RMB166-FLI USERS GUIDE

Version 1.01

OCTOBER 1996

RIGEL CORPORATION
PO Box 90040
Gainesville, Florida 32607
(352) 373-4629
FAX (352) 373-1786
http://www.Rigelcorp.com

Our Policy:
We attempt to up date all software, software manuals, and hardware manuals every 3-6 months. If there is a problem with the software or documentation it is corrected immediately. The newest version is then put on our home page (http://www.rigelcorp.com) with the date noted. Documentation is coded with version number, and a date. The version number is the version of the board (V1.0), and the date (September 1996) refers to the last date the document was written.

We welcome any and all comments about our products.

WARRANTY

RIGEL CORPORATION- CUSTOMER AGREEMENT

1. Return Policy. This return policy applies only if you purchased the RMB-16x 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 (325) 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 RMB166-FLI industrial board features the Siemens SAB 88C166 16-bit high-performance microcontroller in the metric plastic quad-flat pack package. The microcontroller has 32K on board FLASH memory and 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 reset and ROM options chosen. The default configuration is the 64K RAM and no ROM mode. In this mode, the monitor or user program is downloaded to RAM using the SAB 88C166 bootstrap feature. A set of option headers, decoded by a GAL device make the RMB166-FLI a flexible hardware platform.

The RMB166-FLI is compatible with the RMB-165i, RMB-166, RMB-166i, RMB-167, RMB-167i and the RMB167-CRI. The board size, the location, and function of all headers are kept the same. The RMB166-FLI has the following improvements over the RMB-166 and RMB-166i.

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. The RMB166-FLI meets and exceeds the European Community Electromagnetic and Electrostatic Compliance requirements.
2. **FLASH Memory Capability.**
   A. 32K on-chip FLASH Memory
   B. Sockets U5 and U6 accept 29F010 FLASH chips from AMD.
   C. The SAB 88C166 FLASH, and the 29F010 chips may be programmed in circuit using the R166FLI software which is supplied with the board.
3. **Higher Memory Map Resolution.**
   The PALCE22V10-type device has access to A13 to A17 in determining the memory map. The memory map has a resolution of 8 Kilobytes.

1.1 Hardware Overview

- High-performance 16-bit microcontroller, the SAB 88C166
  32K on board FLASH memory
  Bootstrap loading feature
  Runs at 40MHz oscillator frequency with zero wait states, (20MHz internal system clock)
  Internal 10-channel, 10-bit Analog-to-Digital converter
  10 bits of Analog / Digital inputs (Port 5)

Two 16-bit general-purpose input/output ports (P2 and P3)
Two serial ports
16 channels CAPture and COMpare unit

- 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 upto 256K of FLASH ROM (not installed)
- Push buttons for RESET# and NMI#
- GAL-decoded memory map for maximum flexibility
- GAL can be reprogrammed by user or by RIGEL Corporation
- Microcontroller is socketed on the board
- Machine screw sockets under all other IC's
- Power on LED
- LED to indicate SAB 88C166 FLASH burn in progress
- Power consumption is less than 200mA running at 40MHz
- Flexible and embeddable 4" x 6" six layer industrial board
- Mounting holes in all four corners

1.2 Software Overview

1.2.1 READS166 Evaluation Software

READS166 V2.X runs in the MS-Windows 3.1 environment. READS166 supports the bootstrap loader feature and downloads a minimal monitor during bootstrapping. The source code and description of the bootstrap program are included in the documentation. READS166 evaluation software includes: a monitor program, an assembler, and a C-compiler.

<u>RMON166 - The READS166 monitor program</u>

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.

<u>Ra66 - The READS166 Assembler</u>

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 2K.

<u>Rc66 - The READS166 C Compiler</u>

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 a 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 2K.

1.2.2 R166FLI Utility Software

R166FLI Utility Software runs in the MS-Windows 3.1 environment.
It supports the bootstrap loader feature and downloads a minimal monitor during bootstrapping.

R166FLI is a utility program to program the internal FLASH memory of the SAB 88C166 and the external FLASH memory chips on the RMB166-FLI board. The RMB166-FLI must be bootstrapped and the special-purpose monitor program downloaded before any other utility is used. The special-purpose monitor was developed with Rigel's integrated development environment READS166. The R166FLI program should be used only when you wish to burn the FLASH memory. Use the READS166 for code development.

1.3 Parts List

Your RMB166-FLI package includes the following:

Hardware

1. RMB166-FLI board with 64K of static RAM.
2. Serial modem cable with adapter

Software

1. RMON166 monitor program with source code.
2. Evaluation version of READS166 for Microsoft Windows.
3. R166FLI Utility Software

Documentation

1. User's Guide with circuit diagrams
2. Bootstrap file source code.
3. Sample programs.

2. QUICK START TUTORIAL

2.1 System Requirements for READS166 and R166FLI

READS166 and R166FLI are designed to work with an IBM PC or compatible, 386 or better, running MS-Windows 3.1 or later.

READS166 / R166FLI use COM1 to COM4 to talk to the RST166-FLI. These ports are driven using the default interrupt request lines assigned by the Windows setup. 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 / R166FLI. 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.

2.2 Software Installation

2.2.1 READS166

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.

2.2.2 R166FLI

Please follow the above directions to install the R166FLI Utility Software.

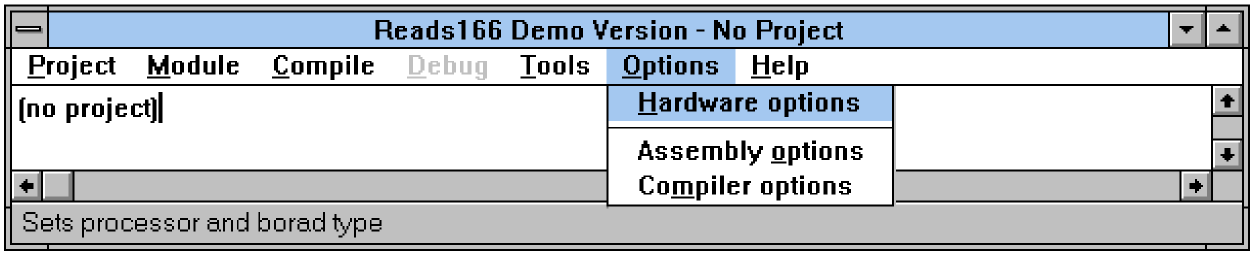
2.3 Start up

1. Connect the RMB166-FLI to a well regulated 5 volt power supply.
2. Connect the RMB166-FLI to the PC host via a serial cable.
3. Check to make sure the bootstrap disable (BTLDIS) and the Bus Active (BUSACT) jumpers are installed. This is the default configuration for the RMB166-FLI 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 READS166 and Initiating Host-to-Board Communications

1. Select the board and processor type using the **Options | Hardware options** menu command. Choose the RMB-166i from the board list which appears.



(no project)

Sets processor and borad type

2. Open the TTY window using the **Tools | TTY** menu command.
3. Select the communication port parameters using the **TTY | Settings** menu command. You will need to select the COM port you are using, and the baud rate. The default configuration for the parameters are as follows: 8 data bits, 9600 baud rate, 1 stop bit and none for the parity bits.

2.5 Bootstrapping

1. Press the reset button on the board and wait 3 seconds.
2. Select item **Tools | TTY** to start the TTY window. Then from the menu in the TTY window select **Bootstrap**. The board will now bootstrap.

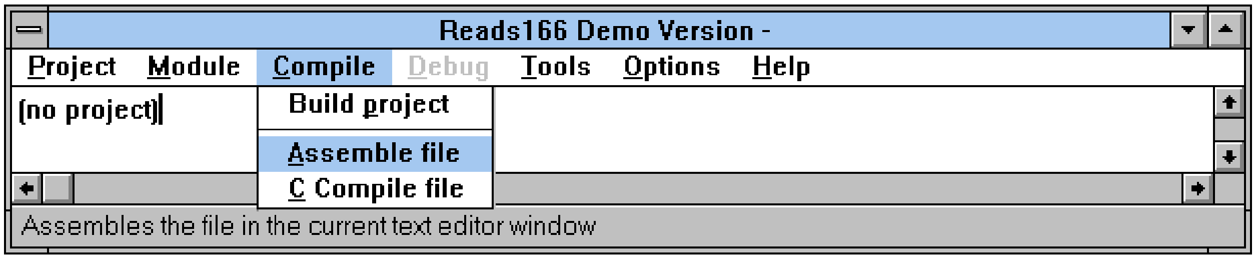
You may observe the bootstrap progress from the status line of the TTY window. The green LED is turned on during bootstrapping, after the EINIT instruction, but before the monitor is downloaded. When bootstrapping is completed, the READS166 monitor prompt appears in the TTY window.

2.6 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.

2.7 Assembling Files

The demo programs which come with the READS166 software will need to be assembled before downloading to the board. Select the menu commands **Compile | Assemble file** to open the screen showing which demo programs are available.



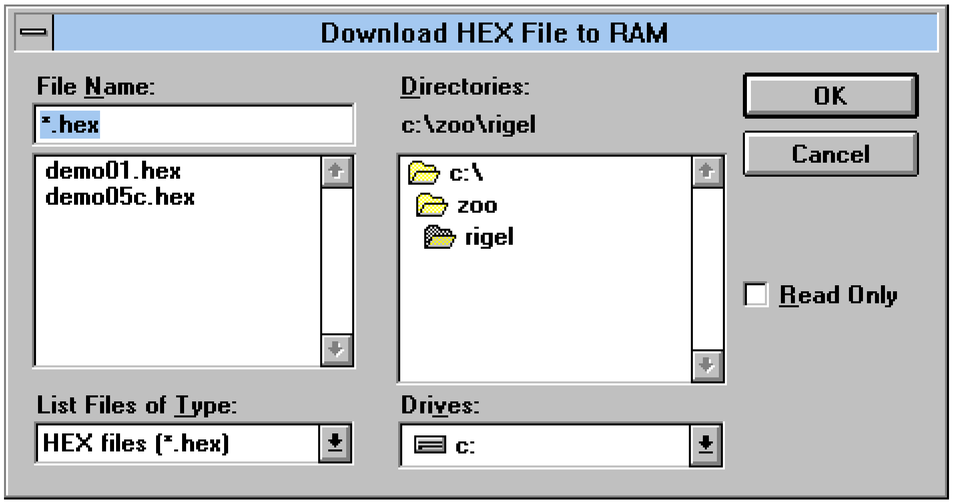
[no project]

Assembles the file in the current text editor window

Select the file demo05C.ASM and press **OK** to assembly the file. A screen will appear which says no errors. Assembling the program produces a HEX file which will be downloaded to the board.

2.8 Downloading Programs

The example program demo05C.ASM repeatedly sends a message to the host in an interrupt driven fashion. Select the **Download | Download to RAM** menu item from the TTY window. A dialog box will open allowing you to select the HEX file you wish to download.



File Name: *.hex

demo01.hex
demo05c.hex

Directories: c:\zoo\rigel

c:\
zoo
rigel

List Files of Type: HEX files (*.hex)

Drives: c:

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 READS166 text editor. First select the menu item **Tools | Text Edit** to open the edit window. Then from the edit menu select the **File | Open** menu command. This will display a list of files which may be opened, edited, and saved.

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 window **Run** menu item. The default address of ***4000*** will appear in 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

You can get more information about READS166 from the help system. To access the help system select **Help | Contents** from the main menu. Once in the Help System, select the topic you are interested in for more information.

3. OPERATING NOTES

The following block diagram of the RMB166-FLI board shows placement of the LEDs, jumpers, headers, switches, and major ICs. It is not drawn to scale and does not show all the components on the board. For a complete top overlay of the board please see Section 13.

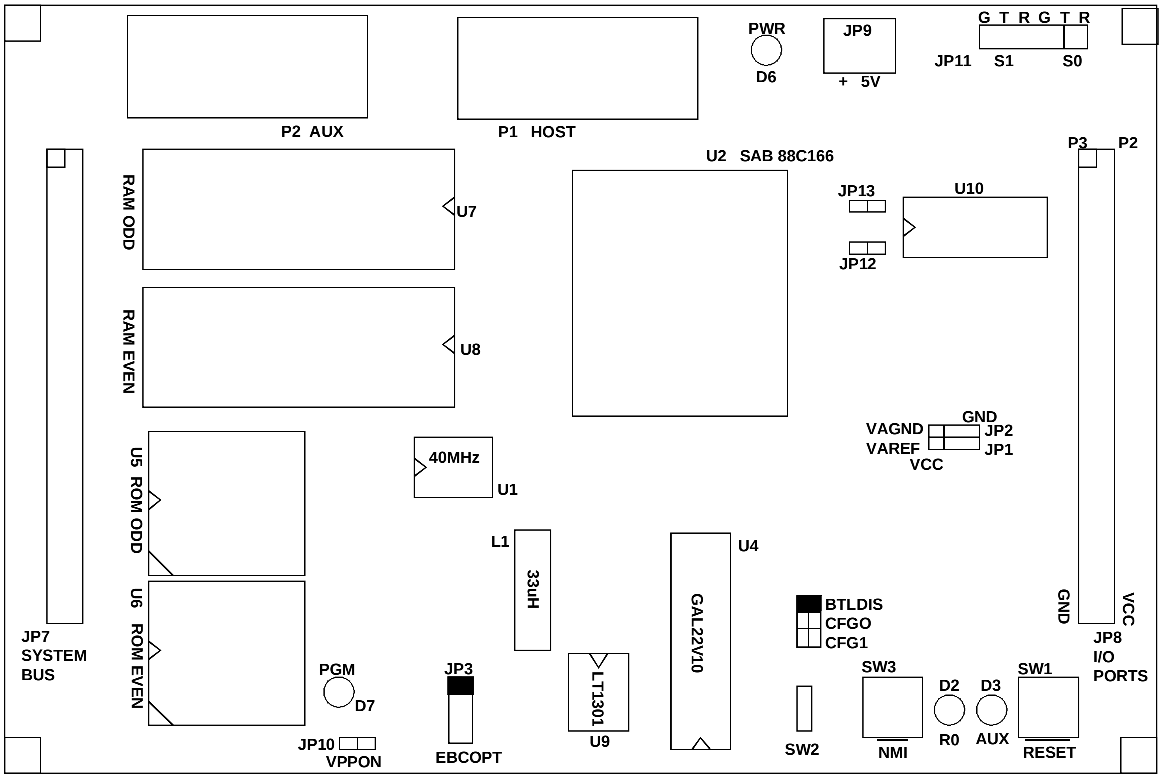


Figure 3.1 Board Layout

The RMB166-FLI needs two connections: to a power supply and to the serial port of a host via a modem cable.

3.1 JP9, Power

Power is brought to the RMB166-FLI board by a two-position screw-type terminal block. A well regulated 5V DC (+/- 5%) source is required. The (+) and (-) terminals are marked on the board.

Note

A +5 volt regulated power supply must be used with the RMB166-FLI board. Lower voltage will not operate the board. Higher voltage will irreversibly damage the active components on the board. An unregulated power supply may cause unpredictable failure conditions. Always check that the power supply is plugged into the board correctly. A diode is placed across the input in reverse. Thus if the power is applied to the

RMB166-FLI board in reverse polarity, the diode will short the power supply attempting to prevent damage to the board.

3.2 Serial Ports

The RMB166-FLI board has two serial ports which may be accessed from the DB9 female connectors labeled P1 (HOST), P2 (AUX), and from the header labeled JP11.

3.2.1 P1, Host

P1, HOST is used to connect the board to an IBM compatible PC. Serial port 0 transmit and receive signals (P3.10 and 3.11) are connected to one channel of a RS-232 level converter. A minimal serial port is constructed on the board with just the 3 lines: transmit, receive, and ground, disregarding all hardware handshake signals. A straight-through modem cable is used to connect with the PC Host. That is a cable connecting pin 2 of the RMB166-FLI 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.

3.2.2 P2, Aux

Serial port 1 transmit and receive lines (P3.8 and P3.9) are connected to the DB9 connector P2. The default configuration of the board uses these lines as general purpose digital input/output ports. To use P3.8 and P3.9 for the second serial port, jumpers must be installed into the headers labeled JP12 and JP13. Once the jumpers are inserted into the headers, RS-232 level signals of serial port 1 are available through the DB-9 connector P2, as well as three of the connectors of JP11.

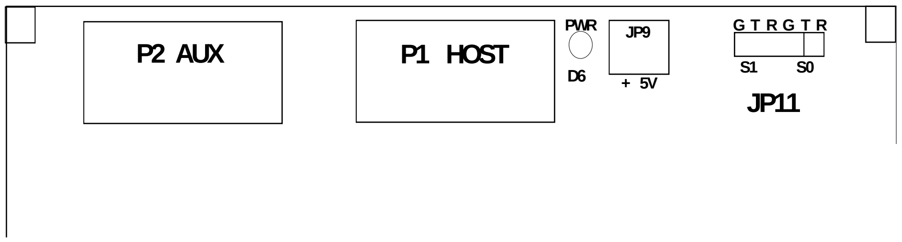


Figure 3.2 Serial Port Connectors

3.2.3 JP11

JP11 provides access to the serial port signals. JP11 is a 6-pin header, which carries the same signals as P1 and P2. JP11 is also denoted by S0 / S1 and its 6 lines by G (Ground), T (Transmit), and R (Receive) on the RMB166-FLI silk-screen. JP11 is intended for embedded uses of the RMB166-FLI when P1 and P2 are not populated.

3.2.4 Troubleshooting the Serial Connection

If after applying power to the board. you are unable to communicate with it using the READS166 software, check the following items:

1. Make sure that the PC serial port is available. See section 2.1 for details.
2. Make sure the cables between the board and the PC, and the power supply and the board, are making good connections.
3. Make sure you are using a straight through serial cable. Do not use a null modem cable.
4. Make sure the READS166 software is installed properly, and that you have selected the correct processor from the menu.
5. Check to make sure that the jumpers for BTLDIS and BUSACT are inserted.
6. After checking these items press the reset button (SW3) on the evaluation board, wait a few seconds, and try to run the software again.
7. Some third party software is designed to be invoked using the NMI or RESET signals. Pushing the RESET or NMI button on the board should generate the correct signals to operate the software. Refer to the third party software for further information.

3.3 LEDs

The RMB166-FLI has four LEDs which provide various information about the board's status.

3.3.1 D6, Power

The red LED, D6, when lit, shows power is connected to the board. If this LED does not light up when power is applied; check for the power at the wall socket, check to make sure you have a good connection at the terminal block, make sure that your power supply is a well-regulated 5 volts, and that it is not plugged into the terminal block backwards.

3.3.2 D2, Reset Out

The green LED, D2, 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.

3.3.3 D3, Auxiliary

The yellow LED, D3, 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 ROM. In the default configuration the yellow LED is nonfunctional.

3.3.4 D7, Voltage Pump

The fourth LED, D7, next to JP10 is lit when the voltage pump is generating the 12V output.

3.4 Push Buttons

3.4.1 Reset (SW1)

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 (SW3)

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).

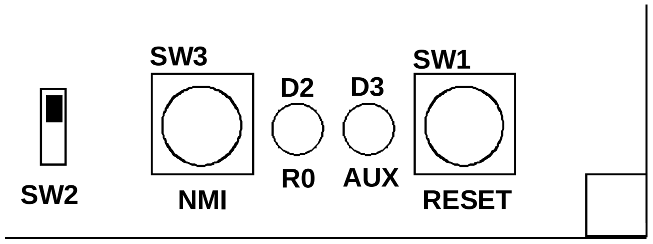


Figure 3.3 Switches

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 Slide Switch (SW2)

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

4. JUMPER CONFIGURATIONS

4.1 Default Jumper Selection

Only two jumpers, across BTLDIS and BUSACT, are needed to use the RMB166-FLI in the default 64K/256K RAM and no ROM mode. Both these jumpers will be installed when shipped from the factory.

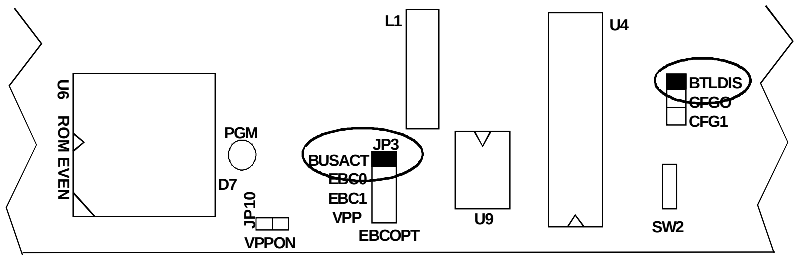


Figure 4.1 Default Jumpers

4.2 BTLDIS

This jumper connects the NMI# input to the bootstrap circuitry. Removing BTLDIS physically prevents the processor from entering the bootstrap mode.

4.3 CFG0 / CFG1

CFG0/ CFG1 are used to select the external FLASH memory mapping. There are two default memory maps for the external FLASH chips. The first is, 48K FLASH and the rest RAM. The second is with 32K FLASH and the rest RAM. By inserting a jumper in CFG1, the first 48K of memory is fetched from external ROM. This allows for the programming of the FLASH chips using the R166FLI Utility Software provided with the board.

	CFG0	CFG1
Default , No FLASH, all external RAM	NO JUMPER	NO JUMPER
First 48K external FLASH, rest is external RAM	NO JUMPER	JUMPER
First 32K external FLASH, rest is external RAM	JUMPER	JUMPER

Inserting a jumper in both CFG1 and CFG0 selects the first 32K of memory to be fetched from external ROM. This configuration can be used to mimic the internal FLASH environment with a ROM-less microcontroller. See Section 5.3 for more information.

4.4 BUSACT / EBC0 / EBC1 / VPP

The external bus configuration and the internal ROM use are determined at reset before the end-of-initialization instruction. There are three processor pins BUSACT#, EBC0 and EBC1, involved in selecting the bus configuration at reset. These pins are brought to the jumper JP3. JP3 has four pairs of posts. The first two are connected to BUSACT# and EBC0. Inserting a jumper into the corresponding position grounds the associated pin. The EBC1 pin of the microcontroller is connected to VCC by a 10K pull-up resistor. EBC1 is brought to the bottom two pins of the jumper JP3. The position of JP3 marked EBC1 is used to connect EBC1 to GND (ground or 0 Volts). The bottom position of JP3 marked VPP is used to connect EBC1 to VPP. The following table is a partial list of the most often used reset options.

Reset Options

	BUSACT#	**EBC0**	**EBC1**	**VPP**
INTERNAL MEMORY DISABLED	JUMPER	NO JUMPER	NO JUMPER	NO JUMPER
INTERNAL ROM ENABLED	NO JUMPER	JUMPER	JUMPER	NO JUMPER
PROGRAM ON-CHIP FLASH	JUMPER	NO JUMPER	NO JUMPER	JUMPER

NOTE

Care must be taken not to populate the two positions, EBC1 and VPP at the same time. Populating both EBC1 and VPP will cause the output of the voltage pump to be grounded.

For more details on the external bus configuration please see the Siemens SAB 80C166 Data Book.

4.5 Serial Port Jumpers and Header

The serial ports are driven by U10, an RS-232 driver. Serial port 0 transmit and receive signals (P3.10 and 3.11) are connected to channel 1 of U10. Serial port 1 transmit and receive lines (P3.8 and P3.9) may be used to drive the second channel of U10 or be used as general purpose digital input/output ports. The default configuration of the board uses these lines as I/O lines.

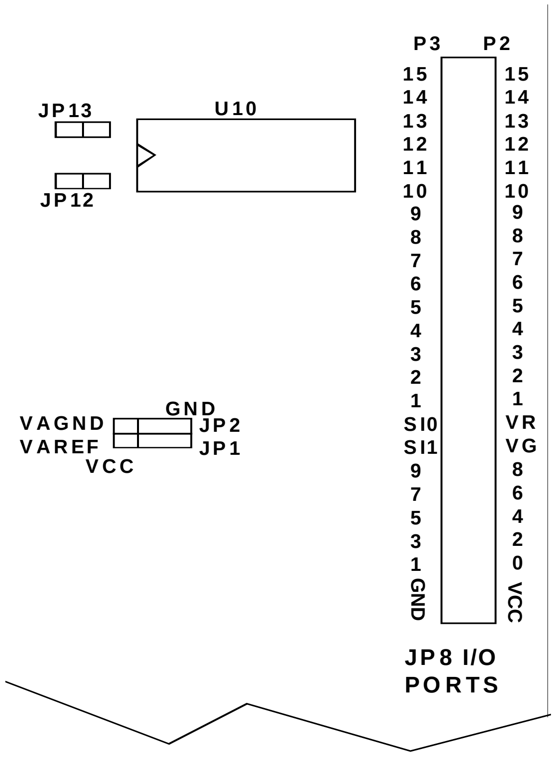

P3
P2
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 SI0 SI1 9 7 5 3 1 GND
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 VR VG 8 6 4 2 0 VCC
JP13
U10
JP12
GND
VAGND
VAREF
VCC
JP2
JP1
JP8 I/O PORTS

Figure 4.2 Serial Port and A/D Jumpers

4.5.1 JP12 and JP13

Jumpers JP12 and JP13 connect serial port 1 transmit and receive signals to channel 2 of U10. Once jumpers JP12 and JP13 are inserted into the headers, RS-232 level signals of serial port 1 are available through the DB-9 connector P2, as well as three of the posts of JP11.

4.5.2 JP11 (S0 / S1)

JP11 provides access to the serial port signals. JP11 is a 6-pin header, which carries the same signals as P1 and P2. JP11 is also denoted by S0 / S1 and its 6 lines by G (Ground), T

(Transmit), and R (Receive) on the RMB166-FLI silk-screen. JP11 is intended for embedded uses of the RMB166-FLI when P1 and P2 are not populated.

4.5.3 SI0 and SI1

Pins S1I and S1O on header JP8 are connected to the input and output of Serial Port 1 after the MAX232 level converter. These signals are identical to those on JP11. The jumpers JP12 and JP13 must be inserted to use S1I and S1O as RS232 level signals.

4.6 Analog-to-Digital Converter Reference Jumpers

The analog-to-digital converter requires a ground and reference voltage to operate.

4.6.1 JP8 (VG and VR)

The reference voltages may be provided either from JP8 lines marked VG (ground reference voltage) and VR (reference voltage), or connected to the +5 volt TTL supply.

4.6.2 JP1 and JP2

Jumpers JP1 and JP2 select the source of reference voltages. The center posts of JP1 and JP2 are connected to the SAB 88C166 VAREF and VAGND inputs. Post 1 of JP1, 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 JP8 from the terminal marked VR. Similarly, the post marked GND of JP2 is connected to the ground of the supply. Connecting the center post of JP2 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 JP8 terminal marked VG.

4.7 JP10, VPPON

The RMB166-FLI board contains a voltage pump to generate a well-regulated 12V from the 5 Volt supply. The voltage pump, built around the LM1301 chip is activated by inserting a jumper at JP10, designated as VPPON on the board. The LED (D7) next to the jumper is lit when the voltage pump is generating the 12V output. (Note that the output of the voltage pump is 5V when the jumper is removed.) Although the voltage pump should probably be disabled when not programming or erasing the FLASH memory, leaving it on will not harm the board.

5. MEMORY BLOCK OPTIONS

There are three blocks of memory available on the RMB166-FLI board. There is the on-chip 32K (internal) FLASH block of memory, there is a RAM block which may hold upto 256K of memory, and there is a ROM block which may hold up to 256K of (external) FLASH memory.

5.1 On-Chip FLASH Memory

The SAB 88C166 has 32K of on-chip FLASH memory. The RMB166-FLI board is designed to provide the 12 volts needed to program the FLASH. The RMB166-FLI is sold with software, the R166 Utility Software, especially written to program the FLASH. The on-chip FLASH has 4 memory banks of 8K each, which may be programmed separately. Please refer to the SAB 88C166 data book for details of the FLASH memory. See Section 10, for directions on how to program the FLASH using the R166FLI Utility Software.

5.2 RAM Memory Options

The RAM block of memory is designed to take static RAMs, either 32K 62C256-type, or 128K 681000-type static RAM chips. Alternately 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 the 32-pin sockets marked U7 and U8. Place 28-pin RAM devices closer to the 2 X 25 header, away from the processor. Programs may be downloaded to the RAMs and run, using the READS166 software which comes with the board. Alternatively you may use third party software to download and run programs on the board.

The SAB 88C166 may be programmed to insert wait cycles during external memory access. However, in order to run the SAB 88C166 at its full potential of 40MHz, the RAMs should be rated at 70 nano seconds or faster.

5.3 ROM Memory Options

The ROM block of memory accepts industry-standard 29F010 Flash EEPROMs providing 256K ROM. Two chips are needed, one for EVEN and the other for ODD addresses. These chips are placed in 32-pin PLCC sockets marked U5 and U6. The chip enable signals for the ROM are generated by the 22V10 GAL. A variety of memory maps are achievable simply by modifying the equations and reprogramming the GAL. The GALs are programmed with two default memory maps which are activated by inserting jumpers into CFG0 and CFG1. When CFG1 is populated with a jumper, the first 48K of memory is fetched from the external ROM. If both jumpers, CFG0 and CFG1 are inserted, the first 32K of memory is fetched from external ROM. In either case, the remainder of the memory is mapped into RAM. Thus, with both CFG0 and CFG1 populated, the memory map is identical to the case of using the evaluation board with the 32K internal FLASH memory. This configuration is useful in simulating the environment with a ROM-less microcontroller.

The industry-standard external Flash EEPROMs are placed into a write mode by writing to addresses around 5555h and AAAAh. These write operations activate an unlock sequence which allows subsequent write operations. If you wish to program the EEPROMs, the R166FLI Utility Software may be used.   See Section 10 for directions.

The default GALs allow the use of upto 48K of external FLASH. The GALs can be reprogrammed to implement a very wide variety of memory maps. Please contact RIGEL Corporation if you have questions about the possibly memory maps, or if you require assistance in programming the GALs.

5.4 Default Memory Setting

The default jumper settings assume 64K of RAM with no ROM. In this configuration, the board is bootstrapped and programs are downloaded into RAM and then run. Except for the BTLDIS, and the BUSACT jumpers, no other jumpers are needed.

6. HEADERS

The RMB166-FLI board has three headers: the input/output port header JP8, the system header JP7, and the serial port header JP11. JP7 contains the address, data and control busses. Ports 2, 3, and 5 are available on JP8. Individual signals of these headers 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. The headers are also labeled on the board to make pin identification easier.

The location of these headers and the signals on the headers remain the same between all of the RMB-16x boards in this series. Any external board designed to plug into the RMB166-FLI board will be pin-to-pin compatible with all of the other RMB-16x boards.

6.1 JP7 - 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.

6.2 JP8 - 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

7. GAL EQUATIONS

A PALCE22V10 is used for the board logic.

```
#TITLE     BDC166
#ENGINEER  SRH
#COMPANY   Rigel Corporation
#REVISION  2
#PROJECT   166-FLI
#COMMENT   10/07/96
#CHIP      U4 - PALCE22V10
```

Settings for CFG0 and CFG1   (0 = no jumper)

```
Normal           0  |  0      “ No ROM, all external RAM
                    |
                    |
First 48K ROM    0  |  1      “ First 48K is external ROM,
                    |         “ rest is external RAM
                    |
First 32K ROM    1  |  1      “ First 32K is external ROM,
                              “ rest is external RAM
  "
  "
  " -------------------- PIN Declarations ---------------------

INPUT A0, A14, A15, A16, A17, BHE_, RSTOUT_, RSTIN_, USERMON,
             ALE, CFG0, CFG1;

OUTPUT RASH_, RASL_, ROSL_, ROSH_, MA16, MA17, PLUG, BTLN, BTLED,
        PLUG_;

  " ------------------- Boolean Equation Segment ---------------

  IF (CFG0 * CFG1) THEN                “ No ROM, all RAM

     RASH_ = BHE_;
     RASL_ = A0;
     ROSH_ = 1;
     ROSL_ = 1;
     MA16  = A16;
     MA17  = A17;

  ELSIF (CFG0 * /CFG1) THEN             “ First 48K is external
                                        “ ROM, rest is RAM.

     ROSH_ = BHE_ + ((A17 + A16 + A15) * (A17 + A16 + /A15 +
            A14));

     ROSL_ = A0 + ((A17 + A16 + A15) * (A17 + A16 + /A15 + A14));


     RASH_ = /(/BHE_ * ROSH_);          “ If it is not external
                                        “ ROM, it is external
RAM
```

```
     RASL_ = /(/A0   * ROSL_);          “ If it is not external
                                        “ ROM, it is external
RAM

     MA16 = A16;

     MA17 = A17;

  ELSIF (/CFG0 * /CFG1) THEN            “ First 32K is ROM, rest
                                        “ is RAM

     ROSH_ = BHE_ + (A17 + A16 + A15);

     ROSL_ = A0   + (A17 + A16 + A15);

     RASH_ = /(/BHE_ * ROSH_);          “ If it is not external
                                        “ ROM, it is external
RAM

     RASL_ = /(/A0  * ROSL_);           “ If it is not external
                                        “ ROM, it is external
RAM

     MA16 = A16;

     MA17 = A17;

  END IF;

  “ Remaining equations implement the bootstrap logic

  PLUG  = /(RSTIN_ * PLUG_);
  PLUG_ = /(/RSTOUT_ * PLUG);
  BTLN  = /ALE * PLUG;
  BTLED = PLUG;
```

8. BOOTSTRAPPING

The SAB 88C166 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 RMB166-FLI 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 RMB166-FLI 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 RMB166-FLI 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 RMB166-FLI logic uses this signal to disable further bootstrap load operations. That is, disable the activation of NMI# every time ALE is low.

The RMB166-FLI logic which performs the bootstrap load operation is embedded in the GAL equations of U4. (Refer to the GAL equation in section 7.)

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 altering the GAL equations.

Once the bootstrap loader is invoked the serial port S0 is used to communicate with the SAB 88C166. The host must first send a 0 byte with 8 data bits, 1 stop bit and no parity bits. The SAB 88C166 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 SAB 88C166 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, 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 SAB 88C166, 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
```

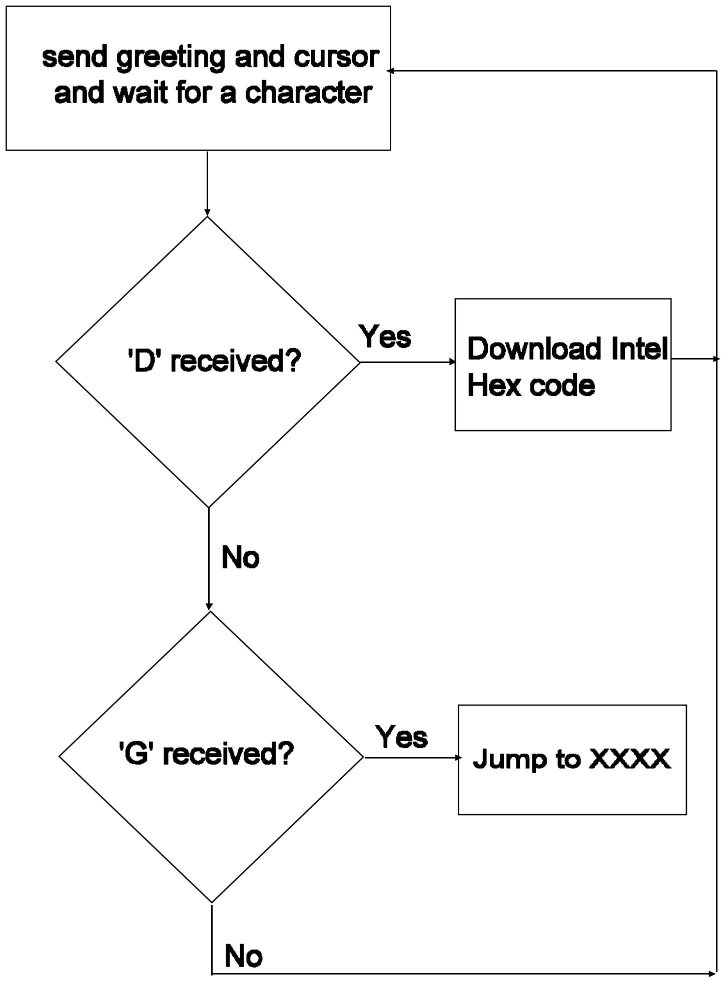
This pattern is used throughout sections 2 and 3. First the word is written to register R1. Then the register is copied to memory. The file BTL.DAT contains the bytes downloaded to the RMB166-FLI board during bootstrapping. The file BTL.SRC contains the source code.

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 above flowchart.

9.2 RMON166 Monitor

The monitor program RMON166 allows inspecting and modifying the first 64K segment of RMB166-FLI 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 RMB166-FLI 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 RMB166-FLI 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 RMB166-FLI. 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 RMB166-FLI 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. R166FLI SOFTWARE UTILITIES

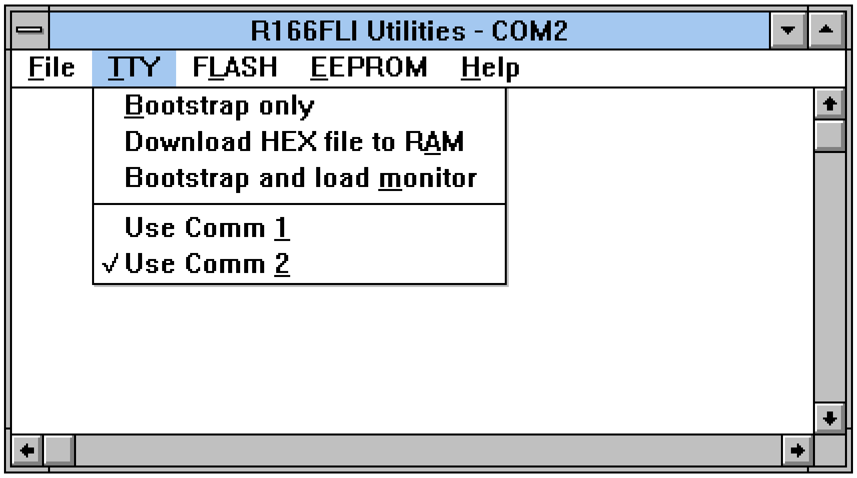
R166FLI is a utility program to program the on-chip FLASH memory and the FLASH memory devices on the RMB166-FLI board. The RMB166-FLI must be bootstrapped and the special-purpose monitor program be downloaded before any other utility is used. The special-purpose monitor was developed with Rigel’s integrated development environment READS166. Many of the functions of the special-purpose monitor are accessible by single-letter commands. Press ’**H’** for a list of single-letter commands the monitor recognizes.

10.1 Programming Internal Flash

Below are step-by-step instructions for using the R166FLI to clear, erase, and program individual words or to download a HEX file into the internal FLASH memory.

10.1.1 Select a COMM Port

Use the **TTY | Use Comm 1** or the **TTY | Use Comm 2** commands to select a communications port. The window title reports the current state of the communications port. The following screen appears showing which comm port was selected.



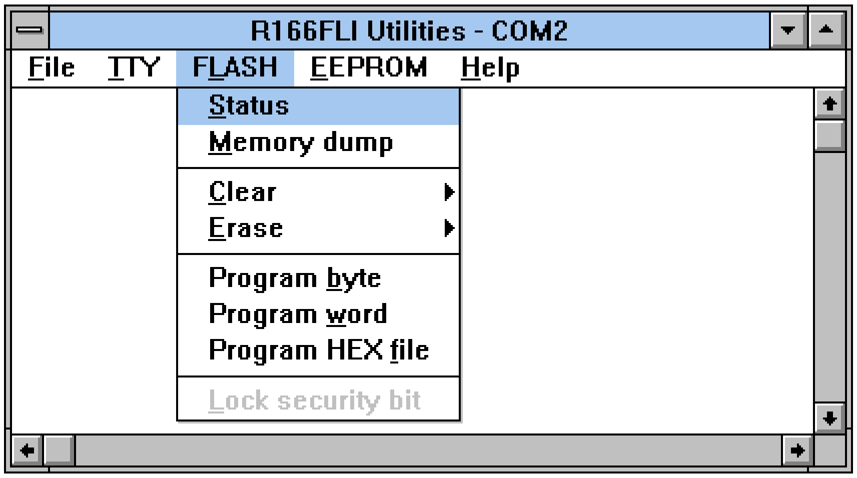
10.1.2 Bootstrapping

Only the two jumpers BTLDIS and BUSACT should be present while bootstrapping and loading the monitor program. Leave these jumpers in place while CLEARing, PROGRAMming, and ERASEing the FLASH memory. Remove the two jumpers, BTLDIS AND BUSACT, to run code from the FLASH memory upon RESET. Use **the "TTY | Bootstrap and load monitor"** command to bootstrap the board. After bootstrapping, the special-purpose monitor program R166FLI.HEX is downloaded to the RMB-166 FLI board. You may verify that the board is responding by pressing the enter key and observing the monitor prompt "10F166 >". You may also type '"**H**" for a brief help screen displayed by the monitor.

10.1.3 Inspect FLASH Status

Use the menu command "**FLASH | Status**" or equivalently, the single-letter command '**S**' to review the current status of the four FLASH banks. Each bank is reported to be in one of three states:

> is CLEARED - all words are 0000
> is ERASED - all words are FFFF
> has DATA - data words are programmed into the FLASH bank

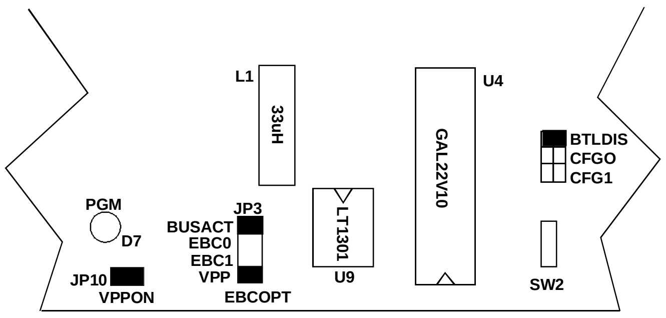


10.1.4 Inspect FLASH Memory

Use the "**FLASH | Memory dump**" menu command to inspect a block of 256 bytes of FLASH memory. Input the block address high byte in the dialog. You may equivalently use the single-letter command '**M**' followed by the address high byte as two hexadecimal digits. For example, "M04" displays the block of memory [0400..04FF].

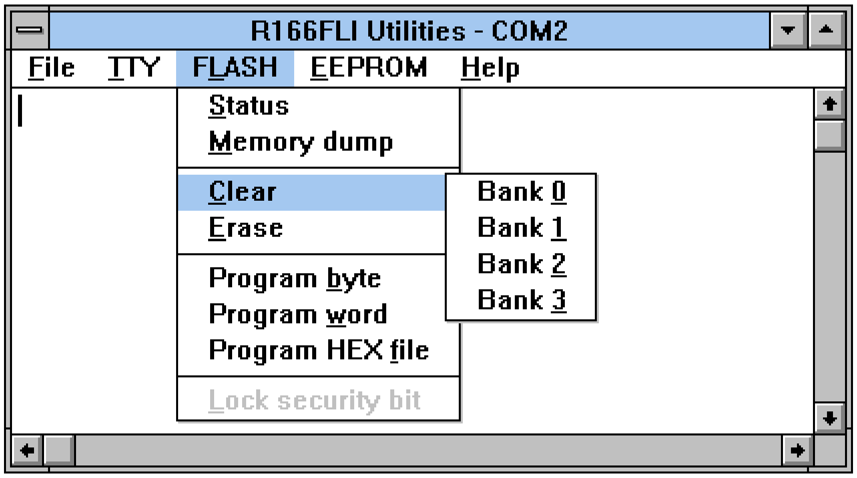
10.1.5 Clear FLASH Bank

Clearing a bank means all words in that bank to 0000. The voltage pump must be turned on by inserting a jumper in JP10, VPPON, and inserting a jumper in VPP on jumper block JP3. The jumpers BUSACT and BTLDIS should remain in place. The LED D7 will light up when the voltage pump is turned on.



10.1 Jumper Settings For Clearing, Programming and Erasing FLASH

Use the "**FLASH | Clear | Bank_n**" menu command to clear any one of the four FLASH banks. The following screen appears allowing you to choose which bank to clear.



10.1.6 Erase FLASH Bank

Erasing a bank means programming all words in that bank to FFFF. The voltage pump must be turned on by inserting a jumper in JP10, VPPON, and inserting a jumper in VPP on jumper block JP3. Inserting a jumper in VPP connects EBC1 to VPP.   The jumpers BUSACT and BTLDIS should remain in place. The LED D7 will light up when the voltage pump is turned on. Use the "**FLASH | Erase | Bank_n**" menu command to erase any one of the four FLASH banks.

Note

You must first clear the bank before you erase the bank. The READS166 demo software will force you to do this.

10.1.7 Program Word

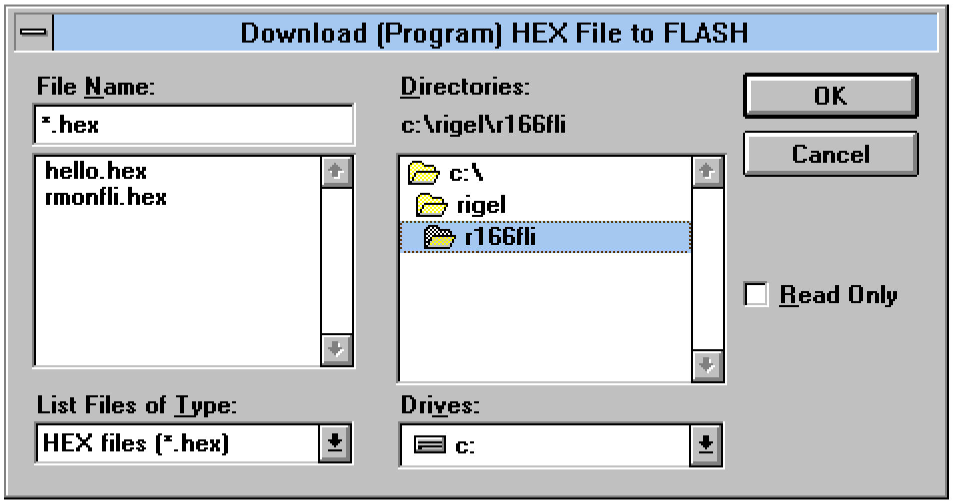
The voltage pump must be turned on by inserting a jumper in JP10, VPPON, and inserting a jumper in VPP on jumper block JP3. The jumpers BUSACT and BTLDIS should remain in place. The LED D7 will light up when the voltage pump is turned on. Use the "**FLASH | Program word**" menu command to program a single FLASH word. Input the address and data in hexadecimal in the dialog box.

You may equivalently use the single-letter command '**B**' followed by the address and word, both as 4-digit hexadecimal numbers. For example, "B00001234" burns the data word 1234 (hex) into the FLASH address 0000.

10.1.8 Program HEX File

The voltage pump must be turned on by inserting a jumper in JP10, VPPON, and inserting a jumper in VPP on jumper block JP3. The jumpers

BUSACT and BTLDIS should remain in place. The LED D7 will light up when the voltage pump is turned on. Use the "**FLASH | Program HEX file**" menu command to download a file into FLASH memory. When this menu item is selected the following screen appears. Input the file name of the HEX file in the dialog box, and press **OK** to program the file to FLASH.


Download (Program) HEX File to FLASH

File Name: *.hex

hello.hex
rmonfli.hex

Directories: c:\rigel\r166fli

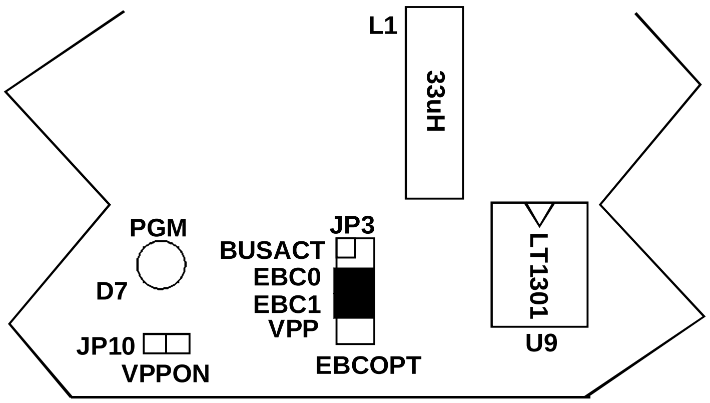
c:\
rigel
r166fli

List Files of Type: HEX files (*.hex)

Drives: c:

10.1.9 Running Code From FLASH Upon Reset

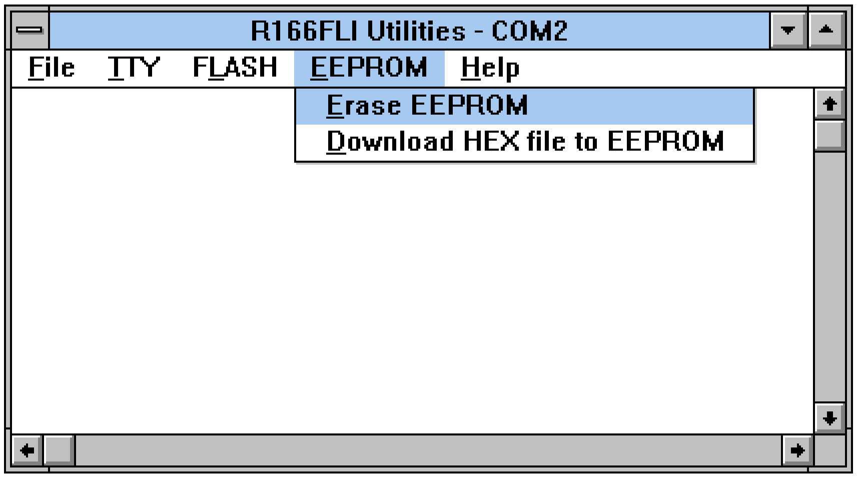
Once an executable program is downloaded into FLASH memory, it may be invoked upon reset. Place only the two jumpers EBC0 and EBC1 on the jumper block JP3. Remove the BTLDIS and BUSACT jumpers. This enables the ROM and maps it to segment 0.



The program must initialize the SYSCON register so that BUSACT is set and that BTYP is 11b. This activates the external bus and selects the 16-bit non-multiplexed external bus configuration used by the RST166-FLI board. Note that the SYSCON register must be modified before the EINIT instruction. Refer to the sample program HELLO.ASM for a demonstration of these steps. You may use HELLO.ASM as a template in developing your own embedded code. HELLO.ASM was written with Rigel's integrated development environment READS166.

10.2 Programming the External FLASH

The industry-standard external Flash EEPROMs are placed into a write mode by writing to addresses around 5555h and AAAAh. These write operations activate an unlock sequence which allows subsequent write operations. If you wish to program the external FLASH chips using the R166FLI Utility Software you will need to place a jumper in the CFG1 position, and leave the jumpers in at the BUSACT and BTLDIS positions. By adding a jumper in the CFG1 position you select the external FLASH memory map of 48K ROM, and the rest RAM.

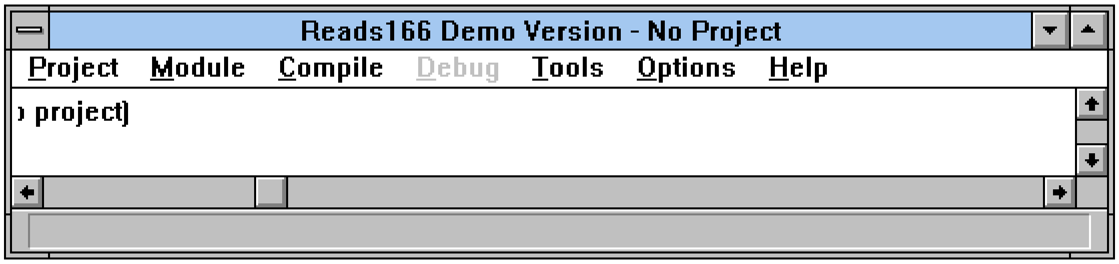


Use the menu option **EEPROM | Download Hex file to EEPROM** or **EEPROM | Erase EEPROM** commands to download or erase the external FLASH memory. The R66FLI Utility Software will only burn 48K of external FLASH.

11. READS166 -- EVALUATION VERSION 2.0

11.1 Overview

READS166 V2.X runs in the MS-Windows 3.1 environment. READS166 supports the bootstrap loader feature and downloads a minimal monitor during bootstrapping. The source code and description of the bootstrap program are included in the documentation. READS166 evaluation software includes: a monitor program, an assembler, and a C compiler.



READS166 V2.X has a more modular look than the previous versions. Although the functionality of the READS166 components remain fully integrated, the user interface has been improved by placing many of the specific commands into sub-menus.

11.1.1 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.

11.1.2 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 2K.

11.1.3 Rc66 - The READS166 C Compiler

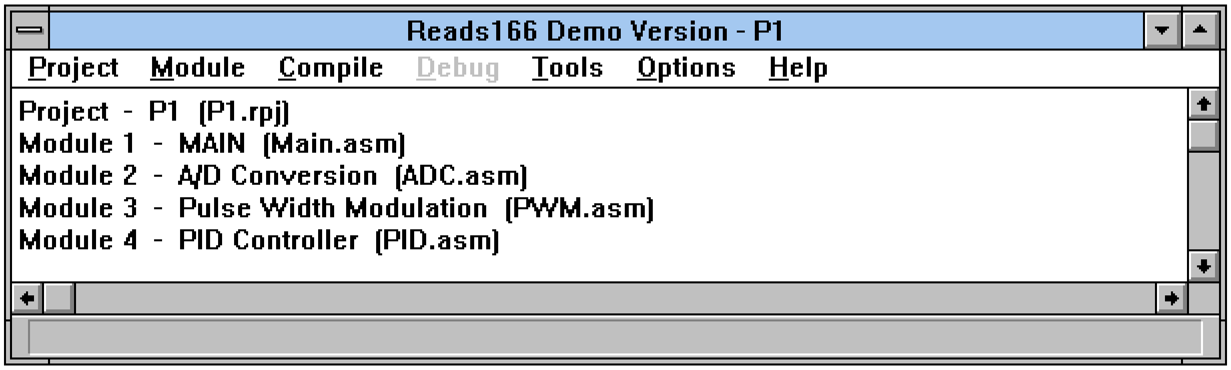
Rc66 is a C Compiler for the C166 family of processors. It compiles code for the tiny memory model which fully resides in the first segment of memory. Rc66 is a designed as a low-cost C compiler which provides a quick development cycle for simpler applications which do not need more than 64K of code, or the use of standard C libraries. Rc66 implements a subset of ANSI C. Currently, structures, unions, enumerated types, and the typedef directives are not implemented. Rc66 in the demo version of READS166 limits the size of code to about 2K. Rc66 works in conjunction with Ra66: first an assembly language program is generated from the C source then a HEX file is created.

11.2 Main Menu Commands

The Main Menu contains the commands for higher-level tasks such as building projects or setting hardware platform options . The major tasks are delegated to the READS166 "**Tools**" which may include editors, host-to-board communications subsystems and code generators. Tools are distinguished by their own environments including sub-menus and accelerator keys. Tools may be minimized when not used or simply closed until needed again.

11.2.1 Project

Projects are collections of source code modules that are compiled as a whole. Use the project menu to create new projects, open existing projects or save projects. Modules written in different languages may be combined in a project.
The use of projects is optional in READS166. It is meant to simplify the bookkeeping of the various components of larger code. For short programs, it is often more practical to simply write the code in the text editor and compile it without first creating a project. (The demo version has limitations on the types and sizes of projects.)


Reads166 Demo Version - P1
Project Module Compile Debug Tools Options Help

```
Project  -  P1  (P1.rpj)
Module 1  -  MAIN  (Main.asm)
Module 2  -  A/D Conversion  (ADC.asm)
Module 3  -  Pulse Width Modulation  (PWM.asm)
Module 4  -  PID Controller  (PID.asm)
```

The Project Window is the space just under the Main Menu. If a project is currently open, a list of modules of the project is displayed. You may resize the Project Window, or use the scroll bars to view the module list. The "**Exit**" command is also under the menu "**Project**" option.

11.2.2 Module

Modules are chunks of code which are combined to constitute a project. An assembly program and an "include" file, for example, may be two modules in a project. READS166 does not require the use of modules. You may "**Create**", "**Edit**", or "**Delete**" modules of the current project using the commands under the "**Module**" option. You may also "**Import**" modules from other (existing) projects. The "**Add Module**" command lets you select assembly or C programs which are currently not a member of any project to be included in the current project.

11.2.3 Compile

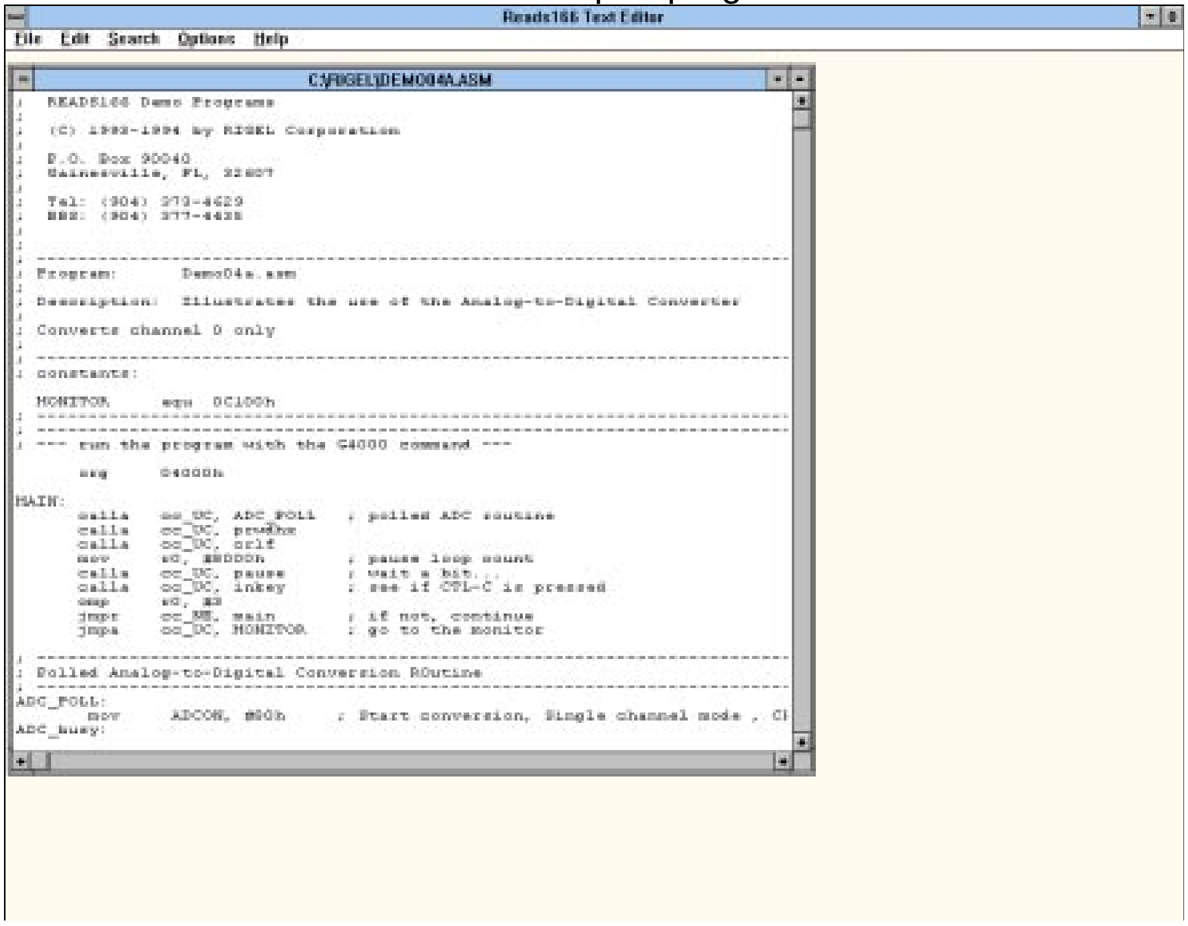
The "**Compile**" menu commands allow you to build the current project, or compile a single file. If the text editor holds a file, this current file is compiled. Otherwise, a dialog box asks for a file to be compiled. Currently, only an assembler and a C compiler are implemented. Other language compilers are being developed.

11.2.4 Tools

Tools are the more powerful subsystems that let you carry out complicated tasks. Tools usually have their own menus and hot-key combinations. Currently there are two tools implemented, the "**Text Editor**," and "**TTY Window**" which is used for communicating with the board.

Text Editor

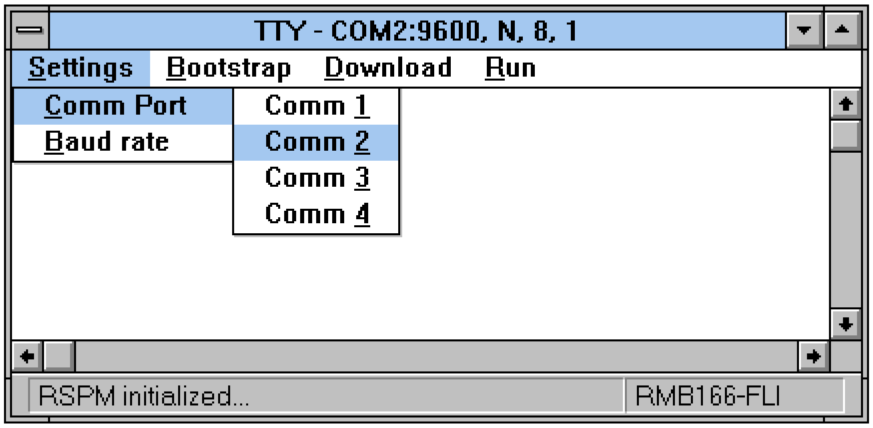
The Text Editor is a MS Windows multi-document interface which holds small text files. The Text Editor has its own menu with the standard File, Edit, and Search commands. The user interface and hot-key combinations are identical to the MS Windows Notepad program.



```
;  READS166 Demo Programs
;
;  (C) 1993-1994 by RIGEL Corporation
;
;  P.O. Box 90040
;  Gainesville, FL, 32607
;
;  Tel: (904) 373-4629
;  BBS: (904) 377-4435
;
;
; -------------------------------------------------------------------------------
; Program:       Demo04a.asm
;
; Description:  Illustrates the use of the Analog-to-Digital Converter
;
; Converts channel 0 only
;
; -------------------------------------------------------------------------------
; constants:

  MONITOR      equ  0C100h
; -------------------------------------------------------------------------------
; -------------------------------------------------------------------------------
; --- run the program with the G4000 command ---

       seg      04000h

MAIN:
       calla    cc_UC, ADC_POLL    ; polled ADC routine
       calla    cc_UC, prwdhx
       calla    cc_UC, crlf
       mov      r0, #0D00h          ; pause loop count
       calla    cc_UC, pause        ; wait a bit...
       calla    cc_UC, inkey        ; see if CTL-C is pressed
       cmp      r0, #3
       jmpr     cc_NE, main         ; if not, continue
       jmpa     cc_UC, MONITOR      ; go to the monitor

; -------------------------------------------------------------------------------
; Polled Analog-to-Digital Conversion ROutine
; -------------------------------------------------------------------------------
ADC_POLL:
        mov      ADCON, #80h      ; Start conversion, Single channel mode , Cl
ADC_busy:
```

TTY Window

The TTY WINDOW encapsulates all host-to-board communications. It has its own menu to set the communications parameters, to bootstrap the board and to download compiled programs into the RAM of the board. The current



RSPM initialized...

RMB166-FLI

communications port and the Baud rate are displayed in the caption (title) of the TTY Window. If the currently selected port is invalid, this condition is also reported in the caption. The Rigel boards use a default of rate of 9600 Baud. Make sure that the serial port you are using is not currently used by another device (such as a serial mouse or a modem card), and that no other peripheral device is competing for the same interrupt as the serial port. You may review or modify the serial port parameters from the MS Windows Control Panel.

Bootstrapping the Board

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 which 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.

Open the TTY Window using the Main Menu command **Tools | TTY**. Verify that the serial port selected is valid as reported in the caption of the TTY Window. Now press the **RESET** button on the board. From the TTY Window menu, select **Bootstrap**. When the board bootstraps, you may press **ENTER** to view the monitor prompt. You may also press **H** (the monitor Help command) to see a short list of available monitor commands.

Monitor Commands

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. This list is also available by issuing the **H** command at the TTY Window.

R E A D S   C O M M A N D S

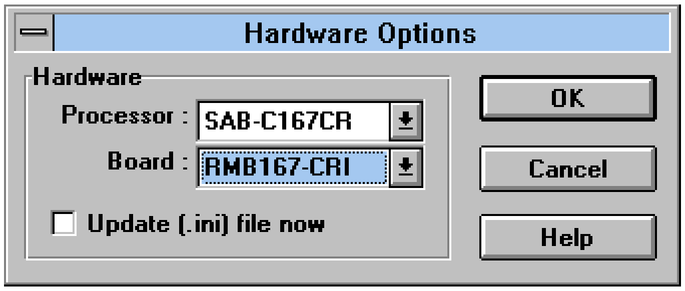
C nn	read port nn Configuration (DPnn)
C nn=mmmm	set port nn Configuration (DPnn=mmmm)
D	Download HEX file
G XXXX	Go, execute code at XXXX
H	Help, display this list
M XXXX	Memory, contents of XXXX
M XXXX=nn	Memory, change contents of XXXX to nn
M XXXX-YYYY=nn	Memory, change block XXXX-YYYY to nn
P nn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

Downloading Programs
The board must be bootstrapped and the monitor program loaded before programs may be downloaded to the board. At the monitor prompt, select the TTY Window menu command **Download | Download to RAM** and select the HEX file you wish to download.

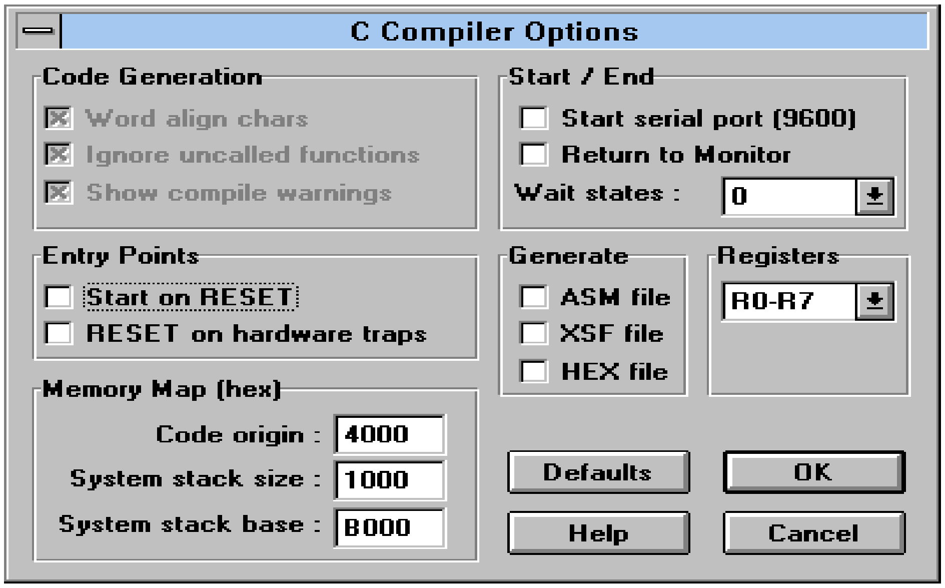
Running programs
Before compiled programs are run, the board must be bootstrapped and the monitor program downloaded. The compiled program must then be downloaded to the board. Each compiled program has an origin. The origin of assembly language programs are determined by the ORG pseudo operation used in the source. The origin of C programs are written to the **Options | Compiler** options dialog.
The monitor command **G** (go) followed by the address is used to run the programs. For example, to run a program whose origin is at 4000h, type

G4000

11.2.5 Options
There are two types of "**Options**." Use the "**Hardware Options**" to select the processor and the board you are using. This informs the READS166 environment which assembly and compile switches to invoke, as well as which bootstrap and monitor programs to send to the boards.



The READS166 assembler and the C compiler also have options which may be set by commands under the "**Options**" menu.



11.2.6 Help

This command invokes the READS166 Help system.

11.3 Using The Ra66, READS166 Assembler

Step 1. Writing an Assembly Language Program

Source code is entered using the Text Editor. Start the Text Editor from the Main Menu using the **Tools | Text Editor** command. From the Text Editor menu, create a new file by clicking on the **File | New** menu item. You are now ready to type in your program.

Enter the following short program:

```
          MONITOR equ 0C000h
          CR      equ 13
          LF      equ 10

          org   4000h              ; code starts at 4000h

          mov   r1, #msgHello      ; specify the string’s address
          calla cc_UC, print       ; the subroutine print is in the
                                   ;  file
     r66util.inc
```

```
          jmpa  cc_UC, MONITOR   ; done -- return to the monitor

          #include "r66util.inc"
     EVEN
     msgHello:
          db  CR, FL, "Hello World", CR, LF, 0
     EVEN
```

This program displays a string on the host. Most of the work is done by the subroutine "print" which is in the include file R66UTIL.INC. This file is included in to the source by the assembler directive "#include". Note that the program starts at address 4000h and returns to the monitor located at C000h after execution.

After typing in your program, save it under the name "TUTOR.ASM".

Step 2. Assembling the program

Now that you have typed in TUTOR.ASM, you may assemble it. Use the Main Menu **Compile | Assemble** file command. Provided that TUTOR.ASM is in the active Text Editor window, it will be assembled. The assembly results are displayed by a dialog.

Step 3. Detecting and correcting the errors

TUTOR.ASM will compile without errors if you did not make any typographical errors in entering the code. You may wish to introduce a few intentional errors to see how these errors are reported. For example, change the line

```
            mov r1, #msgHello
```

to

```
            xmov r1, #msgHello
```

or to

```
            mov r100, #msgHello
```

Step 4. Running the program

When you assemble the program without errors, a HEX file is generated. This file may now be downloaded the board and run. Open the TTY Window using the Main Menu command **Tools | TTY**. Verify that the serial port selected is valid as reported in the caption of the TTY Window. Now press the RESET button on the board. From the TTY Window menu, select **Bootstrap**. When the board bootstraps, you may press **ENTER** to view the monitor prompt. You may also press **H** (the monitor Help command) to see a short list of available monitor commands.

At the monitor prompt, select the TTY Window menu command **Download | Download to RAM** and select the file TUTOR.HEX. The program is now downloaded. Remember that the program had its origin at address 4000h.

From the monitor prompt issue the monitor **G** (go) command followed by the address. That is, type

G4000

to branch to and execute the program.

11.4 Using The Rc66, READS166 C Compiler

Step 1. Writing a C language program

Source code is entered using the Text Editor. Start the Text Editor from the Main Menu using the **Tools | Text Editor** command. From the Text Editor menu, create a new file by clicking on the **File | New** menu item. You are now ready to type in your program.

Enter the following short program:

```
/* ---------------------- */
char *szMsg="Hello World !";
int *pS0TIC=0xFF6C, *pS0TBUF=0xFEB0;

main(void){
 SendStr(szMsg);
}
/* ---------------------- */
void SendStr(char *sz){
 while(*sz) SendChar(*sz++);
}
/* ---------------------- */
void SendChar(char Ch){
 *pS0TBUF=Ch;
 while(!(*pS0TIC&0x80));
 *pS0TIC=0;
}
/* ---------------------- */
```

This program displays a string on the host. Note that the string is sent to the host by placing each character in to the transmit buffer of serial port 0. The function SendChar waits for the character to clear the transmit buffer before returning. After typing in your program, save it under the name "CTUTOR.C".

Step 2. Compiling the program

Before compiling CTUTOR.C, invoke the command **Options | Compiler** options from the Main Menu. Click on the pushbutton titled "**Defaults**" to select the default configuration. Notice that in the default configuration, the code origin is placed at 4000h, and that the program returns to the monitor after execution.

Next issue the **Compile | C Compile** file command from the Main Menu and compile CTUTOR.C. A HEX file of the name CTUTOR.HEX will be created.

Step 3.   Detecting and correcting the errors

CTUTOR.C will compile without errors if you did not make any typographical errors in entering the code. You may wish to introduce a few intentional errors to see how these errors are reported. For example, change the main function line

```
        SendStr(szMsg);
```

to

```
        SendStr(szMessage);
```

and recompile. Observe the errors generated by Rc66.

Step 4.   Running the Compiled Program

When you compile the program without errors, a HEX file is generated. This file may now be downloaded the board and run. Open the TTY Window using the Main Menu command **Tools | TTY**. Verify that the serial port selected is valid as reported in the caption of the TTY Window. Now press the **RESET** button on the board. From the TTY Window menu, select **Bootstrap**. When the board bootstraps, you may press **ENTER** to view the monitor prompt. You may also press **H** (the monitor Help command) to see a short list of available monitor commands.

At the monitor prompt, select the TTY Window menu command **Download | Download to RAM** and select the file CTUTOR.HEX. The program is now downloaded. Remember that the program had its origin at address 4000h. From the monitor prompt issue the monitor **G** (go) command followed by the address. That is, type

G4000

to branch to and execute the program.

12. BILL OF MATERIALS

12.1 Parts List

QUANTITY	PART	DESIGNATORs
1	1nF CAPACITOR	C10
1	10nF CAPACITOR	C1
24	100nF CAPACITOR	C2-9, C11-25
1	1uF CAPACITOR	C26
4	2.2uF CAPACITOR	C34-C37
1	22uF CAPACITOR	C27
2	47uF CAPACITOR	C29, C30
3	100uF CAPACITOR	C33, C32, C31
1	1N4001 DIODE	D5
1	1N4148 DIODE	D1
1	1N5817	D4
4	LEDS	D2, 3, 6, 7
1	PN2907	Q2
1	PN2222	Q1
1	33Uh coil	L1
2	25x2 HEADER	JP7, 8
2	DB 9 (short)	P1, P2
2	PUSH BUTTON	SW2, 3
4	2 HEADER	JP10, 12, 13, HH
4	3 HEADER	JP1, 2,
1	6 HEADER	JP11
1	2X3 HEADER	BTLDIS
1	2x4 HEADER	JP3
1	TERM BLOCK	JP9
1	SLIDE SWITCH	SW2
1	10K GANG RESISTOR	R3, R2
1	100 OHM RESISTOR	R1
4	330 OHM RESISTOR	R6, 7,10, 11
2	1K RESISTOR	R4, 5
1	10K RESISTOR	R12
1	2.2K RESISTOR	R8
1	NOT USED	R9
1	40 MEG Hz OSC	U1
1	SAB 88C166	U2
1	DS1233	U3
1	GAL22V10 15NS	U4
2	M29F040	U5, 6
2	62256 LP	U7, 8
1	LT1301	U9
1	MAX232CPE	U10

12.2 Parts Cross Reference

	Designator	Component	Sheet Number	Reference Sheet
1	C1	10nF	2	R166FCPU.SCH
2	C2	10nF	4	R166FMEM.SCH
3	C3	10nF	4	R166FMEM.SCH
4	C4	10nF	6	R166FVPP.SCH
5	C5	10nF	6	R166FVPP.SCH
6	C6	10nF	6	R166FVPP.SCH
7	C7	10nF	6	R166FVPP.SCH
8	C8	10nF	6	R166FVPP.SCH
9	C9	10nF	6	R166FVPP.SCH
10	C10	10nF	6	R166FVPP.SCH
11	C11	10nF	6	R166FVPP.SCH
12	C12	10nF	6	R166FVPP.SCH
13	C13	10nF	6	R166FVPP.SCH
14	C14	10nF	6	R166FVPP.SCH
15	C15	10nF	6	R166FVPP.SCH
16	C16	10nF	6	R166FVPP.SCH
17	C17	10nF	6	R166FVPP.SCH
18	C18	10nF	6	R166FVPP.SCH
19	C19	10nF	6	R166FVPP.SCH
20	C20	10nF	6	R166FVPP.SCH
21	C21	10nF	6	R166FVPP.SCH
22	C22	10nF	6	R166FVPP.SCH
23	C23	10nF	6	R166FVPP.SCH
24	C24	10nF	6	R166FVPP.SCH
25	C25	100nF	2	R166FCPU.SCH
26	C26	1uF	2	R166FCPU.SCH
27	C27	22uF	2	R166FCPU.SCH
28	C28	1nF	5	R166FPIO.SCH
29	C29	47uF	6	R166FVPP.SCH
30	C30	47uF	6	R166FVPP.SCH
31	C31	100uf	6	R166FVPP.SCH
32	C32	100uf	6	R166FVPP.SCH
33	C33	100uf	6	R166FVPP.SCH
34	C34	2.2uF	7	R166FSER.SCH
35	C35	2.2uF	7	R166FSER.SCH
36	C36	2.2uF	7	R166FSER.SCH
37	C37	2.2uF	7	R166FSER.SCH
38	C38	10nF	6	R166FVPP.SCH
39	C39	10nF	6	R166FVPP.SCH
40	C40	220uF	6	R166FVPP.SCH
41	C41	10nF	6	R166FVPP.SCH
42	D1	1N4148	2	R166FCPU.SCH
43	D2	BOOT LED	3	R166FBTL.SCH
44	D3	AUX LED	3	R166FBTL.SCH
45	D4	1N5817	6	R166FVPP.SCH

46	D5	1N4001	6	R166FVPP.SCH
47	D6	PWR	6	R166FVPP.SCH
48	D7	PGM	6	R166FVPP.SCH
49	D8	BR	6	R166FVPP.SCH
50	JP1	REF	2	R166FCPU.SCH
51	JP2	GND	2	R166FCPU.SCH
52	JP3	EBCOPT	2	R166FCPU.SCH
53	JP4	CFG0	3	R166FBTL.SCH
54	JP5	CFG1	3	R166FBTL.SCH
55	JP6	BTLDIS	3	R166FBTL.SCH
56	JP7	MEMORY	5	R166FPIO.SCH
57	JP8	I/O PORTS	5	R166FPIO.SCH
58	JP9	5V	6	R166FVPP.SCH
59	JP10	VPPON	6	R166FVPP.SCH
60	JP11	RS-232	7	R166FSER.SCH
61	JP12	ATXD	7	R166FSER.SCH
62	JP13	ARXD	7	R166FSER.SCH
63	L1	33uH	6	R166FVPP.SCH
64	P1	HOST	7	R166FSER.SCH
65	P2	S1	7	R166FSER.SCH
66	Q1	2N2222	3	R166FBTL.SCH
67	Q2	PNP	6	R166FVPP.SCH
68	R1	100	2	R166FCPU.SCH
69	R2E	10K	6	R166FVPP.SCH
70	R2D	10K	2	R166FCPU.SCH
	R2C	10K	2	R166FCPU.SCH
	R2B	10K	2	R166FCPU.SCH
	R2A	10K	2	R166FCPU.SCH
71	R3D	10K	3	R166FBTL.SCH
	R3C	10K	3	R166FBTL.SCH
	R3B	10K	3	R166FBTL.SCH
	R3A	10K	3	R166FBTL.SCH
	R3E	10K	3	R166FBTL.SCH
72	R4	1K	3	R166FBTL.SCH
73	R5	1K	3	R166FBTL.SCH
74	R6	390	3	R166FBTL.SCH
75	R7	390	3	R166FBTL.SCH
76	R8	2.2K	5	R166FPIO.SCH
77	R9	10K	6	R166FVPP.SCH
78	R10	330	6	R166FVPP.SCH
79	R11	330	6	R166FVPP.SCH
80	R12	10K	6	R166FVPP.SCH
81	R13	10K	2	R166FCPU.SCH
82	R14	10K	2	R166FCPU.SCH
83	SW1	RESET	2	R166FCPU.SCH
84	SW2	USER/MON	3	R166FBTL.SCH
85	SW3	NMI	5	R166FPIO.SCH
86	U1	40MHz	2	R166FCPU.SCH
87	U2	SAB 88C166	2	R166FCPU.SCH

88	U3	DS1233	2	R166FCPU.SCH
89	U4	GAL22V10	3	R166FBTL.SCH
90	U5	M29F010	4	R166FMEM.SCH
91	U6	M29F010	4	R166FMEM.SCH
92	U7	62256	4	R166FMEM.SCH
93	U8	62256	4	R166FMEM.SCH
94	U9	LT1301	6	R166FVPP.SCH
95	U10	MAX232	7	R166FSER.SCH

13. TOP OVERLAY AND CIRCUIT DIAGRAMS