R-FLIC USER'S GUIDE

Version 1.4
March 20, 2002

RIGEL CORPORATION
PO Box 90040, Gainesville, FL 32607
Phone (352) 373-4629, FAX (352) 373-1786
www.rigelcorp.com, tech@rigelcorp.com

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.

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. You will be responsible for shipping costs. Please call (352) 373-4629 prior to shipping.

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 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 or returned to you as is. 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 CONTENT

1 OVERVIEW

The R-FLIC is an industrial board designed to accept multiple processors in the Infineon and ST Microelectronics 167 family. The board works with the following processors: the C167LM, C167SR, C167CR-LM, ST10R167, and the ST10F168, which have both CAN and 256K FLASH capabilities. The R-FLIC board comes with two software programs written by Rigel Corporation, READS166 and the rFLI Utility Software.

1.1 Basic Features

The R-FLIC board is a 6 layer industrial board with separate VCC and GND planes. The microcontroller is run with a 16-bit nonmultiplexed data bus and a nonmultiplexed address bus. The various configuration options are set by jumpers that determine the reset options of the 167. Most notably, activation of the bootstrap loader, the width of the address bus, and the number of chip select lines are determined by these jumpers.

There are three banks of memory on the board when populated with the ST10F168 processor, RAM, ROM, and 256K internal FLASH. The RAM and ROM banks may hold up to 1 megabyte of memory each, for a total of up to 2 megabytes of external memory. Memory banks are controlled by the chip select signals CS0# and CS1# of the 167. The chip select lines are jumper configurable to allow for maximum flexibility when using the R-FLIC board. The default configuration is the 64K/256K RAM and no ROM mode. In this mode, the monitor program or user program is downloaded to RAM using the 167 bootstrap feature.

The R-FLIC contains the CAN interface chip, the 82C250, for immediate hook up to a CAN network. The R-FLIC has the same standard operating modes and hardware features as the RMB167-CRI. The R-FLIC is designed to run each processor at its fastest speed with no wait states. The R-FLIC is compatible with all of our other 16-bit boards. The board size, the location and function of all headers are kept the same.

Notable Features of the board include:

- **Industrial strength shielding:** The PCB is a six-layered board with separate VCC and GND planes designed to operate in noisy industrial environments.
- **FLASH Memory Capability:** The board uses up to 1MB of 5-volt external FLASH EEPROM. When populated with the ST10F168, there is an additional 256K internal FLASH that may be programmed on-board.
- **External bus system on a header.** The R-FLIC has a 2 x 16 pin header for selection of the operating modes, which are determined during reset by the state of Port 0 bits.
- **Chip Select lines used for memory mapping.** Uses CS0# and CS1# to interface to external memory, allowing greater flexibility for memory mapping.
- **The HiTEX / HiTOOLS ICE connector** designed into the board for those needing emulation capabilities.
- **On Board 5 Volt Power Regulator.**
- **On Board 12 volt Power Regulator.**
- **CAN physical layer built-in to the board.**

1.2 Hardware Features

- 167 Processor (ST Microelectronics, Infineon)
    - Bootstrap loading feature
    - 256K internal FLASH (ST10F168)
    - Built-in CAN interface
    - One serial port
    - One high speed synchronous serial channel
    - Five 16 bit timers
    - Watchdog timer
    - 4K on chip RAM memory
    - 20MHz internal system clock (5MHz clock with internal phase-lock loop)

16 Channel 10-bit A-to-D converter with enhanced operation modes
32 channels CAPture / COMpare unit
8 channels PEC (data transfer channel)
4 channels PWM

- Serial port uses a RS232 driver, terminates at DB-9 connector and a 3-post connector
- 111 bits of general-purpose inputs/outputs
- Chip Select lines used for memory mapping
- Accommodates 64KB - 1MB of SRAM (256K installed)
- Accommodates 256KB - 1MB of FLASH memory (not installed)
- FLASH parts programmable on-board
- Push buttons for RESET# and NMI# (non-maskable interrupt)
- CAN physical layer built into the board
- IC's populated in gold-plated machine screw sockets
- Power on LED
- 12-volt ready LED
- Flexible and embeddable 6 layer industrial board
- Board size 4"x6 ½"
- Power consumption is less than 175mA running at 20MHz
- Mounting holes in corners

1.3 READS166 Overview

READS166, version 3.0x, is Rigel Corporation's Integrated Development Environment for the ST Microelectronics 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 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 8K.

1.4 rFLI Utility Software

rFLI is a utility program used to burn the internal FLASH memory of the processor and the external FLASH memory chips on the 16x/ST10 boards. The rFLI Utility Software runs in the Windows 95/NT environment. It supports the bootstrap loader feature and downloads a minimal monitor during bootstrapping. The board 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 rFLI program is used only when burning the external or internal FLASH memory on Rigel boards. Use READS166 for code development.

1.5 Parts List

Your R-FLIC package includes the following:

Hardware

1. R-FLIC board populated with a 167/168 processor and 64K of static RAM.

Software

1. READS166, Evaluation version for Windows95/NT
2. RMON167 monitor program with source code and a separate MON if board is populated with the ST10F168
3. rFLI Utility Software and rFLI168 for use with the board when populated with the 168 chip.
4. Software from third party vendors

Documentation

1. R-FLIC User's Guide with circuit diagrams
2. READS166 User's Guide on CD ROM
3. Bootstrap file source code on CD ROM
4. Sample programs on CD ROM

2 SOFTWARE

The R-FLIC 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

Two software programs written by Rigel are included with the R-FLIC board, READS166 and rFLI Utility Software. Both are 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 CD-ROM in your drive. Go to the **Rigel Products | 166 Software | READS166 | *.exe** program**.** Click on the exe file and the program will begin to load in your system.
Follow the standard install directions answering the questions with the appropriate answers.

2.3 Software Installation, rFLI Software Utility

Place the CD-ROM in your drive. Go to the **Rigel Products | 166 Software | rFLI | *.exe** program**.** Click on the exe file and the program will begin to load in your system. Load the rFLI168 if your board is populated with the ST10F168 chip.
Follow the standard install directions answering the questions with the appropriate answers.

2.4 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 and Tasking, but may not be their latest version. Updates for their software may be found on their web sites at www.keil.com and www.tasking.com.

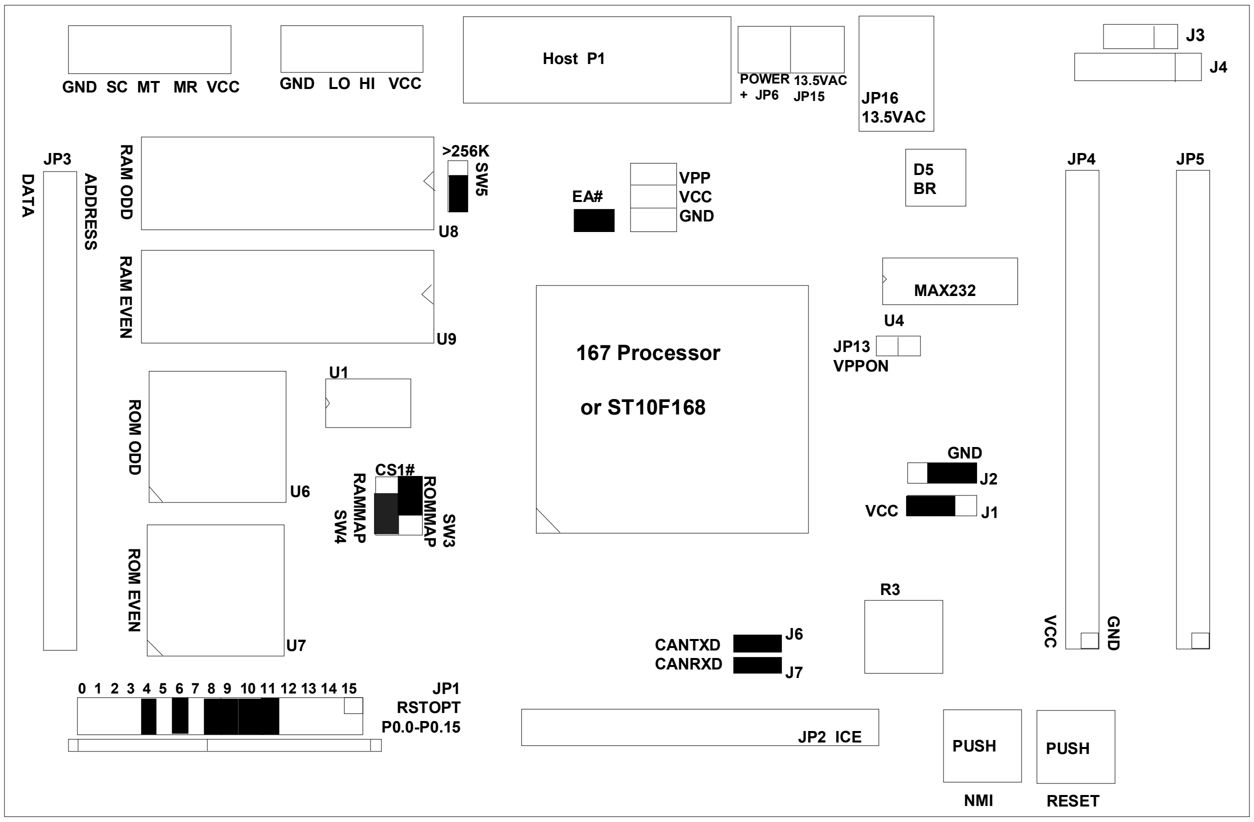
Data sheets on the IC's used on our boards may also be found on Rigel's CD.

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

3 START-UP

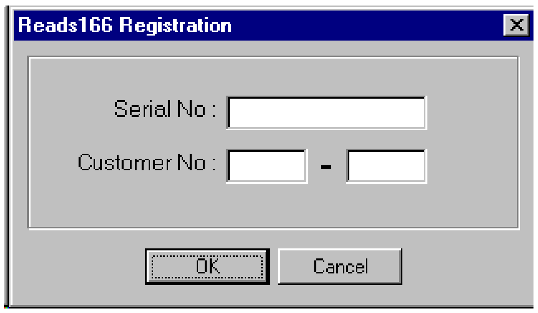
3.1 Connections, Jumper Settings

1. Connect your RIGEL board to the PC host via a serial cable.
2. Connect the board to a power supply.
   If using JP6 a well-regulated 5-volt power supply must be used.
   When using JP15 or JP16 a 9-15 volt (500ma-1 amp) wall transformer may be used.
   The red LED D2 will come on if power is connected correctly
3. Check to make sure the correct jumpers are in place. The following diagram shows the most common configuration for the R-FLIC board. Default jumpers which must be installed are:
   SW4 -- in position CS0#
   SW5 -- in bottom position (with 64K / 256K RAM installed)
   JP1 -- jumpers in P0.4, P0.6, P0.8, P0.9, P0.10, and P0.11,
   EA# jumper
   Other jumpers that may be installed include: SW3, J1, J2, J6, J7, JP1-P0.12.



3.2 Software Initialization

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.


Reads166 Registration

Serial No :

Customer No : -

OK Cancel

3.3 Serial Number

3.3.1 Demo Version

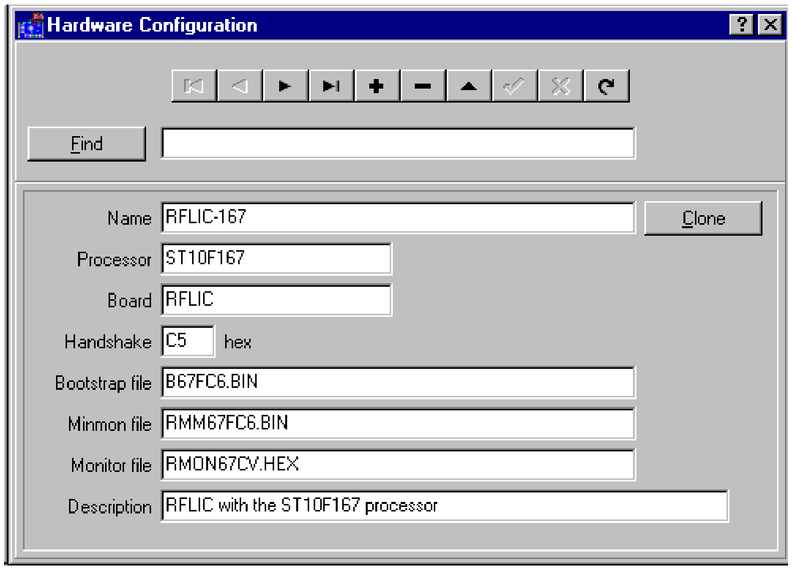
When you run the READS166 software a pop-up registration box will open asking for your serial and customer numbers. If you are using the demo version of the software you may press **CANCEL** and the box will disappear and the About Box will appear. Press the **OK** button in the About Box and the software will run in the Demo mode. The demo mode limits the size of the C files you can compile to about 12K.

3.3.2 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.

3.4 Configuring READS 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 that appears.



Name	RFLIC-167
Processor	ST10F167
Board	RFLIC
Handshake	C5 hex
Bootstrap file	B67FC6.BIN
Minmon file	RMM67FC6.BIN
Monitor file	RMON67CV.HEX
Description	RFLIC with the ST10F167 processor

Note: for the ST10F168 select the following:

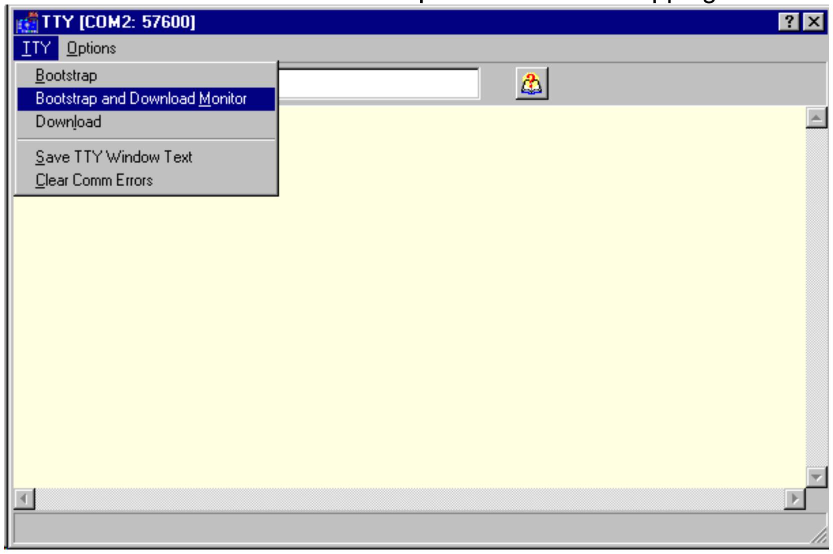
Name:	R168 (user choice)
Processor:	SAB-C167CR-LM
Board:	R168-FLIC
Handshake:	D5
Bootstrap file:	B68X.BIN
Minmon file:	RMM68X.BIN
Monitor file:	RM67J.HEX
Description:	(optional -- user preference)

See Readme.txt in 168 zip file.

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.

3.5 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.

3.6 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.

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.

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

3.7 Troubleshooting

All the boards are functionally tested before shipment. If the system does not bootstrap as expected, review the hardware, software, and PC setup.

3.7.1 Hardware Set-up

1. Make sure that that you have the power connected correctly.
2. Verify that the power-on LED is lit.
3. Review the R-FLIC jumper settings.
   SW4 -- in position CS0#
   SW5 -- in bottom position (with 64K / 256K RAM installed)
   JP1 -- jumpers in P0.4, P0.6, P0.8, P0.9, P0.10, and P0.11,
   EA# jumper inserted
4. Check the modem cable connections to the R-FLIC and to the host PC.
5. Run the READS software and try to bootstrap and download the monitor again.

If the board still won't operate correctly, check the software setup.

3.7.2 Software Set-up

1. Select the TTY menu **TTY Option**.
2. Verify that the COM port correctly matches that of the host PC.
3. Select the Baud rate to be 19.2K
4. Select a large TTY Timeout parameter, one of at least 2 seconds.
5. Note the hardware configuration given in the drop list "**HW Configuration**."
6. From the Reads166 main window, select the menu option **Hardware Configuration** located under the **Tools** menu.
7. Find the record corresponding to the hardware configuration selected in the **TTY Options**.
8. Verify the entries of the record as follows:

Handshake: C5, (D5 for ST10F168) Bootstrap file: B67FC6.bin, MinMon file: RMM67FC6.bin, and the Monitor file: RMON67CV.hex. (For 168 Monitor file is: ???   )

The Name, Processor, Board, and Description fields are for information purposes only. If the fields are not as listed above, you may search for a record which matches the desired fields. Once located, go to the **TTY Options** dialog and select the correct hardware configuration. Alternatively, you may create a new hardware configuration record or modify an existing one from the Hardware Configurations dialog.

If the board still won't operate correctly, check the PC setup.

3.7.3 PC set-up

1. Go to the Windows **Control Panel** usually located under the Settings menu (**Start, Settings, Control Panel**).
2. Click on the **Ports icon**. (If one does not exist, you may need to install some Windows components again.)
3. Select the port you are using. If the port number does not appear in the list add the port by clicking on the "**Add**" button. You need to restart Windows for the new port selection to take effect.
4. Click on the "**Settings**" button.
5. Select 57600 Baud, 8 data bits, no parity and 1 stop bit. Set "**Flow Control**" to none.
6. Click on the "**Advanced**" button. Select the port you are using.
7. Specify the "**Base I/O Port Address**" and the "**Interrupt Request Line (IRQ)**" fields to be "**default.**"
8. You should try both enabling and disabling FIFO.

If these steps do not remedy the problem, please contact Rigel Corporation at (352) 373-4629 or tech@rigelcorp.com.

4 OPERATING NOTES

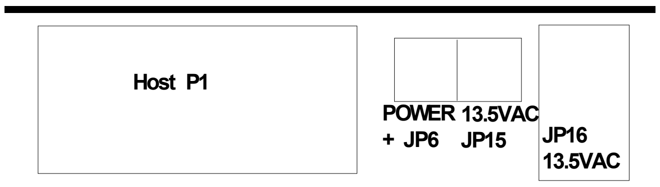
The R-FLIC needs two connections: to a power supply and to the serial port of a host via a modem cable.

4.1 Power

Power can be applied to the board by three different methods. There is a power jack JP16, and 2 two-position screw-type terminal blocks, JP6 and JP15.

4.1.1 JP16, Power Jack

JP16 is the default power connector for use with a 9 to 15-volt 500ma-wall transformer. The transformer will need a 3.5mm plug to mate with the 3.5mm power jack on the board. The board has a built-in regulator that will provide the circuitry with the +5 volts needed for normal operation. When the R-FLIC board is populated with the ST10F168 processor a 12 to 15-volt transformer will be needed for programming the 12-volt internal 256K FLASH. Two jumpers will need to be inserted to activate the 12-volt circuitry for programming the internal FLASH. See Section 5.4 for details.



Power Supply and Serial Port

4.1.2 JP15, Terminal Block

The terminal block JP15 is in parallel with the power jack, providing an alternative connection for the power supply. Use JP15 if your power supply does not have the correct connector on it.

4.1.3 JP6, 5-Volt Terminal Block

JP6 may be used for power if a well- regulated +5V DC (+/- 5%) power source is available. Using JP6 for power bypasses the on-board power regulator. The (+) terminal of JP6 is marked on the board.

Note

When using JP6 for power, a +5 volt regulated power supply must be used. 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 R-FLIC board in reverse polarity, the diode will short the power supply attempting to prevent damage to the board.

4.2 Serial Port 0

Serial Port 0 of the microcontroller is connected to the DB-9 port P1 through an RS-232 level converter. The microcontroller supports the transmit and receive signals. A minimal serial port may be constructed with just 3 lines: transmit, receive, and ground, disregarding all hardware handshake signals. Port P1 (HOST) of the R-FLIC 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 R-FLIC to pin 2 of the host, and similarly pin 3 to pin 3, and pin 5 to pin 5. Note that J3 is a 3-pin header, which carries the same signals as P1. This header is convenient for embedded applications where a modem cable is not needed.

4.3 Serial Port 1

Serial port 1 is used as a synchronous serial channel. It has 3 TTL-level signals, MRST (Master Receive Slave Transmit), MTRS (Master Transmit Slave Receive), and SCLK (Serial Clock). J4 the header, and J8 the terminal blocks, may be used to access serial port 1. Both J4 and J8 are also denoted by SSC (Serial Synchronous Channel) on the silk-screen.

4.4 Push Buttons

4.4.1 Reset (SW1)

The reset button is connected to the reset pin of the processor and resets the board. Before bootstrapping, press and release the reset button and wait 3 seconds to allow the processor to initialize.

4.4.2 NMI (SW2)

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 scheme works as long as the monitor program in RAM is not altered. Pressing the NMI button is usually sufficient to interrupt a user program that is 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.

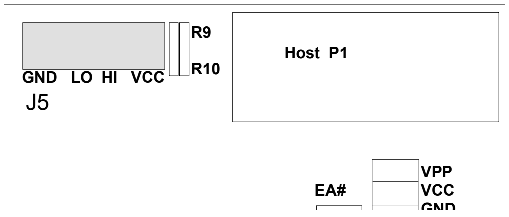
4.5 LEDs

The R-FLIC has three LEDs. D2 is lit whenever power is applied to the board. D3 is lit when 12-volts is active for programming the internal FLASH on the ST10F168 processor. D4 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 D4 will be off and remain off until the bootstrap loader successfully completes loading the bootstrap file into RAM.

4.6 CAN Terminal Block

The CAN LO and CAN HI lines are brought to the terminal block marked J5. A two-wire twisted pair may be used to connect the R-FLIC board to other CAN devices. The terminal block J5 also carries the VCC and GND lines for reference or to provide power.

4.6.1 CAN Physical Layer Considerations



CAN Connectors

The R-FLIC uses the 82C250 chip to provide a physical layer interface to a twisted pair. The outputs of the 82C250 are brought directly to J5 as CAN-LO and CAN-HI. The pull-up or termination resistors are not populated on the board. If your physical layer contains no resistors, for example, if you connect two R-FLIC boards together, pull-up (and pull-down) resistors will improve the line characteristics. The optimum resistor values depend on the transmission speed and the properties of the twisted pair. Note that the resistance and the capacitance of the twisted pair increase with the length of the connection. In addition, the capacitance of the line becomes more important as the transmission speeds increase. Typically, two external resistors (1K to 10K) are sufficient, one pulling CAN-LO to VCC and the other pulling CAN-HI to GND. That is, use a pull-up resistor on CAN-LO and a pull-down resistor on CAN-HI. These resistors may be added to the board at the resistor positions R9 and R10 next to the CAN terminal block.

4.7 Potentiometer R3

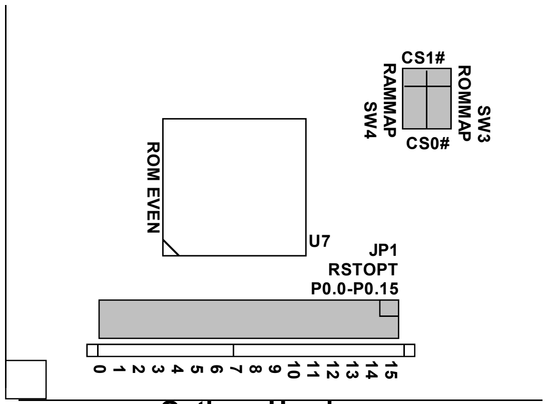
The potentiometer R3 is used by the slope control feature of the 82C250 CAN interface chip. Turning R3 clockwise increases its value. Turned all the way counterclockwise; R3 connects the slope control pin of the 82C250 to ground.

Slope control refers to desensitizing the CAN interface chip to fast acting transients. External noise sources may induce interference on the twisted pair CAN physical layer, which appear as transients to the CAN interface chip. When R3 is about zero, (grounding the slope control pin of the 82C250), the CAN interface chip processes the signals without measuring their slopes. This gives the fastest possible operation of the CAN interface chip. In this case, the physical layer should be coaxial or well insulated from external noise sources. As the value of R3 increases, the CAN interface chip enters its slope control mode. The higher the value of R3 (more clockwise rotation), the less sensitive the chip is to fast acting signals. This is more desirable in a noisy industrial environment when unprotected twisted pair physical media are used.

5 JUMPER CONFIGURATIONS

5.1 JP1 (Reset Options)

JP1 is a 16 x 2 header. Each pair of posts corresponds to a bit of Port 0. The silk-screen is labeled from 0 to 15 (Port0.0 to Port0.15) indicating bit or jumper positions. Some of the 167 operating modes are determined during reset by the state of Port 0 bits. (Refer to the ST / Infineon data book for further information.) Inserting a jumper in JP1 connects the corresponding bit of Port 0 to ground via a 6.8 to 8.2K resistor. This, in turn, sets the operating mode. For example, the Bootstrap mode is invoked when bit 4 of Port 0 is held low at reset. The R-FLIC uses seven of these bits for configuring the board. It is important not to populate the remaining jumpers, since these are either used by the system (for example in testing modes or emulation modes) or are reserved.



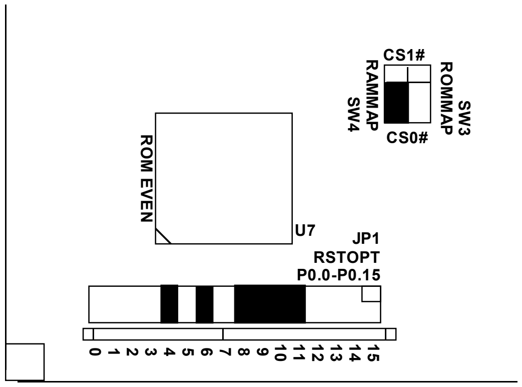
Options Header

5.1.1 P0.4

Bit 4 of Port 0 invokes the bootstrap mode when held low at reset. Inserting a jumper at position 5 (P0.4) on JP1 grounds bit 4, causing the board to bootstrap at hardware reset. The R-FLIC's default configuration is with this jumper P0.4, installed. With P0.4 removed and a user's program with starting address at zero in ROM, the board will run the user's program upon reset.

5.1.2 P0.7 and P0.6

Bits 7 and 6 of Port 0 determine the bus type of the C167/168. In the default configuration P0.7 is not grounded and P0.6 is grounded, resulting in a startup bus type BUSTYP=10. This is the 16-bit nonmultiplexed bus mode. The R-FLIC's default configuration is with this jumper P0.6, installed.



Default Option Jumpers

5.1.3 P0.8

Bit 8 of Port 0 determines the configuration of the WR# and BHE# signals. When P0.8 is grounded, WR# acts as WRL# and BHE# as WRH#. The latter, WRL# and WRH# are then used to select the low and high halves of the memory banks. P0.8 is grounded in the default configuration, by inserting a jumper in P0.8.

5.1.4 P0.9 and P0.10

Bits 9 and 10 are copied to the CSSEL field of the RP0H register. They determine the number of chip select lines the C167 will generate. The default configuration is three chip select lines, which is obtained by grounding both P0.9 and P0.10, with jumpers inserted. The RAM and ROM memory banks use two of the chip select lines. The third chip select line is used in experiments.

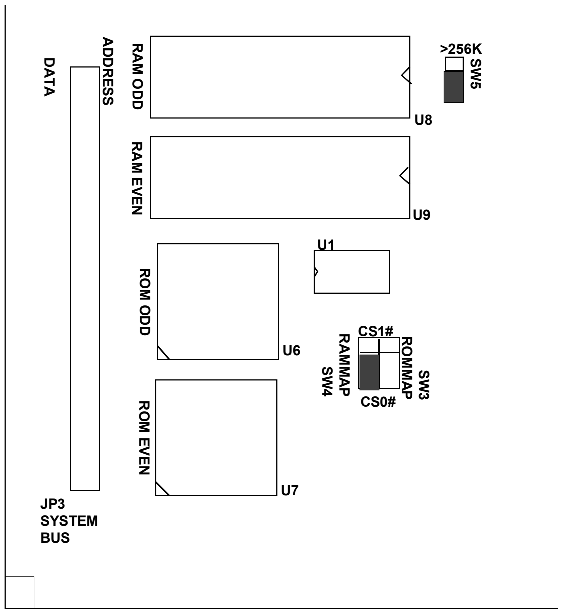
5.1.5 P0.11 and P0.12

Bits 11 and 12 are copied to the SALSEL field of the RP0H register. They determine the width of the address bus the C167 will use. The default configuration is the 24-bit wide address bus. This is achieved by grounding P0.11 but not P0.12.

The on-chip CAN controller uses the address lines A21 and A22 for the CANRxD and CANTxD signals. Thus, when using the CAN controller, the reset options must be set to have no more than the 4 bits of segment address. The 4-bit segment address mode gives 20 address lines (A0 to A19) and makes A21 and A22 available to the on-chip CAN controller. Select the 4-bit segment address mode by grounding both P0.11 and P0.12.

5.2 SW3 and SW4 (RAM and ROM Chip Select Assignment)

SW3 and SW4 are the chip select jumpers used for mapping RAM and ROM. Two chip select lines are available for RAM and ROM mapping, CS0# and CS1#. Note that CS0# is active upon reset. If you wish to run the board with the RAM active at start-up insert SW4 in CS0# position. If you wish to run your own program from external FLASH ROM on start-up insert a jumper in SW3 in the CS0# position. Care should be taken not to map both memory banks to the same chip select signal.



Default Memory Jumpers

5.2.1 SW3, External FLASH ROM

If ROM is not used, you do not need to populate SW3 with a jumper. The default configuration of the board is with RAM mapped to low memory and no ROM.

5.2.2 SW4, RAM

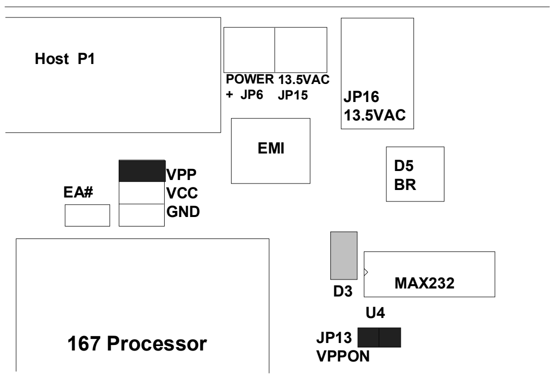
The R-FLIC board uses RAM in low memory for the READS166 monitor program. The default configuration maps RAM to lower memory, controlled by the CS0# signal.

5.3 SW5 (RAM Size)

SW5 selects the size of RAM. This jumper should be placed in the lower position for 32K and 128K devices. Note that with 32K devices, RAM is 64K and with 128K devices RAM is 256K. For 512K devices, which provide 1MB RAM, the jumper needs to be populated in the upper position marked ">256K".

5.4 Internal FLASH Programming Jumpers

Two jumpers control the built-in 12-volt circuitry needed for programming the internal FLASH memory on the ST10F168 processor. JP13-VPPON and JP21-VPP jumpers should only be inserted when you clear, erase, or program the FLASH. During normal operation of the board, these jumpers should be removed.



FLASH Programming Jumpers

5.4.1 JP13, VPPON

JP13 must be inserted to provide the 12-volts needed to program the internal FLASH. When
the jumper is inserted the red LED, D3 will light up showing the 12-volt circuit is active.

5.4.2 JP21, VPP Options

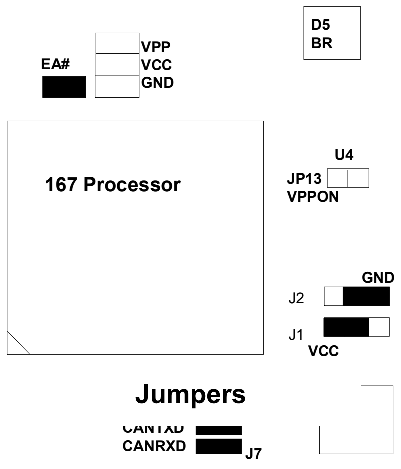
JP21 is a 2x3 position header that connects the VPP pin of the processor to 12-volts (VPP), 5-volts (VCC), or GND. When programming the internal FLASH, the jumper in JP21 must be plugged into the VPP position. With the jumper removed from JP21 the VPP pin of the processor, by default, is connected to 5-volts by a pull-up resistor in series with a diode.

Note: You can damage the board by inserting more than one jumper in JP21. Never insert more than one jumper into JP21.

5.5 JP14, EA#

The EA# jumper is used to connect the EA# pin of the processor to ground. With the EA# jumper inserted, the board uses external memory. With the EA# jumper removed, the board will run from the internal FLASH memory of ST10F168 processor.

Note: The reset options jumper P0.4 must also be removed to prevent the processor from entering the bootstrap mode.



5.6 J6 and J7, CANTxD, CANRxD Jumpers

The on-chip CAN controller uses the address lines A21 and A22 for the CANRxD and CANTxD signals. Note that when using the CAN controller, the reset options must be set to have no more than the 4 bits of segment address. This is done by grounding both P0.11 and P0.12.

The jumpers J6 and J7 connect A21 and A22 to the CAN interface chip. The jumpers are located next to R3, the slope potentiometer. Remove the jumpers J6 and J7 if you are not using the CAN controller and need to use A21 and A22 to access larger banks of (off-board) memory.

5.7 Analog-to-Digital Converter Reference Jumpers

The analog-to-digital converter requires a ground and reference voltage. These reference voltages may be provided either from JP4 or JP5 lines marked VG (GND reference voltage) and VR (reference voltage), or connected to the +5 volt TTL supply. Jumpers J1 and J2 select the source of reference voltages. The center posts of J1 and J2 are connected to the 167 VAREF and VAGND inputs. Post 1 of J1, 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 JP5 from the terminal marked VR. Similarly, the post marked GND of J2 is connected to the ground of the supply. Connecting the center post of J2 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 JP5 terminal marked VG.

5.8 JP2

The header JP2 and jumpers, (JP)8, (JP)9, (JP)10, (JP)11, (JP)12 accommodate the HiTEX/HiTOOLS In-Circuit Emulator (ICE). Please call Rigel if you wish to use the In-Circuit Emulator with this board.

6 MEMORY BLOCK OPTIONS

There are two or three blocks of memory available on the R-FLIC board, (three when populated with the ST10F168) The RAM block, the ROM block, and the on-chip 256K (internal) FLASH block of memory on the ST10F168. The RAM and ROM block can each hold up to 1MB of memory.

6.1 RAM Memory Options

The RAM block is designed to take 3 sizes of static RAMs, 32KB (62C256-type), 128KB (681000-type), and 512KB (628512-type) 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 32-pin sockets marked U8 and U9. Place 28-pin 32K RAM devices closer to the 2 X 25 header, away from the processor. RAM size must be selected with jumper SW5. The 167 may be programmed to insert wait cycles during external memory access. However, in order to run the 167 at its full potential, the RAMs should be rated at 70 nano seconds or faster.

The READS166 software may be used to download and run programs from RAM.

6.2 ROM Memory Options

The ROM block of memory accepts a variety of 5-volt FLASH EEPROMs in the ST Microelectronics, AMD, and Motorola 29F0xx family. Two chips are needed, one for EVEN and the other for ODD addresses. These chips are placed in the 32 pin PLCC sockets marked U6 and U7. The rFLI Utility Software that comes with the R-FLIC board may be used to program the FLASH EEPROMs on the board.
The 167 / 168 may be programmed to insert wait cycles during external memory access. However, in order to run the 167/ 168 at its full potential, the FLASH should be rated at 70 nano seconds or faster.

6.2.1 Program Settings for Running out of External EEPROM

The external FLASH EEPROM may be used to store code, which is to be executed upon reset. The demo project HelloRom illustrates how a C program may be placed into external EEPROM. The compiler settings for code to be executed from external ROM are given as comments at the beginning of the example code. The instructions and settings are repeated below. Please note that parameters such as code origin and stack size may be modified to better suit your application needs.

1. Place a "_startup.inc" file in the project subdirectory containing the following instructions:

```
mov   BUSCON0, #480h
mov   ADDRSEL1, #00C2h
mov   BUSCON1, #480h
```

2. Check the "Start on RESET" box under the "**Entry Points**" tab of the **Projects | Options | Compile Options** dialog box.
3. Check the "Start Serial Port (9600)" box under the "**Start/End**" tab of the **Projects | Options | Compile Options** dialog box.
4. Again, from the Compiler Options dialog, select the "Memory Map" tab and specify the following:

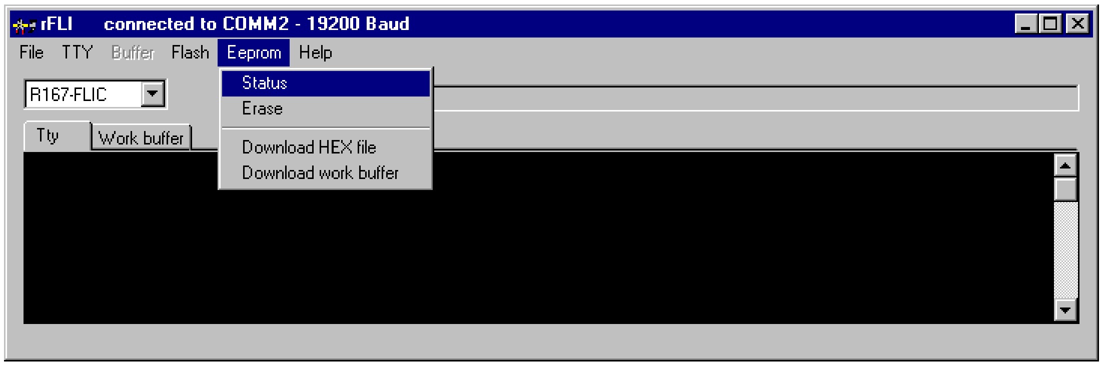
```
Code origin              100
System stack size       1000
System stack base       C800
Syscon                  8080
```

6.2.2 Programming the External EEPROM

Programming the FLASH EEPROMs on the board is very easy using the rFLI Software that comes with the board. The following steps should be followed for programming the FLASH device on the board.

1. Jumpers P0.4, P0.6, P0.9, P0.10, P0.11, EA#, J12, must be inserted and RAM should be mapped to CS0#, and ROM mapped to CS1#. These are the default jumper settings.
2. Apply power to the board. The red LED D1 will light up.
3. The rFLI Utility Software must be installed and used to bootstrap the board.
4. From the rFLI Utility Software select the **TTY | Port** menu command to select the port you wish to use.

5. Use the **TTY | Bootstrap** and load monitor command to bootstrap the board. Alternatively, click the **"Boot"** button at the top of the rFLI window.
6. After bootstrapping, the special-purpose monitor program is downloaded to the board. You may verify that the board is responding by pressing the **'enter key'** and observing the monitor prompt "ST10F167 >". You may also type **'H'** for a brief help screen displayed by the monitor.
7. From the menu commands select **EEPROM | Status** to check the status of the chips. Select **EEPROM | Erase** to erase the EEPROMs.
8. The menu command **EEPROM | Download HEX file** may be used to program the FLASH EEPROMs.
9. The program in the FLASH EEPROM memory may be dumped to the work buffer to be inspected, modified, and reloaded to the memory.



6.2.3 Running Programs from External EEPROM

The program burned into the FLASH EEPROMs can be mapped to run at a variety of locations depending on the chip select signals assigned and on the size of each memory bank programmed into the external bus control registers BUSCON0 and BUSCON1. See Section 8 and the ST / Infineon data books for more information on system programming.
To run the program at start up, the following steps may be followed.

1. Jumpers P0.6, P0.8, P0.9, P0.10, P0.11, EA#, and SW5 (depending on the RAM size), must be inserted. Remove P0.4, insert the jumper from SW4 into the CS#1 position and insert a jumper into SW3 in the CS0# position.
2. Apply power to the board. The red LED D2 will light up.
3. Push the reset button on the board. Your program is now running from low memory. To test this you may use the simple 'Hello World' program given with the READS166 software. Load the program into the external FLASH and from the READS166 TTY window press the reset button on the board. The message 'Hello World' should appear in the TTY window.

6.3 On-Chip FLASH Memory

The ST10F168 has 256K of on-chip FLASH memory. The R-FLIC board is designed to provide the 12 volts needed to program the FLASH. The R-FLIC is sold with software, the rFLI168 Utility Software, especially written to program the FLASH on the ST10F168 processor. The on-chip FLASH has 4 memory banks, two 96K, one 48K, and one 16K, which may be programmed and erased separately. Please refer to the ST10F168 data book for details of the FLASH memory.

6.3.1 Program Settings for Running out of Internal FLASH

The Internal FLASH may be used to store code, which is to be executed upon reset. The demo project HelloFlash168FLIC.hex illustrates how a C program may be placed into internal FLASH. The compiler settings for code to be executed from internal FLASH are given as comments at the beginning of the example code. The instructions and settings are repeated below. Please note that parameters such as code origin and stack size may be modified to better suit your application needs.

1. Place a "_startup.inc" file in the project subdirectory containing the following instruction:

```
mov   BUSCON0, #480h
```

2. Check the "Start on RESET" box under the "Entry Points" tab of the C Compiler Options dialog box.

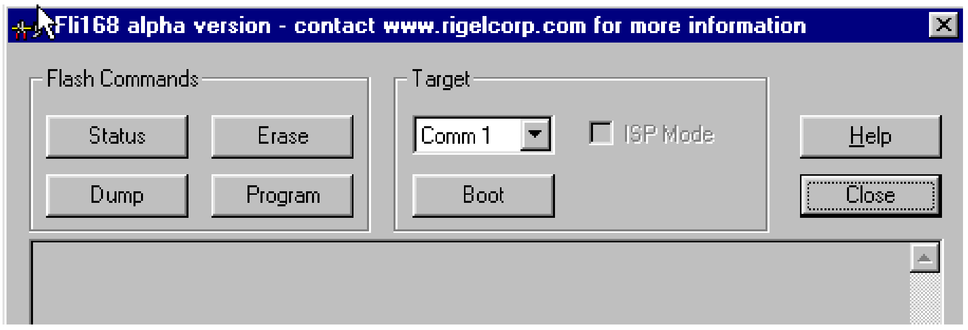
3. Again, from the C Compiler Options dialog, select the "Memory Map" tab and specify the following:

```
Code origin              100
System stack size       1000
System stack base       C000
Syscon                  8480
```

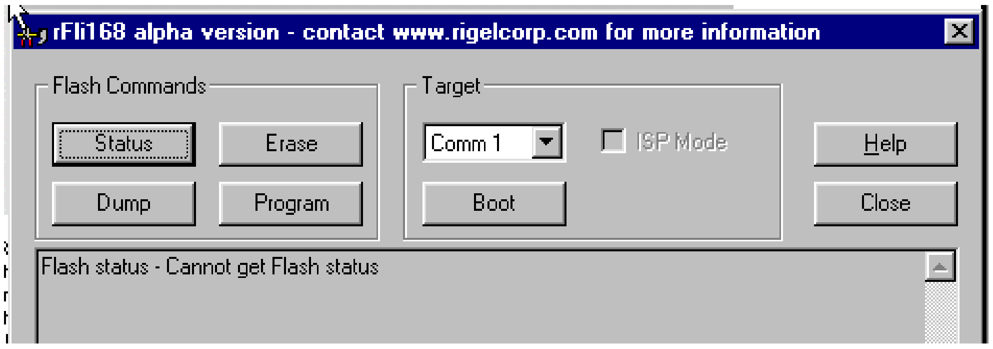
6.3.2 Programming the Internal FLASH

In order to program and run from the internal FLASH on the ST10F168 processor follow these steps.

1. Jumpers P0.4, P0.6, P0.8, P0.9, P0.10, P0.11, EA#, SW4 (CS0#), and SW5 (depending on the RAM size), must be inserted. These are the default jumper settings.
2. Use a 12-15 volt wall transformer in JP15 to power the board, the LED D2 will light up.
3. Reads166 must be closed so that the serial port may be used by the rFLI168 software
4. The rFLI168 Utility Software must be installed and used to bootstrap the board.
5. From the rFLI168 Utility Software select the **Comm Port** you wish to use.


rFli168 alpha version - contact www.rigelcorp.com for more information
Flash Commands
Status
Erase
Dump
Program
Target
Comm 1
ISP Mode
Boot
Help
Close

6. Reset the board and then use the "**Boot"** button to bootstrap the board. During bootstrapping, the special-purpose monitor program is downloaded to the board. At the end of system initialization the LED D4 is turned on. You may verify that the board is responding by observing the monitor prompt "Bootstrapping OK"
7. Use the "**Status"** button to review the current status of the FLASH. The software will report back on status of FLASH:
   "Status OK, Version XXX"
   or "Flash Status – Cannot get Flash Status"

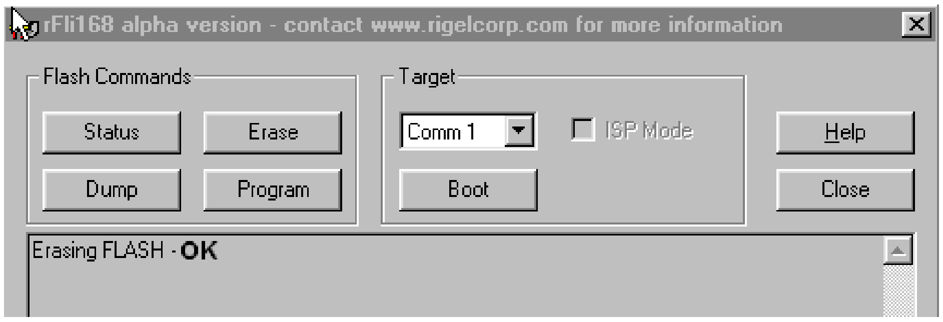

rFli168 alpha version - contact www.rigelcorp.com for more information
Flash Commands
Status
Erase
Dump
Program
Target
Comm 1
ISP Mode
Boot
Help
Close

```
Flash status - Cannot get Flash status
```

Alternatively, you may "Dump" the contents of the internal FLASH, which is then downloaded and written to a file in binary format.
If the memory has data in it, it is recommended that you erase the FLASH before you proceed.

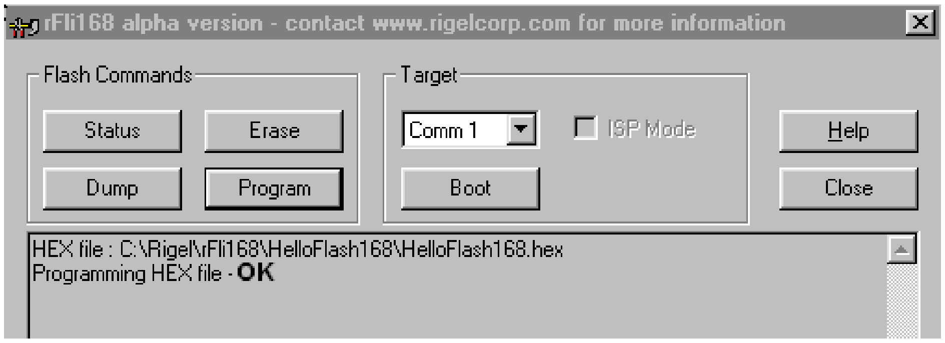
8. JP13-VPPON and JP21-VPP jumpers need to be inserted before you erase, or program the FLASH. The LED D3 will light up when the jumpers are inserted to indicate 12-volts on the board.

9. Once the jumpers are inserted, use the “**ERASE”** button to erase the FLASH memory. When the memory is erased the software sends the message “Erase Flash OK”.


rFli168 alpha version - contact www.rigelcorp.com for more information
Flash Commands
Status
Erase
Dump
Program
Target
Comm 1
ISP Mode
Boot
Help
Close

```
Erasing FLASH - OK
```

10. Use the “**Program”** button to select and download a file into FLASH memory. Input the file name of the HEX file in the dialog box. When the programming is completed the software reports “Programming Hex file OK”


rFli168 alpha version - contact www.rigelcorp.com for more information
Flash Commands
Status
Erase
Dump
Program
Target
Comm 1
ISP Mode
Boot
Help
Close

```
HEX file : C:\Rigel\rFli168\HelloFlash168\HelloFlash168.hex
Programming HEX file - OK
```

11. To check the status of your program in memory you may use the “**Dump”** button. The Dump button downloads your program into a Binary file that allows you to view your program in memory and edit it. You may also download the work buffer back into memory once you've edited it.

6.3.3 Running a Program from Internal FLASH

To run the program at start up, the following steps may be followed.

1. To run the program from internal FLASH remove the jumpers JP13-VPPON and JP21-VPP. The LED D3 should turn off. Remove the EA# jumper. Remove jumper P0.4.
2. Push reset on the board.
3. Your program will now run at start-up. If you’ve used our “HelloWorld” program you may view the prompt “Hello World” by closing the rFLI168 software and opening the Reads166 software and viewing the prompt in the TTY window.
4. You may continue to use the 12-15-volt wall transformer during normal operation of the board.

6.4 Alternate Memory Maps

The RAM, ROM and internal ROM blocks of memory can be mapped in a variety of ways depending on the chip select signals assigned to each bank and on the size of each memory bank programmed into the external bus control registers BUSCON0 and BUSCON1. Refer to the ST / Infineon data books for more information on system programming.

6.5 Default Memory Setting

The default memory map assumes 256K of RAM with no ROM. In this configuration, the board is bootstrapped and run with RAM in low memory. JP1 should have jumpers in locations 4, 6, 8, 9,10 and 11. SW5 should have a jumper in the lower position, and SW4 should have a jumper in the lower RAM position. This is the default configuration for the R-FLIC.

7 HEADERS

The R-FLIC board has five headers: the system bus header JP3, the input/output header JP4, the extra input/output header JP5, and the serial port headers. The system bus header contains the address, data, and control busses, the input/output header contains Ports 2, 3, and 5, and the extra input/output header contains Ports 5, 6, 7, and 8. Individual signals of these jumpers are listed below. Pin 1 may be identified as the post with the square pad on the printed circuit board.

7.1 JP3 - System Bus Header

Signal			Pins		Signal		
167/168	**165**	**166**			**166**	**165**	**167/168**
Ground			1	2	VCC (+5V)		
Ground			3	4	VCC (+5V)		
D0			5	6	A0		
D1			7	8	A1		
D2			9	10	A2		
D3			11	12	A3		
D4			13	14	A4		
D5			15	16	A5		
D6			17	18	A6		
D7			19	20	A7		
D8			21	22	A8		
D9			23	24	A9		
D10			25	26	A10		
D11			27	28	A11		
D12			29	30	A12		
D13			31	32	A13		
D14			33	34	A14		
D15			35	36	A15		
RD#			37	38	A16		
ALE			39	40	A17		
RSTIN#			41	42	WR#	WR# or WRL#	
RSTOUT#			43	44	BHE#		
NMI#			45	46	NC		A18
A21		NC	47	48	NC		A19
A22		NC	49	50	NC		A20

7.2 JP4 - Input/Output Header

Signal			Pins		Signal		
167/168	**165**	**166**			**166**	**165**	**167/168**
Ground			1	2	VCC (+5V)		
Ground			3	4	VCC (+5V)		
P5.0	not used	P5.0	5	6	P5.1	not used	P5.1
P5.2	not used	P5.2	7	8	P5.3	not used	P5.3
P5.4	not used	P5.4	9	10	P5.5	not used	P5.5
P5.6	not used	P5.6	11	12	P5.7	not used	P5.7
P5.8	not used	P5.8	13	14	P5.9	not used	P5.9
VAGND	not used	VAGND	15	16	S1I	not used	
VAREF	not used	VAREF	17	18	S1O	not used	
P2.0	not used	P2.0	19	20	P3.0		
P2.1	not used	P2.1	21	22	P3.1		
P2.2	not used	P2.2	23	24	P3.2		
P2.3	not used	P2.3	25	26	P3.3		
P2.4	not used	P2.4	27	28	P3.4		
P2.5	not used	P2.5	29	30	P3.5		
P2.6	not used	P2.6	31	32	P3.6		
P2.7	not used	P2.7	33	34	P3.7		
P2.8			35	36	P3.8		
P2.9			37	38	P3.9		
P2.10			39	40	P3.10		
P2.11			41	42	P3.11		
P2.12			43	44	P3.12		
P2.13			45	46	P3.13		
P2.14			47	48	P3.14	not used	
P2.15			49	50	P3.15		

7.3 JP5 - Extra Input/Output Header

Signal		Pins		Signal	
167/168	RMB-165i			RMB-165i	167/168
Ground		1	2	VCC (+5V)	
Ground		3	4	VCC (+5V)	
P6.0		5	6	P6.1	
P6.2		7	8	P6.3	
P6.4		9	10	P6.5	
P6.6		11	12	P6.7	
A23	not used	13	14	READY#	
VAGND	not used	15	16	not used	
VAREF	not used	17	18	not used	
P5.0	not used	19	20	not used	P7.0
P5.1	not used	21	22	not used	P7.1
P5.2	not used	23	24	not used	P7.2
P5.3	not used	25	26	not used	P7.3
P5.4	not used	27	28	not used	P7.4
P5.5	not used	29	30	not used	P7.5
P5.6	not used	31	32	not used	P7.6
P5.7	not used	33	34	not used	P7.7
P5.8	not used	35	36	not used	P8.0
P5.9	not used	37	38	not used	P8.1
P5.10		39	40	not used	P8.2
P5.11		41	42	not used	P8.3
P5.12		43	44	not used	P8.4
P5.13		45	46	not used	P8.5
P5.14		47	48	not used	P8.6
P5.15		49	50	not used	P8.7

7.4 Serial Port Headers

The ST10F168 has two serial ports. Serial port 0 is accessed through the RS-232 level converter. Serial port 1 is used as a synchronous serial channel by the R-FLIC.

7.4.1 J3 (S0)

J3 is the secondary access port of serial port 0. J3 is a 3-pin header, which carries the same signals as P1. J3 is also denoted by S0 and its 3 lines by G (Ground), T (Transmit), and R (Receive), on the R-FLIC silk-screen. J3 is intended for embedded uses of the R-FLIC when P1 is not populated. J3 is located above the I/O headers.

7.4.2 J4, J8 (SSC)

Header J4 and the terminal blocks, J8 are the access port for serial port 1. Header J4 carries the 3 TTL-level signals, MRST (Master Receive Slave Transmit), MTRS (Master Transmit Slave Receive), and SCLK (Serial Clock) as well as provides VCC and GND for reference. Header J4 is also denoted by SSC (Serial Synchronous Channel) on the silk-screen. The individual pins of the header are denoted by VCC, R, T, C, and GND, which correspond to VCC, MRST, MTSR, SCLK, and GND, respectively. J4 is located above the I/O headers. The terminal blocks J8 also carry the same signals and are marked as VCC, MR, MT, SC, and GND, which correspond to VCC, MRST, MTSR, SCLK, and GND, respectively.

8 SYSTEM PROGRAMMING

8.1 The SYSCON Register

The system control register (SYSCON) specifies many of the operating parameters of the 167 / 168 microcontroller. While many of these parameters may be changed depending on the application, the BYTDIS bit must be cleared and the Write Configuration Control bit (WRCFG) must be set. With BYTDIS=0, the BHE# signal is generated by the microcontroller. With WRCFG=1, the WR# signal of the microcontroller is used as WRL# and the BHE# as WRH#. These control signals, WRL# and WRH#, in turn are used to enable the odd and even memory devices of the RAM and ROM banks.
The following value is used in the RMON167C monitor program given in the distribution disk,

```
        mov     SYSCON, #01480h  ; 0001:0100:1000:000b
```

which selects the following parameters:

Field Name	Field Description	Value	Function	Remarks
STKSZ	stack size	000	256 words	default
ROMS1	map internal ROM to the first segment	1	map internal ROM in segment 1	
SGTDIS	segmentation disabled	0	disable segmentation (64K memory)	only IP is used (CSP is not used) in calls and returns
ROMEN	ROM enable	1	enable internal ROM	
BYTDIS	Disable BHE#	0	do not disable BHE#	BHE# is used as WRH#
CLKEN	Clock enable	0	CLKOUT disabled	CLKOUT is the alternate function of P3.15
WRCFG	Write configuration	1	WR# is WRL#, BHE# is WRH#	
VISIBLE	Visible mode	0	XBUS accesses are done internally	default
XPER-SHARE	XBUS peripheral share mode	0	External accesses to XBUS peripherals are disabled	default

8.2 The BUSCON0 and BUSCON1 Registers

The width, type of multiplexing, and various wait states of the buses depend on the Bus Control (BUSCON) registers. Each chip select line (CSx#) has a corresponding BUSCONx register. The selection of the various wait states depend on the application and on the speed of the memory devices. Refer to the Infineon Data Book for more information about the fields of BUSCON.

Since CS0# is the chip select line activated upon a reset (a software reset or a hardware reset without bootstrapping), BUSCON0 has the special designation of determining the bus configuration upon reset. The bus type field (BTYP) of BUSCON0 is copied from Port 0 during reset. The values of Port 0 during reset are determined by the jumpers on the R-FLIC board as explained in Section 4.1. If the ROM bank not used, there is no need to program BUSCON1. Otherwise, BUSCON1 should be programmed to have the same bus type as BUSCON0, that is, BTYP=10b for 16-bit demultiplexed. In addition, the bit BUSACT1 of register BUSCON1 must also be set. If BUSACT1 is cleared, CS1# is disabled.

8.3 The ADDRSEL1 Register

Each CSx# line except CS0# has a corresponding ADDRSELx register which determines its offset (beginning address) and its range. CS0# does not have an ADDRSEL register. All addresses outside the ranges collectively defined by ADDRSELx registers are accessed by CS0#.

If you use both the RAM and the ROM banks, then both chip select lines CS0# and CS1# are needed. In this case, register ADDRSEL1 must be set. The following, for example, enables ADDRSEL1 to start at 100000h (offset at 1MB) and have a range of 1MByte.

```
        mov ADDRSEL1, #1008h
```

The actual value depends on the size of RAM and ROM banks. Also note that the SGTDIS bit of the SYSCON register should be cleared if the total memory exceeds 64K

9 BOOTSTRAPPING

The R-FLIC bootstrapping is triggered by grounding P0L.4 at reset. A 32-pin header is used to ground the bits of P0L via a resistor array with a nominal value in the range of 6.8 to 8.2K.
Once the bootstrap loader is invoked the serial port S0 is used to communicate with the 167. The host must first send a 0 byte with 8 data bits, 1 stop bit and no parity bits. The 167 responds with the byte C5h. Then the host expects 32 bytes of code to be downloaded to internal RAM starting at address 0FA40h and run.

Since 32 bytes are not enough to initialize and configure the ST10F168 and then download a user program, a secondary loop is used. This loop is a short piece of code that is placed starting at address 0FA60h, so that when the 32 bytes of primary code are executed, the program continues with the secondary loop. The approach is described in more detail below.
The 32 bytes downloaded are, in hexadecimal,

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

which correspond to the following short code.

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

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

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

The user may alter the number of bytes to be loaded by changing the 21st and 22nd bytes (BB and FC) which give the address (the low byte, followed by the high byte) of the last byte to be read by the loop. Note that there is a practical limit to the number of bytes that can be downloaded by this loop: the PEC source and destination pointers as well as the SFRs which occupy addresses FDE0h and above must not be overwritten by data bytes.
Due to the powerful instruction set of the 167, a lot of functionality can be implemented within 604 bytes of code. The 604 bytes contained in the file BTL67CRI.DAT downloads a minimal monitor program. This

program contains an initialization routine, subroutines to send and receive characters through the serial port, a subroutine to download code in the Intel Hex format, and a subroutine to jump to any location within the 64K segment. The latter two are invoked by single-letter commands.

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

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

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

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

```
            DISWDT
            EINIT
```

whose machine instructions are (`A5 5A A5 A5`) and (`B5 4A B5 B5`),
respectively. The code within the 604-byte download block pokes these bytes starting from address 0. That is, these instructions are placed into memory, one word at a time, as data. The following instructions are used.

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

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

This pattern is used throughout sections 2 and 3. First the word is written to register R1. Then the register is copied to memory. The file BTL67.DAT contains the bytes downloaded to the R-FLIC board during bootstrapping. The file BTL.SRC contains the source code.

The initialization routines configure the SYSCON register. The internal ROM is disabled and the external bus is activated. Next the CSP and DPP registers are initialized. These steps need to be completed before the EINIT instruction. Note that if the watchdog timer is to be disabled, this too must be done before the EINIT instruction.

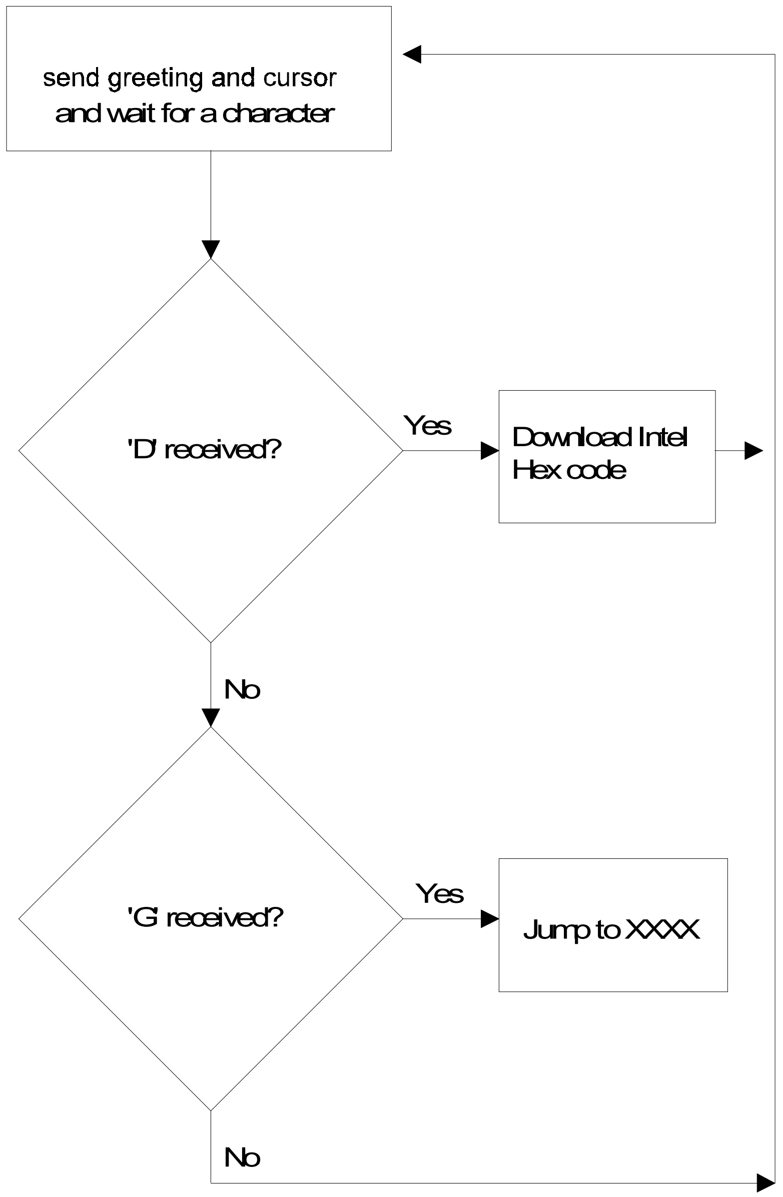
10 THE MONITOR PROGRAMS

10.1 Minimal Monitor

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

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

.



10.2 MON167 Monitor

The monitor program RMON167 allows inspecting and modifying the first 64K segment of R-FLIC memory, configuring the ports, inputting and outputting from the general purpose ports, downloading code in the Intel Hex format, and branching to user code. RMON167 features are invoked by single-letter commands. Serial port 0 is initialized to run at 9600 Baud with 8 bits of data, 1 stop bit and no parity bits. RMON167 is intended to be downloaded after bootstrapping the R-FLIC board. RMON167 is placed starting at address 0C000h. The first 256 bytes are reserved for monitor variables. The entry point to RMON167 is at address C100h. To set up RMON167, initialize READS166 and the R-FLIC board and invoke the Bootstrap command as explained in the previous section. From the TTY menu, select **Download** to download RMON167.HEX. Branch to and execute RMON167 using the **Run** command under the TTY menu. Specify address C100 since the entry point to RMON167 is at 0C000h. Note that RMON167 places a jump to C100 at the nonmaskable interrupt vector. Thus, RMON167 may subsequently be invoked by pressing the NMI pushbutton on the R-FLIC. RMON167 initializes the stack and resets the interrupts. Thus, even after the NMI button is pressed, RMON167 clears the NMI interrupt by executing a dummy 'return from interrupt' instruction.

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

The single-letter commands of RMON167 are explained below.

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

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

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

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

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

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

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

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

W Word Memory

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

11 READS166 VERSION 3.1

11.1 Overview

READS166, version 3.1, is Rigel Corporation's Integrated Development Environment for the ST Microelectronics 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 a cross assembler for the C166 family of microcontrollers. It is intended to be used by the hardware and software products available from Rigel Corporation. Ra66 is a two-pass assembler. Forward references are resolved during the second pass. The second pass is used only when necessary. If no forward references are used, Ra66 completes assembly in a single pass.
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. Rc66 works in conjunction with Ra66: first an assembly language program is generated from the C source then a HEX file is created.

The READS166 software has the following distinctive features:

- Project management for organized software development
- Archive storage for source code modules
- Multiple project management with drag and drop module transfers
- Enhanced graphical user interface for easy monitoring
- Stand alone compiler and editor applications connected to READS166 in a client/server fashion
- Drag and drop code development
- Mixed mode projects (C and assembler)
- Wizard code generator

11.2 READS166 V3.00 CONCEPTS

READS166 introduces a project-oriented code development and management system. The new concepts are defined below.

11.2.1 Project

A project is collection of files managed together. Each file in a project corresponds to a code module. All projects are kept in their individual subdirectories. You may copy or save projects as a single entity. When saved under a different name, a new subdirectory is created and all components of the project are duplicated in the new subdirectory.

By using the long names provided by the 32-bit Windows operating systems, you may use this feature to keep different versions of your software in a controlled manner. For example, the project "Motor Control 07-20-1997" may be saved under the name "Motor Control 07-25-1997" as new features are added. This way, if needed, you may revert to an older version.

A project may either be an “executable project” or an “archive project.”

Executable Projects
Executable projects are meant to be compiled into code which is eventually run on the target system. Components of an executable project are the code modules containing subroutines or functions which make up the entire program.

Archive Projects
Archive projects are never compiled. They are intended to facilitate code reusability by organizing and keeping code modules together. An archive project acts as a repository which you may add modules to, or copy modules from. Executable projects can be quickly constructed using already written and debugged modules from an archive project.

11.2.2 Module

A module is a single file that belongs to a project. Typically modules are either assembly language subroutines or C language functions. You may copy modules from one project to another, or share modules in different projects. For example, you may copy a previously developed module from an archive project to an executable project by simply dragging its icon from one project window to the other. By using existing or previously developed and debugged modules, you may significantly improve code reusability, much in the same manner as libraries. Reusing modules differs from using library functions of existing routines in that modules are kept in source form rather than object form.

For more information on the READS166 software please refer to the READS166 Version 3.0 Users Manual. The entire manual is available in PDF format from our Web site www.rigelcorp.com.

12 PARTS LIST

12.1 Assembly Parts List

QUANTITY	PART	DESIGNATOR	
	CAPICATORS		
1	Surface Mount 1nF	C30	
34	Surface Mount 10nF	C1-C28, C41-47, C49	
1	1uF MONO	C29	
4	1.0uF	C32-35	
1	22uF	C31	
5	47uF	C36-39, C48	
2	100uF	C40, C46	
1	220uF	C44	
	DIODES		
1	1N4001	D1	
2	Surface Mount RED LED	D2, D3	
1	Surface Mount GREEN LED	D4	
1	2N7000	Q1	
1	BRIDGE	D5	
1	EMI FILTER		NOT POPULATED
	CONNECTORS		
3	25x2 HEADER	JP3, 4, 5	
1	3.5MM DC JACK	JP16	
1	DB 9 (short)	P1	
2	SM PUSH BUTTON	SW1, SW2	
4	1X2 HEADER	J6, J7, JP13, JP14	
4	SM TERM BLOCK	JP6, J5, JP15	
3	SM TERM BLOCK	J8	
4	1X3 HEADER	J1, J2, J3, SW5	
2	2x3 HEADER	SW3/SW4, JP21	
1	2X4 HEADER	JP17-JP20	NOT POPULATED
1	1X5 HEADER	JP4	
1	2x16 HEADER	JP1	
2	ICE CONN	JP2	NOT POPULATED
1	1X4 HEADER	10/11	NOT POPULATED
1	1X2 HEADER	12	NOT POPULATED
1	2x3 HEADER	7/8/09	NOT POPULATED
	RESISTORS		
1	330 OHM ½ W	R6	

1	100 OHM 1/4W	R1	NOT POPULATED
1	10K 1/4W	R11	
2	10K 1/4W	R9, 10	NOT POPULATED
1	8.2K GANG (10)	R4	
1	8.2K GANG (8)	R5	
1	20K POT	R3	
1	10K (6 GANG)	R2	
2	1K	R7, R8	
	VARISTOR		
2	ZNR	V1, V2	
	SOCKETS		
1	8 PIN SOCKET	U1	
1	16 PIN SOCKET	U4	
2	32 PLCC	U6, U7	
2	32 PIN SOCKET	U8, U9	
	ICS		
1	ST10F168/ 167CR	U3	
1	PCA82C250TD	U5	
1	DS1233	U2	
1	LM7805-TO220	U10	
1	LM7812-TO92	U11	
1	RS232	U4	
2	62C256 / 6281000	U8. U9	
1	5 MHz CLOCK	U1	

12.2 Part Cross Reference, December 2000

	Designator	Component	Library Reference Sheet
1	C1	10nF	CPU.SCH
2	C2	10nF	PWR.SCH
3	C3	10nF	PWR.SCH
4	C4	10nF	PWR.SCH
5	C5	10nF	PWR.SCH
6	C6	10nF	PWR.SCH
7	C7	10nF	PWR.SCH
8	C8	10nF	PWR.SCH
9	C9	10nF	PWR.SCH
10	C10	10nF	PWR.SCH
11	C11	10nF	PWR.SCH
12	C12	10nF	PWR.SCH
13	C13	10nF	PWR.SCH
14	C14	10nF	PWR.SCH
15	C15	10nF	PWR.SCH
16	C16	10nF	PWR.SCH
17	C17	10nF	PWR.SCH
18	C18	10nF	PWR.SCH
19	C19	10nF	PWR.SCH
20	C20	10nF	PWR.SCH
21	C21	10nF	PWR.SCH
22	C22	10nF	PWR.SCH
23	C23	10nF	PWR.SCH
24	C24	10nF	PWR.SCH
25	C25	10nF	PWR.SCH
26	C26	10nF	PWR.SCH
27	C27	10nF	PWR.SCH
28	C28	10nF	PWR.SCH
29	C29	1uF	CPU.SCH
30	C30	1nF	CPU.SCH
31	C31	22uF	CPU.SCH
32	C32	1uF	SER.SCH
33	C33	1uF	SER.SCH
34	C34	1uF	SER.SCH
35	C35	1uF	SER.SCH
36	C36	47uF	PWR.SCH
37	C37	47uF	PWR.SCH
38	C38	47uF	PWR.SCH
39	C39	47uF	PWR.SCH
40	C40	100uF	PWR.SCH
41	C41	10nF	PWR.SCH
42	C42	10nF	PWR.SCH
43	C43	10nF	PWR.SCH
44	C44	220uF	PWR.SCH
45	C45	10nF	PWR.SCH
46	C46	100uF	PWR.SCH
47	C47	10nF	PWR.SCH
48	C48	47uF	PWR.SCH
49	C49	10nF	PWR.SCH
50	D1	1N4001	PWR.SCH
51	D2	PWR	PWR.SCH
52	D3	VPP	PWR.SCH

53	D4	RSTOUT	ICE.SCH
54	D5	BR	PWR.SCH
55	J1	VAREF	CPU.SCH
56	J2	VAGND	CPU.SCH
57	J3	RS232	SER.SCH
58	J4	SSC	SER.SCH
59	J5	CAN	SER.SCH
60	J6	CANTXD	SER.SCH
61	J7	CANRXD	SER.SCH
62	JP1	RSTOPT	OPT.SCH
63	JP2	ICE	ICE.SCH
64	JP3	MEMORY	PIO.SCH
65	JP4	I/O PORTS	PIO.SCH
66	JP5	XI/O PORTS	PIO.SCH
67	JP6	POWER	PWR.SCH
68	JP7	NMI	CPU.SCH
69	JP8	RIN	CPU.SCH
70	JP9	ROUT	CPU.SCH
71	JP10	RD	CPU.SCH
72	JP11	RDY	CPU.SCH
73	JP12	HOLD	CPU.SCH
74	JP13	VPPON	PWR.SCH
75	JP14	EA#	CPU.SCH
76	JP15	13.5VAC	PWR.SCH
77	JP16	13.5VAC	PWR.SCH
78	JP21	VPPOPT	PWR.SCH
79	L1	EMI	PWR.SCH
80	P1	RS232	SER.SCH
81	Q1	2N7000	ICE.SCH
82	R1	100	CPU.SCH
83	R2C	10K	MEM.SCH
84	R2B	10K	MEM.SCH
85	R2A	10K	CPU.SCH
86	R3	RS	SER.SCH
87	R4	8.2K	OPT.SCH
88	R5	8.2K	OPT.SCH
89	R6	330	PWR.SCH
90	R7	1K	PWR.SCH
91	R8	1K	ICE.SCH
92	R9	10K	SER.SCH
93	R10	10K	SER.SCH
94	R11	10K	CPU.SCH
95	SW1	RESET	CPU.SCH
96	SW2	NMI	CPU.SCH
97	SW3	ROMMAP	MEM.SCH
98	SW4	RAMMAP	MEM.SCH
99	SW5	RAMSIZE	MEM.SCH
100	U1	5MHz	CPU.SCH
101	U2	DS1233	CPU.SCH
102	U3	10F167/168	CPU.SCH
103	U4	MAX232	SER.SCH
104	U5	82C250	SER.SCH
105	U6	AM29F040	MEM.SCH
106	U7	AM29F040	MEM.SCH
107	U8	HM628512	MEM.SCH

108	U9	HM628512	MEM.SCH
109	U10	LM7805	PWR.SCH
110	U11	LM7812	PWR.SCH
112	V1	ZNR	PWR.SCH
113	V2	ZNR	PWR.SCH

13 TOP OVERLAY AND SCHEMATICS