

Copyright 1994 - Remote Processing Corporation. All rights reserved. However, any part of this document may be reproduced with Remote Processing cited as the source.

The contents of this manual and the specifications herein may change without notice.

## TRADEMARKS

RPBASIC-52™ is a trademark of Remote Processing Corporation.

PC SmartLINK™ is a trademark of Octagon Systems Corporation.

Microsoft® BASIC is a trademark of Microsoft Corporation.

BASIC-52 is a trademark of Intel Corporation.

## NOTICE TO USER

The information contained in this manual is believed to be correct. However, Remote Processing assumes no responsibility for any of the circuits described herein, conveys no license under any patent or other right, and make no representations that the circuits are free from patent infringement. Remote Processing makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification. The user must make the final determination as to fitness for a particular use.

Remote Processing Corporation's general policy does not recommend the use of its products in life support applications where the failure or malfunction of a component may directly threaten life or injury. It is a Condition of Sale that the user of Remote Processing products in life support applications assumes all the risk of such use and indemnifies Remote Processing against all damages.

P/N 1085  
Revision: 1.2

# TABLE OF CONTENTS

Chapter 1 Overview		Table 6-1 Connector pin out - J4 . . . . .	21
DESCRIPTION . . . . .	1	COMMANDS . . . . .	22
MANUAL ORGANIZATION . . . . .	1	Chapter 7 Calendar/Clock	
MANUAL CONVENTIONS . . . . .	1	DESCRIPTION . . . . .	23
Symbols and Terminology . . . . .	1	SETTING DATE AND TIME . . . . .	23
TECHNICAL SUPPORT . . . . .	2	GENERATING INTERRUPTS . . . . .	23
Chapter 2 Setup and Operation		COMMANDS . . . . .	23
INTRODUCTION . . . . .	3	Chapter 8 Display Port	
OPERATING PRECAUTIONS . . . . .	3	INTRODUCTION . . . . .	24
EQUIPMENT . . . . .	3	CONNECTING DISPLAYS . . . . .	24
FIRST TIME OPERATION . . . . .	4	WRITING TO THE DISPLAY . . . . .	24
UPLOADING AND DOWNLOADING		PROGRAMMING EXAMPLE . . . . .	24
PROGRAMS . . . . .	4	DISPLAY TYPES . . . . .	25
Uploading programs using PC SmartLink . . . . .	4	DISPLAY CONNECTOR PIN OUT - J6 . . . . .	25
Downloading programs using PC		COMMANDS . . . . .	25
SmartLink . . . . .	5	Chapter 9 Keypad Port	
Other communications software . . . . .	5	INTRODUCTION . . . . .	26
Editing programs and programming hints . . . . .	6	PROGRAMMING EXAMPLE . . . . .	26
WHERE TO GO FROM HERE . . . . .	6	KEYPAD PORT PIN OUT - J5 . . . . .	27
TROUBLESHOOTING . . . . .	7	COMMANDS . . . . .	27
Chapter 3 Saving Programs		Chapter 10 Analog I/O	
INTRODUCTION . . . . .	8	DESCRIPTION . . . . .	28
SAVING A PROGRAM . . . . .	8	CONNECTING ANALOG I/O . . . . .	28
AUTORUNNING . . . . .	8	Analog I/O J1 pin out . . . . .	28
PREVENTING AUTORUN . . . . .	8	Grounding . . . . .	28
Chapter 4 Serial Ports		ACQUIRING ANALOG DATA . . . . .	28
DESCRIPTION . . . . .	10	Reducing noise . . . . .	29
COM0 SERIAL PORT . . . . .	10	Data logging on a timer tick . . . . .	29
COM1 SERIAL PORT . . . . .	10	MEASURING HIGHER VOLTAGES . . . . .	29
RS-422/485 Termination network . . . . .	10	Converting analog measurements . . . . .	29
TWO WIRE RS-485 . . . . .	11	Measuring 4-20 mA current loops . . . . .	30
MULTIDROP NETWORK . . . . .	11	CALIBRATION . . . . .	30
ACCESSING SERIAL BUFFERS . . . . .	12	ANALOG OUTPUT . . . . .	30
ACCESSING COM0 AND COM1 . . . . .	12	Programming . . . . .	31
COMMANDS . . . . .	13	COMMANDS . . . . .	31
Chapter 5 RAM Memory		PROGRAM EXAMPLE . . . . .	31
INTRODUCTION . . . . .	14	Chapter 11 Watchdog Timer	
CHANGING MEMORY . . . . .	14	Description . . . . .	32
BATTERY BACKUP . . . . .	14	Chapter 12 User Jumpers	
RESERVING MEMORY . . . . .	15	DESCRIPTION . . . . .	33
STORING VARIABLES IN RAM . . . . .	15	Chapter 13 Serial EEPROM	
CORRUPTED VARIABLES . . . . .	15	DESCRIPTION . . . . .	34
ASSEMBLY LANGUAGE INTERFACE . . . . .	16	PROGRAM EXAMPLE . . . . .	34
COMMANDS . . . . .	16	Technical Information	
Chapter 6 Digital and Opto Ports		ELECTRICAL SPECIFICATIONS . . . . .	35
INTRODUCTION . . . . .	17	MEMORY AND I/O bank map . . . . .	35
ON-CARD OPTO RACK . . . . .	17	MECHANICAL SPECIFICATIONS . . . . .	35
Description . . . . .	17	JUMPER DESCRIPTIONS . . . . .	36
Installation . . . . .	17	Appendix A RPBASIC -52 Software Supplement	
G4 operation . . . . .	18		
PWM Output . . . . .	18		
G5 operation . . . . .	18		
Converting analog measurements . . . . .	19		
DIGITAL I/O PORT . . . . .	19		
Digital I/O commands . . . . .	19		
High current output . . . . .	19		
Interfacing digital I/O to an opto-module			
rack . . . . .	20		
Interfacing to switches and other devices . . . . .	21		
Digital I/O programming example . . . . .	21		

**DESCRIPTION**

The RPC-52 is an embedded controller with a built in Basic language. Several features make it suitable as a stand alone unit:

- ◆ Built in RPBASIC-52 programming language autoruns at power up. On card EEPROM programmer saves programs to 32K.
- ◆ Four position opto rack accepts G4 and G5 series modules. The G4 series is the industry standard digital I/O module. G5 modules are optically isolated analog.
- ◆ LCD character and graphic display and keypad ports for operator interface.
- ◆ Two RS-232 serial ports, one of which can be configured for RS-422/485.
- ◆ Watchdog timer resets the card if the program "crashes".
- ◆ 24 general purpose digital I/O lines, 8 of which are high current outputs. These lines can connect to another opto rack.
- ◆ Calendar/clock is battery backed and keeps track of date and time even when power is off.
- ◆ 32K, 128K, or 512K RAM is battery backed to save process variables and other data when power is off.
- ◆ A serial EEPROM saves program constants in a secure media.

The RPC-52 uses an 80C552 CPU operating at 22.1184 Mhz. It can operate stand alone or on a network using the RS-485 port. Its 4.7" x 7" size with 4 mounting holes makes it easy to mount in a NEMA box. Compactness is enhanced by the 4 on-board opto module slots and operator interface.

RPBASIC-52 programming language is standard. This language is a reassembled version of the original Intel BASIC-52. It was modified for the RPC-52 for control, data acquisition applications, and on board hardware features.

Program development can take place on your PC, using your word processor, or on the RPC-52. Programs from your PC can be downloaded using PC SmartLINK or other serial communication program.

**MANUAL ORGANIZATION**

This manual provides all the information required to install, configure, and operate the RPC-52. Using this manual you will be able to:

Interface the RPC-52 to your IBM compatible PC or terminal.

Understand the operation of the RPC-52 hardware using RPBASIC-52 programming software.

This manual assumes you are familiar with some type of BASIC programming software. The syntax used by RPBASIC-52 is similar to BASIC-52. If you are not experienced with any BASIC software, you may want to refer to books and training programs available through your local software store. The *BASIC-52 Programming Manual* has information and examples for the original commands. Commands unique or modified by RPBASIC-52 are in the Software Supplement in this manual.

The RPC-52 uses a Signetics/Philips 80C552 processor. Additional information can be obtained from Signetics (800-227-1817), or your distributor. Information about the 80C552 processor is in data handbook IC20.

**MANUAL CONVENTIONS**

Information appearing on your screen is shown in a different type.

Example:

```
RPBASIC-52 V1.0
Copyright Intel (1985) and Remote Processing
Bytes free: 27434
```

**Symbols and Terminology**

**NOTE:** Text under this heading is helpful information. It is intended to act as a reminder of some operation or interaction with another device that may not be obvious.

**WARNING:**

Information under this heading warns you of situations which might cause catastrophic or irreversible damage.

W[-] Denotes jumper block pins.

< xxx > Paired angle brackets are used to indicate a specific key on your keyboard. For example < esc > means the escape key.

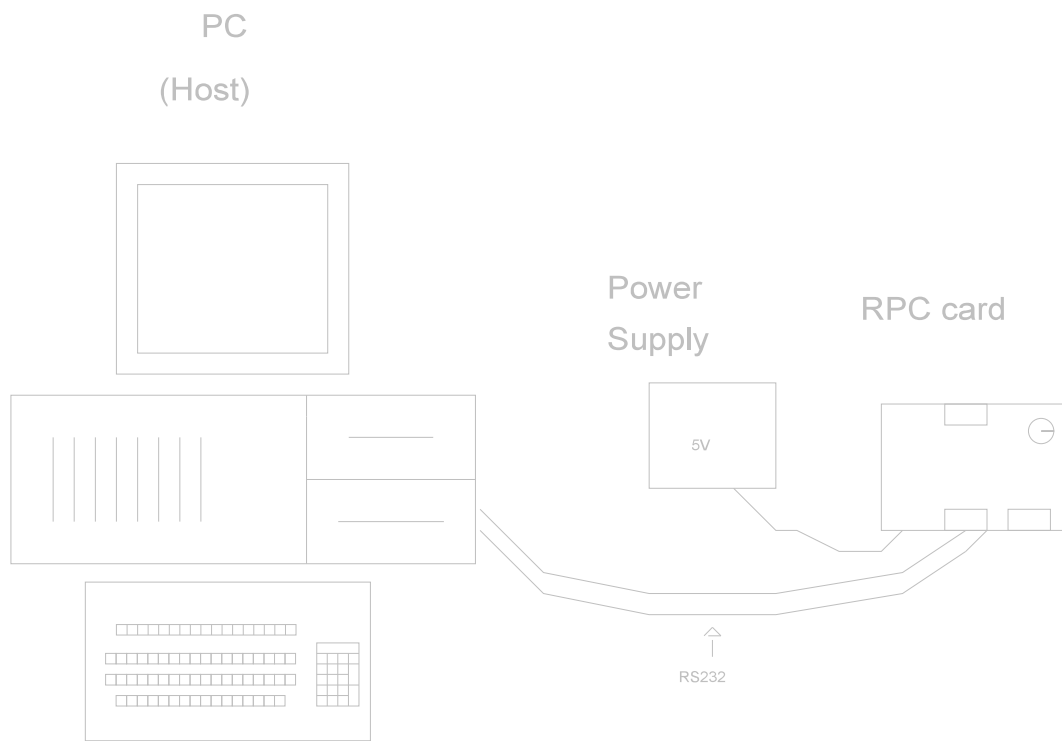
BASIC uses the decimal convention for designating addresses and data. There are times, however, when hexadecimal notation is more convenient to use. The hexadecimal notation used in this manual and by RPBASIC-52 is the 'H' character after the number. A 8CH stands for 8C hexa decimal.

**TECHNICAL SUPPORT**

If you have a question about the RPC-52 or RPBASIC-52 and can't find it in this manual, call us and ask for technical support. Technical support hours are 9 AM to 4 PM mountain time.

When you call, please have your RPC-52 and *BASIC-52 Programming Manual* ready. Many times it is helpful to know what the RPC-52 is used for, so please be ready to describe its application as well as the problem.

Phone: 303-690-1588  
FAX: 303-690-1875



**Figure 1-1 System layout**

**INTRODUCTION**

The RPC-52 is ready to program as soon as you connect it to a terminal or PC and apply power. This chapter describes what is needed to get a sign- on message and begin programming.

Requirements for uploading and downloading programs are discussed. A "Where to go from here" section tells you what chapters to refer to in order to use the various capabilities of the RPC-52. Finally, a troubleshooting section helps out on the most common problems.

**OPERATING PRECAUTIONS**

The RPC -52 is designed to handle a wide variety of temperature ranges at low power. These characteristics require using CMOS components. CMOS is static sensitive. To avoid damaging these components, observe the following precautions before handling the RPC-52.

1. Ground yourself before handling the RPC-52 or plugging in cables. Static electricity

can easily arc through cables and to the card. Simply touching your PC before you touch the card can greatly reduce the amount of static.

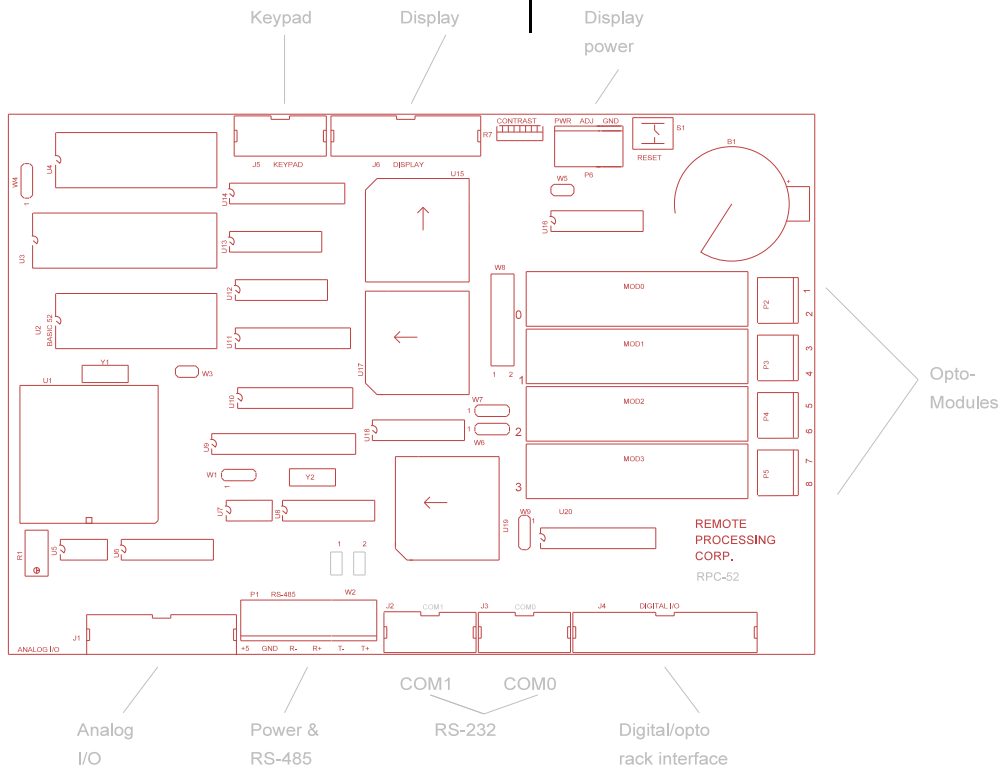
2. Do not insert or remove components when power is applied. While the card is a + 5 volt only system, other voltages generated on the card, which affect other components.

**EQUIPMENT**

You will need the following equipment to begin using the RPC-52:

- RPC-52 embedded controller
- PC with a serial port and communications program (such as PC SmartLINK) or a
- Terminal
- VTC-9F serial cable
- + 5, 300 ma power supply

Refer to *Chapter 4, Serial Ports*, for wiring information to make your own serial cable.



**Figure 2-1 Connector location and function**

## FIRST TIME OPERATION

Become familiar with the locations of connectors before getting started. See Figure 2-1.

RPC-52 jumpers have been set at the factory to operate the system immediately. For first time operation, do not install any connectors or parts unless specified below. Jumpers should be kept in default positions.

### 1. Connect power.

The RPC-52 needs + 5 ±0.25 volts at 200 ma. Any well regulated supply that supplies this will work. Be careful when using "switching" power supplies. Some of these supplies do not regulate properly unless they are adequately loaded. Don't forget that power requirements increase when opto modules are installed. G4 opto modules require up to 20 ma each. The G5 series requires about 130 - 150 ma per module.

Make sure power is off. Connect the power supply to the appropriately marked terminals on the RPC-52.

### 2. Hook up to a PC or terminal.

You can use either a PC or CRT terminal to program the RPC-52. Connect one end of the VTC-9F connector to the 10 pin COM0 port on the RPC-52. Refer to Figure 2-1 for connector location. (COM0 may be marked as COM1 on the silkscreen).

#### Using a PC

Connect the VTC-9F serial cable to the PC's COM1 or COM2 port. You may need a 9 pin male to 25 pin female adapter. The VTC-9F is designed to plug directly into the 9 pin serial port connector on a PC.

Start up your serial communication program (PC SmartLINK or other). Set communication parameters to 9600 baud, 8 data bits, no parity, 1 stop.

#### Using a Terminal

Follow your terminal instructions to set the baud rate to 9600 baud, 8 data bits, no parity, and 1 stop. You may need a 9 pin male to 25 pin male adapter to connect the VTC-9F. If you are using a cable from the terminal, check the connector's sex to determine the type of adapter needed.

### 3. Power up.

Turn on your power supply. On power up a copyright message is printed.

RPBASIC-52 V1.03  
Copyright Intel (1985), Remote Processing (1993)  
Bytes free: 28208

If a nonsense message appears, your terminal or PC may not be set to the appropriate communication parameters. If the system still does not respond, refer to TROUBLE SHOOTING later in this chapter.

### 4. Testing.

The system is now in the "immediate mode" and is ready for you to start programming. Type the following program:

```
10 FOR X=0 TO 2
20 PRINT "Hello ",
30 NEXT
40 PRINT
```

Now type RUN

The system will display:

```
Hello Hello Hello
READY
>
```

You may terminate a program by typing a < Ctrl> -C.

## UPLOADING AND DOWNLOADING PROGRAMS

Downloading programs means transferring them from your PC (or terminal) to the RPC-52. Uploading means transferring them from the RPC-52 back to the PC. This section explains how to do both of these procedures using PC SmartLink. Generalized instructions for other terminal programs are given at the end of this section.

### Uploading programs using PC SmartLink

In the previous section, you wrote a test program. To upload that program to a PC (using PC SmartLink) and save it to disk:

1. Press the < F1> key. A window with the main menu will appear.
2. Press the letter U (upper or lower case). Your program will begin to transfer from RAM to the PC. When menu appears.
3. To save a program to disk, type the letter S. You are prompted for a file name. Enter the file name you want the program saved under.
4. Press < F2> to return to the immediate mode.

**NOTE:** Some versions of PC SmartLINK have pull down menus or will operate differently. Refer to the SmartLINK manual for the version you are using.

### Downloading programs using PC SmartLink

To practice downloading a program, type

```
new<return>
```

Perform the following when using PC SmartLINK:

1. Press the < F1> key to view the main menu.
2. SmartLINK has a buffer which is used to temporarily store the program. If you followed these instructions without exiting SmartLINK, the previously uploaded program is in the buffer and may be downloaded. However, lets assume you just started SmartLINK. Press the L key to get the program from the disk.
3. Enter the filename you saved it under then press the < F2> key.
4. Press D to download the program.
5. Press the < F2> key to return to the programing mode. List the program by typing:

```
list
```

**NOTE:** Some PC SmartLINK versions drop the first line of uploaded code. To make sure the entire program is uploaded, make the first program line a REM.

### Other communications software

The following is general information when using another terminal emulation program (Procomm, Windows Terminal, etc.).

When uploading or downloading files, select ASCII text format. XMODEM, YMODEM, or other formats are not used.

RPBASIC-52 does not know when you are typing in a program or if something else (laptop or mainframe) is sending it characters. The upload and download file does not contain any special codes; they are simply ASCII characters.

Uploading programs is simply a process of receiving an ASCII file. You or your program simply need to send "LIST" to receive the entire program. The default baud rate (9600) is rather high. Make sure your PC and communications software can work at these baud rates. PROCOMM was tested on a 12 Mhz 286 PC and it worked fine. Windows Terminal on the same PC had problems at much slower baud rates.

Downloading a program requires transmitting an ASCII file. As you type in (or download) a line, RPBASIC-52 tokenizes that line. The time to do this depends upon its complexity and how many lines of code have been entered.

RPBASIC-52 must finish compiling a line before starting the next one. When a line is compiled, a "> " character is sent. This should be your terminal programs pacing character when downloading a program.

If your communications program cannot look for a pacing prompt, set it to delay transmission after each line is sent. A 100 ms delay is usually adequate, but your program may be long and complex and require more time. A result of a short transmission time is missing or incomplete program lines.

COM0 on the RPC-52 does not recognize the CTS or RTS lines. The CTS line is pulled high on the RPC-52. The effect of not recognizing these lines is your PC or terminal cannot hold off the RPC-52's transmission. Converse, the RPC-52 cannot hold off the host from sending it data.

## Editing programs and programming hints

Files uploaded or downloaded are simply ASCII DOS text files. No special characters or control codes are used. You may create and edit programs using your favorite word processor or editor. Just be sure to save files in DOS text format.

A technique used to further program documentation and reduce code space is the use of comments in a downloaded file. For example, you could have the following in a file written on your editor:

```
REM Check position

REM Read output from the pot and
REM calculate the position

2200 a = ain(0) :REM Get position
```

The first 3 comments downloaded to the RPC-52 are ignored. Similarly, the empty lines between comments are ignored. Line 2200, with its comment, is a part of the program and could be listed. The major penalty by writing a program this way is increased download time.

**NOTE:** Some versions of PC SmartLINK may optionally strip comments before downloading. Check your manual to see if this option is available.

Notice that you can write a program in lower case characters. RPBASIC-52 translates them to upper case.

Some programmers put "NEW" as the first line in the file. During debugging, it is common to insert "temporary" lines. This ensures that these lines are gone. Downloading time is increased when the old program is still present.

If you like to write programs in separate modules, you can download them separately. Modules are assigned blocks of line numbers. Start up code might be from 1 to 999. Interrupt handling (keypad, serial ports) might be from lines 1000 to 1499. Display output might be from 1500 to 2500. The programmer must determine the number of lines required for each section.

RPBASIC-52 automatically formats a line for minimum code space. For example, you could download the following line of code:

```
10 for a= 0to5
```

When you listed this line, it would appear as:

```
10 FOR A=0 TO 5
```

Spaces are displayed but not stored. The following line:

```
10 for a = 0 to 5
```

will be compressed and displayed as in the second example above. Spaces are removed. However, spaces as part of a remark or PRINT are not removed.

Instead of uploading and downloading programs, you can save them to the on card EEPROM. This is useful if you are using a terminal to write programs. Simply type SAVE. To retrieve a program, type LOAD.

## WHERE TO GO FROM HERE

If you want to do this:	Turn to Chapter
Save a program	3
Run a program at power up or reset (autorun)	3
Know more about serial ports	4
Install a different RAM memory chip	5
Using RAM to save variables	5
Run an assembly language program	5
Configure digital I/O lines	6
Detect on/off switch status	6
Use high current outputs	6
Use on board opto rack	6
Connect an external opto rack	6
Learn to use G5 module	6
Use the calendar/clock	7
Displays	8
Keypad	9
Analog I/O	10

Refer to the table of contents for a more detailed listing.

**TROUBLESHOOTING**

You would probably turn to this section because you could not get the sign on message. If you are getting a sign on message but can't enter characters, then read the end of this section. The following are troubleshooting hints when you can't get anything.

1. Check the power source. If it is below 4.65 volts at the input power terminal, the RPC-52 will reset. Power is  $5 \pm 0.25$  volts. Make sure it is a clean 5 volt source. If it dips intermittently to 4.65 volts (due to switching noise or ripple), the card will reset for about 100 ms. If the noise is frequent enough, the card will be in permanent reset. Check U16, pin 15. If it is low (about 0 volts), then it is in reset. This line should be high (about + 5 volts).
2. Check the COM0 port (J3). Remove the connector from COM0. Refer to the outline drawing earlier in this chapter. Connect an oscilloscope (preferred) or a voltmeter to pin 3 (Txd) and ground. Pin 3 should be -6 volts or more negative. (Pin 1 is designated by the  $\nabla$  symbol on the connector. Pin 3 is next to it, nearer the key opening.) If you have -6 volts or more, press the reset switch. If you have a scope attached, you should see a burst of activity. With a volt meter, you should see a change in voltage. Using a Fluke 8060A set to measure AC, you should see a momentary reading above 2 volts. Press reset several times to make sure it captures it.
3. Install the cable and make sure the voltages and output activity are still there. Output is from pin 3 on the VTC-9F. If not, check to make sure something is not shorting the output.
4. Check the serial parameters on your PC or terminal. They should be set to:  
  
9600 baud, no parity, 8 data bits, 1 stop
5. If you are receiving a sign on message but not able to enter characters, check U10, pin 5 for at least -6 volts. When it is near 0 volts, the terminal or PC's Tx line is not connected to the card. When you press a character on the terminal or PC, you should see the voltage go positive.

If all of this fails, call technical support listed at the front of the book.

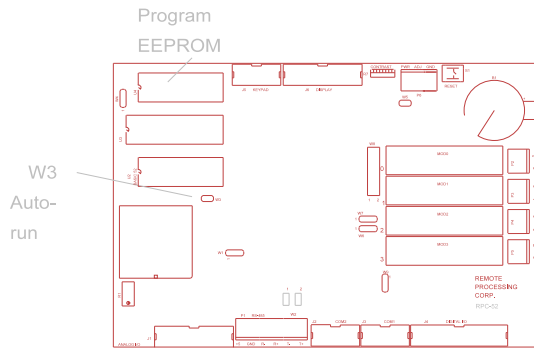
**INTRODUCTION**

Programs are stored in an EEPROM in socket U4. You can store one program up to 32K bytes. A general rule to determine program memory requirements is one line requires 40 bytes. 32K bytes would store 800 lines of code. Your application could be significantly more or less, depending upon the number of commands/line, comments, and print statements.

Despite the fact you may have a 128K or 512K RAM installed, the maximum program size RPBASIC-52 can run is about 60K (including room for variable storage). Only one program can be stored on the EEPROM, and this is limited to about 32K.

An EEPROM is non-volatile (retaining data even when power is disconnected), having an unlimited number of read cycles and a limited number of write cycles (about 1,000). A program is not run from EEPROM. It is transferred to RAM and run from there. Programs in RAM can be modified. They can be saved to EEPROM for auto execution later.

The RPC-52 can be set to autorun on power up or reset by installing a jumper (W3). When autorun is on, the program in EEPROM is loaded into RAM and begins to execute immediately.



**Figure 3-1 W3 autorun jumper**

The RPC-52 has two EEPROMs. One is used for program storage (U4). This is the one under discussion in this chapter. The other is a serial EEPROM used to save various RPBASIC-52 and user parameters (U7). The serial EEPROM is discussed in chapter 13.

This chapter discusses saving programs to EEPROM (U4) and program autoexecution.

**SAVING A PROGRAM**

For this example, assume you wanted to save the following program:

```
10 FOR N= 0 TO 2
20 PRINT "Hello ",
```

```
30 NEXT
40 PRINT
```

If this program is not already in, type it in now (or, if you prefer, use your own program).

Type in the following command:

```
SAVE
```

RPBASIC-52 responds with:

```
Saving 35 bytes
Verifying --- OK
```

The time it takes save a program depends upon the length and complexity of the program. Programming rate is about 600 bytes/second. If the program does not successfully save to the EEPROM, an error message will appear.

Saving a program overwrites the previous one. There is no way to recover the old one since both occupy the same space.

**AUTORUNNING**

To autorun a program:

1. Make sure there is a program in EEPROM (from above).
2. Install jumper W3.

If you push the reset button, the program should autoexecute. If there are any errors, the program will stop (assuming you have not trapped them with ON ERROR) and display the error message.

**PREVENTING AUTORUN**

When troubleshooting a program, it's not always convenient for an autoexecute file to run. This is especially true if the program has been configured to ignore the < ESC> or < Ctl-C> keys. To prevent autorun, remove jumper W3 before power up or reset.

**LOADING A PROGRAM**

There are times when you may wish to temporarily modify or otherwise test out a change to a program. Since the program is loaded into RAM in autorun, modifications can be made without affecting the program in EEPROM. Use the LOAD command to transfer the EEPROM program to RAM.

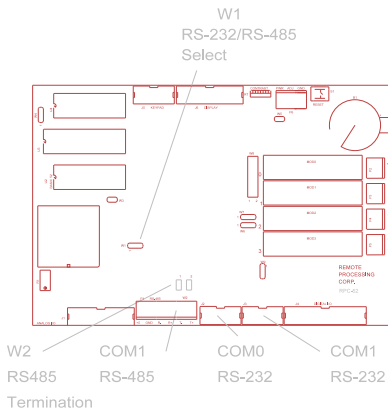
If you find out that modifications are not desirable or did not work, you can restore the original program to RAM using the LOAD command.

**DESCRIPTION**

The RPC-52 has two serial ports that can be used for interfacing to a printer, terminal, RS-485 network, or other serial devices. This chapter describes their characteristics and how to use them. Frequent references are made to commands listed in the *BASIC-52 Programming Manual* or *RPBASIC-52 Software Supplement* in this manual. Please refer to these manuals for more information about these commands.

Serial ports are numbered COM0 and COM1. COM0 is RS232 only and is used for program development. During run time, it can be used for other functions. COM1 is a general purpose port and can be used as either RS-232 or RS-422/485.

Each port has a 256 character interrupt driven input and output buffer. This allows characters to be sent out (using PRINT) without slowing down program execution. However, if the PRINT buffer fills, program execution is suspended until all characters are in the buffer. Both ports have a 256 character input buffer. When more than 256 characters are received, excess ones are ignored.



**Figure 4-1 Serial port and jumper locations**

Your circuit board may have COM0 and COM1 marked as COM1 and COM2. If this is the case, the silkscreen is wrong and this manual should be followed.

**COM0 SERIAL PORT**

This port uses a VTC-9F serial cable to connect external serial devices to the port. The cable consists of a 10 pin IDC connector wired one-to-one to a DB-9 connector. Line 10 is simply cut off. The pin out is designed so it plugs directly into the 9 pin serial port connector on a PC.

COM0 does not use hardware handshake lines. The CTS line is pulled high in case external equipment uses this line.

This port is normally used for programming. During run time it may be used as a general purpose serial port. When used for programming or with the INPUT

statement, it will accept ASCII character values from 0 to 127. When used with the GET function, it will return ASCII values from 0 to 255.

**COM1 SERIAL PORT**

COM1 is either an RS-232 or RS-422/485 port. A VTC-9F serial cable, described above, is used for RS-232 level communications. RS-485 is from screw terminals. COM1 is identical to COM0 except that COM1 has 2 hardware handshaking lines, CTS and RTS. When RTS goes low, the RPC-52 is held off from transmitting out COM1. The status of this port is read by the LINE B statement. The example below returns the status of the RTS line:

```
100 B = LINEB(2,5).AND.64
```

If B = 64, transmission is held off.

The CTS line may be set high or low to hold off communication. Line 400 sets CTS high and 500 sets it low, or to hold off.

```
400 LINEB2,2,0A5H
500 LINEB2,2,0B5H
```

Jumper W1 determines if COM1 receive is RS-232 or RS-422/485.

[1-2]	RS-485
[2-3]	RS-232 (de fault)

COM1 default is RS-232. Use the CONFIG BAUD statement to set it to RS-422 or RS-485. When set to RS-422, the transmitter is always on. RS-485 mode turns on the transmitter only when sending.

**RS-422/485 Termination network**

When the RPC-52 is the last physical unit on a network (RS-485), or it is the only unit (RS-422), the receiver must be terminated to prevent ringing. Jumper block W2 installs or removes this network. Set W2 according to the table below:

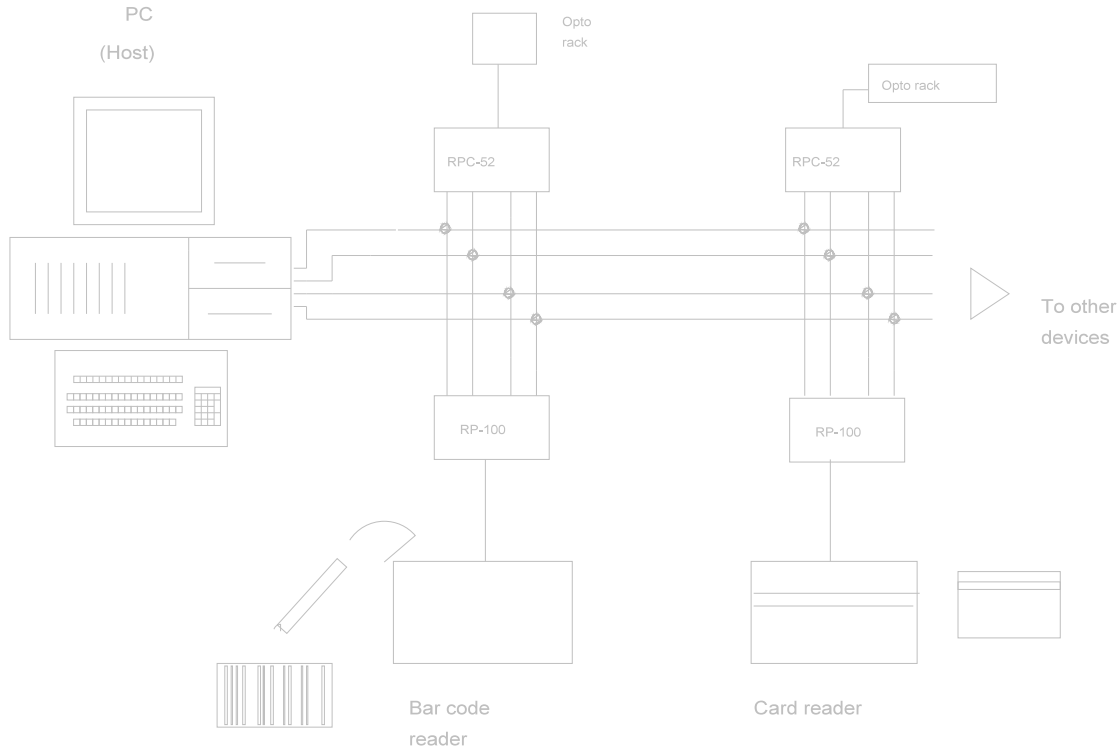


Figure 4-2 Network diagram

- [1-3],[2-4] Termination network installed
- [3-5],[4-6] Termination network removed

Only one slave device on a RS-485 network should have a terminator installed. The host transmitter should also have a 100 ohm resistor in series with a 0.1 mfd capacitor. The terminator on the RPC-52 includes pull up and pull down resistors to prevent lines from floating and generating erroneous characters.

**TWO WIRE RS-485**

The RS-485 port on the RPC-52 is set up for 4 wire mode. 2- wire mode will cause the transmitted data to be received. To use the RPC-52 in this mode, your code should "flush" the received data or otherwise remove transmitted information.

Mechanically, to make a 2- wire system, simply connect T+ to R+ and T- to R-. Make sure CONFIG BAUD is set up for RS-485 mode.

**MULTIDROP NETWORK**

You can use the RPC-52 in a multidrop network by using COM1's RS-422/485 port. You can connect up to 32 units (including other RPC-52's) over a 4,000 foot range.

Figure 4-2 shows an example of a multidrop network. This network includes a host and one or more devices. The host transmits data packets to all of the devices, or nodes, in the network. A data packet includes an address, command, data, and a checksum. See figure 4-3. The packet is received by all devices, and ignored by all except the one addressed.

The relationship described below between nodes and the host is a master-slave. The host directs all communication. Nodes "do not speak unless spoken to". Peer to peer communication, while possible with the RPC-52, is not discussed here.

There are many communication protocols. For this example, a protocol might look something like this:

> 22MB1

The protocol starts with the < cr> character. This character synchronizes all units and alerts them that the next few characters coming down are address and data. In this case, "> 22" is the units address. "M" is the command and "B1" is the checksum. The command is terminated with a < cr> character.



Figure 4-3 Data packet

A response depends upon the nature of the command. Suppose command M means "return a digital I/O port status". The RPC-52 could read the port and respond with AA2< cr> . The first A is an acknowledge, that is no errors were detected in the message. The data, A2, can be broken down as follows:

```
Bit/line    7 6 5 4 3 2 1 0
Status      1 0 1 0 0 0 1 0 = A2
```

Lines 1, 5 and 7 are high while the others are low.

### ACCESSING SERIAL BUFFERS

You can access COM0 and COM1 buffers in three ways:

1. INPUT statement. This removes all characters in the buffer up to the terminator character and puts them into a variable.

When using the INPUT statement, program execution is suspended until a < cr> (Enter key) is received. Whether this is a problem depends on your particular application.

INPUT strips bit 7. This means ASCII characters from 0 to 127 are received. The INPUT statement can return a maximum string length of about 150 characters.

2. GET function. Characters are removed one at a time as a numerical value. A 0 is returned when the buffer is empty. Use the COM function to determine if the buffer is empty or if a 0 is a data value.

If you don't read the buffer and the buffer fills, all subsequent characters are discarded. GET may be

used anywhere in the program.

3. COM\$(n) retrieves all characters in the buffer, including other control codes (except CR).

### ACCESSING COM0 AND COM1

The port INPUT and GET functions retrieve data using the UIn command. UI0 routes inputs to COM0 while UI1 routes inputs to the COM1 port. PRINT outputs are set by the UOn command. UO0 prints out COM0 while UO1 outputs COM1.

The following show how UIn and UOn work.

```
100  UI0      Set to COM0
110  INPUT A  Get data from COM0 port

520  UI1      Switch to COM1 port
530  INPUT B  Get data from COM1 port

800  UO0      Print to COM0
810  PRINT "Temperature:",T

900  UO1      Print to COM1
910  PRINT "Set pressure at:",CA
```

Power up default is set to COM0. UIn has no effect upon COM\$(n) function.

**COMMANDS**

The following is a list of RPBASIC-52 commands used for serial I/O. Variations for many commands are not listed here. These commands and functions are explained in the *BASIC-52 Programming Manual* and *RPBASIC-52 Software Supplement* in this manual.

Command	Function
CLEAR COM\$	Clears serial input buffer
COM\$	Returns string from buffer
COM	Returns number of characters in buffer
CONFIG BAUD	Sets serial port parameters
GET	Returns a character from the serial buffer
INPUT	Receives string from port
LIST	Outputs program listing
PRINT	Outputs data in various formats
SPC	Print out n number of spaces
TAB	Tabs to predetermined positions
UI0	Reroute inputs to COM0
UI1	Route inputs to COM1
UO0	Reroute PRINT statement to COM0
UO1	Route PRINT statement to COM1
USING	PRINT formatting statement

**SERIAL PORT PIN OUT**

Pin outs for J2 and J3 are shown below. COM0 (J3) only has TXD and RXD active. Unused pins are open.

COM0 J3	COM1 J2	Name	Direction from card
3	3	Tx	Out
	4	RTS	In
5	5	RXD	In
6*	6	CTS	Out
9	9	Ground	
10	10	+ 5	

\*COM0 pin 6 has a 4.7K resistor to + 5V.

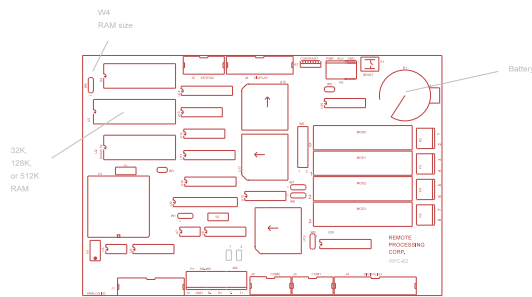
**INTRODUCTION**

RPC-52 models are available with 32K or 128K of battery backed RAM. RAM may be changed at any time. RAM is in socket U3.

RAM is automatically backed up when it is installed. The battery is shared with the clock and controlled by the reset/watch dog timer. Battery life will depend upon RAM size, its power consumption, and amount of time the board is operating. Generally, a battery life from 5 to 10 years can be expected.

This chapter discusses changing RAM, saving and retrieving variables, running assembly language programs, and battery maintenance. Figure 5-1 shows the location of U3, jumper W4, and the battery.

Increasing RAM size does not increase the program size RPBASIC-52 can handle. Maximum program and variable size is 60K. Additional RAM does increase the amount of space available for PEEK and POKE storage.



**Figure 5-1 RAM chip, W4 jumper, & battery**

**CHANGING MEMORY**

Different types of memory can be installed at any time. RPC-52 models come with either 32K or 128K of RAM installed. Up to 512K can be installed.

To change a memory chip, you need to remove the original chip, install the new one, and set jumper W4.

To install a new memory chip:

1. Turn off power to the RPC-52.
2. Remove the memory chip from U3.
3. Orient the chip so pin 1 is towards the card edge.

If installing a 32K RAM, place the chip at the bottom of the socket (memory chip pin 14 goes into socket pin 16). The top two socket pins in each row will be empty.

If installing a 128K or 512K, install the chip into the socket.

4. Check and change, as necessary, jumper W4 to conform to the new memory.

RAM size	Jumper
32K	[2-3]
128K	[2-3]
512K	[1-2]

**BATTERY BACKUP**

The RPC-52 battery operates the clock and backs up the RAM when power is off. Battery life will depend upon RAM size, type, and time the RPC-52 has power applied to it. You can expect the battery to last between 5 to 10 years.

**NOTE:** Do not place the RPC-52 circuit on a metal surface, even with the power off, without standoffs. Voltage is present on the circuit side of the board and it is possible to short out the battery supply through the circuit traces.

Battery voltage is approximately 2.7 volts. The voltage is measured by placing a volt meter between ground and the battery clip.

The battery may be replaced by the following type or equivalent:

Panasonic BR2325

To replace the battery, lift up the holder and push the battery from behind. To install, simply reverse the procedure. The battery may be replaced while power is on. If you replace the battery with power off, be sure to reset the date and time. Also, data stored in RAM will be lost.

## RESERVING MEMORY

Normally, RPBASIC-52 uses the first 30K of RAM for program and variable storage. However, additional memory can be reserved for PEEK and POKE variables using RPBASIC-52's CONFIG MTOP statement.

When only a small number of variables need to be stored (or a small assembly language program run), a 32K RAM system may be adequate. If the combined program and data size exceed 30K, a 128K or 512K RAM is necessary. The additional RAM may be necessary if your program has large arrays and/or string storage requirements.

The CONFIG MTOP statement is not necessary when you do not use RPBASIC-52 memory for variable storage. This is possible when a 128K or 512K RAM is installed. However, you may want to set MTOP to the top of RAM using the CONFIG MTOP command. Highest MTOP value is 65535.

## STORING VARIABLES IN RAM

The term "variables" in this context includes numbers, strings, arrays, recipes, or formulas as applied to your application.

Programs and RPBASIC-52 variables reside in segment 0. Variables are generally stored in segment 1 and higher (a segment is 64K of memory). See memory map figure 5-2. "Extended memory" is segment 1 or higher.

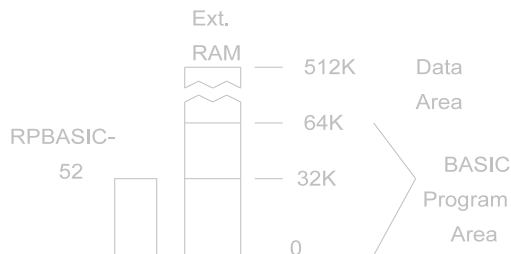


Figure 5-2 RPBASIC-52 memory map

PEEK and POKE commands store and retrieve values from memory. For example:

```
20 POKE B1,12,A
```

puts the value of A into segment 1, address 12.

Use the PEEK statement to retrieve the variable:

```
50 B = PEEKB(1,12)
```

You can store and retrieve strings and variables in this way. There are many variations of PEEK and POKE statements. Refer to the RPBASIC-52 Software

Supplement in this manual for additional information and examples. A list of commands appears at the end of this chapter.

## CORRUPTED VARIABLES

The RPC-52's RAM is automatically battery backed up. User defined data can be saved when the board is powered off then on. When your application must rely on the accuracy of this data after power up, corrupted variables becomes a possibility.

The nature of RAM is it is easily written to. Any POKE'd data is susceptible to corruption. This is especially true when the board is powered down. The RPC-52 has an intelligent reset circuit which minimizes data corruption. However, when POKEing long data, such as strings, a reset could interrupt a saving process. The result is information is corrupted.

Since it is impossible to predict or delay a reset, a work around is to duplicate or triplicate POKEd values. That is, you would have to save the same information in two or three different places. For purposes of discussion, POKEd variables are called sets because data can consist of a mixture of variables and strings.

On power up, your program should compare values from one set to the other one or two. If the two (or three) agree, then there was no corruption and the program can reliably use the values. At run time, you would read information from set 1, but would save data to all two or three.

The use of duplicate or triplicate sets depends upon what the system must or can do if data is corrupted. When using a duplicate set, a corrupted set indicates that default values (from serial EEPROM or the program) should be used, since it is uncertain if the first or second set is corrupted. Both data sets are then re-initialized.

A triplicate set is used to recover the last set or indicate that the data in the first set is valid.

Data is written to each set in a specific and consistent order (data to an entire set does not have to be written to, just that element). For example, a calibration constant is saved (POKE'd) in three different places. Assume that the constant was assigned address 0, 100, and 200 in segment 1. The data is POKEd to address 0 first, then 100, then 200.

Upon reset, the calibration value is checked. If the value at address 0 agrees with address 100 and 200, then no corruption occurred. When address 0 and 100 agree but not 200, then this indicates that a reset occurred at first updating the third set. The first data set can be trusted. The third data set simply needs to be updated.

When the first two sets do not agree, then you know that the first data is corrupted. If the second and third set agree, then, depending upon the system requirements, the first set could be "corrected" using the old data. The user or other device could be alerted that a calibration (or whatever) must be performed again. When all three

sets disagree, then you must take action appropriate to the situation.

Another technique to check for valid memory is checksums. Simply writing a program to add the values in RAM and compare it against a number is a good check. However, you cannot tell which data element was corrupted.

Instances of data corruption are rare. They do increase as the board power is cycled or reset. You should be aware that data corruption is not impossible and there are methods to detect and correct it.

### ASSEMBLY LANGUAGE INTERFACE

Assembly language programs must be placed in the RPBASIC-52 EPROM. Programs should start at address 6000H or higher up to 7FFFH.

Documented assembly language interface calls listed in the Intel *MCS BASIC-52 Users Manual* may work with RPBASIC-52. This is because RPBASIC-52 has been reassembled and code has been shifted around.

### COMMANDS

The following is a list of RPBASIC-52 commands used with RAM.

Command	Function
CALL	Calls an assembly language routine
CBY	Returns code memory data
DBY	Returns or assigns internal memory
MTOP	Sets top of RAM memory
PEEK B	Returns a byte
PEEK W	Returns a 16 bit value
PEEK \$	Returns a string
POKE B	Stores a byte
POKE W	Stores a 16 bit value
POKE \$	Stores a string
XBY	Returns or assigns external memory

**INTRODUCTION**

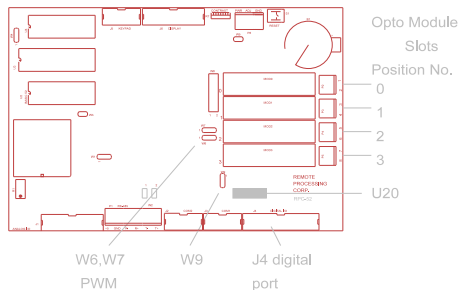
Digital I/O lines are used to interface with opto-module racks, switches, low current LED's, and other TTL devices. The RPC-52 has 24 of these lines available through J4. 8 of these lines are high current outputs, capable of sinking 75 to 200 ma. Additionally, there are 4 opto-module sockets on the card itself.

On-card opto-module slots accept G4 and G5 series opto modules. G4 series opto modules are used to sense the presence of AC or DC voltages or switch them. Maximum switching current is 3 amperes.

G5 series are optically isolated analog input or output modules. The modules connect to thermocouples, RTD's, load cells, 4-20 ma current loops, and general purpose voltage inputs. They can also output voltages and currents. These modules are supported by the G5MOD command. Input modules return a number from 0 to 255 in a manner similar to an A-D. Conversion time is 7 milli-seconds.

In addition to the 24 I/O lines from J4, the display port can be used as digital I/O. Refer to chapter 8 for more information.

Two on card opto rack slots may be jumpered for PWM output.



**Figure 6-1 Digital I/O**

**WARNING:**

Apply power to the RPC-52 before applying a voltage to the digital I/O lines to prevent current from flowing in and damaging devices. If you cannot apply power to the RPC-52 first, contact technical support for suggestions appropriate to your application.

This chapter is divided into two sections. The first section is about the on-card opto rack. The second section refers to the digital I/O port J4.

**ON-CARD OPTO RACK**

**Description**

The on-card opto rack accepts the G4 series opto modules (manufactured by Opto-22, Grayhill, and others). These modules can switch AC or DC voltages from 5 to 240 volts at 3 amperes. They can also sense input voltages of the same type and range.

Opto channels 0 and 1 can be jumpered for PWM outputs. PWM output is also used to generate analog output.

The RPC-52 also accepts the Grayhill G5 series. These modules measure voltage, current, thermocouple output, or RTD resistance and return it as a frequency. Additionally, modules can output a voltage or a current.

RPBASIC-52 supports the G5 series through the G5MOD command.

**Installation**

G4 and G5 modules are installed in the same manner as an opto rack. A screw at the top is used to secure the module to the board. Modules may be installed in any order and types can be intermixed.

A hole for a standoff near the modules is provided to keep the board from bending during installation or removal.

Input and output lines are fastened by the two position terminal in front of the opto module. The module number is in each module position and behind the module. This module position is used in conjunction with the LINE statement.

Refer to the appropriate module data sheet for additional hookup information, if required.

**G4 operation**

G4 modules are accessed using the LINE command and function. Line numbers are from 0 to 3. To turn on a module, execute the following statement:

```
100 LINE 2,ON
```

To return the status of a line, execute the following:

```
200 A = LINE(3)
```

**PWM Output**

Opto module positions 0 and 1 may be jumpered for PWM output. Power up frequency and duty cycle are set using the CONFIG PWM statement. The duty cycle is changed during run time using the PWM statement. See the RPBASIC-52 Software Supplement for information on these commands. The frequency is adjustable from approximately 170 Hz to 40 KHz. The frequency is the same for both channels. Duty cycle is adjustable with a 1/255 resolution. Each channel is set

using the PWM statement.

Jumpers W6 and W7 set the PWM output for opto positions 0 and 1.

Jumper	Position	Description
W6	[1-2]	Position 0 normal mode
W6	[2-3]	Position 0 PWM output
W7	[1-2]	Position 1 normal mode
W7	[2-3]	Position 1 PWM output

When either position is jumpered for PWM output, none of the LINE commands will work for that position.

PWM outputs are used to generate analog outputs. When using one of the PWM positions, the corresponding analog output should not be used.

### G5 operation

The G5MOD statement can refer to an input or output module, depending upon how it was used. Its use as an input is discussed first.

Values from G5 modules are returned using the G5MOD function. The syntax is

$$A = G5MOD(slot)$$

The *slot* number is from 0 to 3 or 100 to 123, corresponding to the position on or off the board. This function returns a number from 0 to 256, corresponding to a resolution of 8 bits. The G5MOD function takes about 7 ms to convert the input data.

### WARNING:

Conversion time is approximately 7 ms. During this time, all interrupts are turned off. Serial characters will be missed at baud rates at and above 9600.

When installing and operating a G5 module in a particular slot, make sure it is not turned "ON" as for a G4 digital output module. If an output is turned on, then no reading is received. When executing the CONFIG LINE 0 statement, the slot the G5 module goes into should be set as an input (1).

The program below takes 100 samples and stores it in an array. The time to read and store 100 samples is 0.7 seconds, or 7 ms per sample.

```
10 DIM G(100)
20 FOR X = 0 TO 99
30   G(X) = G5MOD(1)
40 NEXT
```

Some applications require that measurements be made at fixed intervals. The ONTICK construct can be used to take samples at timed intervals. The program below reads 2 channels every tenth of a second and stores it into an array. When the array fills, the tick timer stops.

```
10 DIM G(100)
20 DIM H(100)
```

```
30 ONTICK .1,100
40 REM This is a dummy loop
50 GOTO 40
100 G(I) = G5MOD(0)
110 H(I) = G5MOD(1)
120 PRINT I,G(I),H(I)
120 I=I+1
130 IF I=100 THEN ONTICK 0,100
140 RETI
```

The G5MOD statement also outputs to a module on an external opto rack. It is not possible to use the internal opto rack due to the electrical driver required. The output number is from 0 to 4095.

To output to a G5 module, execute the following command:

$$1000 \quad G5MOD \text{ channel,value}$$

*channel* is from 100 to 123 and corresponds to the external opto rack position. *value* ranges from 0 to 4095.

### Converting analog measurements

Input readings are converted to usable units of measurement by performing scaling calculations in the program. The G5MOD function returns values from 0 to 256. To change these readings to other units, use the following calculation:

$$\text{variable} = K * G5MOD(slot)$$

K is a scaling constant. It is obtained by dividing the highest measurement unit number by 256.

Example:

You want to measure a 0 to 200 PSI pressure transducer with a 0 to 5 volt output. Divide 200 by 256 to obtain the value of K.

$$K = 200 / 256$$

$$K = .78125$$

To obtain the final value for the equation in PSI:

$$100 V = .78125 * G5MOD(n)$$

## DIGITAL I/O PORT

Digital I/O lines on the RPC-52 go to connector J4. I/O interface is through an 82C55 chip.

This port can be used to interface additional opto modules (using the MPS series racks), drive small relays, solenoids, motors, or lamps, and provide general purpose TTL I/O to other logic devices or mechanical switches. The LINE command is used to access and control this port.

The lines on J4 are divided into 3 eight bit groups. Ports A and B can be configured as all inputs or outputs. Port C can be programmed as one group of 8 inputs or outputs or as two groups of four lines (upper and lower C). The four lines in upper and lower C can each be programmed

as all inputs or outputs.

When a line is configured as an output, it can sink a maximum of 2.5 ma at 0.4V and can source over outputs sink 15 ma at 1.0V. This will drive opto modules. Port B is connected to a high current sink through U20. See "High current output" later in this chapter.

Digital I/O lines at J4 are pulled up to + 5 volts or ground through a 10K resistor pack using jumper W9.

Jumper W9 for pull up or down configuration is as follows:

W9[1-2] Pull up  
W9[2-3] Pull down

Setting W9 for pull up makes interfacing to switches and "open collector" TTL devices easy. See "Interfacing to Switches and other devices" below.

**Digital I/O commands**

The CONFIG LINE statement is used to configure the 82C55 ports. This statement stores input and output parameters in serial EEPROM for recall at power up or reset. CONFIG LINE should not be executed very often as the serial EEPROM has a limited number of write cycles (100,000). Factory default is: Port A - Inputs, Port B - Outputs, Port C - Inputs.

LINE function and statement is used with opto modules. It accesses a module according to the position number printed on the board. Lines are numbered from 100 to 123. The opto module number used in this command is computed by adding 100 to the board position number.

The LINE B function and statement is used to access digital I/O lines 8 bits at a time. The address for port A is 0, B is 1, and C is 2. The bank number is 3.

LINE # function and statement accesses lines according to the pin number at J4. Lines are numbered from 101 to 125. The line number used in this command is computed by adding 100 to the connector pin number. Line 102 is not allowed as it is the + 5V supply. See table 6-1 to correspond a pin number to a port and opto rack position.

LINE, LINE B and LINE # return the 'true' logic level. A '1' indicates + 5 volts or high and a '0' is low or ground. LINE B and LINE # output true logic levels. LINE, however, outputs inverted logic. In order to turn on an opto module, a line must go low. However, turning on a module using LINE, you must specify a '1'.

**High current output**

Eight lines at J4 can be used as high current drivers. These outputs will switch loads to ground. Outputs are controlled by Port B on the 82C55.

Logic outputs are inverted. That is, when a 1 is written

to the high current port, the output is switched on and goes low.

The output driver chip, U20, can be replaced with a DIP shunt jumper so it is like the other lines at J4.

**NOTE:** Outputs at the high current lines are not compatible with TTL logic levels and should not be used to drive other logic devices.

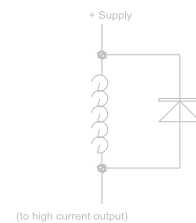
Each of the high current outputs can sink 500 ma at 50V. However, package dissipation will be exceeded if all outputs are used at the maximum rating. The following conservative guidelines assume the number of outputs are on simultaneously:

# of outputs on	Maximum current per output
1	500 ma
2	400 ma
3	275 ma
4	200 ma
5	160 ma
6	135 ma
7	120 ma
8	100 ma

The thermal time constant of the package is very short, so the number of outputs that are on at any one time should include those that overlap even for a few milliseconds.

Incandescent lamps have a "cold" current of 11 times its operating current. Lamps requiring more than 50 ma should not be used.

Protection diodes must be used with inductive loads. Refer to figure 6-2



**Figure 6-2 Inductive load protection**

Do not parallel outputs for higher drive. This results in damage since outputs do not share current equally.

The outputs at U20 are pulled either up or down through 10K resistors, according to the status of jumper block W9.

**Interfacing digital I/O to an opto-module rack**

I/O lines can be interfaced to an MPS-8, 16, or 24 position opto module rack. Lines not going to an opto module connect to a screw terminal on the MPS-XX series boards. This feature allows you to connect switches or other TTL type devices to the digital I/O

lines. The MPS-XX series boards accept G4 and/or G5 series modules.

A CMA-26-18 connects J4 on the RPC-52 to the MPS-XX board. Cable length should be less than 2 feet. Excessive cable lengths will cause a voltage drop and consequently unreliable operation. Make sure you connect + 5 V and ground to the MPS-16 and -24 opto racks. The MPS-08 rack obtains its power through the ribbon cable.

Before a line can be accessed or read, the 8255 chip must be initialized. This is done using the CONFIG LINE statement. Refer to Table 6-1 for Opto module position, port number, and connector pin out. If opto channels 16-23 are used, U20 should be replaced by a DIP shunt jumper.

The LINE and LINE # commands are used to control and access opto modules and lines. These commands are both functions and statements, depending upon how they are used.

```

100 LINE 100,0      Statement
110 LINE #103,0    Statement
120 A = LINE(100)  Function
130 A = LINE#(103) Function
    
```

Program line 100 turns external opto module rack position 0 off. Program line 110 sets J4, pin 3, to a logical 0 level. Program line 120 returns the status of external opto module rack position 0. If the module is "off", a 1 is returned (assuming it is an output module). Program line 130 returns the status of J4, pin 3 as a 0 or 1.

Example: To turn on opto module in slot position 8, the following command is executed:

```
LINE 108,1
```

A '1' turns on a module while a 0 turns it off. (In actual fact, a 0 is written at the port.)

**Interfacing to switches and other devices**

The STB-26 terminal board provides a convenient way of interfacing switches or other digital I/O devices. Lines at J4 are connected to the STB-26 with a CMA-26 cable. Digital devices are then connected to the screw terminals on the STB-26. The MPS-XX series opto racks also provide a way to access digital I/O lines.

Switches may be connected directly to a line. When jumper W9 configures the resistors as pull ups, a switch closure to ground at a line is read as a 0 using the LINE # function.

When W9 configures the input resistors as pull downs, one end of the switch must be tied to + 5 volts. If this is not possible or convenient, a 1K resistor can be tied between an input and + 5 volts to force it high when a switch is open.

**Digital I/O programming example**

The following example reads a switch at port A, bit 3 (J4-25) (program line 200), reads opto module channel 1 (program line 210) and turns on opto module at channel 5 (program line 220). A LED is controlled through the high current port at J3-10 (port B, bit 0) (program lines 230 and 240).

```

200 D = LINE #(125)
210 F = LINE(1)
220 LINE 105, 1
230 LINE #110,1 :REM Turn on LED
240 LINE #110,0 :REM Turn off LED
    
```

Note that the LINE statement is used to control both opto modules and individual lines.

Table 6-1 Connector pin out - J4

Pin #	82C55	Description	Opto Channel
19	Port A, line 0		8
21	Port A, line 1		9
23	Port A, line 2		10
25	Port A, line 3		11
24	Port A, line 4		12
22	Port A, line 5		13
20	Port A, line 6		14
18	Port A, line 7		15
10	Port B, line 0	High current	16
8	Port B, line 1	High current	17
4	Port B, line 2	High current	18
6	Port B, line 3	High current	19
1	Port B, line 4	High current	20
3	Port B, line 5	High current	21
5	Port B, line 6	High current	22
7	Port B, line 7	High current	23
13	Port C, line 0	Lower C	0
16	Port C, line 1	Lower C	1
15	Port C, line 2	Lower C	2
17	Port C, line 3	Lower C	3
14	Port C, line 4	Upper C	4
11	Port C, line 5	Upper C	5
12	Port C, line 6	Upper C	6
9	Port C, line 7	Upper C	7
26		Ground	
2		+ 5V	

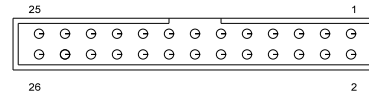


Figure 6-3 Digital I/O connector pinout (viewed from top)

**COMMANDS**

The following tables shows the RPBASIC-52 commands used for digital I/O.

Command	Function
CONFIG LINE	Configures I/O ports
G5MOD	Function returns analog value from an opto module slot.
G5MOD	Statement writes to an analog opto module at an opto module slot.
LINE	Function returns status of an opto module as a 0 or 1.
LINE	Statement turns on or off an opto module.
LINE B	Function returns 8 data bits from any I/O type device.
LINE B	Statement writes 8 data bits to any I/O type device.
LINE #	Function returns status of line at J4 connector as a 0 or 1.
LINE #	Statement writes data to a line at J4 connector as a 0 or 1.
PWM	Sets PWM duty cycle for channel 0 or 1.

**DESCRIPTION**

The RPC-52 has a built in battery backed Calendar/clock. When used in conjunction with the DATE and TIME commands, the current date and time can be set and read. Additionally, the clock can be programmed to interrupt the CPU at specific intervals with a 1 second resolution.

Battery life depends upon the power consumption of RAM in U3 and the time the board is on. Generally, you can expect a battery life of 5 to 10 years.

The clock chip, U14, contains a built in crystal. Accuracy is about 1 minute/month and is not adjustable.

Hours are expressed in 24-hour format.

Refer to the RPBASIC-52 Software Supplement for more command information.

**SETTING DATE AND TIME**

The date and time can be set while running a program or in the immediate mode. Date and time are treated as numbers and not strings. To set the date and time:

```
DATE 5,22,93
TIME 13,23,43
```

The time is set to 1:23:43 PM.

To retrieve date and time as part of a program:

```
100 PRINT "Time: ",
110 FOR N=0 TO 2
120 PRINT TIME(N),
130 NEXT
140 PRINT "Date: ",
150 FOR N=0 TO 2
160 PRINT DATE(N),
170 NEXT
180 PRINT CR,
190 GOTO 100
```

run

```
Time: 13 24 12 Date: 5 22 93
```

**GENERATING INTERRUPTS**

The clock chip generates an interrupt every second, minute, or hour. These interrupts may be used in conjunction with the ONTICK construct. Clock interrupts are captured using ONTIME construct. Refer to the RPBASIC-52 Software Supplement in this manual for more information about ONTIME.

**COMMANDS**

The following is a list of RPBASIC-52 commands for the calendar/clock.

Command	Function
DATE	Sets date
DATE(n)	Returns date
TIME	Sets time
TIME(n)	Returns time
ONTIME	Interrupt handler

**INTRODUCTION**

RPBASIC-52 and the RPC-52 can interface to a variety of displays:

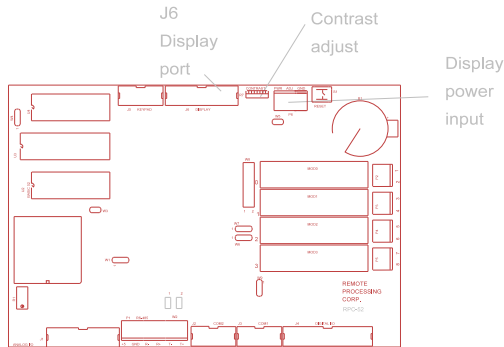
- VF (vacuum florescent) character
- LCD (liquid crystal) character
- LCD graphics

Character display sizes range from four lines by 20 characters to four lines by 40 characters. The graphics display supports 160 x 128 pixels. Remote Processing supplies these displays with appropriate cables.

A contrast adjustment for LCD character displays is built into the card. All displays connect to J6. An appropriate cable connects a display to the RPC-52.

If a display is not used, J6 may be used for general purpose digital I/O. Port A and part of port B from an 82C55 are available.

The cable length to a display depends upon the amount of current it requires. A significant amount of voltage drop occurs with a long cable. VF and LCD graphics cables should be less than 2 feet. A character LCD display cable should be less than 5 feet.



**Figure 8-1 Display interface**

**CONNECTING DISPLAYS**

The display port is designed to supply all the lines necessary for VF and LCD displays. A custom cable connects the RPC-52 to the display.

Displays purchased from Remote Processing include a cable. You simply connect the 20 pin connector to the RPC-52 LCD display port and the other end into the display.

Additional wiring is usually required for LCD graphic and VF character displays. This information is included with the display. Information content is display dependent. Below is general information on both.

Graphic displays require additional voltages not generated on the RPC-52. These must be supplied

externally. An external contrast adjustment may be necessary. You may be able to connect these through screw terminal block P6.

VF character displays require + 5 volts and ground to be brought to connector P6. This may in the form of external wires from the main power connector on the board. Power is not supplied from the board due to the danger of ground loops.

Additional information for commands mentioned in the following text may be found in the RPBASIC-52 Software Supplement in this manual.

**WRITING TO THE DISPLAY**

The display type must first be set using the CONFIG DISPLAY command. The DISPLAY command is used to print information. The display type is stored in the on card serial EEPROM. CONFIG DISPLAY needs to be done only once.

**PROGRAMMING EXAMPLE**

The example below is for a four line by 40 character LCD display. Even though DISPLAY statements do not end with a comma (,), a < cr> < lf> sequence is not sent. Use CR to force a return. A CR does not scroll characters on a display. You must position the cursor to the next line.

```
CONFIG DISPLAY X
10 STRING 200,30
20 $(0) = "Remote Processing display"
30 DISPLAY (1,2),$(0)
```

**DISPLAY TYPES**

RPBASIC-52's software driver is based upon the characteristics of the display family. Compatible VF and LCD displays are shown below:

Manu fact.	Model	Type
Optrex	DMC 40457	LCD 4 x 40
Optrex	DMC 40202	LCD 2 x 40
IEEE	3601-90-080	VF 4 x 20
Optrex	DMF 682N	LCD 160W x 128D

**DISPLAY CONNECTOR PIN OUT - J6**

The display port uses an 82C55 for data and control. The table below lists a pin number and its intended function. A display may not use all lines even though they are available.

J6 Pin	8255 Port/line	Function
1		Logic + 5V
2		Digital ground
3	A/4	D4
4		Contrast voltage
5	A/6	D6
6	A/5	D5
7	B/4	Reset (Open collector inverter)
8	B/3	Write
9	B/2	Read
10	A/7	D7
11	A/1	D1
12	A/0	D0
13	A/3	D3
14	A/2	D2
15	B/7	CS (Open collector inverter)
16	B/6	Command/ data
17	B/5	Halt
18		Contrast adjust
19		Alternate power
20		Power ground

J6 is available for additional I/O if a display is not used. Port A may be configured as an input or output. Port B must be configured as an output if a 17 key or larger keypad is used. Use the LINE B command to access this part.

Pins 18, 19, and 20 are for the LCD-5003 and other graphic displays.

**COMMANDS**

The following RPBASIC-52 commands are used for the display.

Command	Function
CONFIG DISPLAY	Specifies the display type to use
DISPLAY	Prints the string at the row and column specified

**INTRODUCTION**

16, 20, or 24 position keypads are plugged into keypad port J5. Keys are arranged in a matrix format. A key is recognized when a row and a column connect.

RPBASIC-52 scans and debounces the keypad every debounce time as defined by CONFIG KEYPAD. Keypad presses are returned as a number from 1 to 24 using the KEYPAD function.

Keypads from Remote Processing simply plug into J5.

The keypad cable length should be limited to less than 5 feet.

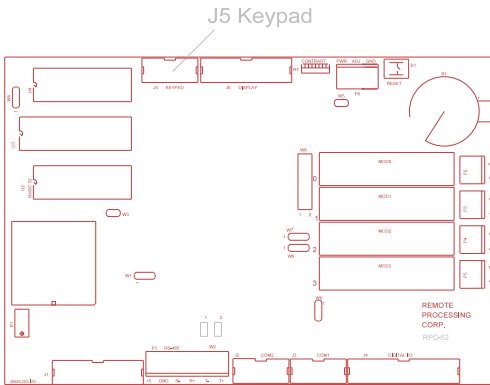


Figure 9-1 Keypad connector

**PROGRAMMING EXAMPLE**

The following example sets up RPBASIC-52 to scan a 16 position keypad. The results are echo'ed when a key is pressed. CONFIG KEYPAD is entered in the command mode. It need be entered only once. Press the 'D' key to enter.

```

CONFIG KEYPAD 5

10 STRING 200,20
20 $(0) = "123A456B789C*0#D"
30 P = 1
40 PF = 0
50 PRINT "Enter a number from the keypad",

REM Rest of program continues
REM Scan keypad and update display

200 GOSUB 500
210 IF PF = 0 THEN 200
220 PRINT
230 PRINT "Entered string is: ",$(2)
240 PF = 0
250 GOTO 50

500 A = KEYPAD(0)
510 IF A = 0 THEN 500
520 IF A = 12 THEN 600 : REM Process clear
530 IF A = 16 THEN 700 : REM process enter
540 A=ASC$(0),A)
550 PRINT CHR(A),
560 ASC$(2),P) = A
570 P = P + 1
    
```

```

580 ASC$(2),P) = 13
590 RETURN
600 REM Clear input string
610 $(2) = ""
620 P = 1
630 RETURN
700 REM Enter processing
710 P = 1
720 PF = 1
730 RETURN
    
```

**Program explanation**

Line 20 defines the keypad legend. Letters may be redefined as necessary.

Line 30 sets the position counter used to insert characters into the string.

Line 200 waits for a key press. The entered string is printed.

Line 500 checks the keypad. If a character is available, it processes it.

Lines 540-590 update the input string and position. A < CR> is inserted to mark the end of string.

**KEYPAD PORT PIN OUT - J5**

The keypad port uses ports B and C from an 82C55. Lower port C is configured as an input. Upper port C and port B bits 0 and 1 are outputs. The table below lists J5's pin out, 82C55 port and bit, and its intended function.

Pin	82C55 Port/bit	Function
1	C/0	Row 1
2	C/6	Column 3
3	C/5	Column 2
4	C/1	Row 2
5	C/2	Row 3
6	C/4	Column 1
7	C/7	Column 4
8	C/3	Row 4
9	B/0	Column 5
10	B/1	Column 6

**COMMANDS**

The following is a list of RPBASIC-52 commands for the keypad.

Command	Function
CONFIG KEYPAD(n)	Sets keypad parameters Returns last key from keypad port

**DESCRIPTION**

The RPC-52 has 8 single ended analog input channels. These channels can be used to measure voltages from transducers, 4-20ma current loops, thermistors, etc. Input voltage range is 0 to 5 volts with 10 bit (1024 count) resolution. In addition to the inputs, there are 2 analog outputs that are shared with the PWM positions. These outputs may be used to control the speed of motors or provide an analog indication of a level or position. Output voltage can be varied in 255 steps from 0 to 5 volts.

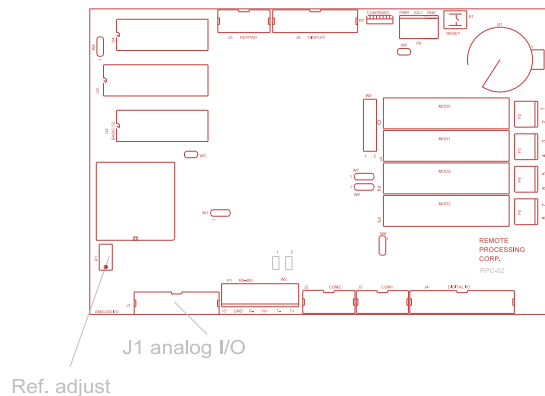
This chapter begins with basic information on connecting and using analog inputs. Later, descriptions of how to measure voltages other than 0 to 5 volts, data logging, and calibration are explained. Generating analog outputs are then discussed.

The analog inputs on this card are above any others used in the opto module slots. Inputs and outputs discussed here are not optically isolated.

**CONNECTING ANALOG I/O**

All analog I/O interfaces through connector J1. An STB-20 and CMA-20-24 ribbon cable can be used to provide screw terminal interface. Screw terminals accommodate 12-22 gaging wiring.

Additional components, such as resistors and capacitors, may be connected directed to the screw terminals.



**Figure 10-1 Analog I/O**

**Analog I/O J1 pin out**

Use the following table to connect to the appropriate input or output. Pin numbers correspond to those on the STB-20.

Description	J1 pin
Channel 0 in	1
Analog ground	2
Channel 1 in	3
Analog ground	4
Channel 2 in	5
Analog ground	6
Channel 3 in	7
Analog ground	8
Channel 4 in	9
Analog ground	10
Channel 5 in	11
Analog ground	12
Channel 6 in	13
Analog ground	14
Channel 7 in	15
Analog ground	16
Analog 0 out	17
Digital ground	18
Analog 1 out	19
Digital ground	20

**Grounding**

Analog ground is somewhat isolated from digital ground. While the ground plane is connected between the two, analog ground is a virtual "island" connected only in one place to digital ground. To minimize noise pickup, the sending device should be connected to analog ground. When both analog and digital grounds come from the same device, you will have to play around with the grounds to determine which scheme provides the best performance for your system.

**ACQUIRING ANALOG DATA**

Analog data is accessed with the AIN function. The syntax is:

$$A = \text{AIN}(\text{channel})$$

This function assigns the analog value of a channel to the variable; A in this case. The value returned is always in the 0 to 1023 range because the converter is 10 bits. A 0 corresponds to 0.000V and 1023 corresponds to 4.99V.

To view the result of a conversion in the command mode, type:

```
print ain(0)
```

The result at channel 0 is returned.

The AIN function requires about 1 ms to convert the data. Additional time is needed to store the data. The example below takes 255 data samples and stores them into an array which requires 6 bytes per entry. The second example takes only two bytes per entry, can save to extended memory, but requires a longer time to get a

data point.

The program below takes about 2.4 mS per data point.

```
10 DIM A(254)
20 FOR X=0 TO 254
30 A(X) = AIN(0)
40 NEXT
```

This next program saves data above MTOP. MTOP was previously set by CONFIG MTOP to 30000. However, if you have 128K or more RAM, you can POKE into segment 1 or higher. It takes approximately 3.6 mS per data point and is not affected by the memory location to save to.

```
10 A = 30000
20 FOR X=0 TO 999
30 POKE W0,A,AIN(0)
40 A=A+2
50 NEXT
```

Data is retrieved using the PEEK W command

**Reducing noise**

An input channel can appear to be noisy (change readings at random) if unused inputs are allowed to float. To minimize noise, connect all unused inputs to ground.

A high impedance is, by definition, sensitive to voltage pickup. Noise is minimized by running wires away from AC power lines. A low impedance voltage source helps to reduce noise pick up. Shielded cable can help reduce noise from high impedance sources. Make sure the shield is not used for power ground. Using the shield for power ground defeats its purpose.

**Data logging on a timer tick**

Some applications require that data is read at fixed intervals. The ONTICK construct can be used to take data in intervals from 0.01 to 327 seconds. The example below takes 1 sample per second until 100 samples have been obtained.

```
10 DIM A(100)
20 ONTICK 1,500
30 REM THE REST OF YOUR PROGRAM
40 REM CONTINUES
80 GOTO 30
500 A(N) = AIN(3)
510 N=N+1
520 IF N = 100 THEN ONTICK 0,500
530 RETI
```

**MEASURING HIGHER VOLTAGES**

Voltages higher than + 5V can be measured by inserting a series resistor to the input. A resistor can be connected directly to the STB-20.

The table below shows resistor values for typical input voltages.

Maximum Input Voltage	Resistor
-----------------------	----------

6	20K
12.5	150K
24	380K

The following formula is used to determine the series resistance necessary for a maximum voltage input:

$$R_s = V_i * 20000 - 100000$$

$R_s$  is the resistor value in ohms in series with the input.  $V_i$  is the maximum input voltage. When the result of your calculation is negative or zero, a series resistor is not necessary.

**NOTE:** When an input voltage exceeds + 5 or is less than 0 volts, other channel values are affected.

**Converting analog measurements**

Inputs can be converted to engineering units of measurement by performing scaling calculations in the program. The AIN function returns values from 0 to 1023. To change these numbers into something more meaningful, use the following formula:

$$var = K * AIN(n)$$

$n$  is the analog channel to read.  $K$  is the scaling constant.  $K$  is obtained by dividing the highest number in the range of units by the maximum AIN count (1023).

Example 1: To measure the results of an A/D conversion in volts and the voltage range is 0 to 5V, divided 5 by 1023 to obtain  $K$ .

$$K = 5/1023$$

$$K = .004887$$

Your program could look something like:

```
1000 C = .004887 * AIN(N)
```

Example 2: You want to measure a 0 to 200 PSI pressure transducer with a 0 to + 5V output. Divide 200 by 1023 to obtain the constant  $K$ .

$$K = 200 / 1023$$

$$K = .1955$$

The code can then look like:

```
1000 B = .1955*AIN(0)
```

**Measuring 4-20 mA current loops**

Current loops is a convenient way to transmit a value and still assure the integrity of the signal. If the line should break, a 0 volt (or nearly so) is returned.

A 4-20 ma current loop is converted to 1 - 5V by placing a 250 ohm resistor across the input of the channel to ground.

Current loop readings are converted to engineering units by performing scaling as described earlier. Since the

measurement range is 1 to 5V, the count range is reduced by 20% to 818. The constant K is computed as:

$$K = 5/818$$

$$K = .006188$$

The above equation is assuming the 4-20 ma loop is returning a value that represents a range of 0 to 5V. As in the previous example, if pressure were measured:

$$K = 200/818$$

$$K = .24449$$

There is one addition factor. Since the lowest value read is 1 V, this offset is subtracted from all readings. A 1 V offset is 1/5 of 1023 counts, or 205. The program line then becomes:

```
200 A=.006112*(AIN(N)-205)
```

Note that if the current loop line breaks, a negative value is returned.

## CALIBRATION

The A/D comes factory calibrated for a 0 to 5V input. This span can be changed by adjusting R1. You can adjust the span to 5.12V. This is useful when the input is 0 - 5V and you want to know when the input is over-range.

To calibrate or adjust the voltage reference:

1. Connect the voltmeter ground to any even numbered pin on J1. Make sure there are no other connections to the analog ground.
2. Connect the voltmeter '+' lead to U5, pin 6.
3. Adjust R1 for 5.00 VDC or other voltage as desired. Do not exceed 5.2 V.

## ANALOG OUTPUT

The two analog output channels share the PWM outputs at the digital port. Analog output 0 is controlled by PWM 0 and output 1 by PWM 1. When a PWM output is used by an opto channel, it should not be used as an analog output.

To make sure PWM is not used by an opto channel, set the following jumpers:

```
W6[1-2]   PWM 0
W7[1-2]   PWM 1
```

One channel may be used as a PWM output while the other is used as an analog output. Both outputs are independently programmable as to duty cycle but not frequency.

Analog output is generated from the PWM outputs by simply placing a low pass filter in the output. The result is an output with a 10K series resistance. The filter capacitor is 10 Mfd. These values provide reasonable

ripple filtering and response over the PWM's output frequency.

The 10K output resistance may be high for some devices. Check the device you are interfacing to. If the impedance is less than 1 Meg ohm, and certainly 100K, then the accuracy and maximum output will be affected. If the full 5 volt range is not necessary, then the lower impedance may not be a problem. The output may be buffered using an external OP amp.

To make sure the output voltage is at the desired level, the output can be connected to the analog input. Then, by reading the voltage the output can be adjusted to the desired level. The maximum output voltage will be reduced when using this method, however.

The output voltage is proportional to the PWM channel's duty cycle. Thus, the output voltage will change in steps of 1/255, or approximately 20 mv/step. This effectively makes the analog output act like an 8 bit D/A converter.

The accuracy of the output depends upon the accuracy of the + 5 V supply. When the supply is exactly 5 volts, the output will go nearly to + 5 and ground at the extreme ranges. When the output is loaded, + 5 volts may not be reached. With the output loaded at 100K, output voltage is reduced by about 10%.

### Programming

The PWM statement is used to set the output voltage. CONFIG PWM can be used to set the output and frequency on power up.

The output voltage is proportional to the duty cycle:

$$V_o = V_{cc} - (V_{cc} * \text{duty cycle} / 255)$$

For example, a duty cycle of 80 produces an output of 3.43 volts. This assumes the power supply (Vcc) was exactly 5 volts and the output was not loaded. The following example ramps both outputs:

```
100 FOR N=0 TO 255
110 PWM 0,N
120 PWM 1,N
130 NEXT
140 GOTO 100
```

The outputs will ramp from + 5 volts to ground then return to + 5V to repeat the cycle.

**COMMANDS**

The following RPBASIC-52 commands are used for analog I/O. More information is found in the appendix of this manual.

Command	Function
AIN(n)	Returns analog value.
PWM	Changes duty cycle, which changes output voltage.
CONFIG PWM	Sets power up default values for PWM.

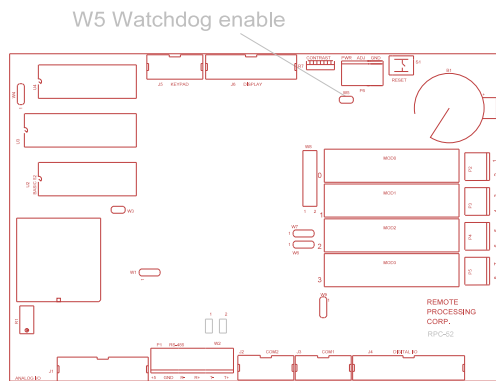
## DESCRIPTION

The watchdog timer is used to reset the RPC-52 if the program or CPU "crashes". When jumper W5 is installed, the WDOG command must be executed at least once every 1.2 seconds to avoid a reset. The timeout is not adjustable.

The watchdog should not be used if using a RPBASIC-52 INPUT statement. Also, loops which do not end quickly or are of indeterminate duration should be avoided unless a timer reset pulse is included. An example of an indeterminate loop is one that waits for a port condition to change.

The watchdog is enabled by jumpering W5. The timer is reset by executing a WDOG command.

The watchdog timer is part of a voltage monitor, battery backup controller, and reset chip U16.



**Figure 11-1 W5 Watchdog jumper**

## PROGRAM EXAMPLE

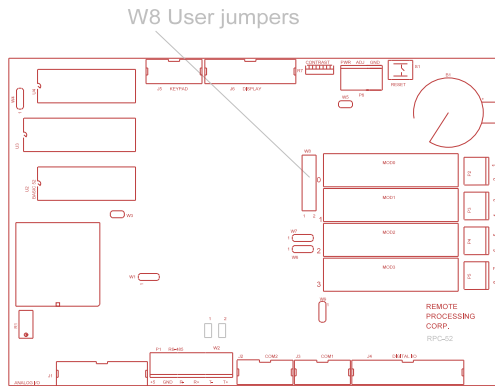
The following program fragment resets the timer while the program is running.

```
5000 WDOG
```

**DESCRIPTION**

Eight jumpers are available at W8. These jumpers may be read as part of a program to determine a boards function or configuration. It is up to you to determine what the jumpers mean. A common use is to set the boards address in a RS-485 network.

Jumpers are at bank 6, address 1, bits 0-7.



**Figure 12-1 W8 user jumpers**

Jumper W 8 is mapped to the following bit numbers.

Jumper	Bit No.
[1-2]	0
[3-4]	1
[5-6]	2
[7-8]	3
[9-10]	4
[11-12]	5
[13-14]	6
[15-16]	7

Jumper status is read using the LINEB function. A '0' indicates a jumper is installed.

```
100 A = LINEB(6,1)
```

Jumpers may be read to determine a card address. Use the following map to assign a value to a jumper. When a jumper is installed, its value is 0. When removed, its value is shown to the right. The program example shows how a jumper configuration can be converted into a number and a string. The values below are returned for a position if other jumpers are installed.

Jumper	Value
[1-2]	1
[3-4]	2
[5-6]	4
[7-8]	8
[9-10]	16
[11-12]	32
[13-14]	64
[15-16]	128

Assume W8[7-8] and [1-2] are not jumpered and all others are. The program would return the following value.

```
10 A = LINEB(6,1)
20 PRINT "Jumper value is:",A

RUN

Jumper value is: 9
```

Even if all jumpers are not installed, you can .AND. out jumpers not of interest.

**DESCRIPTION**

The serial EEPROM is a 128 byte, non-volatile device that stores various RPC-52 power up configurations. These include baud rate, I/O port, display and keypad type. 80 bytes are available to the user using the SPROM statements.

Information such as calibration constants, recipes, RS-485 address, or other "soft" information that may change over time should be stored here.

An EEPROM is more secure than battery backed RAM because it is more difficult to write to it. Several microprocessor instructions must take place before a byte is changed. RAM, on the other hand, requires only a momentary pulse to modify its memory.

Each byte can be written to 100,000 times and read from any number of times. The EEPROM could be updated once a day for over 27 years before this limit is exceeded.

Do not constantly store information to the EEPROM. That is, do not continuously write to it once a second as part of your program. This is an electronic part you can "wear out".

**PROGRAM EXAMPLE**

The following program example saves and retrieves a byte of data from the EEPROM.

```
100 SPROM 0,A  
110 A=SPROM(0)
```

## ELECTRICAL SPECIFICATIONS

### CPU

82C552, 22.1184 Mhz clock

### Memory

RPBASIC-52, 32K ROM

Programming and data is 32K or 128K RAM standard, 512K Optional.

RAM is battery backed up. Battery life is 5-10 years depending upon RAM size, type, and operating time.

Maximum program is 32K EEPROM

### Digital I/O

The RPC-52 has 24 digital I/O lines. 24 are from J4, which is a general purpose port.

The specifications below are for all digital I/O except for the eight high current lines at J4.

Drive current	2.5 ma maximum per line, sink or source. TTL compatible.
Output low voltage	0.45V max at 2.5 mA, 1V max at 15 mA for opto rack.
Output high volts	2.4V minimum, sink or source at rated current.

All digital input lines are TTL compatible.

### High current output at J4

8 of the 24 lines can drive up to 500 ma at 50V. Refer to *CHAPTER 6, DIGITAL AND OPTO PORTS* for limitations.

### Keypad input

10 lines accept a 16 position matrix keypad. Scanning and debounce performed in RPBASIC-52.

### Display output

14 digital and 6 power and ground lines used to control LCD, VF, and LCD graphics displays. Displays supported in RPBASIC-52.

### Serial ports

Two RS-232D serial ports. All have RxD and TxD lines. COM0 has only these lines. COM1 also has CTS and RTS lines. COM1 configurable to RS-232 or RS-422/485. Termination network for RS-422/485 available. Baud rates from 300 to 38.4K. COM1 programmable baud rates only. COM1 programmable 7 or 8 data bits, parity even, odd, or none.

### EEPROM and programmer

Accepts 29C256 or equivalent EEPROM. Size: 32K

### Opto module rack

Four position accepts G4 or G5 series I/O modules

### Calendar/Clock

Accuracy to 1 minute/month

Supported by RPBASIC-52

Battery backup standard. Expected life 5 to 10 years depending upon RAM installed and operating time.

### Watchdog timer, reset

Watch dog timer resets card for 100 ms minimum when enabled.

Time between resets is 1.0 to 2.0 seconds

Push button reset included.

### Power requirements

+ 5 ±5% at 90 ma operating.

RS-232 voltages generated on card.

Current consumption does not include any opto-modules or other accessories.

## MEMORY AND I/O bank map

### Memory

Description	Address
RPBASIC-52, U2	0000H - 7FFFH
RAM, U3, 32K	00000H - 07FFFH
128K	00000H - 1FFFFH
512K	00000H - 7FFFFH

### I/O Bank

I/O Bank	Bank No
RAM (U3)	0
EEPROM (U4)	1
UART (U9)	2
Digital I/O (U19)(J4)	3
Display/keypad (U15)(J5&6)	4
Clock/calendar (U14)	5
Opto/user jumpers (U17)	6

## MECHANICAL SPECIFICATIONS

### Size

4.7" x 7.0"

# TECHNICAL INFORMATION

---

## JUMPER DESCRIPTIONS

A \* after a jumper position indicates factory default is jumpered.

Jumper	Description
W1[1-2]	COM 1 is RS-422/485
W1[2-3]*	COM 1 is RS-232
W2[3-5]	RS-485 terminated
W2[3-1]*	RS-485 not terminated
W2[4-2]	RS-485 terminated
W2[4-6]*	RS-485 not terminated
W3[1-2]	Autorun
W4[1-2]	A17 for 512K RAM
W4[2-3]*	VBAT for 32K & 128K RAM
W5[1-2]	Watchdog timer
W6[1-2]*	On board opto module 0 controlled by LINE statement
W6[2-3]	On board opto module 0 controlled by PWM
W7[1-2]*	On board opto module 1 controlled by LINE statement
W7[2-3]	On board opto module 1 controlled by PWM.
W8[1-2]	User jumper 0
W8[3-4]	User jumper 1
W8[5-6]	User jumper 2
W8[7-8]	User jumper 3
W8[9-10]	User jumper 4
W8[11-12]	User jumper 5
W8[13-14]	User jumper 6
W8[15-16]	User jumper 7
W9[1-2]*	J4 resistors pulled up
W9[2-3]	J4 resistors pulled down