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

TRADEMARKS

Signetics is a registered trademark of Phillips Semiconductors.

PC SmartLINK® is a trademark of Octagon Systems Corporation.

Intel is a copyright of Intel Corporation.

Windows, Windows 95, and Windows Terminal are trademarks of MicroSoft Corporation.

Remote Processing is a registered trademark of Remote Processing Corporation.

Remote Processing® Corporation
7975 E. Harvard Ave.
Denver, Co 80231 USA
Tel: (303) 690 - 1588
Fax: (303) 690 - 1875
email: getinfo3@rp3.com
internet: www.rp3.com

NOTICE TO USER

The information contained in this manual is believed to be correct. However, Remote Processing assumes no responsibility for any of the software or 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.

FCC NOTICE

The RPC-220 was not tested for EMI radiation. When operated outside a suitable enclosure, the board and any cables coming from the board will radiate harmful signals which interfere with consumer and industrial radio frequencies. It is your responsibility to properly shield the RPC-220 and cables coming from it to prevent such interference.

P/N 1736
Revision: 1.4

TABLE OF CONTENTS

TABLE OF CONTENTS

OVERVIEW SECTION 1

DESCRIPTION

The RPC-220 is a small embedded controller programmable in C or assembly. Included is a monitor to facilitate program downloading and debug. Programming is usually done on a PC using any 8051 compatible assembler or C compiler. Additional hardware features include:

- On card flash EPROM allows program updates in the field without removing any parts.
- LCD character port with contrast adjustment. A keypad may also be connected.
- Second software serial port
- Watchdog timer resets card
- Nineteen + individually programmable digital I/O lines
- Two PWM outputs (shared with analog outputs)
- Eight channel, 10 bit resolution A-D converter
- Two analog outputs (shared with PWM)
- Real time clock with interrupts
- Low power modes and wide input voltages

The RPC-220 uses a Phillips 80C552 CPU operating at 22.1184 Mhz. This CPU is software compatible with the Intel 8051 series of CPUs.

A monitor is standard with the flash. The monitor facilitates program development and, when development is complete can be replaced by the final program. The ROM can also be replaced by other third party software interfacing to debugging software.

The RPC-220 is available in 3 different models.

P/N	Description
1710	Full featured with 512K battery backed RAM, real time clock, 32K flash type EPROM, 10 bit, 8 channel A-D, 2 channel D-A (derived from PWM signal), counter/timers. Low power mode draws less than 5 ma. 5.0/5.4 to 21 volt power input without a heat sink. Operating current is about 90 ma.
1715	Similar to P/N 1710 except RAM is 128K. Low power mode is 40 ma. Operating current is about 150 ma.
1720	Low cost version. No clock, battery backup, analog output, or reset switch.

MANUAL ORGANIZATION

This manual, along with the application disk, provides information to write C code and download it to the card. Software examples use the Dunfield Micro-C, but should be adaptable to any other software vendors.
Each chapter has one or more ready-to-run program showing how to use a feature. Writing your program should be a simple matter of combining parts of programs into your own. The RPC-220 programming disk has both source code and output for all examples. Sample programs run stand alone, meaning there is some measurement and display going on or an output.

This manual assumes you are familiar with 8051 C and/or assembly language. If you are not experienced with C, you may want to refer to books and training programs available through many colleges. The C compiler vendor you are using to develop programs will have programming information specific to the 8051.

80C552 INFORMATION

There are several PDF files which have detailed information about the 80C552 CPU. These files are viewed using a PDF reader such as Adobe Acrobat, version 2.1 or 3 or later. You may print these out to get more information as they are referred to frequently.

8XC552OV.PDF	Programming information.
80C552X.PDF	Electrical information for the 80C552. This is a subset of FAMHDWR.PDF.
FAMHDWR.PDF	80C51 hardware information.
PROGGUI.PDF	80C51 programming guide.
AN418.PDF	83C552 timer 2 application note.
93017.PDF	8XC552 analog input application note.

NOTE: There is an error in the 8XC552OV.PDF data file. Figures 6 and 7 are in error. The RTE register in Figure 6 resets bits P4.0 to P4.5 on a match to CM1, not CM2. The STE register in Figure 7 sets bits P4.0 to P4.5 on a match to CM1, not CM2.

RTC INFORMATION

The optional real time clock file is 1689-93.PDF. As with the 80C552 information, these are read and printed using a PDF reader.

MANUAL CONVENTIONS

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

```
Remote Processing Debug Monitor
For RPC-220
Version 1.0
Type 'H' for command summary
```

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.

Wx[a-b] Denotes jumper block pins. [a-b] are the pins to connect.

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

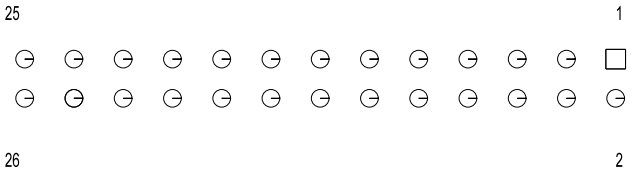
Jx-N Designates a pin number on a connector.

'C' convention generally uses hexadecimal to designate addresses and decimal for data. Any hexadecimal numbers are represented by standard 0xNN notation. Decimal does not have any special notation.

Connector Orientation and Numbering Scheme

The RPC-220 uses IDC type (0.1" center) connectors. Shrouded connectors are keyed to protect against reverse installation. They also take up a lot of room. The RPC-220 has silk screen marks on the board to indicate the key on the connector. Pins 1 and 2 are indicated on all of the larger connectors.

IDC type connectors (J1-5) follow a pin out numbering scheme shown below. View is component side. Square pad (pin 1) is found on the circuit side.



26 Pin Connector Pin Out

Counting scheme is the same for 10, 16, and 20 pin connectors.

Terminology

"A-D"

Shorthand for analog-to-digital converter. The A-D "measures" a voltage and converts it into a binary number. On the RPC-220, this number is from 0 to 1023.

"flash EPROM"

Technically, the EPROM used in the RPC-220 is a PEROM. Unlike a flash, a sector does not have to be erased before written to. Unlike an EEPROM (electrically erasable PROM), 64 bytes are written at a time. EPROM, flash, and flash EPROM are used interchangeably in this manual.

"MOVX"

refers to the 8051 assembly language instruction to read or write to external memory. This manual uses the term to refer to any external RAM or I/O access.

"I/O"

are input-output devices. On the RPC-220, this includes the RTC, expansion port, RAM segment set, and flash EPROM. I/O is an accessing mode enabled when CPU port P4.0 is low. When not in the I/O mode, external RAM is selected.

"RTC"

is the optional real time clock in U8.

"segment"

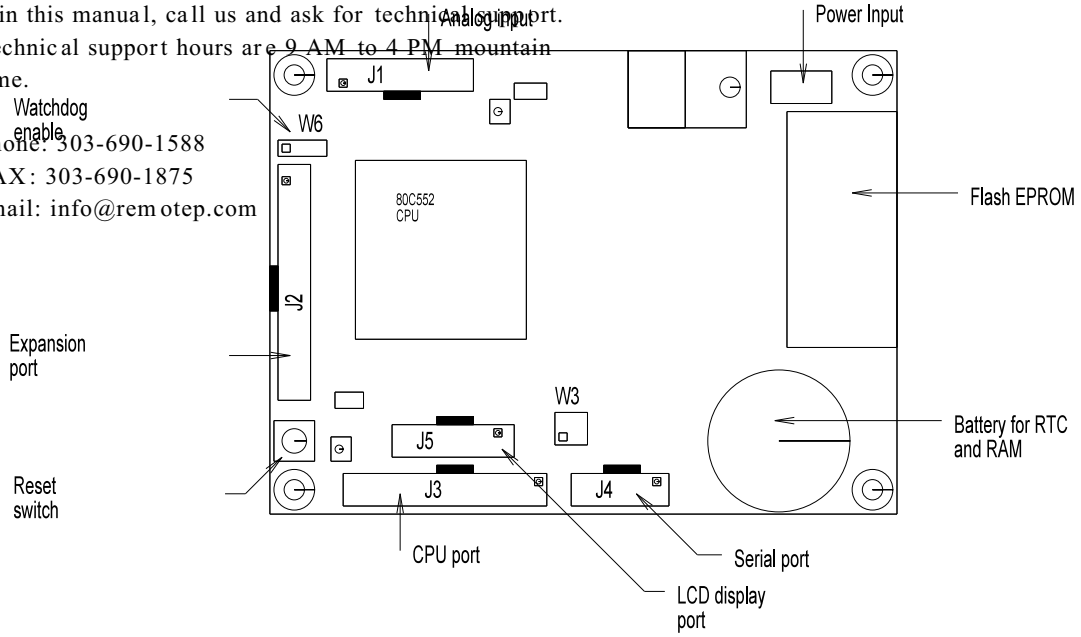
is a 64K block of RAM. Up to 8 segments, numbered 0 - 7, are available. The 8051 CPU series accesses RAM in 64K blocks. The limitation is due to the limited number of address lines. The RPC-220 bank switches RAM in 64K segments.

This practice is common among 8 bit controllers.

TECHNICAL SUPPORT

If you have a question about the RPC-220 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.

Phone: 303-690-1588
FAX: 303-690-1875
email: info@remotep.com



RPC-220 Component Location

INTRODUCTION

The RPC-220 is ready as soon as you connect it to a PC and apply power. This section describes what is needed to get a sign on message. A demonstration program will also be downloaded.

"Development System Setup" describes hooking up using components from our development system.

OPERATING PRECAUTIONS

The RPC-220 is designed to handle a wide temperature ranges at low power. These characteristics require using CMOS components. CMOS is static sensitive. To avoid damaging these components, observe the following precautions:

1. Ground yourself before handling the RPC-220 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 cables or components when power is applied.

EQUIPMENT

You will need the following equipment to begin using the RPC-220. If you ordered the development system, the '●' items are supplied.

- ●RPC-220 embedded controller
- ●VTC-9F serial cable
- ●Power supply, 6 to 15 VDC @ 300 ma or + 5, 300 ma
- PC with a serial port and communications program (such as Microsoft Windows Terminal or Procomm)

Refer to SECTION 4, "SERIAL PORT PIN OUT", for wiring information to make your own serial cable.

The application disk is necessary to run the demonstration program. This disk is included in the development system.

DEVELOPMENT SYSTEM SETUP

A development system has most of the software and hardware needed to get your card operating. All you really need is a PC with a serial port and a serial communications program. A development system includes terminal boards with cables to easily access analog and digital I/O.

You can install the C compiler at any time.

You may want to put in the stand-offs to the board. This keeps the board from touching metal or other surfaces where the battery could discharge.

Refer to Figure 2-1 on the next page. Connect the power cable assembly (P/N 1725) to "Power In" on the board. The RED wire goes to P1 terminal marked "V". Black goes to P1 terminal marked "G".

You may plug the power supply into a 120 VAC wall outlet or power strip. Power may be applied or removed from the RPC-220 board by plugging the 3.5mm connector from the power supply to the power cable assembly.

Continue with "First Time Operation" below. Since the power supply is set up, skip step 1.

Application Programs

To use the application and demonstration programs, you must put them to your hard drive first. They can be any place you want. Make up a subdirectory and run 'RPC-200.BAT' to extract the files. This program expands files in the current DOS directory and creates additional subdirectories.

DEVELOPING UNDER WINDOWS

You can write, compile, and debug under Microsoft Windows 95 or NT. An easy way is to create a folder and put your editor (Word, Wordpad, Note pad, or other), terminal program, and MS-DOS icon in it. Open the editor and terminal programs and use the task bar to switch between the two.

The Windows 'Terminal' program configuration file, mon220.trm, is supplied on the applications disk. You may need to change the COM port.

MS-DOS properties are changed to invoke the compiler. (Properties are selected by first right clicking on the MS-DOS icon). 'Cmd line:' box is changed to:

```
C:\mc\CC51 \rpc220\subdir\fname.c -p -i
```

'subdir' is the directory you are writing your program in. 'fname.c' is the file you are compiling. Additional command line options, such as '-p -i', are at the end.

The 'Working:' directory is nearly the same:

```
c:\rpc220\subdir
```

An inconvenience using some of the Microsoft Windows editors is a lack of line number indication. Should you ever make an error writing C code, the compiler returns the line numbers of the error(s). Using a DOS editor, such as "Edit", you can quickly go to the offending line. Wordpad or Notepad do not show you line numbers. Word indicates the line on a page. Also, Word tends to leave the file it is editing open. When it is open, the compiler cannot access it. You will need to save and close the file before compiling. Wordpad or Notepad do not have this problem.

FIRST TIME OPERATION

Become familiar with the locations of connectors before getting started. See Figure 2-1. RPC-220 jumpers have been set at the factory to operate the system immediately using a 6 - 21 V supply. If you have a 5V supply, then remove jumper W4. 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-220 needs + 5 volts or 7 to 21 volts at 300 ma. The RPC-220 has its own regulator which supplies 5V to the rest of the card when power is applied to the 'V' terminal.

   Be careful when using "switching" power supplies. Some supplies do not regulate properly unless they are adequately loaded.

   Make sure power is off. Connect the power supply to one of the appropriately marked terminals on the RPC-220. Power connector P1 is designated as '5' for 5 volts, 'V' for the 6-21 volt input, and 'G' for ground.

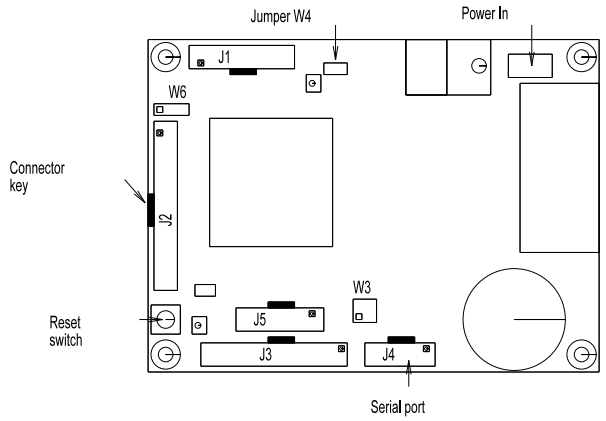


Figure 2-1 Start up connector locations

2. Connect the serial ports.
   Connect one end of the VTC-9F connector to the 10 pin 'COM' port (J4) on the RPC-220. The VTC-9F 'key' on the 10 pin connector faces the inside of the card. Refer to Figure 2-1 for connector location. Connect the DB-9 end 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. Set communication parameters to 19200 baud, 8 data bits, no parity, 1 stop. If using Microsoft Windows Terminal, you may load up 'mon220.trm' file (located on the applications disk root directory) to quickly configure the program. Make sure you set the "connector" to COM1 or COM2 in the Communications window.

3. Power up.
   Turn on or connect the power supply. On power up a message is printed.

```
Remote Processing Debug Monitor
For RPC-220
Version 1.0
```

   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 TROUBLESHOOTING later in this section.

4. Testing.
   Press the "Enter" key on your PC to verify the '> ' symbol returns. Type the letter "h" to view the monitor help menu. If the system responds, you are

now ready to send a program.

DOWNLOADING PROGRAMS

When downloading programs, select ASCII text format. XMODEM, YMODEM or other formats are not used. Transfers are done a line at a time. The pacing character is '> '. A delay is not necessary. Programs generated by the assembler or compiler should be in standard Intel hex format. Extended addressing records are ignored.

Download and Run a Demo Program

The file "demo.hex", should be in the root directory where you put the demonstration programs. This program displays digital status at J3 and voltage inputs at J1.

To download the program, prepare the RPC-220 for hex download by simply typing

```
R< Enter>
```

'R' is the hex download command. The monitor automatically converts lower case to upper case.

After you have hit the < Enter> key, you are ready to send a text file. How this is done depends upon the terminal program you are using. For most terminals, this is a simple ASCII transfer. Refer to SECTION 3, MONITOR COMMANDS, Miscellaneous, R Read Intel Hex File for setup information on different programs.

For Windows Terminal, select 'Transfers', then 'Send Text File'. Select "DEMO.HEX". Appending a LF after the CR is optional.

Downloading takes about a minute.

After download, type in:

```
G 8000
```

The program will execute starting at address 0x8000. You should see a continuous display of analog input and digital I/O status. At this point you can connect the STB-20 terminal board to the analog connector J1. Apply a battery or other < 5 volt input to one of the channels and you should see the result. Use the STB-26 terminal board for J3 CPU port to change port status. Refer to SECTION 6, DIGITAL LINES, Table 6-1 for pin outs. You may bring the a line high or low and see the result on the screen. The port status in hex is displayed.

TROUBLESHOOTING

You would probably came 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 5 in this section. The following are troubleshooting hints when you can't get anything.

1. Check the power source.
   There are two board power options. The first one supplies 5 volts from an external source. If this is below 4.65 volts at the input power terminal, the RPC-220 is in reset. Power is 5 ±0.25 volts. Make sure the 5V supply goes to the pin marked '5' on P1 terminal strip.

   Make sure jumper W4 is removed if using a 5V supply and installed for higher voltages (using the regulator). Default is jumper is installed.

   Make sure the 5 volt supply is "clean". 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 the center pin on U7. If it is low, then the card is in reset. This line should be high (about 5 volts). Some switching power supplies require a minimum load to operate. Check your power supply specifications. The RPC-220 draws from 80 to 140 ma, depending upon the model.

   The second option supplies 6 to 21 volts from an external source. Power is applied to P1 terminal marked 'E'. Verify the voltage at P1-3 (terminal marked '5') is between 4.75 and 5.25 volts. If the voltage is missing, verify jumper W4 is installed. Make sure the supply voltage is relatively clean. It may have some ripple on it, provided it does not go below 6 volts. Make sure you are not using an AC power source.

2. Check the COM0 port (J4).
   Make sure the VTC-9F serial cable is oriented correctly. The key on the cable corresponds to a silk screen area on the board.

   Remove the connector from COM0. Refer to the outline drawing earlier in this section. Connect an oscilloscope (preferred) or a voltmeter to pin 3 (Txd) and ground. Pin 3 should be -6 volts or more negative. (Pin 1 and pin 2 are marked on the board). Pin 3 is next to pin 1. If you have -6 volts or more, reset the board. 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.

3. Check the cable.
   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. Wiring is in SECTION 4, SERIAL PORT PIN OUT.

4. Check the serial parameters on your PC. They should be set to:

   19200 baud
   No parity
   8 data bits
   1 stop

   Make sure you have set the communications port on your PC to the one the VTC-9F is plugged into.

5. Receiving a sign on message and can't enter characters.
   Check U9, pin 14 for at least -6 volts with the serial cable connected to the PC. When it is near 0 volts, the terminal or PC's Tx line is not connected. When you press a character on the terminal or PC, you should see the voltage go positive on the oscilloscope.

If all of this fails, call technical support listed in SECTION 1.

INTRODUCTION

The RPC-220 is programmed in C or assembly languages. Virtually any 8051 type compiler and assembler may be used to generate code. An assembler is provided on the application disk. This section provides information to set up your compiler/ assembler.

OPERATING MODES

The RPC-220 executes code in one of two modes called development and normal. The defining difference between the two modes is how the memory map changes. In normal mode, the CPU accesses its code from U3 flash EPROM exclusively. All of RAM in U4 is accessed using MOVX type commands. In development mode, code is also accessed from RAM starting at address 0x8000.

I/O mode accesses devices (RTC, expansion port, and flash EPROM). This mode is accessed by setting the I/O bit (CPU port P4.0) low.

During development mode (CPU port P4.1 low), RAM from address 0x8000 to 0xffff *can*, but does not have to, execute code. This memory can be accessed as I/O using MOVX commands.

When setting up your compiler or assembler to develop code, set the ORG (originate) to 8000H. The monitor vectors interrupts to base address + 8000H. For example, when there is a serial interrupt the CPU vectors to address 0x23. The monitor has a jump instruction to address 0x8023.

When you are done writing code, simply change the ORG to address 0 and re-compile or assemble.

Detailed Description

Development/normal and I/O modes are controlled by two lines directly from the CPU. On power up or reset, these lines go high, placing the card in normal operating mode capable of accessing external RAM. When the DEVL line (port P4.1) goes low, the card is in development mode. When the I/O line (port P4.0) goes low, non-RAM devices are accessed. These lines are mutually exclusive, except when programming the flash EPROM.

Development mode simply allows CPU code to be accessed from RAM at addresses 0x8000 to 0xffff. MOVX type commands always access all of RAM (assuming the I/O control line is high). U3 flash EPROM is accessed as code from address 0x0000 to 0x7fff.

The I/O control line complicates things just a bit. The purpose of this line is to allow access to non-RAM devices (such as the real time clock and expansion boards) while still allowing full access to RAM. The PEEL in U6 controls RAM, I/O, and flash access. It is designed to allow code access in the memory map (Figure 3-1 below), RAM and I/O at the same time. This is how code runs from RAM while accessing the RTC. It also allows RAM to be modified in the code area.

Generally, the I/O control and development/normal lines operate independently. The only time both lines operate together is during flash programming.

MEMORY MAPS

Memory maps are controlled by one of two bits on the CPU port. Port P4.1 controls development/normal mode. On power up or reset, this line goes high placing the board in normal mode. A low configures the board for development mode.

Port P4.0 selects between RAM and other I/O such as the RTC, expansion port, and flash EPROM. On power up or reset, this line goes high allowing access to RAM. A low maps out the RAM for MOVX type instructions only. Code can be accessed if the development mode is selected (P4.1 is low). When P4.0 is low, the flash eprom, expansion port, and RTC are accessible using MOVX type commands.

During normal mode, up to 64K is accessible by making a PCB modification and replacing U3 with a 27C512 type EPROM. As delivered, 32K of code is accessible. See Programs larger than 32K later in this section.

The memory map changes, depending upon the status of the development and I/O control lines. Figure 3-1 shows the code map during development mode.

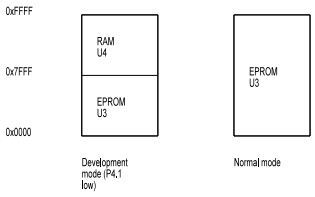


Figure 3-1 Code Memory Map

Code starting at 0x8000 should be exactly what you would have if you were starting in EPROM at address 0, except there is an 0x8000 offset. Interrupts are handled by the monitor by jumping to 0x8000 + interrupt base.

Change your C compiler start up code so it starts at 0x8000. If you ordered the development system, then the code needed to use the Dunfield compiler has been included. Simply install the Dunfield compiler first, then replace the files with those on the supplemental disk. These files are 8051RLPT.ASM, 8051RLPM.ASM, and 8051RLPL.ASM.

The I/O map is more complicated because there are more devices to select. All I/O is accessible regardless of the setting of the development/normal mode bit. Figure 3-2 is the external map.

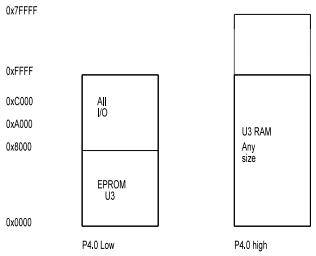


Figure 3-2 I/O and RAM Map

Each area is selected by the following table:

Device/area	Address
Flash EPROM	0x0000 - 0x7fff
Expansion port	0x8000 - 0x9fff
Real time clock (RTC)	0xa000 - 0xbfff
RAM segment	0xc000 - 0xffff

Several demonstration programs access I/O. Directories where programs are located:

MEM220
RTC
FLASH

ACCESSING I/O AND RAM

I/O devices and RAM share the same addresses. Access to them is controlled by CPU port P4.0. When this line is high, RAM is accessed using MOVX type commands. When this line is low, the expansion port and real time clock are accessed.

NOTE: The I/O control line, CPU port P.0, should be low for as short a period of time as possible. Additionally, interrupts should be turned off while the line is low especially if the routine accesses external memory. The following are suggested steps to using the I/O port.

1 Set up pointers and registers as necessary to perform the I/O operation.
2 Turn off interrupts (if used)
3 Set CPU port P4.0 low
4 Perform the read or write
5 Set CPU port 4.0 back high
6 Enable interrupts.

Step 4 above may consist of several reads and/or writes. The idea is to keep this time short.

Alternately, you could write interrupt service routines to check the I/O line before accessing RAM. The line would be restored after servicing. The reason this method is not recommended is the amount of time needed to read, store, and restore the status is longer than the time it takes to code straight through. If step is 4 long (say monitoring a clock status bit), perhaps the extra overhead might be worth it.

See the MEM220.C program to see how RAM and I/O are accessed. Specialized *speek* and *spoke* functions are written in assembly language. This was done not only to speed up memory access and reduce interrupt off time but to ensure correct access. Note that parameters to the function are passed in the stack and returned in registers A and B. Make sure your compiler passes parameters through the stack in the memory model you are using. You may have to adjust parameter passing for your compiler.

MONITOR ROM

The monitor ROM allows you to download code, set breakpoints, examine and modify RAM and I/O. The monitor occupies the same space as the final program will. Generally, the final code replaces the monitor when the project is done. The monitor can be re-installed under two conditions:

1 Your code allows the flash to be written to. This is a routine that must be included in your program. See "SAVING YOUR PROGRAM TO FLASH".

2 The flash is removed from the socket and programmed using an external programmer.

Monitor source code and hex file are under the MON220 directory.

Some internal registers and ports are initialized upon power up and program break. These include registers associated with COM0, RAM segment, and timers. A partial list of modified registers include: S0CON, TMOD, TCON, TL1, TH1 and all of the registers in bank 0. Review the source code, under the programs "MAIN220.ASM" and "START220.ASM" to determine which registers are used and how they might affect your code.

External RAM from address 0x7fc0 to 0x7fff are used by the monitor to store break points and registers. Make sure your program does not write in this area if you intend to use these.

Effects of the Monitor on a Program
The monitor program minimally modifies RAM and I/O ports during its normal operation. There are some modifications the monitor does to operate. These effects are most noticeable during breaks.

RAM segment is reset to 0 upon entry. The RAM examine command (DX) temporarily modifies the segment during access only.

I/O mode is reset to RAM.

Interrupts are turned off via the IEN.7 bit in the interrupt register.

The monitor sets its stack to address 0x60. 'register' or internal RAM variables at or above this address may get corrupted or otherwise not reliable using the DI command.

After a reset, the 8051 stack is automatically set to internal address 7. 'register' or internal RAM from 0x08 to 0x0c get modified. The only time this is a problem is when you reset the card and want to look at data in this area. To solve this problem in C, simply declare 4 'dummy' integer registers before the others.

Ports and registers are restored for Goto commands except P4.1. This bit controls the development mode and is always set low on GOTO.

MONITOR COMMANDS

The following describe command function and operation in greater detail. The monitor displays a summary when 'H' or '?' is entered.

All numeric data is entered as HEX. An address is up to 4 bytes long. Any address less than this assumes leading 0's.

Pressing the < esc> key at any time aborts a command.

Commands with parameters in [brackets] are optional. If nothing entered, 0 is assumed for that and any remaining parameters. For example, if an address and segment are parameters of a command but nothing is entered for either, then 0 is assumed for both.

Commands with multiple parameters are separated by a comma (,).

An Enter (< cr>) executes a command in most cases. Some commands, such as DI and H, execute as soon as the key is hit.

Display Commands
There are 5 display commands:

DX	Display eXternal memory
DI	Display Internal memory
DP	Display Port or I/O data
DR	Display Registers
DS	Display Special function registers (80H - FFH)

Data is displayed 16 lines at a time (except for registers). The segment, if applicable, followed by the address starts off a line. Up to 16 bytes of data are displayed, followed by printable ASCII characters. Printable characters are those from 20H - FFH.

DX Display external memory
External memory is RAM on the RPC-220 card. It is accessed using a segment:memory type scheme. Command format is:

```
DX [address[,segment]] <ret>
```

If DX is simply entered, the monitor assumes address and segment 0. If an address is entered but no segment, then segment 0 is assumed.

DI Display internal memory
Internal memory is RAM internal to the CPU. It is fixed at 256 bytes. Command format is:

```
DI
```

All internal memory from 0 to FFH are displayed.

DP Display I/O port

Addresses for I/O devices (and monitor EPROM) are shown in the I/O memory map above. This command automatically sets the I/O control CPU bit P4.0. The DP command is used to access I/O devices (RTC, expansion). Command format is:

```
DP address<cr>
```

You can view the contents of the monitor EPROM in U3 using this command. The *address* is from 0 to 0x7fff.

DR Display Registers

CPU registers A, B, R0-R7, DPTR, CPU ports P1 and P3, stack pointer (SP), program status word (PS), interrupt (IE), and program counter (PC) are displayed. These registers are displayed after a program break is encountered and are loaded on a GOTO command.

When using this command after a power up or push button reset, certain registers default to a variety of states. The SP is always 7, DPTR and PSW (PS) are 0. P1 and P3 are set to a state that forces bank 0 and memory segment. Additionally, the PC (program counter) is not valid. Command format is simply:

```
DR
```

The terminal outputs:

```
A B R0 R1 R2 R3 R4 R5 R6 R7 P1 P3 PS SP DPTR IE PC
12 34 56 78 9A BC DE F0 12 34 C7 CE 04 4E 0123 00 2114
```

Numbers under the registers are for illustration only. Upon reset or program break, register values are stored in external RAM. Register values after reset reflect default conditions. PC is not reliable and IE is always 0. Registers may be modified using the MX command. The starting address is displayed on a break command.

DS Display SFR's

Special function registers are accessed by the DS command. Format is:

```
DS
```

A < cr> is not necessary.

Special function registers are those between 80H - FFH. These include timers, A and B accumulators, PSW, and other registers. Since there are only 128 locations, 8 lines are displayed. This command is different from DR in that the current contents is displayed. DR shows those saved after a break or reset.

NOTE: A short program is run from RAM at address 0xfffa to directly access these registers. You may have to reload your program after executing this command.

Modify Commands

There are 4 commands to modify memory, registers, or I/O:

MX	Modify eXternal memory
MI	Modify Internal memory
MS	Modify Special function registers
MO	Modify I/O

Data or status to be modified is displayed one byte at a time. The memory segment, if appropriate, is displayed at the start of the line followed by its address. The current data is then displayed. If the data is modified or skipped (by pressing the space bar), the next address and its data is displayed. A maximum of 4 addresses and data are displayed on a line.

For example, MX can display the following at address 0x0000:

```
00:0000 0A     0001 0D 03  0002 52 43  0003 43 51
```

Segment:address is first displayed followed by the current data at that address. If < space> is hit, nothing is modified and the next address and data are displayed. To modify, simply enter the new value.

Addresses and data are displayed in hex notation. Modifications are entered in hex.

Data is modified or not modified depending upon the number of data characters entered, if any, and type of continuation character hit. Pressing < space> always continues to the next address. If any characters were entered, then data at the address displayed gets modified by the value displayed. If no characters were entered, then no modifications are made. Pressing < Enter> ends the modification routine. If any digits were entered, then data at the last address is modified.

MX Modify external memory

External memory is RAM on the RPC card. It is accessed using a segment:memory type scheme. Command format is:

```
MX [address[,segment]] <ret>
```

MI Modify internal memory

Internal memory is the RAM internal to the CPU. It is fixed at 256 bytes. Command format is:

```
MI [address]
```

If an address is not entered, 0 is assumed.

MP Modify I/O ports

I/O devices such as real time clock, digital ports, D-A converters, counters, etc. are modified using this command. Format is:

```
MP [address]
```

If an address is not entered, 0 is assumed for both parameters.

MS Modify Special Function Registers

The 80C552 has several special function registers. They are accessed in the range of 80H to FFH. Refer to the Phillips data sheet (file 8XC552OV.PDF) for register specific information. Command format is:

```
MS [address]
```

When [address] is not entered, 0 is assumed. The monitor program does not check addresses.

Miscellaneous

F Fill External Memory

External memory is filled with a value. Command format is:

```
F start_address,end_address,value[,segment]
```

'start_address' must be lower than 'end_address'. Failure to do this will cause most of memory to be written. Address values are not checked.

R Read Intel Hex file

Code and data files are downloaded using the Intel MCS-86 object format, commonly known as Intel Hex. This format supports both 16- and 32-bit extended addresses. This monitor supports only the 16 bit address. A record type 04, extended linear address record, does not produce an error but is ignored. Any RAM memory from 0 to FFFFH may be loaded to. The command format is:

```
R [offset]
```

'offset' is any value from 0 to FFFFH. 'offset' is added to the address in the hex file. Data is stored to offset + address. Use offset when you want to load code starting at another address. Keep in mind that any address loaded between 0 and 0x7FFF is kept in external RAM and is not accessible as code. Data loaded between 0x8000 and 0xFFFF is accessible as either code or data.

If a checksum error is received during a download, the last address stored is printed on the screen along with a prompt to press the < esc> key to stop printing the message.

Downloading code is done using any number of modem communication programs. This includes PC SmartLink, Procomm, and Windows Terminal. No matter which communication program you use, first type the letter "R" followed by the offset, if any. No value entered assumes a 0. The monitor is now waiting for a ":" to begin processing a line. After a line is sent, the monitor returns a "> " character as a pacing prompt. If there is an error in the checksum, the program will continuously send out a message until the < esc> key is pressed. Instructions for three communication programs are given below.

Procomm

Make sure Procomm's ASCII download is set up using a "> " as a pacing character. Delays between characters and lines are not necessary. Select Upload and then ASCII as the type. Enter the file name then < Enter> . You will see the Hex file as it is sent.

Windows Terminal

Under the "Settings" menu selection, make sure "Text Transfers" are set to "Line at a time" and under "Transfer a line at a time" is set to "Wait for Prompt String". The prompt string is the "> " character. To download the file, select "Transfers" then "Send text file". Select the file you wish to transfer. The "Strip LF" option can be checked off to see what is transferred.

If the download process gets stuck, you can press the < esc> key at any time after the download is aborted. You can also press the reset button. Pressing the < esc> key at any time during the download process aborts the receiving process.

G GOTO Run Program
Once a program is downloaded into RAM and breakpoints are optionally set, it is executed using the 'G' command. The application program has complete control. The only way to stop execution is to press the reset switch or the program hits a breakpoint. Command syntax is:

```
G [address]
```

When GOTO is executed, register values from the last break or reset are loaded first. Then the program is executed. This way you can stop program execution, examine registers, and continue. Registers may be modified before executing. This is done using the MX command. Register values are stored beginning at the address shown when a Break is executed. Values are stored in the same order as they are displayed.

Using G into an interrupt section may be tricky, depending upon its type. The problem is when interrupts are re-enabled, it may execute the same program you executed a break. Unless your program is recursive, variables may be unreliable.

B Set / Clear Breakpoint
A program may be interrupted and the contents of certain registers displayed when a specified address is reached. Up to 8 breakpoints, numbered 0-7, are allowed. Command syntax is:

```
B number[,address]
```

number is from 0 to 7. A number larger than 7 aborts the command. If address is 0 or left out, the contents of RAM at the breakpoint number is restored to its original values. This is the same as clearing a breakpoint.

This command works by replacing 3 bytes in external RAM memory at address and replacing it with a CALL to the monitor break point handler. When a break point is encountered, interrupts are turned off and contents of registers A, B, DPTR, P1, P3, PSW, SP, and R0-R7 are saved and displayed. Additionally, the address where the break occurred is also displayed. RAM at the break point address is not restored until a 'B number' is executed.

As a general rule, external memory is not affected by a push button reset or break point. Thus, if your program hangs up you may restore a break point and replace it at a new address. The following illustrates an example. Suppose your code begins as 8000H and you want to set break points at 8112H and 8215H. Execute the following commands:

```
B 0, 8112
B 1, 8215
G 8000
```

When the code at either 8112H or 8215H is reached, the break point message is displayed.

```
**Break Register information at address 7FC0
A B R0 R1 R2 R3 R4 R5 R6 R7 PS SP DPTR PC
03 00 98 00 00 00 00 00 00 21 41 4F 2147 2112
```

Both breakpoints are still active. To continue execution from this point you must enter:

```
B 0
```

You may then enter a new break point or 'G 2112' to continue.

When a break point is set, 3 bytes from the original program is stored in RAM. Additionally, the break address is also stored.

When a program break is executed, certain registers and I/O conditions are set to a known state. See "REGISTER USAGE AND INITIALIZATION" above for more information. You may modify these registers using the MX command. Values for registers are in order as displayed above. The starting address for the registers is displayed when a break is encountered. The above examples is 7FC0.

S Save to flash EPROM
This command is described in detail below.

SAVING YOUR PROGRAM TO FLASH

Your program is saved to flash EPROM when development is done or when you wish to make sure a program works properly.

Usually, but not always, the monitor is replaced. This means the save to flash feature will be gone UNLESS you put it in your program. See "WRITING CODE FOR UPDATES" below for taking care of this situation.

You can put code or data in a portion of the flash using the monitor save command. However, if you overwrite starting at address 0 to about 1900H, the monitor is effectively destroyed.

The 'S' command transfers the actual prom burning routine RAM starting at address 0xff00 and executes from there. Make sure none of your program is located in this area.

While programming is in progress, the letter 'P' is printed for every 512 bytes programmed (about 100 milli-seconds). The letter 'V' is printed on the next line for every 512 bytes verified. If there is a verify error, a letter 'E' is printed at the 512 byte block it was verifying and the programming process stops.

After verifying good, code jumps to address 0. You may want to press reset to simulate power on conditions.

Saving Process

You can use the monitor code (mon220.hex) to get a feel for saving code. Save this code to an EPROM address not used (such as 0x4000). Then use the DP command to view the contents of the EPROM.

The saving process consists of 2 steps:

First, down load code to RAM using the 'R' command. Do not specify an offset. Begin saving at address 0.

Second, save to flash by executing the 'S' command. Its syntax is:

```
S RAM address, EPROM address, length
```

All parameters are in hex format. RAM and EPROM address are usually 0. The only real figuring to do is length. A quick way to check is first 'F' ill RAM with all 0's from 0 to 0x7fff. Down load the code. Then use the 'DX' command to view your code. When data is all 0's, you can take the highest address and use it as your length.

NOTE: The flash EPROM is programmed in 64 byte blocks. The saving routine stops when the length is reached. FF's are programmed in locations not specified to round out the block.

NOTE: Since the flash EPROM is programmed in 64 byte blocks, you should start on an even byte boundary. An 'even' boundary is 0xXX00, 0xXX40, 0xXX80, or 0xXXc0.

WRITING CODE FOR UPDATES

There are two ways to field update code. One method is listed in the /LOADER directory. There, a loader program always resides in memory. You just set a line low and the loader program is invoked. Programs are stored starting at address 400H. Review the readme files for more information.

The second way is to embed a save routine in your code. This is discussed further below.

A sample routine in FLASH\PFLASH.C is complete code to prompt the field person to download and burn a new program. It receives Intel Hex code, puts it to RAM, uploads the flash burn program into RAM, and runs it. How you get to this routine is up to you. You could check a line status or check a password through the serial port or any number of different means.

In-circuit Reprogramming

The general strategy is to temporarily go into development mode. The ROM code puts programming code in RAM, sets the development mode bit, and jumps to it. Sample code is under the "flash" directory.

Most likely new code is transferred through the serial port. You should allow for some character code or sequence to put the card into programming mode. Using the "FLASH.C" program as an example, the code receives lines of hex, buffers them, and programs the flash 64 bytes at a time. After programming, the code resets the development bit and jumps to CPU address 0.

You cannot read or execute from flash while it is writing new data into itself. Therefore, you must suspend all other processes. Make sure you turn off interrupts.

WRITING FOR C

Memory Models

Most C compilers have different memory models. They are usually referred to as SMALL, COMPACT, and LARGE. The Dunfield compiler has 5 models while Keil has 3. The differences between the memory models have to do with where and how much data is stored and how they are accessed. When selecting a memory model, do not select one where code and data are overlapped into a single 64K address space.
Do not use the SMALL or COMPACT models in the Dunfield compiler. The TINY, MEDIUM, and LARGE models are OK.

Memory Use

Most programs written for 8051 CPUs require relatively little RAM for control (data logging does require a lot more). C compilers use both internal and external RAM to store variables. Internal RAM is faster to access, although there is much less of it. You have about 32K of external RAM for variable space during development.

Set up your compiler so external RAM starts at address 0x0000. During development, you can use up to address 0x7FBF. Addresses 0x7FC0 to 0x7FFF are used by the monitor. Addresses 0x8000 to 0xFFFF is where code is executed in RAM. All of RAM in segments 1-7 is available for data storage, although this is not directly supported by the compiler.

The RPC-220 can access up to 512K of RAM. This is done by selecting a RAM segment the reading or writing to it. See MEM220.C under MEM220 directory, functions *speek, spoke, spokew, speekw* for example access.

Other Considerations for Writing and Saving Programs

Starting address
All demonstration programs use a starting address of 0x8000. You can put programs into RAM anywhere from 0x8000 to 0xff00 (the last 256 bytes are used by the SAVE command).

Interrupts
The monitor ROM vectors all interrupts to RAM. Simply add 0x8000 to the CPU's interrupt base address. When an interrupt is serviced, the monitor ROM vectors off to RAM. The jump code in RAM then jumps to the appropriate location. When compiling for the final version, the jump codes in RAM are replaced with vector addresses only.

Reprogramming
See "WRITING CODE FOR UPDATES" above for more information.

USING DEMONSTRATION PROGRAMS

Demo programs and batch files were written for Dunfield Development systems C compiler. A freeware assembler, ASM51, is included and is in the root directory. If you use another compiler you must modify the code accordingly.

Demonstration programs are in separate directories. Each directory has source, output (hex), and batch files for C. All are ready to run.

Batch and source files assume the Dunfield compiler is used. Switches used in the command line may not work with other compilers or assemblers. When you order a compiler or development system from us, we will send you library and include files to make the start up code. In general, startup code was modified to instruct the compiler where to begin assembly and include interrupts particular to the CPU type used.

PROGRAMS LARGER THAN 32K

The largest program that can be developed on the RPC-220 is 32K. This is because the RAM allocated for development is 32K and the flash EPROM is 32K.
It is possible to develop and put up to 64K of code in the RPC-220. You must do three things to make this happen:

1 Use an EPROM emulator. Remove the flash from U3 and install the emulator. Set the emulator for a 64K EPROM (27C512). Make sure you make the modifications described in 2 below.

2 Cut a trace (write line) on the circuit side to Jumper W5. Refer to Figure 3-3 below.

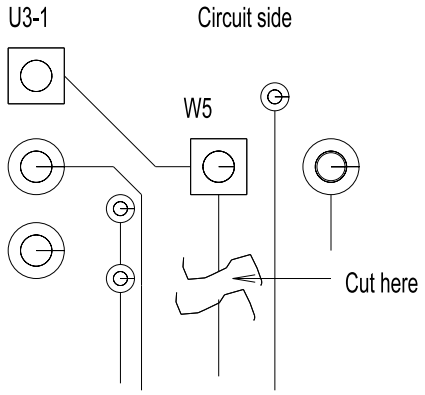


Figure 3-3 Modification for 64K EPROM

Solder a jumper at W5[1-2]. This connects address line A15 to U3, pin 1.

3 Install a 27C512 EPROM when you are done. This EPROM is programmed using an external programmer.

Develop your code in the normal mode. Since code runs from 64K of its own space, all of RAM is available for data storage.

APPLICATION PROGRAMS

The following is a list of programs, locations, and function.

Directory & File	Function
mem220\mem220.c	Writes and reads memory in all segments. It is also a quick RAM test. Written for 512K RAM. Will work with 128K RAM. There will be memory errors in segments 2-7.
flash\dflash.c	Demo program that copies data from RAM to flash. You are prompted for address and length.
flash\pflash.c	Demo program that receives a program in Intel hex format through a serial port and programs the EPROM.
loader\loader.asm	Resident loader program is invoked when a line is low. Used to update or replace existing program. Executes your program when high. Invoke condition can be changed. This program is used with the NOICE debugger.
loader\monload.asm	Program called from loader.asm to reload the monitor routine.
loader\readme1.doc	More information on the loader programs.

INTRODUCTION

The RPC-220 has 1 hardware serial port. A second software serial port is available. If the second port is not needed, then these lines may be used for CTS/RTS control on COM 0 or high voltage I/O to external devices such as proximity sensors. The second serial port is useful for sending to printers, displays, and other non-time critical devices. Jumper block W3 is used to enable CTS/RTS or the second serial port.

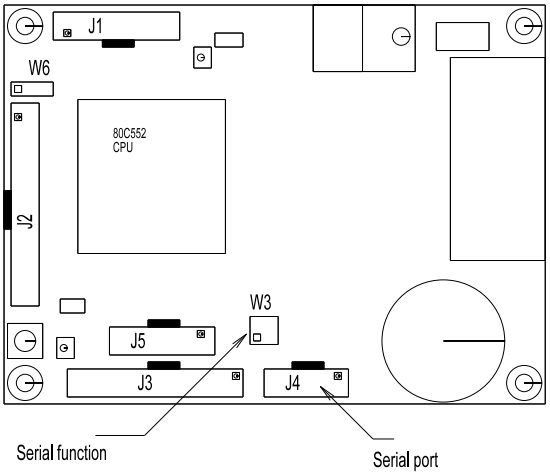


Figure 4-1 Serial Port Jumpers and Connectors

COM 0

COM 0 is a hardware UART directly from the CPU. It may be programmed in any number of modes. Refer to the 80C552 data sheet file "8XC552OV.PDF" for more programming information on the different modes.

All demo programs use COM0. Most operate the serial port in polled mode. SERIAL/SERINT.C operates in interrupt mode. Timer divider factors for different baud rates are also given.

When using timer 1 as the baud rate generator, be sure to initialize TL1 and TH1. When TL1 is not initialized there is a delay before the first character is sent out after a power up or reset. Usually, this is not a problem. But if the first character must start right after reset, then TL1 must be initialized. The delay could be 30 milli-seconds.

Control Lines

Software CTS and RTS lines are available. These lines are shared with other functions and ports on the card. CPU ports P4.3 and P4.4 are used for CTS/RTS, TX1/RX1, or general purpose I/O at CPU port J3.

Jumper W3 connects the CPU to the RS-232 port U9. Set the jumpers according to the table below.

W3	CPU port	Function
W3[1-2]	P4.3	CTS or Tx1
W3[3-4]	P4.4	RTS or Rx1

The large Figure below, 4-2, shows W3 pin locations. Square pad pin 1 is viewed from the component side.

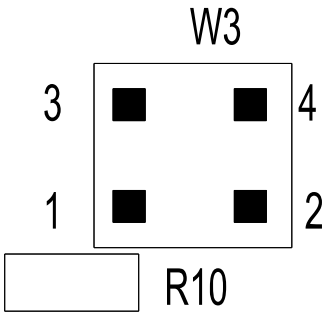


Figure 4-2 Jumper W3 Pin Numbering

SOFTWARE SERIAL PORT

J4 is also a second software serial port. These lines may be used as CTS and/or RTS for COM 0. If one or more of these lines is used for COM 0, then the software port is not available.

Transmit and receive lines are taken from J4-4 (Receive) and J4-6 (transmit). When using COM 0, these lines must be broken from the cable.

Jumper W3 must be set to use either or both lines. Lines P4.3 and P4.4 come from the CPU and also go to CPU port J3. Set W3 as follows:

W3	CPU port	Function
W3[1-2]	P4.3	CTS or Tx1
W3[3-4]	P4.4	RTS or Rx1

Review Figure 4-2 for jumper pin locations.

POWER CONTROL

The RS-232 driver chip U9 has a shut down control to reduce power (about 10 ma). This line is controlled by CPU port P4.2. Examples are in RTC-3.C in the RTC directory.

On power up or reset, the control line automatically goes high, enabling RS-232. You can save about 10 ma. current by bringing P4.2 low.

Allow about 1 ms recovery time for U9 before transmitting characters when P4.2 is brought back high (from IDLE mode).

I²C BUS

The I²C bus uses two wires SDA (available at J3-17) and SCL (available at J3-15) to transfer information between devices connected to the bus. Refer to the 8X552OV.PDF data sheet for more programming information. Detailed information starts on page 3-481.

This bus uses ports P1.6 and P1.7 which are shared with the LCD display port.

SERIAL PORT PIN OUT

Pin out for J4 is shown below. Unused pins are open. You can easily make your own cable from 10 pin header and female DB-9 IDC connectors to a PC. Take a 9 wire ribbon cable and align pin 1 to pin 1 on each connector and crimp.

J4 pin	Name	Direction from card	DB-9 pin
3	Tx 0	Out	2
4	RTS (Rx 1)	In	7 (3)
5	Rx 0	In	3
6	CTS (Tx 1)	Out	8 (2)
9	Ground		5
10	+ 5		

Alternate functions and pin out for RTS and CTS are shown above in parentheses. When used as transmit and receive, these lines must be broken from the cable and wired directly to a connector.

The corresponding signal names from the DB-9 are the opposite of what is coming out of the RPC-220. Thus, Tx becomes Rx on the PC. CTS becomes RTS on the PC.

Rx1 and Tx 1 are alternate uses for RTS and CTS lines. These are the COM1 software serial port lines. The VTC-29F serial cable connects to the RPC-220 and breaks out COM0 and COM1 into two connectors.

APPLICATION PROGRAMS

The following programs are in the SERIAL directory.

File name	Description
SOFT.asm	Uses software serial port. Baud rates from 300 to 9600. Should not be used with interrupts.
COM00.C	Interrupt driven COM 0 serial routine. Transmit and receive are buffered.

This section discusses accessing RAM and battery backup.

ACCESSING RAM

RAM is accessed using MOVX type instructions. The I/O control bit (CPU port P4.0) must be high. This bit is high on power up or reset.

The 8051 series of CPU's access RAM 64K bytes at a time. Additional RAM is accessed using a bank selection scheme. Bank selection in the RPC-220 is done by writing to an I/O address.

Demonstration program MEM220.C accesses all RAM segments. Byte and word wide accesses are through functions *speek, spoke, speekw*, and *spokew*.

On power up or reset, RAM segment is set to 0. To change the RAM segment, do an I/O write to address *0xcYzz*

Where:

- *0xc* is the constant part of the address
- *Y* determines the segment selected according to the formula below.
- *zz* is don't care data.

Simply writing to this address sets the segment. No data is involved. Use the following formula to calculate the address based on the segment desired:

address = 0xc000 + segment * 0x800

The best way is in assembly language. This is shown in the *speek* and *spoke* type routines in MEM220.C under the MEM220 directory. Essentially, DPH is loaded with the address after shifting (multiplying) the segment.

You can use either compiler supplied peek and poke functions or a pointer to access RAM in segment 0. Use caution when using these functions to access other segments. Normally, C stores variables in segment 0. If you have code like:

```
CLR P4.0            // set I/O mode
poke(addr1,value1); // set segment
SETB P4.0           // Back to RAM mode
poke(addr2,value2); // Save memory
```

All of the variables, unless they are register, are bogus since they were obtained from the wrong segment in RAM.

WARNING:

Turn off interrupts before accessing RAM in segments 1-7 unless you make your handlers switch to segment 0. You can use seriously wrong variables if the program assumes they are in segment 0 and you are working another segment 1.

WARNING:

When you are logging data and are in the development mode, you can easily modify the code in segment 0. Code starts at 0x8000. Start logging data in segment 1.

BATTERY BACKUP

The RPC-220 comes in two versions with battery backed RAM. Battery backup is controlled by the RTC chip U8. The supplied battery can expect to back up RAM and keep the clock going for about 4-6 years with a 128K RAM, the unit completely off, and sitting at 25°C. Battery life degrades quickly, about 50%, at 50°C. You can extend battery back up life by installing your own. See SECTION 7, EXTERNAL BATTERY, for more information.

RAM SIZE JUMPER

Jumper W1 configures RAM U4 for 128K or 512K. It is preset at the factory.

APPLICATION PROGRAMS

The following programs are in the MEM220 directory.

File name	Description
MEM220.C	Extended memory access routines.
MEMTST22.C	A slow, but extensive memory test. All 512K RAM checked. Primarily to check for overlapping memory segments.

INTRODUCTION

There are 19 digital lines available. All are from the CPU, have alternate uses, and are available at J3.

Alternate uses include:

- LCD display
- PWM output
- Interrupts
- Counting/timing
- Serial I/O

Some lines have 3 uses.

When configuring your system, you will have to make use of lines based on needed functions and availability.

ELECTRICAL CHARACTERISTICS

Digital ports sink and source a limited amount of current. Each port is different as shown in the table below:

Ports	Current/condition
P1.0 - P1.5	1.6 milli-amp sink
P1.0 - P1.5	60 micro-amp/source
P1.6 - P1.7	3.0 milli-amp/sink
P3, P4	1.6 milli-amp/sink
P3, P4	60 micro-amp/source
PWM 0,1	3.2 milli-amp/sink
PWM 0,1	400 micro-amp/source

Digital inputs (except P1.6 and P1.7) have a soft pull up internal to the CPU. For a more robust signal, add a pull up resistor to the outputs.

Ports P1.6 and P1.7 are open drain output and do not have internal pull up. To use these as inputs or output, make sure the device you are connected to has a pull up resistor. Otherwise, add one.

Refer to the 80C552 data sheet (file) for more information on each port.

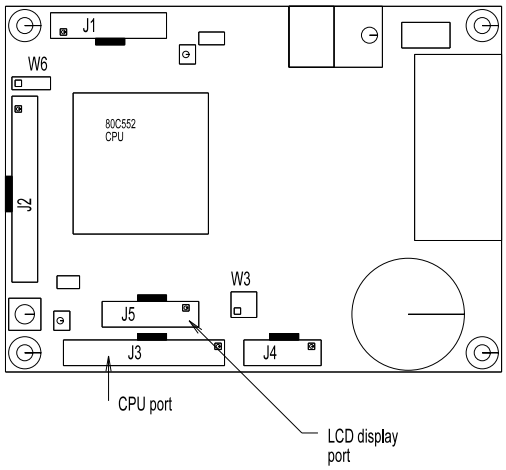


Figure 6-1 CPU Port

CPU PORT PIN OUT

Table 6-1 below lists the pin out for CPU port J3. Most of the lines at this port come from the CPU.

J3-Pin	Primary CPU port name
1	P1.0
2	P4.3
3	P1.1
4	From RTC, square wave output
5	P1.2
6	INT0 (P3.2)
7	P1.3
8	+ 5V supply
9	P1.4
10	System Reset
11	P1.5
12	Ground
13	P4.5
14	Ground
15	P1.6
16	P4.4
17	P1.7
18	+ 5V supply
19	P3.4
20	PWM 1 output
21	P3.5
22	PWM 0 output
23	INT 1 (P3.3)
24	P4.6
25	RTC IRQ
26	P4.7

USING ANALOG INPUTS

Analog inputs may be used as digital inputs only. Use CPU port designation P5 or read as an analog input. Any value above 512 counts is considered a '1'.

ALTERNATE PIN FUNCTION TABLE

Table 6-2 lists each available digital I/O line and alternate functions.

Table 6-2 Alternate CPU pin

CPU Pin	Function 1	Function 2
P1.0	Capture timer CT0I	LCD display
P1.1	Capture timer CT1I	LCD display
P1.2	Capture timer CT2I	LCD display
P1.3	Capture timer CT3I	LCD display
P1.4	T2 event input	LCD display
P1.5	Timer 2 reset	
P1.6	Clock, I_2C bus	LCD display
P1.7	Data, I_2C bus	LCD display
P3.2	INT0	
P3.3	INT1	
P3.4	Timer 0 external input	LCD Display
P3.5	Timer 1 external input	
P4.3	Timer 2 compare set/reset 3	2nd serial transmit
P4.4	Timer 2 compare set/reset 4	2nd serial receive
P4.5	Timer 2 compare set/reset 5	
P4.6	Compare 0	
P4.7	Compare 1	
PWM0	PWM 0 output	Analog 0 output
PWM1	PWM 1 output	Analog 1 output

INTRODUCTION

An optional real time clock (RTC) incorporates an industry standard DS1287 with several enhanced features. Of note, these include a silicon serial number, century register, power elapsed time counters, and power cycle counter.

This optional RTC is battery backed and also backs up RAM in U4. It can generate an interrupt or wake up the CPU from low power mode at periodic time intervals or preset time.

This section of the manual addresses many of the parts features and capabilities. However, not all are discussed. Review the data sheet, 1689-93.PDF, for more information.

YEAR 2000

The RTC has a century counter which rolls over from 1999 to 2000. Run the demo program RTC-1.C and set the date/time to late December 31, 1999. You can watch the date roll over to year 2000.

ADDRESSING

The RTC's base address is 0xa000. The part is accessed when I/O control bit P4.0 is low and any MOVX type access between 0xa000 and 0xa0ff is performed. Clock registers are accessed by adding 0xa000 to the address.

Review the application programs in the RTC subdirectory for programming examples. RTC-1.C accesses most all functions.

Note that there are two register banks in this chip. Both banks allow access to timekeeping, control, and some RAM. Bank 0 has an additional 64 bytes of RAM while bank 1 has serial number, extended control, and other timers. Bank control is through register A, bit 4. See page 10 of the 1689-93.PDF file for register A information.

The debug monitor is also useful for accessing the various registers. Command MP is used to display and modify a register. Command DP displays all registers and RAM addresses. Monitor use is shown under "SQUARE WAVE OUTPUT" below.

PROGRAMMING NOTES

The optional RTC oscillator is turned off and date, time, and counters are reset to zero as shipped from the factory.

The crystal used is 12.5 pf. Bit 5 must be set in extended control register 4B. This bit is set in RTC-1.C demonstration program.

LOW POWER MODES

The CPU can enter two kinds of low power modes: Power down and idle. Power down draws only 4 ma board current at 5 volts. Idle mode draws about 10 ma. The PCON register determines the mode. Refer to CPU manual, page 3-523 in 8XC552OV.PDF, for more information.

The demonstration program RTC-3.C demonstrates both modes. RTC operation is the same. The only real difference is how PCON register is set (programmed), interrupt handed, and jumpers configured.

Power Down

The only way to "wake-up" the CPU from a power down mode is with a reset. A RTC interrupt, through jumper W2, makes this happen.

When W2 is installed, make sure you do not have INT1 and the RTCIRQ lines jumpered together on J3 (pins 23 and 25). No damage will result, but you will never get interrupts since the card will always reset.

CPU register PCON, bit 1, controls power down mode. Setting this bit high shuts off the CPU its crystal. The demonstration program configures the RTC and causes a reset in 5 seconds. When the RTC sends out its pulse, the card stays in reset for about 350 ms.

Review RTC-3.C demonstration program. Make the few modifications as explained in the code for the different power down modes.

Developing in Power Down Mode

The only way to effectively run and test power down mode is to put the program in EPROM and run it, since the card resets.

As an alternative, program PCON for idle mode. You can use INT1 to trigger an interrupt and run a reset simulation (ie GOTO 0x8000 after the interrupt is handled). Power levels in idle mode are about 10 ma instead of 4 but at least you can test reset recovery

without having to reprogram the flash every time. One potential downside is CPU I/O ports (and registers) are changed after a reset. Running a program through interrupts does not account for a reset.

IDLE
This mode uses more power (about 10 ma) but lets you wake up on an interrupt and continue execution. Essentially, this is the same as interrupt mode described under "RTC INTERRUPTS" below.

INTERRUPTS

The RTC is can generate an interrupt due to several conditions. These are described on page 7 of the DS1689-93.PDF file. Three of the conditions (kick start, RAM clear, and wake up) are effectively disabled on the RPC-220. This leaves you with periodic, alarm (specific date and time), and update interrupts.

Two of the interrupts, periodic and alarm, are described in application programs RTC-1.C and RTC-2.C in the RTC subdirectory. To use the interrupts, jumper J3[23-25]. This brings the interrupt from the clock to CPU interrupt INT1.

The RTC can generate periodic interrupts every month, day, hour, minute, and second. It can also generate an interrupt on a specific day, hour, minute, and/or second by simply programming the appropriate alarm registers.

Another periodic interrupt, using the square wave output, is described next.

SQUARE WAVE OUTPUT

The RTC can output a square wave at J3-4. Frequency ranges from 2 Hz to 32.768Khz. Maximum useful sink current is 25 ma while source current is 10 ma.

Square wave output can be used to as another clock interrupt. To enable, set jumper J3[4-6]. This puts the periodic interrupt to CPU INT0. Make sure you are not using INT0 on expansion port J2.

This output can also be used to trigger periodic A-D conversions. The analog input must be enabled for external triggers. Jumper J1[16] to J3[4] to do this. See demo program AIN220-3.C.

You can get a square wave output at J3-4 in the monitor mode using the MP command. Simply enter 0x31 at I/O address 0xA00A and 0x8f at address 0xA00B. Square wave output at J3-4 is 256 Hz. To change the frequency, refer to Table 2 in the DS1689-93.PDF file for the proper value to write to RS0-RS3.

Step by step instructions are:

Screen	Description
MP a00a<Enter>	Your entry to access register A. Response is
A00A 20 31	Indicates oscillator off. Enter 31 for 256 Hz, select extended registers. Press space bar.
A00B 03 0a<Enter>	Set register B to 0A. This enables square wave and 24 hr mode.
MP a04b	Select extended control register B.
A04B 40 20< Enter>	Indicates 32768 output. Enter 20 for crystal bit at 12.5 pF, disable 32Khz output, enable square wave.

Other registers are accessed similarly. See RTC-1.C for program example that accesses most of the registers.

OTHER REGISTERS

The DS1689 has a few registers that may be of interest. See demonstration code in RTC-1.C for access examples.

There are two kinds of elapsed time counters. The first one, located at 0xa054 - 0xa057, bank 1, counts elapsed time power has been applied. The second counter located at 0xA058 - 0xa05B records the total number of seconds the clock has been running, both battery and external powered supplied.

These counters may be useful for determining when equipment needs maintenance, end of battery life, or other similar situations. Since these are easily read and reset, they can be used as secondary timers for relatively long events.

A power cycle counter at addresses 0xa05c-0xa05d, bank 1, counts the number of times power is applied. This can also be useful for maintenance alerts.

Each card with a RTC has a unique 64 bit serial number located at addresses 0xa040 - 0xa47, bank 1. You can embed this serial number into your code to uniquely identify a system and prevent unauthorized copying.

EXTERNAL BATTERY

The battery included on two RPC-220 versions is, in theory, good for 10 years of power off backup.[1] In practice, you will achieve 4 to 6 years of life. This is true of any lithium button cell. Factors that cause life to decrease are elevated temperatures, high and low humidity. A batteries' self discharge doubles when its environment temperature changes from 25°C to 50°C. This alone reduces battery life from 10 years to 5. Lower temperatures reduce self discharge rate.

Low humidity promotes the evaporation of electrolyte from the battery. High humidity helps discharge it.

If battery life is not long enough, you may connect an external 3 volt battery at W7. You must remove the existing battery first. A Tadiran[2] model TL-5276/SL is a utility meter battery and will, in theory, provide backup for over 100 years.

Battery current is about 1.5 micro-amps with board power off.

Remote Processing can build versions without the battery and with W7 installed. Minimum quantities will apply.

STABILITY AND TEMPERATURE

Real time clock's crystal initial tolerance is ±20 ppm. (Tolerance is specified as Δf/f.) This tolerance changes due to the following factors: Temperature, humidity, and operating environment.

Reference temperature is 25 °C. Frequency drift is an additional 40 ppm at -10°C and + 60°C. Peak temperature coefficient is -0.035 ppm/°C.

Another factor that causes the clock to run faster is very high humidity. When moisture condenses on the board, there is an increase of apparent capacitance. Increased capacitance causes the crystal to oscillate at a higher frequency.

When the operating environment is dirty, or contains conductive "dust", the crystal can change frequency.

A way to prevent these problems is to coat the areas around the clock crystal with a non-conductive sealant.

APPLICATION PROGRAMS

The following application programs are in the RTC directory.

File name	Description
RTC-1.C	Prompts you for the current date and time then continuously prints it out. Also prints chip serial number, power cycle count, battery elapsed time and power on elapsed time. Bypass entering date & time by entering 0 for year.
RTC-2.C	Run RTC-1 prior to running this program. Demo generates an interrupt every 3 seconds from the RTC. Can be modified to generate interrupts at an combination of date and time. Uses INT1.
RTC-3.C	Dual example program puts CPU in IDLE or power down mode. Default is IDLE. Power down mode possible by changing a jumper and PCON register. Run RTC-1 first to set date and time.

Footnotes

[1] Information from Renata catalog and other sources.

[2] Tadiran is in Port Washington, NY. Ph: 516 621 4980

INTRODUCTION

The 80C552 CPU chip on the RPC-220 has many counter/timers. These are used to count pulses or measure pulse widths. Some of these timers are multi-purpose, and are shared with other functions. Timer 1 is used as the baud rate generator for COM 0. Some lines used for timing are also used for other functions, such as LCD display and software serial port.

The optional RTC can generate periodic interrupts on INT1. See SECTION 7, INTERRUPTS for more information.

Timer operation is described in detail in two PDF files. FAMHDWR.PDF describes timers 0 and 1 and 8XC552OV.PDF describes timer 2. Review the application programs listed at the end of this section. Timers 0 and 1 operate the same as on the 80C51 CPU.

TIMER 0

Timers 0 and 1 are very similar to each other. Since the serial port is used for most applications, timer 1 is not described.

Timer 0 uses an external or internal clock. When operating on an external clock, it is effectively a counter. Pulses go into port P3.4, available at J3-19. Pulses are gated by the TCON and TMOD registers and INT 0 pin on J3-6 and J2-6. These pins have "soft" pull ups and may require external resistors to operate reliably.

Maximum clock frequency to timer 0 is 921.6 Khz and the minimum pulse width is 542 nano-seconds. The timers operate in one of 4 modes and generate an interrupt on a terminal count. Refer to the FAMHDWR.PDF file for a complete description of these modes.

J3 CONNECTOR PIN OUT

The table below is the pin out for J3 and associated timing and alternate functions.

Table 8-1 J3 timing pins

J3 Pin	Name	Timing function
1	P1.0	CTOI
2	P4.3	CMSR3
3	P1.1	CT1I
4	RTC	Square wave
5	P1.2	CT2I
6	P3.2	T0 gate
7	P1.3	CT3I
8	+ 5V	
9	P1.4	T2 clock
10	Reset	
11	P1.5	T2 reset
12	Ground	
13	P4.5	CMSR5
14	Ground	
15	P1.6	
16	P4.4	CMSR4
17	P1.7	
18	+ 5V	
19	P3.4	T0 clock
20	PWM1	
21	P3.5	T1 clock
22	PWM0	
23	P3.3	T1 gate
24	P4.6	CMT0
25	RTC IRQ	
26	P4.7	CMT1

TIMER 2

Timer 2 is used to measure time between edges, generate timed interrupts, or set, clear, and toggle certain bits at timed intervals. CPU port P1 (located on J3 and shared with the LCD display port) and P4 are mostly used for this purpose.

Timer 2 can use an external (at CPU port P1.4) or internal clock. The internal clock frequency is 1.8432 Mhz. and can be divided down via TM2CON register.

Review 8XC552OV.PDF file for more timer 2 information. Text starts on the second page (3-474). AN418.PDF is an application note for timer 2.

NOTE: There is an error in the 8XC552OV.PDF data file. Figures 6 and 7 are in error. The RTE register in Figure 6 resets bits P4.0 to P4.5 on a match to CM1, not CM2. The STE register in Figure 7 sets bits P4.0 to P4.5 on a match to CM1, not CM2.

APPLICATION PROGRAMS

The following programs are in the subdirectory as shown. All programs can use the RTC output for counting and pulse width measurement to try them out. See SECTION 7, SQUARE WAVE OUTPUT for generating information.

The programs list the input and control lines.

File name	Description
timer0\timer0a.c	Timer 0 in counter mode. Displays current count using J3-19 input.
timer0\timer0b.c	Timer 0 in timing mode. Measures high pulse time (35 ms maximum) at J3-6.
timer2\timer2a.c	Timer 2 used to generate timed interrupts. Can also use to generate timed pulse outputs.
timer2\timer2b.c	Measure pulse widths (rising edge to rising edge). Prints out time in micro-seconds.
timer2\timer2c.c	Measure positive pulse width. Faster interrupt routine for shorter pulses. Prints out number of counter ticks.
timer2\timer2d.c	Outputs delayed pulses. Need oscilloscope to view.

A watch dog timer resets the CPU if it enters erroneous processor states or "loop forever" code. The timer is reloaded in periods that are shorter than the watchdog interval. The timer is built into the CPU and is enabled or disabled by jumper W6.

OPERATION

When the watchdog timer is enabled and overflows, a short reset pulse, internal to the CPU, is generated. A short output pulse is also generated but does not have any effect on the reset signal at J2 or J3.

Information about the watchdog starts on page 3-480 in the 8X0C552OV.PDF file.

The timer consists of an 8 bit timer and an 11 bit prescaler. The 8 bit timer is loaded to prevent a reset pulse. Watchdog timer interval is determined by the following formula:

interval = 1.111 ms * timer_value

The watchdog service routine is very short:

```
watchdog:
        ORL PCON,#10H        ;set condition
        MOV T3,timer_value   ;load with
                                ;interval
        RET
```

This should be in assembly language even when using C to prevent it from wrongly manipulating PCON. The address of T3 is 0xff.

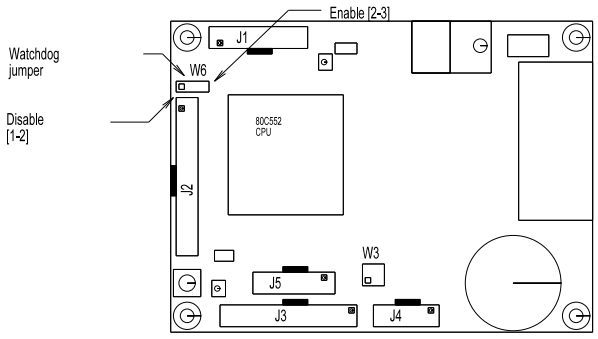


Figure 9-1 Watchdog Jumper Location

USE DURING DEVELOPMENT

The watchdog jumper W6 is set to [1-2] to disable it. If you want to enable it for debugging purposes, you can set the jumper to enable (W6[2-3]) while your program is running. If a watchdog crash happens, the program will reset and the monitor program

When your program is in flash, you can set W6[2-3] all of the time.

On power up or reset, the watchdog timer is reset to 0, which is a timeout interval of 283 ms. This is enough time for basic system initialization (memory and I/O states). If you are performing extensive initialization (such as clearing 512K of RAM) then you should put the watchdog routine as part of the program.

INTRODUCTION

There are external CPU interrupts: INT 0 and INT 1. If you do not use these interrupts, they may be used for general purpose I/O or timing. See SECTION 6, DIGITAL LINES, for current capabilities and pin outs.

Both interrupts are 8051 compatible.

INT 0

This interrupt line goes to both J2-6 and J3-6. This line has a soft pull up, internal to the CPU. This allows wire OR with other expansion boards. Software will have to poll the expansion boards to determine which one caused the interrupt.

INT 1

INT 1 is meant to be a local interrupt for the RTC. In addition, it goes to J3-23 so it may be triggered by another external device.

APPLICATION PROGRAM

Interrupts are part of the CPU process. For setup example, see the program below.

File name	Description
RTC\RTC-3.C	Uses INT 1 to wake up from idle mode. INT 0 is similarly programmed.

INTRODUCTION

There are eight single ended analog to digital (A-D) input channels on the RPC-220. These channels are used to measure voltages from transducers, transmitters, 4-20 ma current loops, thermistors, etc. Input voltage range is 0 to 5 volts with 10 bit (1024) count resolution. Input impedance is 100K ohms to ground. Conversion time is about 28 micro-seconds. Reference is adjustable to 5.00 ±.20 volts.

This section begins with basic information to connect and use analog inputs. Later, descriptions of how to measure voltages other than 5 volts and calibration are presented.

CONNECTING ANALOG INPUTS

All analog inputs interface through connector J1. A STB-20 terminal board and CMA-20 ribbon cable can be used to provide screw terminal interface. Additional components, such as resistors and capacitors, may be connected directly to the screw terminals.

For greatest accuracy, connect unused inputs to ground. In environments with high voltage or static electricity, unused inputs can bleed over to other channels.

R4 may be adjusted to trim accuracy and/or maximum voltage to your system. See *Calibration* later in this section for more information.

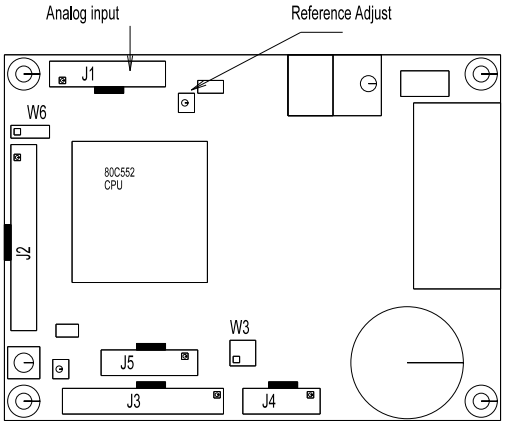


Figure 11-1 Analog Input and Reference Adj.

ACQUIRING ANALOG DATA

Analog data is acquired in polled or interrupt fashion. Polled mode takes about 50 micro-seconds/channel when written in C. A-D conversion time is about 28 micro-seconds. Read INPUT SIGNAL CONSIDERATIONS below for signal requirements.

INPUT SIGNAL CONSIDERATIONS

Slew Rate

Slew rate is the rate of signal change. It is usually expressed as volts/second, but other units are also used.

Ideally, signals presented to the RPC-220 are steady, DC signals. However, this is not a real world situation. Input signals are always changing.

To get a good reading from the ADC, the input signal should be relatively stable during the sampling time. Sampling time begins when conversion starts. To obtain 1/2 LSB (least significant bit) stability, the input signal should not change more than 2.44 mv in 3.3 micro-seconds or a rate of 750 volts/second.

The effect of a fast slew rate is a probable conversion result of 0x3ff. The problem is that a rapidly changing signal couples through to the input of the internal ADC comparator, saturating it. This is true when any channel, selected or not, changes rapidly. The input signal may change rapidly before any conversion. The critical time when the signal must be stable is during conversion.

A way to limit slew rate is to connect an RC filter to the analog inputs. The time constant should be 500 micro-seconds or more. If a 5 volt P-P sine wave signal is applied, the maximum frequency is about 600 Hz.[1]

Signal Source

The output resistance (impedance) of the signal source should be as low as possible. If the source impedance is too high, there will be a voltage drop over the source resistance.

The voltage over the output impedance is primarily due to the 100K ohm pull down resistors at the analog inputs. For 1/2 LSB accuracy, source impedance should be 50 ohms or less.

The resistor network R1 can be removed to allow an increase in source resistance. If you do this, then all unused inputs must be grounded.

EXTERNAL TRIGGER

External trigger (CPU signal STADC) at J1-16 causes an A-D conversion on a positive or rising pulse. Tie this line to ground if you do not use this signal.

This line can be tied to the optional RTC output at J3-4. Programmable, periodic acquisitions from 2 to about 32,000 times/second are possible. Execution time is saved because conversions are started outside of a program.

See application program AIN220-3.C for programming information.

NOISE

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

A high impedance input 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. Wire pairs can also be twisted. 5-6 twists/foot provides a good amount of noise cancellation.

Noise is defined in this section as any random change from a known input. The amount of noise you can expect under normal operating circumstances is ±3 counts for any input range.

One way to compensate for noise is to take a number of samples and average the results. Taking 7 or more samples does, in theory, cancel out any effects of noise. A problem with this is noise tends to group together. Taking 7 readings at one time might show no change from the norm. Another 7 readings might be all high. If possible, try to spread out readings over a period of time.

Noise is, by definition, random. If you were to plot out the deviations from a norm, it would roughly resemble a bell shaped curve. Experiments on the RPC-220 have shown that 99% of the readings are within the ±3 count reading. Noise readings were made with all inputs shorted to ground.

MEASURING HIGHER VOLTAGES

Voltages higher than + 5V are 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 using the 0-5V range.

Maximum Input Voltage	Resistor
6	20K
12.5	150K
24	380K

Use the following formula to determine the series resistance necessary for a maximum voltage input:

Rs = Vi * 20000 - 100000 0 - 5V range
Rs = Vi * 40000 - 100000 0 - 2.5V range

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

Adding a series resistor increases the source impedance beyond recommended values. A way to compensate for this is to put a capacitor at the line at J1. This effectively makes an RC filter. A 0.001 mfd or greater capacitor is adequate. This does limit the systems frequency response.

NOTE: When an input voltage exceeds the input range, other channel values are affected.

Measuring 4-20 mA Current Loops

Current loops are a convenient way to transmit a value and still assure the integrity of the signal. If the line should break, a 0 volts (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.

Since the measurement range is 1 to 5V, the count range is reduced by 20% to 819.

CALIBRATION

R4 is used to adjust the maximum, or reference voltage. Its location is shown in Figure 11-1 above. Normally, the reference is 5.000 volts. You can adjust R4 so the reference voltage is between about 4.8 and 5.2 volts. Adjusting the reference voltage to 5.12 volts makes easy binary to voltage conversion (5 mv/bit) for display or over range condition on a sensor output.

To calibrate or otherwise adjust the reference, connect a volt meter between J1-2 and R3, pin 1. If the voltage you are measuring is about 9 volts, then you probably have the wrong side of R3. Adjust R4 for the desired voltage. Adjustment is best accomplished using a 3 mm screw driver slot.

APPLICATION PROGRAMS

Programs are in the ANALOG directory.

File name	Description
AIN220-1.C	Simple polled mode. All channels printed out. Prints out as a real voltage. i.e. 3.74
AIN220-2.C	Same as above except interrupt driven. Int handler gets data and puts it into a global variable.
AIN220-3.C	Interrupt driven, but triggered externally at J1-16 (CPU STADC pin) on a rising pulse. Result for channel one only printed.

Functions are named AIN(*channel*), where *channel* is from 0 to 7. Return values are always in the 0 to 1023 range.

To convert a returned number to a voltage (reference set to 5.000 volts), multiply the result by .004883. The Dunfield C compiler does not support floating point directly. All application programs show a way to print out a value as if it were floating point by using 'long' data types.

Reference

[1]Phillips Semiconductors Application note EIE/AN93017, page 6-196 ff.

J1 ANALOG I/O CONNECTOR PIN OUT

The table below is a pin out for analog connector J1.

J1 Pin	Name
1	Analog input 0
2	Ground
3	Analog input 1
4	Ground
5	Analog input 2
6	Ground
7	Analog input 3
8	Ground
9	Analog input 4
10	Ground
11	Analog input 5
12	Ground
13	Analog input 6
14	Ground
15	Analog input 7
16	Start convert input
17	Analog output 0
18	+ V (≈8.5 volts)
19	Analog output 1
20	-V (≈ -8 volts)

INTRODUCTION

The RPC-220 has two programmable PWM outputs. These outputs are the source for the two analog outputs. Thus, when using an analog output, its corresponding PWM output may not be used. PWM frequency is programmable for approximately 162 Hz to 43Khz. Duty cycle is variable from 0 to 100% in 256 steps. The frequency is set for both PWM output channels while each channel had independently programmable duty cycles.

Analog output takes the PWM signal and filters it. Output voltage range is programmable from 0 to about 5V in 256 steps. It is effectively an 8 bit D-A.

PROGRAMMING PWM

PWM output signals come directly from the CPU at port J3. J3-22 is designated as PWM0 while J3-20 is PWM1. There is no buffering. Drive levels are TTL compatible. Lines must be buffered to directly drive opto modules.

Repetition frequency for both outputs is given by:

fpwm = 43369/(1+ PWMP)

Where:

PWMP is the prescaler frequency control register value at special function register address 0xfe. Its range is 0 to 255.

The duty cycle (in percent) for both outputs is given by:

Dcy = 100 * (1 - (PWMn/255))

Where:

PWMn is from 0 to 255. PWM0 is at special function register address 0xfc and PWM1 is at 0xfd.

100% duty cycle is defined as always high while 0% is always low.

Quick PWM Programming Example

PWM outputs are at J3-22 for PWM0 and J3-20 for PWM1. To program them, use monitor command 'MS' to modify the registers. For example type in the following command.

```
MS fc
```

Address 0xfc is PWM 0 duty cycle register.
The board responds with

```
00FC 00
```

Type in a 40. This makes channel 0 a mostly high pulse. Enter the data shown below. This programs PWM outputs for 4.8 Khz repetition rate pulses. View the outputs at J3-22 for PWM0 and J3-20 for PWM1.

```
00FC 00 40 00FD 00 c0 00FE FF 08 <cr>
```

Address 0xfd is PWM 1 duty cycle register and address 0xfe determines both channels repetition rate (frequency).

Example program PWM.C under directory PWM generates a ramp at the analog outputs by continuously varying the duty cycle.

ANALOG OUTPUTS

Output voltage is proportional to the PWM channel's duty cycle. Thus, the output voltage changes in steps of 1/255, or approximately 20 mv/step. This makes the analog output act like an 8 bit D-A converter. Accuracy depends upon the 5V supply. The analog outputs are proportional to the PWM duty cycle.

Using on card regulators, maximum output voltage can be 0.25 from the ideal 5.0V. Thus, the output can be as much as 5% off.

When analog output is used as part of a system with feedback, absolute accuracy is generally not critical. This is because the output changes according to demand requirements. When the output is used as a reference (i.e. 2.5V = 1/2 speed) and there is no feedback, then accuracy can suffer. In situations where accuracy is important, it is better to use a card with accurate analog output (such as the RPC-210), or adjust the output by reading the input.

The output may be "adjusted" by bringing the outputs to one of the A-D's on board. (The A-D uses a fixed reference so accuracy is higher.) Simply read the analog output and adjust the duty cycle as necessary to bring the voltage to the correct level.

Response Time

Since the analog outputs are derived from digital pulses, there are a couple of things you have to take into account. First, there is some ripple on the output. The amount of ripple depends upon the PWM frequency. Generally, the higher the frequency, the better. The CPU PWMP register should be programmed for 0x10 or less. Lower PWM frequencies increase ripple.

Second is step response time. Pulses are filtered through a 1 pole filter. This means there is some delay time from when the duty cycle changes to output change. Generally, this is about 100 ms for small steps and 200 ms for larger (1/2 scale) steps.

Analog outputs are at J1. J1-17 is analog output 0 while J1-19 is analog output 1. Output channels are controlled by their corresponding PWM channel.

Example program PWM.C under directory PWM generates a ramp on the analog outputs by continuously varying the duty cycle.

Measuring outputs using on card A-D

The analog outputs can be set to a more precise level by reading them at the A-D input. Due to interactions between the analog output and A-D circuits, a 20K ohm or greater resistor should be placed between A-D input and output. If the resistor is not installed, A-D conversions intermittently read full scale (1023 counts).

Output levels and noise

The operational amplifier used to buffer outputs gets its power from the RS-232 generator. If you turn of the RS-232 output, its quite probable analog output will be affected.

There is some noise (ripple) on the analog outputs. This noise comes from two sources: First is the PWM signal it self. If the CPU PWMP register is programmed for a low frequency, there will be ripple on the output. To correct this problem, decrease the value written to PWMP to 0x10 or lower.

The second noise source is the RS-232 generator. The RS-232 chip U9 generates both + and - voltages in the 8 volt range. The positive voltage is somewhat filtered. The negative is not. To clean up the negative voltage, put a 10 micro-farad or larger capacitor on the -V pin at J1-20. Capacitor voltage should be 16 volts or higher.

PIN OUTS AND REGISTER INFORMATION

The following tables are partial pin outs for J1 and J3. Full pin out for J1 is in SECTION 11 and J3 in SECTION 6.

Pin outs

Conn	Function
J1-17	Analog output, channel 0
J1-19	Analog output, channel 1
J3-22	PWM output, channel 0
J3-20	PWM output, channel 1

PWM registers

Register	Address (in special function area)	Function
PWMP	0xfe	Prescaler
PWM0	0xfc	Duty cycle
PWM1	0xfd	Duty cycle

APPLICATION PROGRAM

The following program is in the PWM directory. A brief description follows:

File name	Description
PWM.C	Initializes PWM prescaler and varies duty cycle from 0 to 100%. Used also to view voltage ramp from analog outputs.

INTRODUCTION

Port J5 supports character LCD's and some VF displays in 4 bit data mode. Seven lines from CPU port 1 and one from CPU port 3 are used to control the display. Angle/contrast adjustment via R9 is provided as are + 5V and ground. Figure 13-1 on the next page shows the location of the connector and R9.

CPU I/O lines for the display are shared between J3 and J5. See Table 13-1 for a list of CPU lines used.

Any number of character displays may be used. Program examples assume a HD44780 controller is used. This controller is used on Optrex, AND, Stanley, and other displays. Some VF displays from Noritake are LCD interface compatible.

DISPLAY PIN OUTS

Pin outs for a size and type of LCD are generally standard. However, pin outs for different display sizes are not. The mechanical pin out for a 4 x 40 display is different from a 4 x 20, which is different from a 2 x 20. Some are dual 16 pin, others are dual 14 pin, and some are single line 14 pin. To further complicate matters, the numbering scheme changes whether you take the signals from the top or bottom of the display. Electrically, all of these displays are the same or very similar.

The following tables are pin outs for J5 on the RPC-220 and the two most popular display types. Table 13-2 is for a 4 line by 40 character display. Table 13-3 is for most other displays. This table is usually valid for 14 pin displays.

J5 pin out was optimized for a Optrex 4 x 40 display. A simple 16 pin IDC connector from J5 to the display is all that is needed. The connector from this display must mount on the bottom of it.

Other 14 pin, dual in line displays may also directly connect using a simple IDC connector. The display connector mounts on the top side and the ribbon cable is reversed (display pin 1 goes to pin 14 on J5 side).

Connector Orientation

J5 is an open header connector. Pins 1 and 2 are marked on the board. As a further aid, a "key" outline is also on J5.

CONTRAST/ANGLE ADJUSTMENT

Pot R9 is used to adjust display contrast and viewing angle. Its location is shown in Figure 13-1 on the next page. The pot is adjusted to 0V at the factory. Its range is approximately + 5V to -7V, depending upon current draw.

Adjust R9 for the best display after it is initialized. Use a 3 mm slot screw driver.

Table 13-1
LCD Port J5 pin out, CPU line, and function

J5 pin	CPU Line	LCD function
1	P1.2	Data
2	P1.3	Data
3	P1.0	Data
4	P1.1	Data
5	NC	no connect
6	NC	no connect
7	NC	no connect
8	NC	no connect
9	P1.6*	Read/write (0 = write)
10	P3.4	E signal strobe
11	none	Angle/contrast adjust
12	P1.7*	Register/data select
13	none	+ 5V
14	none	Ground
15	NC	no connect
16	P1.4	Enable 2

* CPU lines are open drain.

Table 13-2 4 x 40 LCD signal pin out

J5 pin	Disp. pin	Symbol	Function
2	1	DB7	Data bus
1	2	DB6	Data bus
4	3	DB5	Data bus
3	4	DB4	Data bus
6	5	DB3	no connect
5	6	DB2	no connect
8	7	DB1	no connect
7	8	DB0	no connect
10	9	E1	Enable 1
9	10	R/~W	Read, write
12	11	RS	Reg. select
11	12	Vee	Contrast V
14	13	Gnd	Ground
13	14	+ 5V	+ 5V
16	15	E2	Enable 2
15	16	NC	No connect

Table 13-3 Other LCD pin out

J5 pin	Disp. pin	Symbol	Function
14	1	Gnd	Ground
13	2	+ 5V	+ 5 power
11	3	Vee	Contr. adj
12	4	RS	Reg. select
9	5	R/~W	Read/write
10	6	E	Enable
8	7	DB0	no connect
7	8	DB1	no connect
6	9	DB2	no connect
5	10	DB3	no connect
3	11	DB4	Data bus 4
4	12	DB5	Data bus 5
1	13	DB6	Data bus 6
2	14	DB7	Data bus 7

APPLICATION PROGRAM

LCD display program is in the LCD directory.

File Name	Description
LCD440.C	Demo/driver for 4 line x 40 character LCD display. Specifically, for P/N 1723.

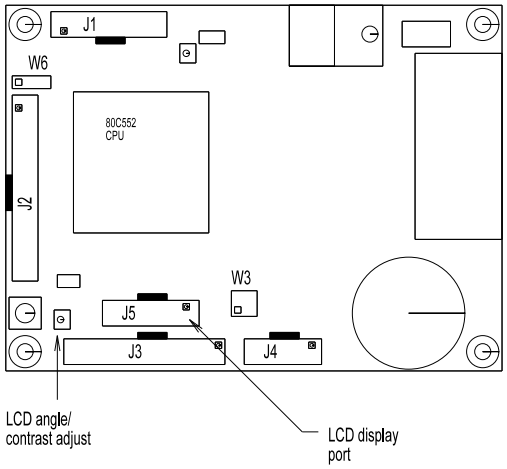


Figure 13-1 LCD Connector and Angle Adjust

INTRODUCTION

Expansion port at J2 allows you to connect other I/O type cards for expanded capability. Power connector P1 is for ground, + 5, and higher voltage inputs.

EXPANSION PORT

Keep the connector length between J2 and expansion card connectors as short as possible (4 inches or less).

Power and ground are available at this connector. Since the ribbon cable is small gauge and has high resistance, keep power currents as low as possible (less than 100 ma) to prevent ground loop problems. Ground loop problems manifest themselves as random resets, lockups, and inaccurate A-D readings.

J2 expansion port pin out is shown at the end of this chapter.

EXTERNAL POWER

The RPC-220 accepts different voltage ranges, depending upon the card. Recommended operating voltage is 5 ±0.25V, and is standard for all cards. Absolute maximum "5 volt" supply to the card is 6 volts. This means you can hook up 4 ni-cad, carbon-zinc, or alkaline batteries to the 5 volt input.

Higher voltages are applied to the 'V' terminal on P1. Jumper W4 to connect regulator output to the rest of the card. External voltages of 6 to 16 volts are for P/N's 1715 and 1720. 5.4 to 21 volts apply to P/N 1710 only. Higher voltages may be applied provided a heat sink under U10 is used. See "Heat Sink" below. Operating current is 100 ma for the low power board (P/N 1710) and 160 ma for others.

Board + 5 and ground are applied through appropriate pins at J2, J3, J4 or P1.

P/N 1710 has reverse voltage protection if power is applied to externally. The board itself does not have 5V reverse protection.

The maximum external voltage of 16 or 21 volts is due to regulator power dissipation (see "Heat Sink" below). Up to 30 volts may be applied if an adequate heat sink is used under voltage regulator U10. The voltage may be stepped down from higher voltages yet (up to 40 volts) by putting a regulator (such as a 7824) to P1. Depending upon the external voltage, you may need a heat sink on the extra regulator.

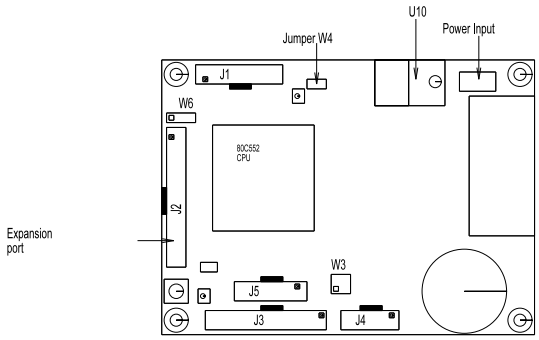


Figure 14-1 Power Input, Jumper, and Expansion Connectors

Heat Sink

A heat sink under U10 is normally not necessary. Conditions when you need a heat sink depend upon the amount of power you expect U10 to dissipate and ambient temperature.

First step is to determine the power you expect U10 to dissipate. This is calculated as follows:

$P(MAX) = (V_{BATT} - 5) * I$

Where: V_{BATT} is battery or supply voltage

$I =$ current into the board.

Current into the board depends upon its model number. Use 100 ma for P/N 1710 and 170 ma for all others. Don't forget to add current for other devices connected to the bus, I/O ports, LCD display, and if you are taking power from the regulated output.

No heat sink is necessary if power(P(MAX)) is less than 1.6W and the board operates at 25°C. This means a 21 volt supply can be used with P/N 1710 and 14 volts with all other models (assuming no additional current is used).

There is a small heat sink on the RPC-220. In marginal situations, you can screw down U10 to the board to get slightly better heat dissipation.

If you expect to operate the board at higher temperatures, use the following formula to determine maximum power dissipation:

$$P(MAX) = \frac{125 - T_A}{60}$$

Where: T_A is ambient temperature.

Use a heat sink for a TO-220 IC. Suggested heat sink types from Aavid are:

AAvid P/N	Height	P(MAX) @ 25°C	P(MAX) @50°C
577002B00000	0.250"	2.85	2.14
577102B00000	0.375"	3.46	2.63
577202B00000	0.500"	3.64	2.73

Other heat sink types and styles may be used depending upon your mounting considerations. Heat dissipation is greatly increased with air flow around the regulator.

Be sure to use silicone grease to increase heat flow. An insulator is normally not necessary unless you want to keep the heat sink from shorting to an external case.

LOW POWER MODES

Low power modes are discussed in SECTION 7, REAL TIME CLOCK, LOW POWER MODES.

Expansion Port Pin Out

J2 Pin	Description
1	Data bus 0
2	I/O chip select (active low)
3	Data bus 1
4	High voltage (5.4 to 24 V) power
5	Data bus 2
6	INT 0
7	Data bus 3
8	Reset (active low)
9	Data bus 4
10	+ 5V supply
11	Data bus 5
12	Power ground
13	Data bus 6
14	Power ground
15	Data bus 7
16	Address A4
17	Address A0
18	+ 5V supply
19	Address A1
20	Address A5
21	Address A2
22	no connect - leave open
23	Address A3
24	ALE signal from CPU
25	Read strobe (active low)
26	Write strobe (active low)

CPU

Phillips 80C552 operating at 22.1184 Mhz

Power Supply

Inputs: 5 ±0.25VDC or
6-16V (RPC-220 P/N 1715, 1720) or
5.4 to 21V (RPC-220 P/N 1710)
Higher input voltage possible with heat sink.
Current: 180 ma (P/N 1715, 1720) maximum.
100 ma (P/N 1710) maximum.
Outputs not loaded.
5 ma maximum, lowest power mode (P/N 1710)

Currents measured without RS-232 attached.

Screw terminals provided for + 5V and external high voltage input.

Memory

External RAM -

128K or 512K, depending upon model. 32K accessible as code during development mode.

Code address during development mode is 0x8000 to 0xffff. All external RAM accessed using MOVX type commands and by segment selecting.

RAM is battery backed up when real time clock option is installed. Backup time varies with usage and operating temperature. Backup time is estimated at 4 to 6 years at 25°C, board not powered. Backup time reduced by 50% when operating at 50°C.

Program EPROM -

32K, Atmel type 29C256. This is a PEROM and does not need sector erase. However, data is written in 64 byte blocks.

Monitor included. This is written over after program development.

A 64K UV EPROM (27C512 type) is possible. See SECTION 3, PROGRAMS LARGER THAN 32K for modification instructions.

Analog Input

Channels: 8
Resolution: 10 bits (about 5 mv/count)
Range: 0 - 5V
Input impedance: 100K ohms
Conversion time: 28 micro-seconds
Reference: Adjustable from about 4.8 to 5.2 volts

Analog Output

Channels:	2
Resolution:	Effectively 8 bits (20 mv)
Range:	0 - 5V Maximum output voltage, and accuracy, dependent upon 5V supply
Ripple:	20 mv P-P maximum at 50% duty cycle, 4.8 Khz PWM frequency
Output current:	2 ma maximum/channel, 4 ma total.
Response time:	300 milli-seconds or less

Analog outputs use PWM outputs. When an analog output is used, its corresponding PWM output cannot be used.

PWM Output

Channels: 2
Duty cycle: 0 to 100% 256 steps
Frequency: programmable from about 163 to 42 Khz.
Output current: 4 ma maximum, TTL compatible

PWM outputs generate analog outputs. When analog output is used, its corresponding PWM output cannot be used.

Real Time Clock

Type: DS1689

Accuracy:	1 min/month. This will vary depending upon operating temperature. Accuracy specified at 25°C
Interrupts:	May generate an INT1 when J3-23 is jumpered to J3-25. Also used to wake up card from low power mode.
Battery backup:	Battery backs up RAM. Connector for external battery provided when special ordered without battery.
Operating life:	Four to 6 years, depending upon temperature and humidity.

Reset

CPU resets when 5V supply falls below about 4.65 volts. Held in reset until supply rises to about 4.65 volts. Reset time is about 300 ms. A pulse from the real time clock resets CPU to wake it up from low power mode.

LCD Display Port

Interfaces to RPC P/N 1723 character display. Contrast adjustment provided.

Serial Ports
Type: RS-232
Ports: 2. One hardware from CPU, other is software.
Control lines: Software lines may be used for RTS/CTS control.

MECHANICAL

Size: 2.85" x 3.85"
Mounting: Mounting hole centers are 0.150" from each corner. Hole sizes are 0.125" dia. in 0.250 pad. Use 0.250" standoffs for mounting.

SHOCK AND VIBRATION

The RPC-220 board should withstand 5 g's random vibration, 5 to 500 Hz if some moderate amount of precautions are taken.

Secure voltage regulator U10 to the board and/or heat sink using a 4-40 or 6-32 nut and screw. The pad under U10 is not connected to any signal. It is there to aid in power dissipation.

U3 is plastic and has relatively little mass. It is not as susceptible to shock and vibration as a ceramic chip. It can be secured by using a nylon tie wrap around the chip. Another method is to carefully apply silicon rubber to each end of the chip and socket.

The battery is soldered to the board. However, in continuous vibration environments the welds could break. Metal fatigue is possible in the bands connecting the battery to the board. You will have to put a non-conductive piece between the battery and IC under it. Keep in mind that the clock and RAM draw micro-amps of current. Any conduction between the + and - sides of the battery drains it. Be careful of chemical adhesives as they may initially draw a lot of current unit they cure.

Another way around the battery problem may be to remove B1 and solder leads to jumper W7. W7 is for an external battery.

Secure the settings for the reference and contrast pots R4 and R9. These pots are relatively open, so a high viscosity fluid should be used.

MOUNTING TO A MOTHER BOARD

The RPC-220 can plug into a "mother board" which has power, signal and other functions on it. Figure 15-1 shows pin 1 connector locations. Mother board connectors are dual row, female, on a 0.1" x 0.1" matrix, for 0.025" square posts. Suggested connector types are:

Manufacturer	P/N and size
3M	929852-01-26 26 position
3M	929852-01-20 20 position
3M	929852-01-10 10 position
AMP	2-17314508 26 position
Oupiin	204-126GS PCB mount, 26 pin
Oupiin	204-120GS PCB mount, 20 pin
Oupiin	204-110GS PCB mount, 10 pin
Samtec	SSM series (surface mount)
Samtec	SSW, SSQ series (through hole)
Samtec	ESW, ESQ series (elevated)

Samtec connectors come in a variety of plating and lengths. Contact them at 800-726-8329 for more information.

Maximum component height is 0.350". However, other components differ in height around the board. Take this into account when laying out your board.

Remote Processing can, on special order and in quantity, put connectors on the circuit side of the board.

Connector Positions

Leaders in Figure 15-1 show the location of pin 1 for J1 - J5. Pin 2 is shown immediately next to pin 1 as a circle. View is from component side of the RPC-220.

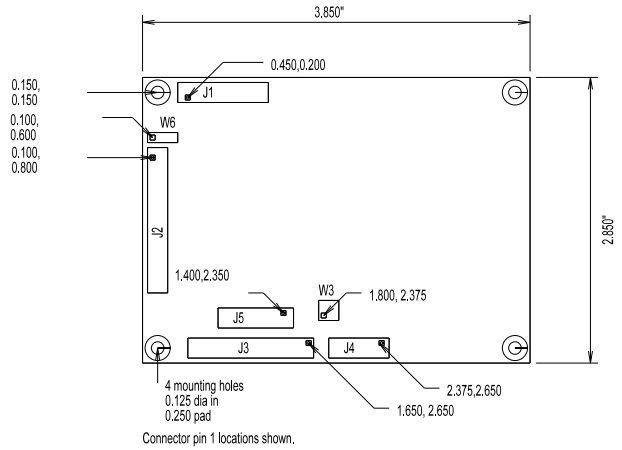


Figure 15-1 Board Size and Connector Placement

Four mounting holes are 0.150" from each corner. Hole diameter is 0.125" and is in a 0.250" diameter pad.