief explanation of the reason for the return.

Return merchandise in original packaging.

All returned products are subject to a \$15 restocking charge. "Custom Items" are not returnable.

**Repair Policy.** If you encounter problems with your board or software after the 60 day warranty period, please call Rigel Corporation at (352) 373-4629 or email [tech@rigelcorp.com](mailto:tech@rigelcorp.com) for advice and instruction.

Rigel Corporation will test and attempt to repair any board. You will be responsible for shipping costs and repair fees. If you send a detailed report of the problems you encountered while operating the board, Rigel Corporation will inspect and test your board to determine what the problem is free of charge. Rigel Corporation will then contact you with an estimated repair bill. You will have the choice of having the board fixed, returned to you as is, or purchasing a new board at a reduced price. Rigel Corporation charges repair fees based on an hourly rate of \$50.00. Any parts that need to be replaced will be charged as separate items. Although Rigel Corporation will test and repair any board, it shall not be responsible for malfunctions due to customer errors, this includes but is not limited to, errors in connecting the board to power or external circuitry.

**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 send you an assembled and tested board and bill you the price difference. 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.

**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

<b>1</b>	<b>OVERVIEW</b> .....	<b>1</b>
1.1	HARDWARE OVERVIEW.....	1
1.2	READS166 OVERVIEW.....	1
1.3	PARTS LIST.....	1
<b>2</b>	<b>QUICK START TUTORIAL</b> .....	<b>3</b>
2.1	SYSTEM REQUIREMENTS .....	3
2.2	SOFTWARE INSTALLATION, READS166 .....	3
2.3	THIRD PARTY SOFTWARE .....	3
2.4	START UP.....	3
2.5	SERIAL NUMBER.....	3
2.5.1	Demo Version .....	3
2.5.2	Full Version .....	3
2.6	CONFIGURING READS166 AND INITIATING HOST-TO-BOARD COMMUNICATIONS.....	4
2.7	BOOTSTRAPPING.....	5
2.8	VERIFYING THAT THE MONITOR IS LOADED .....	5
2.9	RUNNING A PROGRAM .....	5
2.10	USING HELP.....	6
<b>3</b>	<b>OPERATING NOTES</b> .....	<b>7</b>
3.1	POWER .....	7
3.2	SERIAL PORT 1.....	7
3.3	DEFAULT JUMPER SELECTION.....	7
3.4	LEDS.....	7
3.5	PUSH BUTTONS.....	7
3.5.1	Reset (S1) .....	7
3.5.2	NMI (S2) .....	7
3.6	SLIDE SWITCH.....	7
<b>4</b>	<b>JUMPER CONFIGURATIONS</b> .....	<b>8</b>
4.1	BTLDIS - BOOTSTRAP ENABLE JUMPER .....	8
4.2	ROM - EPROM ENABLE JUMPER.....	8
4.3	AUX1, AUX2, MON AND BTL JUMPERS .....	8
4.3	JP4, JP5 - SERIAL PORT JUMPERS.....	8
4.4	JP8, JP9 - ANALOG-TO-DIGITAL CONVERTER REFERENCE .....	8
<b>5</b>	<b>HEADERS</b> .....	<b>9</b>
5.1	SERIAL HEADERS .....	9
5.2	JP1 .....	9
5.3	JP2 .....	10
<b>6</b>	<b>MEMORY BLOCK OPTIONS</b> .....	<b>11</b>
6.1	DEFAULT MEMORY SETTING .....	11
6.2	RAM MEMORY OPTIONS .....	11
6.3	ROM MEMORY OPTIONS .....	11
6.4	DOWNLOADING AND RUNNING PROGRAMS IN EPROM.....	11
<b>7</b>	<b>PAL EQUATIONS</b> .....	<b>13</b>
7.1	U4 EQUATIONS .....	13
7.2	U11 EQUATIONS .....	13
<b>8</b>	<b>BOOTSTRAPPING</b> .....	<b>15</b>

<b>9</b>	<b>THE MONITOR PROGRAMS</b> .....	<b>18</b>
9.1	THE MINIMAL MONITOR .....	18
9.2	RMON166 MONITOR.....	19
<b>10</b>	<b>PARTS LIST</b> .....	<b>21</b>
<b>11</b>	<b>TOP OVERLAY</b> .....	<b>22</b>







# 1 OVERVIEW

The RMB-166 evaluation board contains the Infineon (Siemens) 80C166 16-bit high-performance microcontroller. The microcontroller is run with a 16-bit nonmultiplexed data bus and an 18-bit nonmultiplexed address bus. The board may be configured in several different ways depending on the type of reset and EPROM options. The default configuration is the 64K RAM and no EPROM mode. In this mode, the monitor program or user program is downloaded to RAM using the 80C166 bootstrap feature. A set of option headers, decoded by PAL devices make the RMB-166 a flexible hardware platform.

## 1.1 Hardware Overview

- High-performance 16-bit microcontroller in the metric package
  - Bootstrap loading feature
  - Runs at 40Mhz with zero wait states
  - Internal 10-bit analog-to-digital converter
  - 10 bits of analog or digital inputs (Port 5)
  - Two 16-bit general-purpose input/output ports (Port 2 and Port 3)
  - Two serial ports
- Serial ports are driven with a MAX232 and terminate at DB-9s and a 6-post header
- Accommodates 64K or 256K of SRAM (64K installed)
- Accommodates 64K of ROM (not installed)
- Push buttons for RESET# and NMI#
- GAL-decoded memory map for maximum flexibility
- GALs 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" six layer industrial board
- Mounting holes in all four corners

## 1.2 READS166 Overview

READS166, version 3.0x, is Rigel Corporation's Integrated Development Environment for the SGS Thomson 16-bit processors. READS166 includes an editor, a host-to-board communications system, an assembler, and a C compiler. READS166 is completely rewritten in native 32-bit code to run on Windows95 and WindowsNT. READS166 includes a sophisticated project management system to simplify code reusability and version control. The C compiler is rewritten to support a full debugger. The debugger allows you to step through your code with breakpoints and variable watches as the compiled code runs on the target board, similar to the operation of an in-circuit emulator.

### **RMON166 - The READS166 monitor program**

RMON 166 is downloaded after bootstrapping (or it may be placed into ROM) and supports basic memory and port functions. RMON166 allows downloading and running applications programs. The complete source code for user modifications or upgrades is included on disk.

### **Ra66 - The READS166 Assembler**

Ra66 is an assembler for the C166 family of controllers. It is a multi-pass absolute assembler which generates HEX code directly from assembly source code. The assembler in the demo version of READS166 limits the size of code to about 8K.

### **Rc66 - The READS166 C Compiler**

Rc66 is a C Compiler for the C166 family of processors. It compiles code for the tiny memory model which fully resides in the first segment of memory. Rc66 is designed as a low-cost C compiler which provides a quick development cycle for simpler applications which do not need more than 64K of code, or the use of standard C libraries. Rc66 implements a subset of ANSI C. Rc66 works in conjunction with Ra66: first an assembly language program is generated from the C source, then a HEX file is generated. Currently, structures, unions, enumerated types, and the typedef directive are not implemented. The C-compiler in the demo version of READS166 limits the size of code to about 8K.

## 1.3 Parts List

Your RMB-166I package includes the following:

**Hardware**

1. RMB-166I board with a 64K of static RAM.

**Software**

1. RMON166 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 Infineon (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

The RMB-166 board comes with a variety of software from Rigel Corporation and third party vendors, whom we work very closely with. Rigel's own software for the 166 family may be found on the CD-ROM in the Rigel Products file, under 166 Software.

### 2.1 System Requirements

READS166 is designed to work with an IBM PC or compatible, 486 or better, running Windows 95 or Windows NT.

### 2.2 Software Installation, READS166

Place the first READS166 disk or the CD-ROM in your drive.

To install from Windows 95 /NT select **START | Settings | Control Panel**. Choose the **Add/Remove Programs**, and follow the standard install directions answering the questions with the appropriate answers.

### 2.3 Third Party Software

A variety of third party software is available on Rigel's CD-ROM. The software may be found in the Strategic Partners Folder under Keil, Tasking, and PLS.

Data sheets on the IC's used on our boards may also be found on Rigel's CD.

Please note that software updates occur frequently. Please check our web site (and the third party vendors web sites) occasionally to make sure you have the most recent versions of the software.

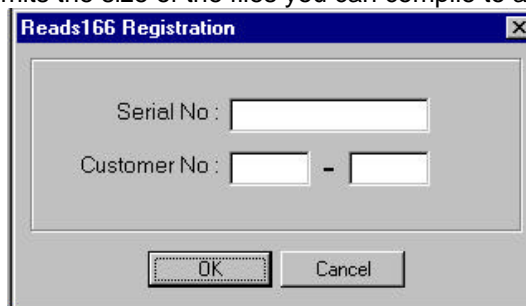
### 2.4 Start up

1. Connect the RMB-166 to a well-regulated 5 Volt supply.
2. Connect the RMB-166 to the PC host via a modem cable.
3. Check to make sure the bootstrap disable jumper (BTLDIS - JP10) is installed. This is the default configuration for the RMB-166.
4. Run the READS166 host driver by selecting **Start | Programs | READS166**. You may also start READS166 by double clicking on the READS166 short cut icon if installed.

### 2.5 Serial Number

#### 2.5.1 Demo Version

When you run the READS166 software a pop-up registration box will open asking for your serial and customer numbers. If you are using the demo version of the software you may press **CANCEL** and the box will disappear and the About Box will appear. Press the **OK** button in the About Box and the software will run in the Demo mode. The demo mode limits the size of the files you can compile to about 8K.



Reads166 Registration

Serial No :

Customer No :  -

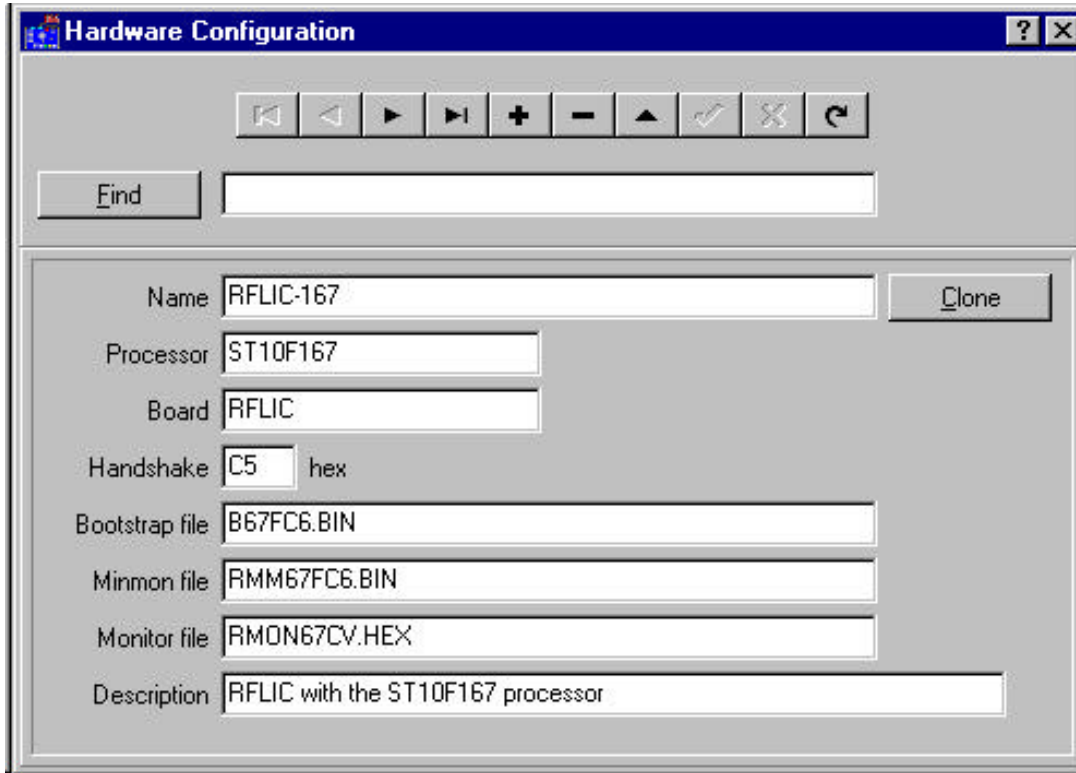
OK Cancel

#### 2.5.2 Full Version

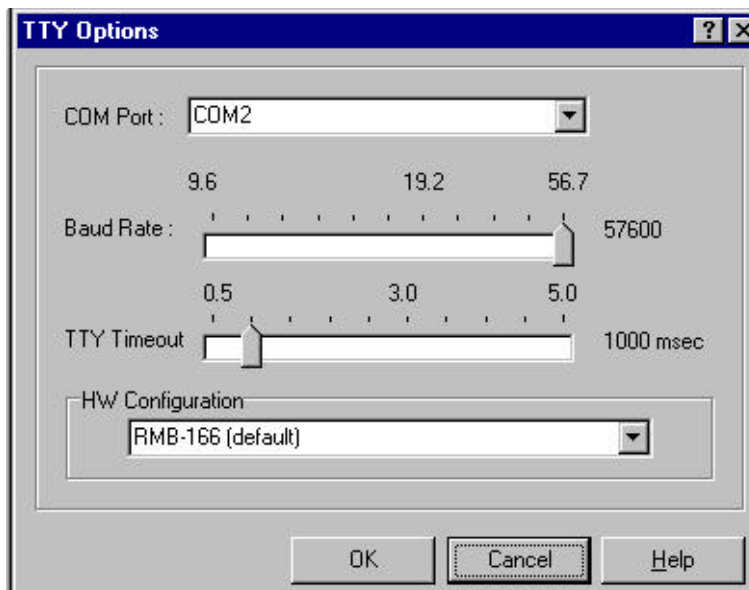
If you've purchased the READS166 software a registration sheet with the serial number and customer number will be included with the READS166 User's Guide. When you run the READS software insert these numbers in the correct boxes, press **OK** and then press **OK** in the About Box. Your software is registered and ready to use.

## 2.6 Configuring READS166 and Initiating Host-to-Board Communications

1. Press the **Projects | Options | Project Properties | HW Configuration** to select the board you are using from the list. Or using the **Tools | Hardware Configuration** menu command, choose the name of the board you are using from the board list which appears.



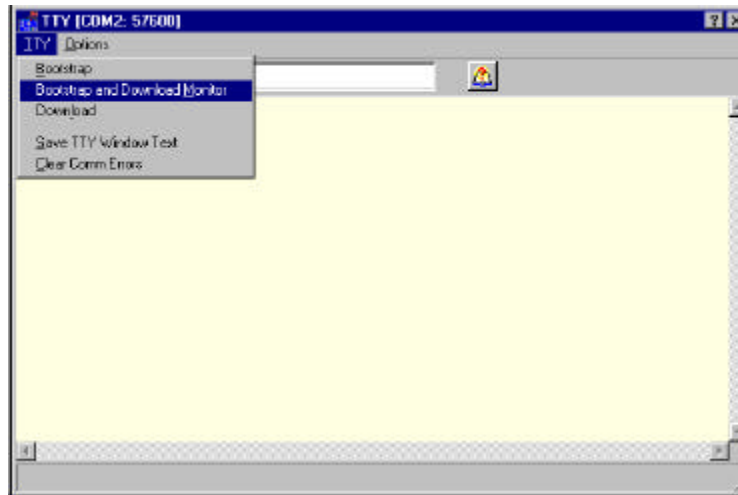
2. Open the TTY window using the **Tools | TTY** menu command. Select the communication port parameters using the **Tools | TTY | TTY Options** menu command. You will need to select the COM port you are using, and the baud rate.



## 2.7 Bootstrapping

In the default configuration, all monitor programs are downloaded to the boards after the boards are bootstrapped. That is, there is no ROM on the board that is executed upon reset. Bootstrapping loads a small monitor, called MinMon, which in turn loads a larger monitor RMON16x. Once the monitor program is loaded, the monitor commands are available to the user.

1. Press the reset button on the board.
2. From the menu in the TTY window select **TTY | Bootstrap and Download Monitor**. The board will now bootstrap and download the monitor program.



You may observe the bootstrap progress in the status line of the TTY window. When bootstrapping is completed, the READS166 monitor prompt appears in the TTY window.

## 2.8 Verifying 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.

The READS monitors use single-letter commands to execute basic functions. Port configurations and data, as well as memory inspection and modifications may be accomplished by the monitor. Most of the single-letter commands are followed by 4 hexadecimal digit addresses or 2 hexadecimal digit data bytes. The following is a list of the commands.

### READS COMMANDS

---

C nn	read port nn Configuration (DPnn)
C nn=mmmm	set port nn Configuration (DPnn=mmmm)
D	Download HEX file
G XXXX	Go, execute code at XXXX
H	Help, display this list
M XXXX	Memory, contents of XXXX
M XXXX=nn	Memory, change contents of XXXX to nn
M XXXX-YYYY=nn	Memory, change block XXXX-YYYY to nn
P nn	read Port nn (Pnn)
P nn=mmmm	write to Port nn (Pnn=mmmm)
W XXXX	Word memory, contents of XXXX
W XXXX=mmmm	Word memory, change contents of XXXX to mmmm
W XXXX-YYYY=mmmm	Word memory, change block XXXX-YYYY to mmmm

## 2.9 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.

## 2.10 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-166I needs two connections: to a 5 volt 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-166 board by a two-position screw-type terminal block or a DC jack. A well regulated 5V DC (+/- 5%) 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-166 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-166 draws less than 175 mA.

### 3.2 Serial Port 1

Serial port P1 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. P1 (HOST) of the RMB-166 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-166 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. JP3 is a 6-pin header, which carries the same signals as P1 and P2.

### 3.3 Default Jumper Selection

Only one jumper across BTLDIS (JP10) is needed to use the RMB-166 in the default 64K/256K RAM and no EPROM mode. This jumper connects the NMI# input to the bootstrap circuitry. Removing JP10 physically prevents the processor from entering the bootstrap mode.

### 3.4 LEDs

The RMB-166 has three LEDs. The red LED, when lit, shows power is connected to the board. 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 nonfunctional.

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. The LED RO will be off and remain off until the bootstrap loader successfully completes loading the bootstrap file into RAM.

### 3.5 Push Buttons

#### 3.5.1 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.

#### 3.5.2 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.

### 3.6 Slide Switch

The slide switch is inactive on the board with the factory GALs installed. The switch is intended to be used in an application specific manner. The user may burn GALs to implement the switch.

## **4 JUMPER CONFIGURATIONS**

### **4.1 BTLDIS - Bootstrap Enable Jumper**

Only one jumper across BTLDIS (JP10) is needed to use the RMB-166 in the default 64K/256K RAM and no EPROM mode. This jumper may be found in the 2 x 6 block of headers next to U11. This jumper connects the NMI# input to the bootstrap circuitry. Removing JP10 physically prevents the processor from entering the bootstrap mode.

### **4.2 ROM - EPROM Enable Jumper**

To enable the EPROMs on the board, Jumper 13 - ROM, must be inserted in the 2 x 6 block of jumpers. See Section 6 for details.

### **4.3 AUX1, AUX2, MON and BTL Jumpers**

The AUX1, AUX2, MON and BTL jumpers located on the 2 x 6 header next to U11 are not used. The board is designed to so these jumpers may be implemented for custom use by changing the GAL programs.

### **4.3 JP4, JP5 - Serial Port Jumpers**

Serial ports are driven by U1, an RS-232 driver. Serial port 0 transmit and receive signals (P3.10 and 3.11) are connected to channel 1 of U1. Serial port 1 transmit and receive lines (P3.8 and P3.9) may be used to drive the second channel of U1 or be used as general purpose digital input output ports. Jumpers JP4 and JP5 connect serial port 1 transmit and receive signals to channel 2 of U1, respectively. Once jumpers JP4 and JP5 are inserted into the headers, RS-232 level signals of serial port 1 are available through P2 and the three of the connectors of JP3. The serial port 1 signals are marked G, T, and R, corresponding to the ground, transmit, and receive signals. These three lines are denoted by S1 on the silkscreen.

Note that JP3 provides access to all serial port signals. JP3 is intended for embedded uses of the RMB166 when P1 and P2 are not populated.

### **4.4 JP8, JP9 - Analog-to-Digital Converter Reference**

The analog-to-digital converter requires a ground and reference voltage. These reference voltages may be provided either from JP2 lines marked VG (ground reference voltage) and VR (reference voltage), or connected to the +5 volt TTL supply. Jumpers JP8 and JP9 select the source of reference voltages. The center posts of JP8 and JP9 are connected to the 80C166 VAREF and VAGND inputs. Post 1 of JP8, marked VCC is connected to the +5 volt supply. Thus, connecting this post with the center post selects VAREF to be the same as the +5 volt supply. In the alternate position, VAREF is to be supplied from JP2 from the terminal marked VR.

Similarly, the post marked GND of JP9 is connected to the ground of the supply. Connecting the center post of JP9 with the post marked GND selects VAGND to be the same as the ground of the supply voltage. In the alternate position, the ground reference is to be supplied from the JP2 terminal marked VG.

## 5 HEADERS

The RMB-166 board has two headers: the system header JP1 and the input output port header JP2. JP1 contains the address, data and control busses. Ports 2, 3, and 5 are available on JP2. 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.

### 5.1 Serial Headers

Pins S1I and S1O at header positions 16 and 18 are connected to the input and output of Serial Port 1 after the MAX232 level converter. These signals are identical to those on P2 and JP3. The jumpers JP4 and JP5 must be inserted to use S1I and S1O as RS232-level signals.

### 5.2 JP1

#### JP1 - System Header

Signal	Pins		Signal
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#
RSTOUT#	43	44	BHE#
NMI#	45	46	
	47	48	
	49	50	

Note that pins 46-50 are not connected to any signals.

### 5.3 JP2

JP2 - Input/Output Header

Signal	Pins		Signal
Ground	1	2	VCC (+5V)
Ground	3	4	VCC (+5V)
P5.0	5	6	P5.1
P5.2	7	8	P5.3
P5.4	9	10	P5.5
P5.6	11	12	P5.7
P5.8	13	14	P5.9
VAGND	15	16	S1I
VAREF	17	18	S1O
P2.0	19	20	P3.0
P2.1	21	22	P3.1
P2.2	23	24	P3.2
P2.3	25	26	P3.3
P2.4	27	28	P3.4
P2.5	29	30	P3.5
P2.6	31	32	P3.6
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
P2.15	49	50	P3.15

## 6 MEMORY BLOCK OPTIONS

There are two blocks of memory available on the RMB-166 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.

### 6.1 Default Memory Setting

The default memory map assumes 64K of RAM and up to 64K EPROM. In this configuration, the board is bootstrapped and run. Except for the bootstrap disable jumper BTLDIS (JP10), no other jumpers are needed when using RAM. Programs are downloaded into RAM and then run. To use EPROMs on the board see below.

### 6.2 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 U6 and U7. 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 C166 may be programmed to insert wait cycles during external memory access. However, in order to run the SAB C166 at its full potential of 40MHz, the RAMs should be rated at 70 nano seconds or faster.

### 6.3 ROM Memory Options

The RMB-166 accepts 27C256-type EPROMs in the ROM memory location. Two chips are needed, one for EVEN and the other for ODD addresses. These chips are placed in 28-pin sockets marked U9 and U10. The EPROMs must have the program burned into them using a separate EPROM burner. To enable the EPROMs on the board, Jumper 13 - ROM, must be inserted in the 2 x 6 block of jumpers. The RMB-166 will not burn the EPROMs. Using EPROMs in the ROM memory location is time consuming, but due to the low cost of the EPROMs it is an inexpensive method of adding non-volatile memory to the board. EPROMs have one large advantage over battery-back RAMs, FLASH, 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. Also any variables in the program must be put into RAM since the EPROM program cannot be changed in circuit. Once the program is debugged it may be burned into the EPROMs using a stand alone EPROM burner.

The SAB C166 may be programmed to insert wait cycles during external memory access. However, in order to run the SAB C166 at its full potential of 40MHz, the EPROMs should be rated at 70 nano seconds or faster.

## 6.4 Downloading and Running Programs in EPROM

### Programming the EPROMs

The EPROMs must have the program burned into them using a separate EPROM burner. 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.

- With the power disconnected populate U9 and U10 with the pre-programmed EPROMs.
- Connect the power to the board.
- Bootstrap the board as usual.
- Insert JP13 to select the alternate memory map with ROM block in low memory.
- Select the **TTY | Run** menu item and type the program's starting address at the address field.
- Press **OK** to run the program.
- 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.**

- With the power disconnected populate U9 and U10 with the pre-programmed EPROMs.
- Remove BTLDIS jumper. This prevents the processor from entering the bootstrap mode when the reset button is pressed.
- Insert JP13 to select the alternate memory map with ROM block in low memory.
- Connect the power to the board.
- The program will start from address zero.
- If the board is powered-up you can restart your program by pressing the reset button.

## 7 PAL Equations

Two PALCE16V8 are used. The first U4 is responsible for the bootstrap loader logic, and the second U11, for the memory decode logic. Many memory decoding schemes are possible by reprogramming the PAL devices. Please contact RIGEL Corporation for more information.

### 7.1 U4 Equations

```
;U4 Bootstrap Loader Logic
;----- PIN Declarations -----
; --- inputs ---
PIN 2      ALE      COMBINATORIAL ;
PIN 3      RSTIN_   COMBINATORIAL ;
PIN 4      RSTOUT_  COMBINATORIAL ;
; --- outputs ---
PIN 12     BTLEN    COMBINATORIAL ;
PIN 13     LED_     COMBINATORIAL ;
PIN 14     PLUG_    COMBINATORIAL ;
PIN 15     PLUG_    COMBINATORIAL ;

PIN 16     NCO      COMBINATORIAL ;
PIN 17     NC1      COMBINATORIAL ;
PIN 18     NC2      COMBINATORIAL ;
PIN 19     NC3      COMBINATORIAL ;
;----- Boolean Equation Segment -----
EQUATIONS
PLUG = /(RSTIN_ * PLUG_)
PLUG_ = /(RSTOUT_ * PLUG)
BTLEN = (/ALE * PLUG)
LED_ = PLUG
NCO=1
NC1=1
NC2=1
NC3=1
```

### 7.2 U11 Equations

```
;PALASM Design Description
; !!! 0-7FFFH RAM / EPROM SELECTED BY JP13 (ROM) !!!
; !!! INSERT JP13 TO ENABLE EPROM !!!
CHIP _r166u11 PALCE16V8 ; RMB-166 U11

; --- PIN Declarations -----
; --- inputs ---
PIN 1      A0      COMBINATORIAL ;
PIN 2      A15     COMBINATORIAL ;
PIN 3      A16     COMBINATORIAL ;
PIN 4      A17     COMBINATORIAL ;
PIN 5      BHE_    COMBINATORIAL ;
PIN 6      BTL     COMBINATORIAL ;
PIN 7      EVA     COMBINATORIAL ; not used
PIN 8      EPROM   COMBINATORIAL ;
PIN 9      ESEG0   COMBINATORIAL ; not used
PIN 11     ESEG1   COMBINATORIAL ; not used
PIN 19     MON_USER COMBINATORIAL ; not used
PIN 18     PLUG    COMBINATORIAL ; not used
; --- outputs ---
PIN 12     RAMSELL_ COMBINATORIAL ;
PIN 13     RAMSELH_ COMBINATORIAL ;
PIN 14     ROMSEL_  COMBINATORIAL ;
PIN 15     MA15     COMBINATORIAL ;
PIN 16     MA16     COMBINATORIAL ;
```

PIN 17            MA17            COMBINATORIAL ;

; --- Boolean Equation Segment ---

EQUATIONS

RAMSELL\_ = A0 + ( /EPROM \* /A15 \* /A16 \* /A17 )

RAMSELH\_ = BHE\_ + ( /EPROM \* /A15 \* /A16 \* /A17 )

ROMSEL\_ = EPROM + A15 + A16 + A17

MA15 = A15

MA16 = A16

MA17 = A17

;------

**MA15** = **A15**

**MA16** = **A16**

**MA17** = **A17**



## 8 Bootstrapping

The 80C166 bootstrap loader is invoked by the following sequence of signals after a hardware reset:

1. Pull ALE high
2. Activate the non-maskable interrupt by a high to low transition

These two signals are generated by the RMB-166 hardware. The ALE is connected to a 1K pull-up resistor. Upon reset, while the ALE is sampled, it is read by the microcontroller to be at logic level high. Next, the microcontroller configures the ALE as an output. The ALE is driven low, to be pulsed high for address latches. The RMB-166 logic detects this high to low transition (input to output configuration) of ALE and uses this signal to drive NMI# (non-maskable interrupt) to the logic level low. The RMB-166 logic also inspects the state of RSTIN# and RSTOUT#. The activation of RSTIN# also triggers the events described above. Some code is downloaded to the microcontroller during bootstrap. This code contains an EINIT instruction. The execution of EINIT activates the RSTOUT# signal. The RMB-166 logic uses this signal to disable further bootstrap load operations. That is, disable the activation of NMI# every time ALE is low.

The RMB-166 logic which performs the bootstrap load operation is embedded in the PAL equations of U4. (Refer to the PAL equations below).

Note that the PAL 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 altering the PAL equations.

Once the bootstrap loader is invoked the serial port S0 is used to communicate with the 80C166. The host must first send a 0 byte with 8 data bits, 1 stop bit and no parity bits. The 80C166 responds with the byte 55h (the ASCII character 'U'). 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 80C166 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
W0:   mov     R0, #0fa60h
      jnb    SORIR, W0
      movb   [R0], SORBUF
      bcl r  SORIR
      movb   SOTBUF, [R0]
      cmpi  1 R0, #0fcbb ; read 604 bytes
      jmp r  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, executes the NOP operations and then starts 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 80C166, a lot of functionality can be implemented within 604 bytes of code. The 604 bytes contained in the file BTL.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 BTL.DAT contains the bytes downloaded to the RMB-166 board during bootstrapping. The file BTL.SRC contains the source code.

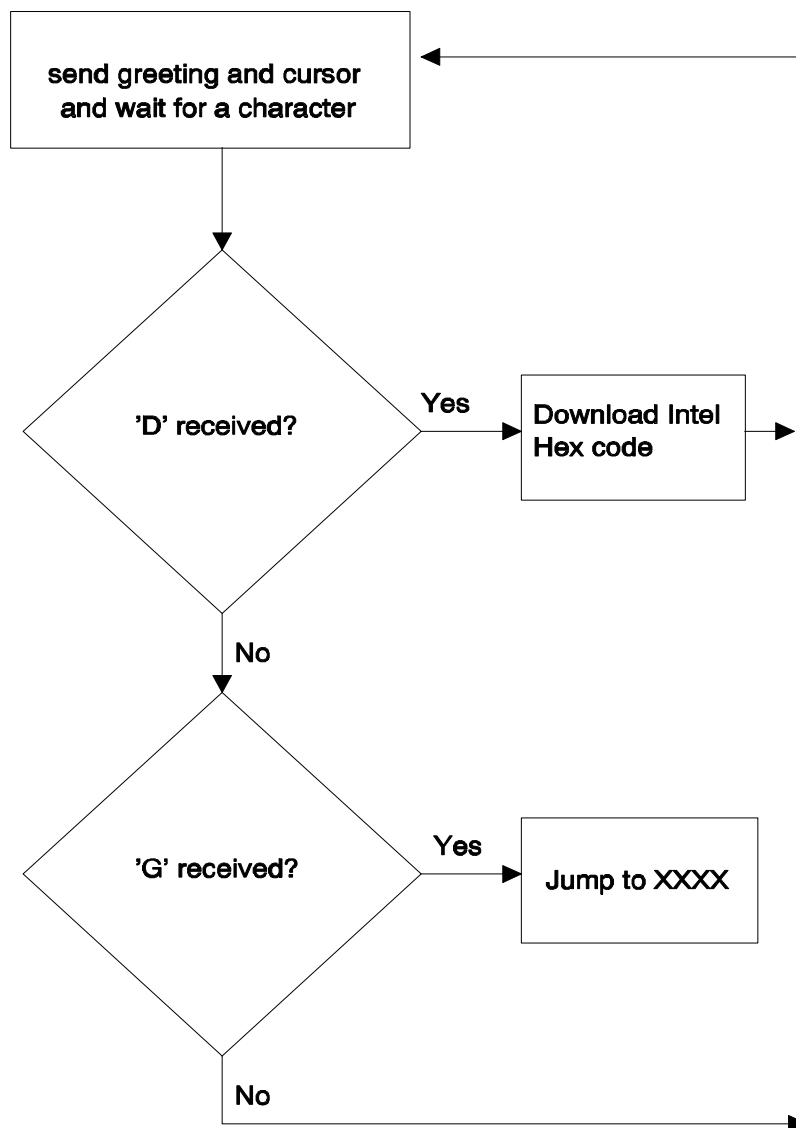
The initialization routines configure the SYSCON register. The internal ROM is disabled and the external bus is activated. Next the CSP and DPP registers are initialized. These steps need to be completed before the EINIT instruction. Note that if the watchdog timer is to be disabled; this too must be done before the EINIT instruction. The final step of initialization consists of configuring port bit 3.13 as an output port. This pin is used as the WR# signal to put data into external RAM.

## 9 THE MONITOR PROGRAMS

### 9.1 The 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 RMON166, 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



## 9.2 RMON166 Monitor

The monitor program RMON166 allows inspecting and modifying the first 64K segment of RMB-166I memory, configuring the ports, inputting and outputting from the general purpose ports, downloading code in the Intel Hex format, and branching to user code. RMON166 features are invoked by single-letter commands. RMON166 assumes a 40Mhz system crystal. Serial port 0 is initialized to run at 9600 Baud with 8 bits of data, 1 stop bit and no parity bits.

RMON166 is intended to be downloaded after bootstrapping the RMB-166I board. RMON166 is placed starting at address 0C000h. The first 256 bytes are reserved for monitor variables. The entry point to RMON166 is at address C000H or C100h. To set up RMON166, initialize READS166 and the RMB-166I board and invoke the **Bootstrap** command as explained in the previous section. From the TTY menu, select **Download** to download RMON166.HEX. Branch to and execute RMON166 using the **Run** command under the TTY menu. Specify address C000H or C100h since the entry point to RMON166 is at 0C000h. Note that RMON166 places a jump to C000H or C100h at the nonmaskable interrupt vector. Thus, RMON166 may subsequently be invoked by pressing the NMI pushbutton on the RMB-166I. RMON166 initializes the stack and resets the interrupts. Thus, even after the NMI button is pressed, RMON166 clears the NMI interrupt by executing a dummy 'return from interrupt' instruction.

Alternatively, RMON166 may be placed in the ROM memory block and invoked upon reset. The source code for RMON166 is given on the distribution disk. RMON166 is not optimized for speed or size, but rather for clarity and pedagogical value. Legal users are encouraged experiment with, make modifications to, or use portions of the RMON166 in their applications.

The single-letter commands of RMON166 are explained below.

### D Download HEX file

The D command places RMON166 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 RMON166 by a branching instruction to address 0C000h. RMON166 initializes the stack, thus, either a jump or a call instruction may be used to return to RMON166.

### H Help

The H command displays a summary of available monitor commands.

### M Memory

The first 64K segment of the RMB-166I 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, RMON166 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.

## 10 PARTS LIST

Quantity	Part	Designator
1	1nF Capacitor	C1
13	10nF Capacitor	C7-C10, C12 - C20
4	1 uF Capacitor	C2, C3, C4, C5
1	47uF Capacitor	C21
2	100uF Capacitor	C6, C11
1	100 Ohm Resistor	R3
3	330 Ohm Resistor	R2, R7, R8
2	1K Resistor	R5, R6
1	2.2K Resistor	R1
1	4.7K Gang Resistor	R9
1	10K Resistor	R4
1	2N2222	Q1
1	1N4001 Diode	D1
1	Red Led	D2
1	Green Led	D3
1	Yellow Led	D4
2	DB9 Connector	P1, P2
2	Push Buttons	S1 S2
1	Slide Switch	S3
2	1 x 2 Headers	ARXD-JP5, ATXD-JP4
1	1 x 6 Header	RS-232 JP3
2	1 x 3 Header	VCC-JP6, VCC-JP7
2	2 x 3 Header	JP8, JP9
1	2 x 6 Header	BTLDIS-JP10, BTL- JP11, EVA-JP12, EPROM-JP13, SEG0-JP14, ESEG1-JP15,
2	2 x 25 Header	JP1, JP2
1	MAX232	U1
1	40MHz Clock	U2
1	80C166 CPU	U3
2	GAL16V8	U4, U11
2	62256 RAM	U5, U8
2	27256 EPROM	U9, U10

## 11 TOP OVERLAY