RTC-52PLUS
(80C52- with Basic Interpreter)
Real Time Controller

Technical Manual
Rev 2.0

Micromint, Inc

RTC-52PLUS

RTC series Microcontroller
Using Micromint's 80C52
Masked BASIC Interpreter

Technical Manual

Release 2.0
03/27/01

Micromint, Inc.
902 Waterway Place
Longwood, FL. 32750

COPYRIGHT

Micromint, Inc.
902 Waterway Place
Longwood, FL. 32750

DISCLAIMER

Devices sold by Micromint are covered by the warranty and patent indemnification provisions in its Terms of Sale only. Micromint makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Micromint makes no warranty of merchantability or fitness for any purposes. Micromint reserves the right to discontinue production and change specifications and process any time without notice. This product is intended for use in normal commercial applications. Applications requiring extended temperature and unusual environmental requirements, or applications requiring high reliability applications, such as military, medical life support or life-sustaining equipment, are specifically NOT recommend without processing by Micromint for such application.

Occasionally in this manual we refer to other manufacturers' products. Such reference does not constitute an endorsement of these products, but are included for the purpose of illustration or clarification. We do not intend such technical information and interface data to supersede information provided by individual manufacturers.

Conditions of Sale and Product Warranty

Micromint, Inc. and the Buyer agree to the following terms and conditions of Sale and Purchase:

1. Micromint, Inc. extends the following warranty: a factory manufactured circuit board or assembly carries with it a one-year warranty covering parts and labor. Any unit, which is found to have a defect in materials or workmanship, will, at the discretion of Micromint, Inc., be repaired or replaced.

2. Products distributed, but not manufactured by Micromint, Inc. carry the original manufacturers warranty, usually 90 days. Such products include, but are not limited to: power supplies, sensors, I/O modules, MOSARTs, LCD displays, battery-backed RAM modules, and disk drives.

3. A minimum inspection fee must be prepaid for the repair of units that are no longer under warranty. Call Micromint, Inc. for a current list of fees.

4. Micromint, Inc. will not be responsible for the repair or replacement of any unit damaged by user modification, negligence, abuse and mishandling, or improper installation.

5. Micromint, Inc. is not responsible to the Buyer for any loss or claim of special or consequential damages.

6. All units returned for repair must first receive prior authorization from Micromint, Inc.. A return authorization number can be obtained by phone, letter, or email. Please retain a record of this number, since most subsequent correspondence will refer to this authorization. Under no circumstances should any product be returned to Micromint, Inc. without such authorization, and Micromint, Inc. assumes no responsibility for returns unaccompanied by an authorization number. All returns must be shipped prepaid and should be insured. Micromint, Inc. is not responsible for losses or damage during shipment. Repaired units will be returned postage- and insurance-paid.

7. Micromint, Inc. reserves the right to alter any feature or specification at anytime. This right extends to fees, charges, and any other conditions or warranties contained herein.

REV. 10/89

1.0 Mirocontroller Evolution

The 8031 microcontroller is probably the most widely used core for embedded processing in the industry. Its price/performance ratio as a general-purpose controller helps to keep it a popular choice for new designs. It is not surprising that manufactures have based many of today's newer processors on the 8031 core. For many years Intel had produced a masked 8032 part with a BASIC interpreter. Many products took advantage of this on-board language since it offered users a high-level language, which was easy to use, and required NO special development software. Some where along the line Intel decided to cease production of the chip. Micromint quickly took up the slack by having a compatible version masked into a CMOS part. This CMOS part operates down to 0Hz, being truly static. Lower operating currents make this part even more attractive.

The masked BASIC language allows the user to sit down at his/her PC connected to the RTC-52 PLUS (using serial communication software) and literally start programming immediately. BASIC is easy to learn and very powerful. Full floating-point numbers means you're not limited to just using integer arithmetic in your calculations. Print formatting (and strings) is available for your output.

Developing BASIC programs in real-time means easier debugging. You can run your program and make necessary changes immediately. Completed BASIC programs can be put into an EPROM for non-volatile storage, which can automatically run the program upon power-up or reset. Optionally, with the addition of a non-volatile RAM module, your program can be placed within the (non-volatile) RAM to replace the need for an EPROM. Since non-volatile RAMs are RAMs they can be written too by mistake, so use them with care!

The RTC-52PLUS is a plain-brown-wrapper 80C32 controller that is optimized with the most requested I/O on-board for minimal configuration applications. While the RTC-52 PLUS may be perfect for mast single board applications, it does contain the RTC expansion bus for adding on those special bits of I/O which might be necessary for your specific application. If that special I/O isn't available as a standard expansion board, it can be easily added to an RTC prototyping expansion board.

The RTC system measures only 3.5 inches square and uses vertical stacking connectors for I/O expansion. The RTC-52 PLUS processor board contains the 80C52 BASIC processor, EPROM and RAM memory, address decoding and buffering, 24-bits of parallel I/O, a 2-channel 12-bit A/D converter, 1 full duplex serial port and a serial printer port (available through Port 1 bit 7). Each vertically stacked expansion board only increases the system height by .75 inches.

2.0 80C52 Pin Description

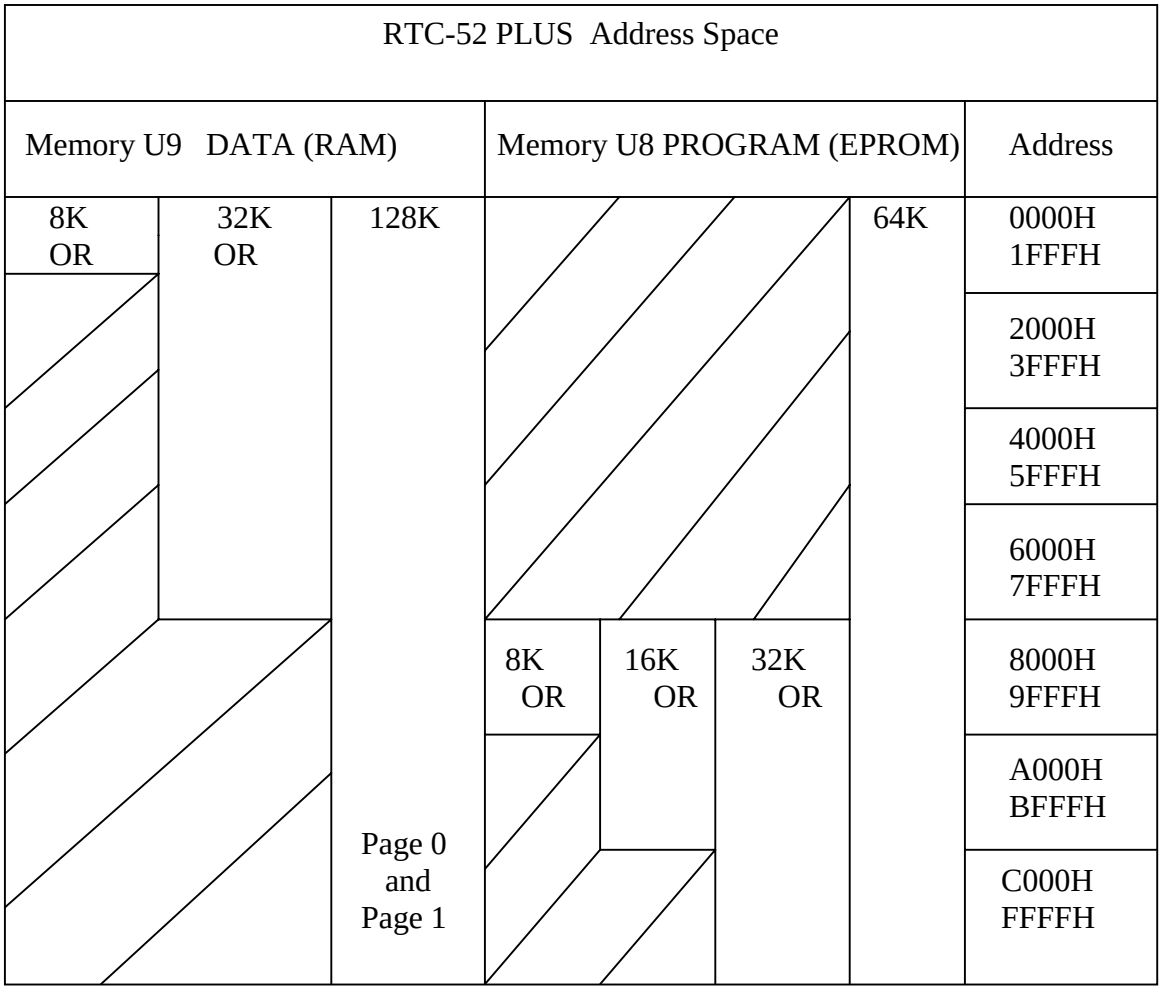
Port 0	Pins 36-43	8-bit open drain bi-directional I/O (multiplexed low-order data/address for external memory).
Port 1	Pins 2-9	8-bit bi-directional I/O.
Port 2	Pins 24-31	8-bit bi-directional I/O. (High-order address for ext. memory)
Port 3	Pins 11 & 13-19	8-bit bi-directional I/O Secondary options are as follows: **RXD/data** Serial channel's receiver **TXD/data** Serial channel's receiver ***INT0** Interrupt 0/ counter gate 0 input ***INT1** Interrupt 1/ counter gate 1 input **T0** counter 0 input **T1** counter 1 input ***WR** write for external data memory ***RD** read for external data memory
XTAL1	Pin 20	Crystal in
XTAL2	Pin 21	Crystal out
Vcc	Pin 44	+5V
Gnd	Pin 22 & 23	Ground
ALE	Pin 33	Address latch enable
***PSEN**	Pin 32	Read for external program memory
***EA**	Pin 35	Tied to logic high for executing code masked within the 8x5x series processors with internal ROM. If tied to logic low this disables internal ROM, all instructions are fetched from external program memory.
NC	Pin 1, 12 & 34	Not connected

3.0 RTC-52 PLUS External Addressing Space

The RTC-52 PLUS microcontroller can directly address 64K of external memory. That is 64K of DATA memory and 56K of PROGRAM memory. The *RD and *WR lines control DATA memory (read/write and I/O) and the *PSEN line controls PROGRAM memory (read only memory). Overlapped space occurs when the *RD and *PSEN lines are combined, a combined DATA/PROGRAM space is useful when code must be executed out of RAM for various reasons.

Many combinations of 8K through 128K RAMs and 8K through 64K EPROMs are possible on the RTC-52 PLUS board. The first 16 K of the address space (0000H-3FFFH) is always separated into DATA space for RAM and code space for EPROM. The next 16K of address space (4000H-7FFFH) can be separated or combined spaces. The last 32K of address space (8000H-FFFFH) can also be assigned as either separate of combined DATA and CODE space.

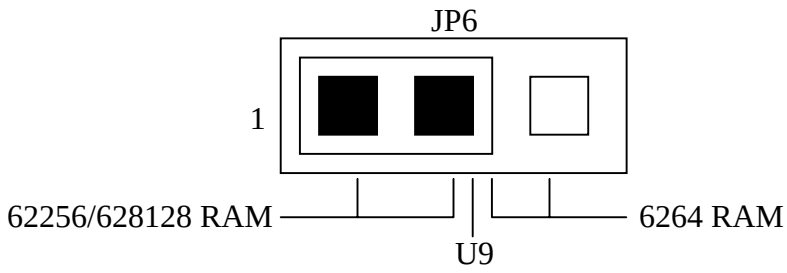
The following chart shows conventional addressing for the memories used on the RTC-52 PLUS:



4.0 RTC-52 PLUS Memory Device Configuration

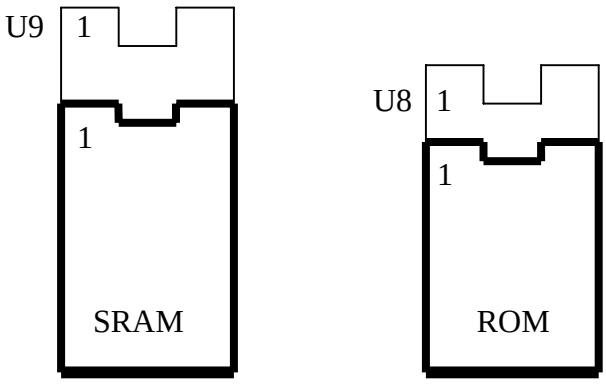
4.10 SRAM Size

The RTC-52 PLUS provides IC sockets for two memory devices. Socket U9 has been designated as the SRAM position and U8 as the EPROM position. SRAMs of sizes 8K to 128K by 8 may be used in the RTC-52 PLUS. However, the user must know how much memory is installed and access only that area because the RAM will wrap around and be duplicated in multiple address blocks. Only one jumper needs to be placed to configure the SRAM socket U9 for your SRAM chip, JP6.



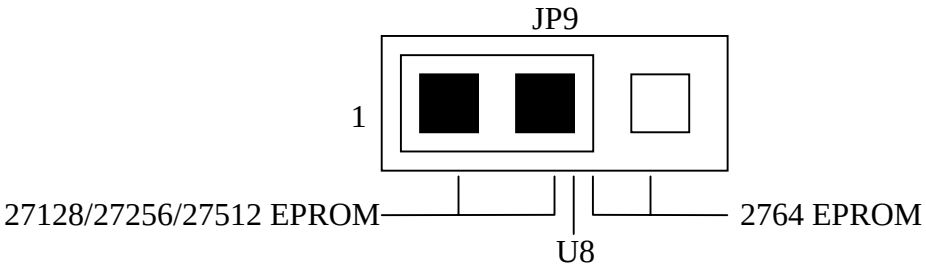
SRAM size configuration jumper JP6 is shown configured for 628128 (128Kx8) SRAM.

Note: Smaller memory devices should be lower justified in the IC sockets as shown below.

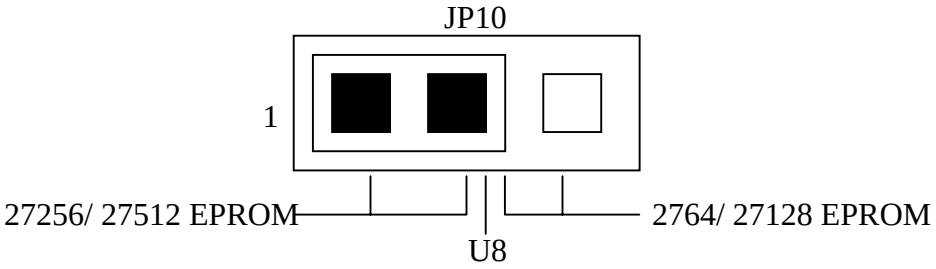


4.20 EPROM Size

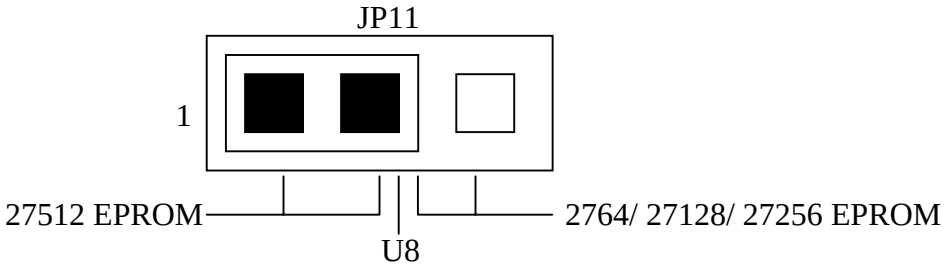
Four different size EPROMs may be used in the RTC-52 PLUS ROM socket U8. As with the RAM, EPROMS smaller that 64K will continually wrap around and be duplicated in multiple address blocks. Three jumpers need to be placed to configure the EPROM socket U8 for your EPROM chip; JP9, JP10, and JP11.



The EPROM configuration jumper JP9 is shown configured for a 27512.



The EPROM configuration jumper JP10 is shown configured for a 27512.



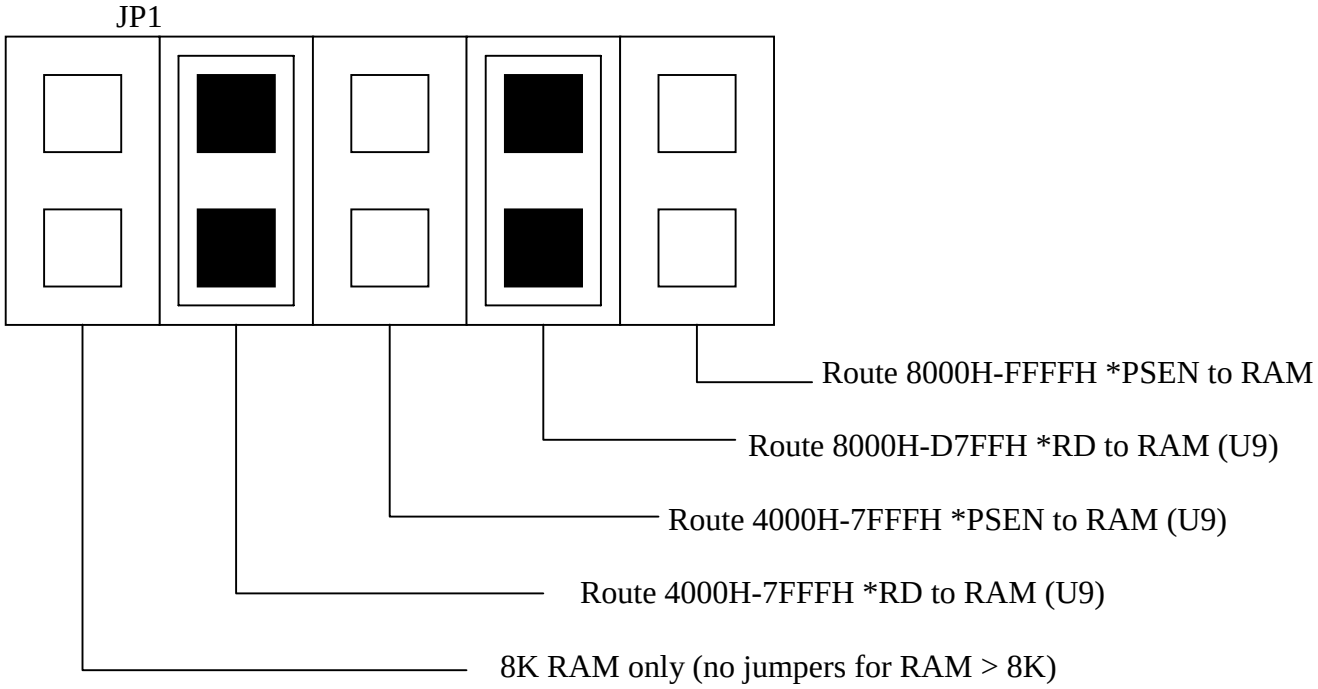
The EPROM configuration jumper JP11 is shown configured for a 27512.

4.30 Defining Data and Code Space

The 64K address space is divided into three areas: 0000H-3FFFH, 4000H-7FFFH, and 8000H-FFFFH. The first area (first quadrant) is permanently separated into non-overlapping data and code spaces. Any access to code space (*PSEN 2000H-3FFFH) goes to the U8 EPROM socket and any access to data space (*RD and *WR 0000H-3FFFH) goes to the SRAM socket. NOTE: Code space 0000-1FFFH is not accessible when using the 80C52 BASIC chip as this address space is internal to the processor.

The second area 4000H-7FFFH (second quadrant) can be configured as either overlapping or non-overlapping data and code spaces. The *PSEN (code read) line can be directed toward either the SRAM or the EPROM allowing a code fetch for execution from either device. The *RD (data read) line can be directed toward either device as well (although, this may not make logical sense to you). NOTE: Since the RTC-52 PLUS has room for only two memory devices, and one must be a RAM, if your BASIC program (which must be less than 24K) will be in an EPROM which is less than 64K, you will probably not use code space in either the first or the second quadrant.

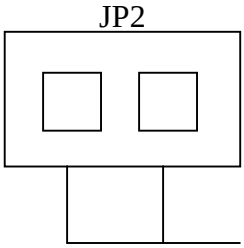
The third area 8000H-FFFFH (third and forth quadrants) can be configured as either overlapping or non-overlapping data and code spaces. The *PSEN line can be directed toward either the SRAM or the EPROM allowing a code fetch for execution from either device. The *RD line can be directed toward either device as well (although again, this may not make logical sense to you).



JP1 is shown configuring all *RD signals to RAM and all *PSEN signals to EPROM. This separates the data and code spaces providing non-overlapping spaces (64K [potential] data space and 64k [potential] code space).

5.0 External Code Selection

The 80C52 BASIC microcontroller requires the *EA (pin 31 on the microcontroller) to be pulled to a logic high level. This instructs the processor to start executing machine language code starting internally at address 0000H (code space).

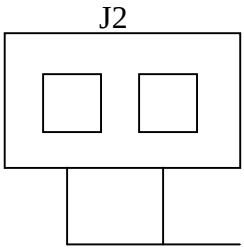


Enable external EPROM code at 0000H

JP2 shows the microcontroller enabled for internal code execution of the BASIC interpreter.

6.0 Resetting the RTC-52 PLUS

Power-on reset of the RTC-52 PLUS occurs whenever power reaches approximately 4 volts. Manual reset is accomplished by momentarily shorting the pins of J2 together or connecting a normally open push button switch to J2 and pressing it.



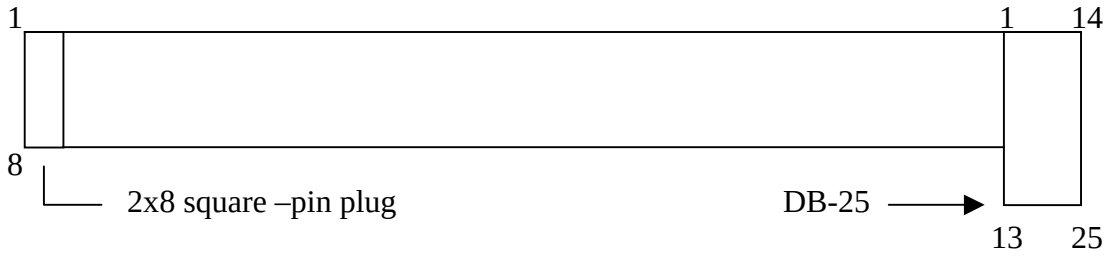
Momentarily short pins to reset board

Use J2 for connecting a normally open push button switch as an external system reset.

7.0 RS-232 Communications

The user's standard communications interface is through the RS-232 console interface port. This port can be used to send and receive programs and data to and from the RTC-52 PLUS. The RTC-52 PLUS will auto-baud detect (a space character) from 300-9600 baud or can be programmed to autostart a saved program at a particular baud rate. An RS-232 driver converts the TTL-level serial signals to +10-volt RS-232-compatible signals.

An RS-232 connection is made to your terminal device using a simple straight through cable. A ribbon cable with a DB-25 at one end and a 2x8 square pin connector at the other end is easily made using locally available parts, or can be purchased from Micromint Inc.

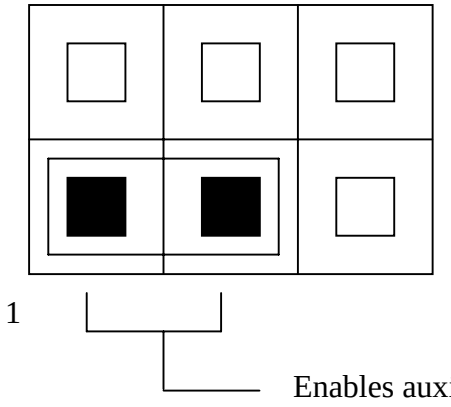


Cable is required for RS-232 communication.

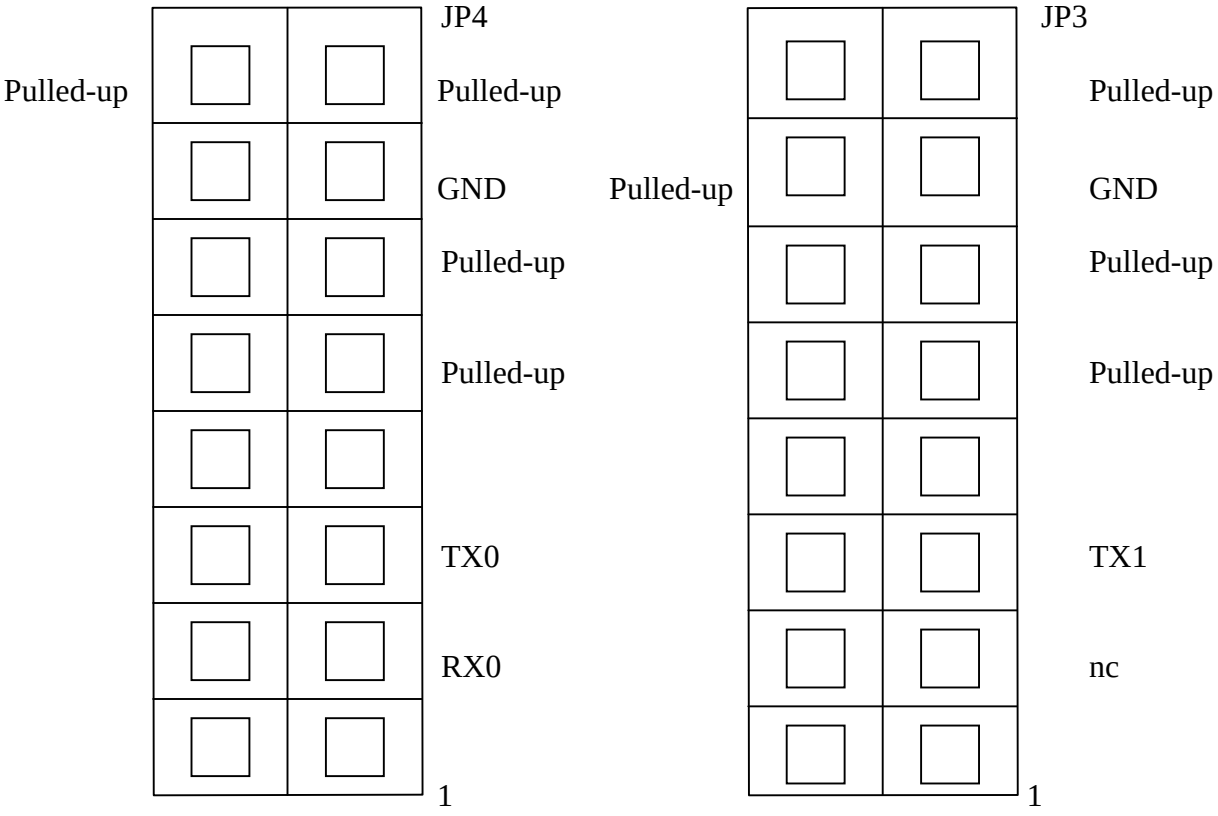
The RTC-52 PLUS is also equipped with an auxiliary serial output. This output port is frequently referred to as a serial printer port. The BAUD[expr] statement is used to set the baud rate for this printer port. In order for this statement to properly calculate the baud rate, the crystal (special function operator-XTAL) must be correctly assigned (e.g. XTAL = 9000000). BASIC assumes a crystal value of 11.0592 MHz if no XTAL value is assigned.

The main purpose of the software line printer port is to let the user make a "hard copy" of program listings and/or data. The command LIST# and the statement PRINT# directs outputs to the serial printer port. If the BAUD[expr] statement is not executed before a LIST# or PRINT# command/statement is entered the output to the software serial printer port will be at about 1BPS and it will take a long time to output something. It is necessary to assign a baud rate to the serial printer port before using LIST# or PRINT #.

NOTE: To eliminate unwanted noise from the RX input to the processor, remove the unused 75176 line driver chip.



JP8 must have a jumper installed as shown to enable the auxiliary serial printer port.



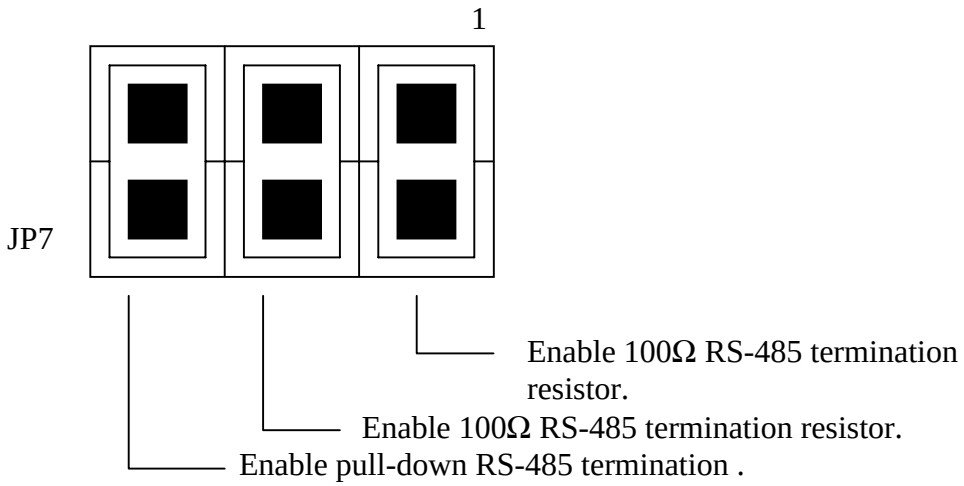
JP3 and JP4 are RS-232 interface connectors. JP3 is the auxiliary printer port and JP4 is the console serial port.

8.0 RS-485 Communications

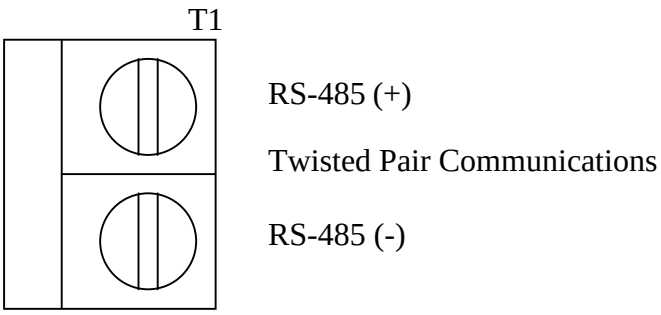
RS-485 communications over a single twisted pair can include multiple (up to 32) devices. Since each device can transmit and receive, certain protocols must be adhered to so message collision can be prevented. The simplest being “listen to the line and transmit only if free”. (The protocol you use will depend on the application and is beyond the scope of this manual.) JP7 enables a termination resister across the twisted pair and should be installed only on the microcontrollers located at the extremes of the twisted pair (one at each end). If low power operation is of great concern and the RS-485 is not being used, current consumption can be reduced by removing the 75176. (Actually the 75176 should be removed whenever RS-485 is not being used.)

RS-485 requires termination resistors on the network for it to operate properly. On any RS-485 network the first and last device on the network need to have terminating resistors and a 100-ohm resistor between the two lines. All devices in the middle of the network also need a 100-ohm resistor between the two lines.

A set of screw terminal blocks (T1) is provided for RS-485 twisted pair communication.



JP7 shows termination of the RS-485 lines enabled.



The transmit line on the RS-485 driver is turned on by setting bit P3.4 (T0 in the 80C52). Clearing bit 4 of Port 3 on the processor turns off the RS-485 transmitter. When using the 80C52's BASIC interpreter this bit is not accessible through BASIC. It can only be accessed through an assembly language CALL statement. A two-statement machine language routine is all that is necessary.

To enable transmit driver:

```
0D2H 0B4H        SETB T0
22H              RET
```

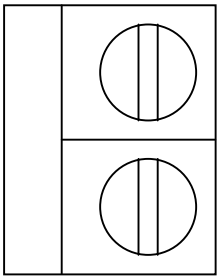
To disable transmit driver:

```
0C2H 0B4H        CLR T0
22H              RET
```

The code for these two routines (6 bytes) can be hard coded into your EPROM or poked into RAM using DATA statements from a BASIC program.

9.0 RTC-52 PLUS Power Requirements

A set of screw terminal blocks is provided for +5 volt power and ground. 150mA are required for 11MHz operation.



Ground

Power Inputs

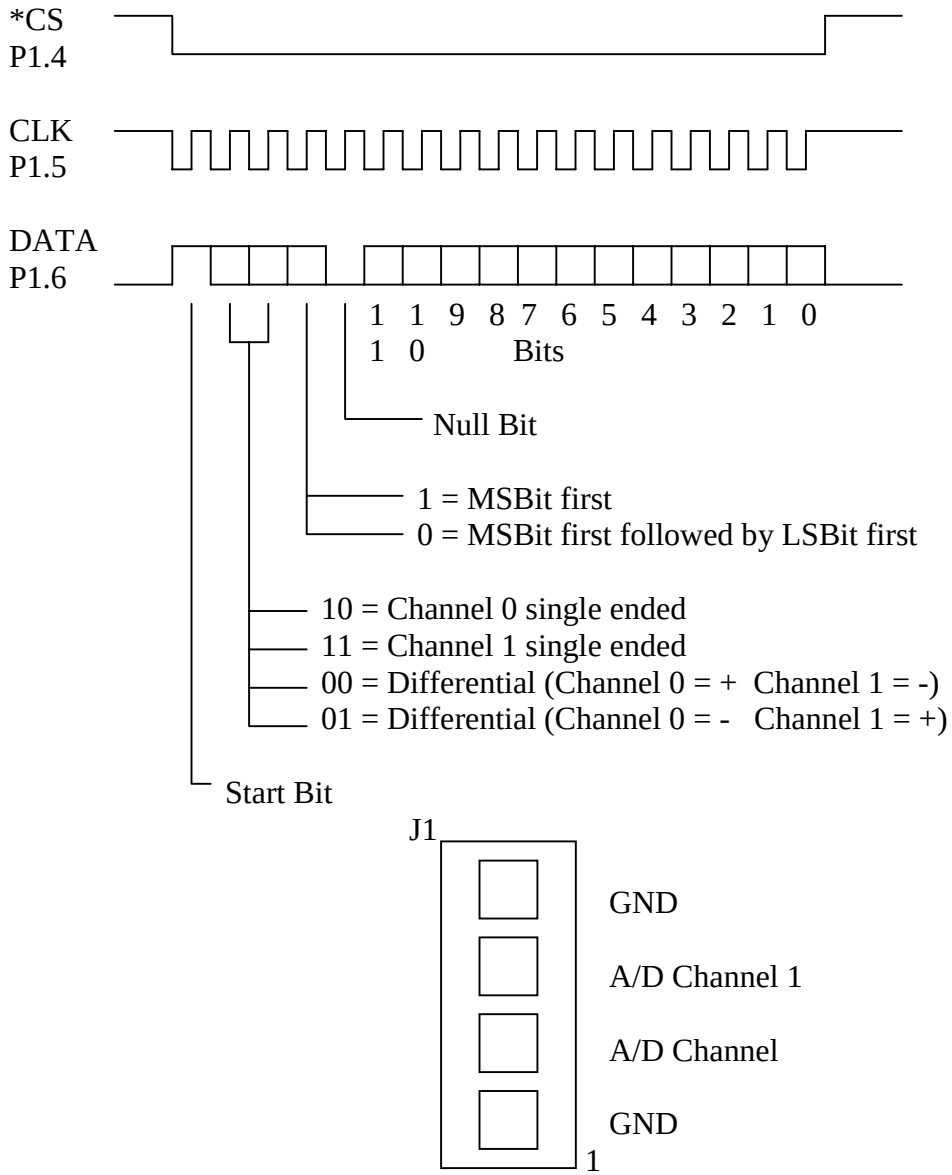
+5 Volts

10.0 Stand-Alone I/O Connections

In an attempt to allow the RTC_52 PLUS to be more widely used in stand-alone mode, the most popular I/O was added. Additional I/O includes 24-bits of digital I/O plus a two channel 12-bit analog-to-digital converter.

10.10 A/D Converter

A three wire serial A/D is used to provide the RTC-52 PLUS with two channels of 12-bit analog to digital conversion. The conversion is based on the clock speed; the maximum clock speed is 2.5µS high and 2.5µS low. Input impedance looks like 500Ω in series with 20pF. At maximum speed (about 90µS cycle time) the DC input current is about 1.56µA. A 750Ω source impedance will cause about 1 bit of full-scale error. If the source resistance can not be small, then the clocking speed can be reduced by a factor of 10 or even 100. This is actually an advantage if the A/D routine is written in BASIC since a BASIC program line executes about 100 to 1000 times slower than assembly language.



Analog input channels are available on J1, a 1x4 square pin header.

A/D Sample Program in BASIC

```
10 MTOP=2FFFH
20 REM* VALUES USED TO TOGGLE CHIP SELECT ON ADC
30 XBY(6007H)=0C2H : XBY(6008H)=094H : XBY(6009H)=022H
40 XBY(600AH)=0D2H : XBY(600BH)=094H : XBY(600CH)=022H
50 REM* VALUES USED TO TOGGLE CLOCK ON THE ADC
60 XBY(600DH)=0C2H : XBY(600EH)=095H : XBY(600FH)=022H
70 XBY(6010H)=0D2H : XBY(6011H)=095H : XBY(6012H)=022H
80 REM* VALUES USED TO TOGGLE THE DATA IN/OUT LINE ON THE ADC
90 XBY(6013H)=0C2H : XBY(6014H)=096H : XBY(6015H)=022H
100 XBY(6016H)=0D2H : XBY(6017H)=096H : XBY(6018H)=022H
110 REM* VARIABLE USED TO TOGGLE IN THE DATA FROM THE ADC
120 DA=040H
130 REM* CALLS ASM ROUTINES TO 'BIT BANG' THE READING FROM THE ADC FRO CH0
140 REM* Low the ADC's Chip Select
150 CALL 06007H
160 REM* Low the ADC's Clock
170 CALL 0600DH
180 REM* High the ADC's Data In/Out
190 CALL 06016H
200 REM* High the ADC's Clock
210 CALL 06010H
220 REM* Low the ADC's Clock
230 CALL 0600DH
240 REM* High the ADC's Data In/Out
250 CALL 06016H
260 REM* High the ADC's Clock
270 CALL 06010H
280 REM* Low the ADC's Clock
290 CALL 0600DH
300 REM* Low the ADC's Data In/Out
310 CALL 06013H
320 REM* High the ADC's Clock
330 CALL 06010H
340 REM* Low the ADC's Clock
350 CALL 0600DH
360 REM* High the ADC's Data In/Out
370 CALL 06016H
380 REM* High the ADC's Clock
390 CALL 06010H
400 REM* Low the ADC's Clock
410 CALL 0600DH
420 REM* High the ADC's Data In/Out
430 CALL 06016H
440 REM* High the ADC's Clock
450 CALL 06010H
460 REM* Low the ADC's Clock
470 CALL 0600DH
480 REM* READS IN THE CONVERSION COUNT FROM THE ADC FOR CH0
490 T=0 :  FOR X=11 TO 0 STEP -1
500 REM* High the ADC's Clock
510 CALL 06010H
520 REM* PUTS THE COUNT IN ORDER FROM MSB TO LSB
```

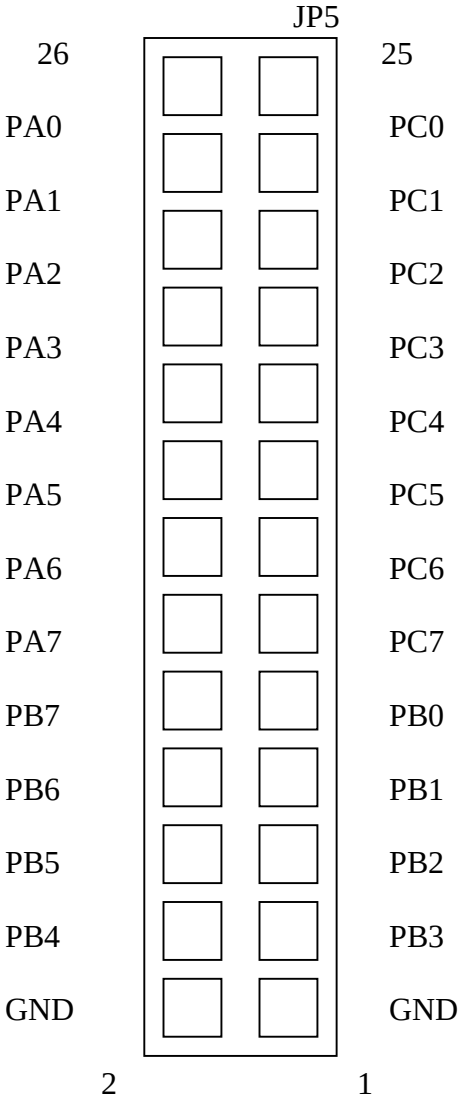
```
530 IF (PORT1.AND.DA)=DA THEN T=T+2**X
540 REM* Low the ADC's Clock
550 CALL 0600DH
560 NEXT X
570 REM* High the ADC's Clock
580 CALL 06010H
590 REM* High the ADC's Chip Select
600 CALL 600AH
610 REM* Low the ADC's Data In/Out
620 CALL 06013H
630 PRINT "The A/D conversion value is ",T
640 PRINT "Based on a Vcc of 5.00 volts that equals ",
650 PRINT USING(#.###), 5*(T/4095) : REM Formatted outputs
660 PRINT USING(0), : REM Cancel formatted output
670 PRINT
680 GOTO 120 :  REM Do another conversion
```

This program shows how assembly language routines can be added to a BASIC program to allow bit access of PORT 1. CALL 6007H and 600AH toggle the chip select (P1.4) of the LTC1298. CALL 600DH and 6010H (P1.5) is used to toggle the clock line (P1.5) of the LTC1298. CALL 6013H and 6016H are used to toggle the data-in/data-out line (P1.6) of the LTC1298.

10.20 8255 Programmable Peripheral Interface

The 8255 is user programmable through the MODE port at address 0DC03H. The I/O is split into three ports. Port A is available through address 0DC00H. Port B is available through address 0DC01H. Port C is available through address 0DC02H. Upon power-up the 8255 is configured as all input bits. The user can change the configuration at any time by writing to the MODE port. NOTE: any change to the MODE port resets all outputs to logic low.

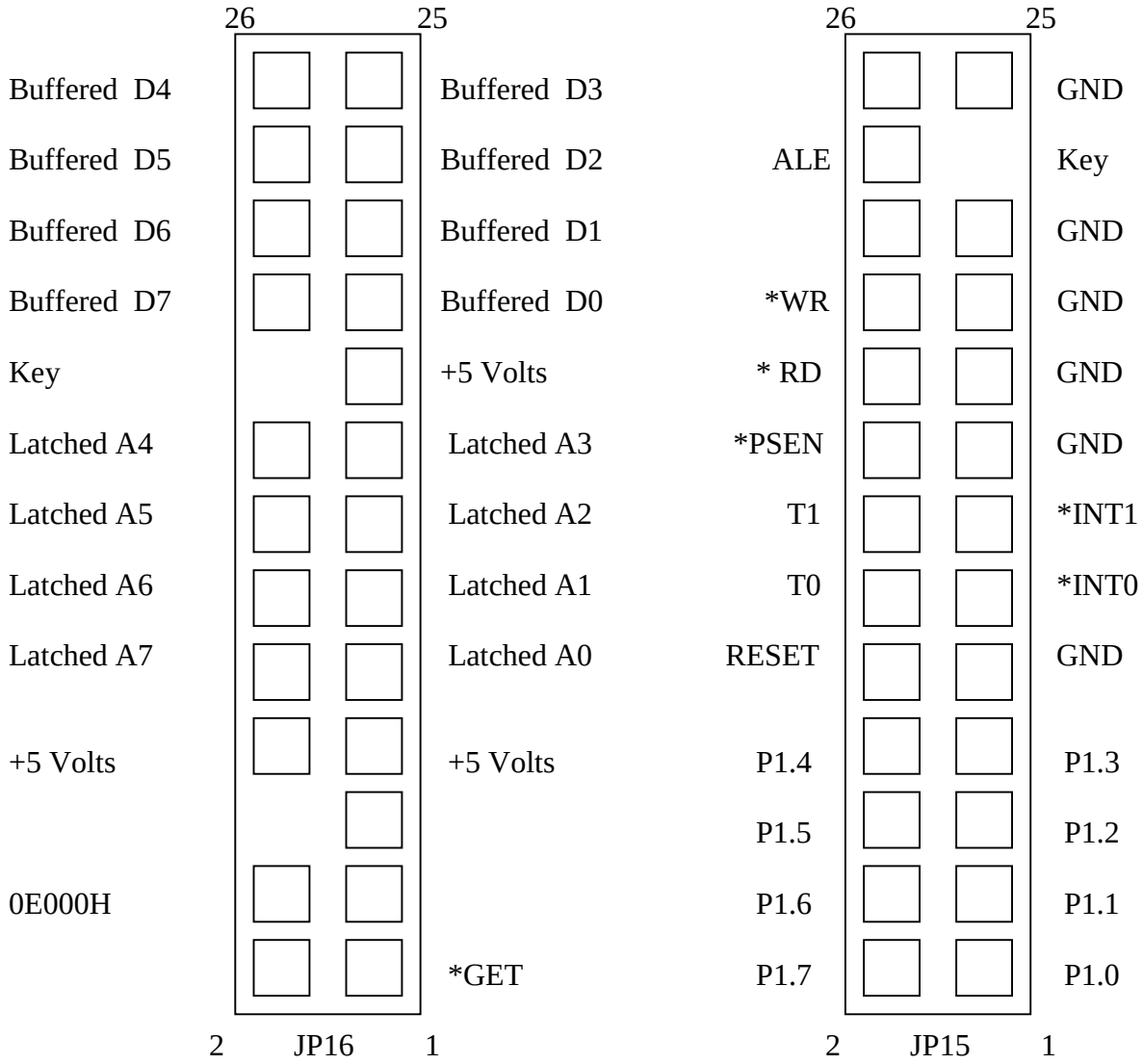
8255 Configuration				
Port A	Port B	Port C upper nibble	Port C lower nibble	MODE value
IN	IN	IN	IN	09BH
IN	IN	IN	OUT	09AH
IN	IN	OUT	IN	093H
IN	IN	OUT	OUT	092H
IN	OUT	IN	IN	099H
IN	OUT	IN	OUT	098H
IN	OUT	OUT	IN	091H
IN	OUT	OUT	OUT	090H
OUT	IN	IN	IN	08BH
OUT	IN	IN	OUT	08AH
OUT	IN	OUT	IN	083H
OUT	IN	OUT	OUT	082H
OUT	OUT	IN	IN	089H
OUT	OUT	IN	OUT	088H
OUT	OUT	OUT	IN	081H
OUT	OUT	OUT	OUT	080H



Digital I/O connections are available through JP5, a 2x13 square pin header.

11.0 Vertical-Stacking Expansion Headers

The small size of the RTC-52 PLUS microcontroller board in not compromised by expanding I/O through the expansion connector. The footprint remains the same as each I/O board only adds ¾ of an inch to the height of the system. I/O expansion is obtained through a vertical header system making a backplane unnecessary. The data bus and latched low-order address bus are passed through the expansion header along with control lines and power. In place of the upper address, the upper 8K block is decoded and passed through the expansion header as a block select.



12.0 Getting Started

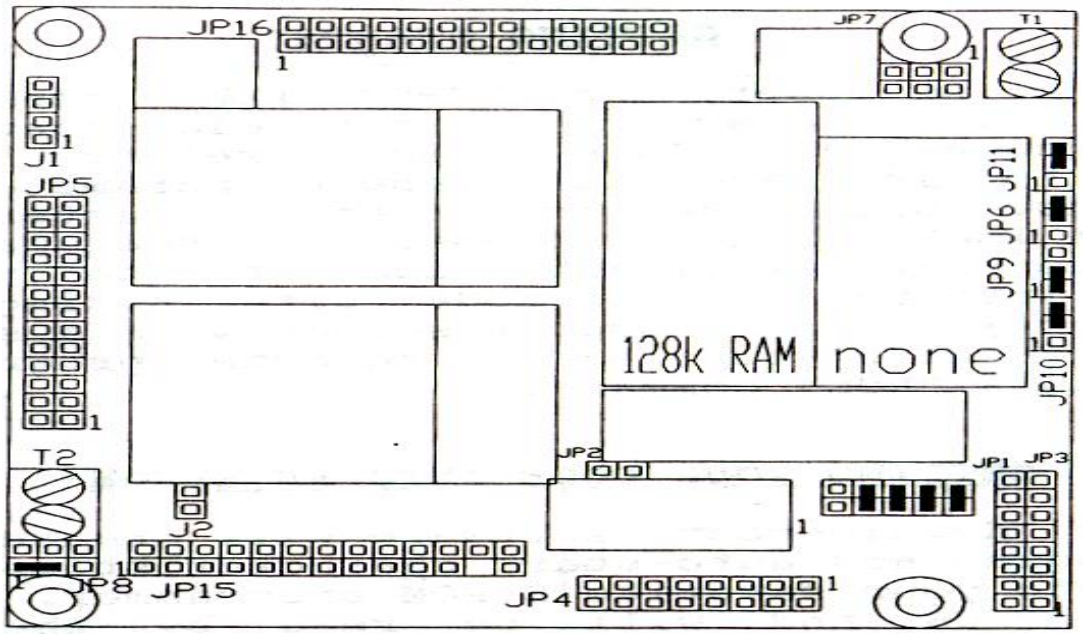
Upon power-up (or reset) the RTC_52 PLUS looks at address 8000H for a special character. This special character might be an ASCII ‘1’ through’6’. This number relates to the PROGx command used to indicate autoexecute parameters. If this location does not hold one of these characters, the BASIC interpreter will look for a key press. Tap the space bar at this time and the interpreter will auto baud rate to your connected speed and go into command mode (after sizing the available RAM in the system). Command mode responds with a sign-on message and a ‘>’ prompt. Here you are free to enter program lines and execute them with a ‘RUN’ command.

Getting Your Program Into an EPROM

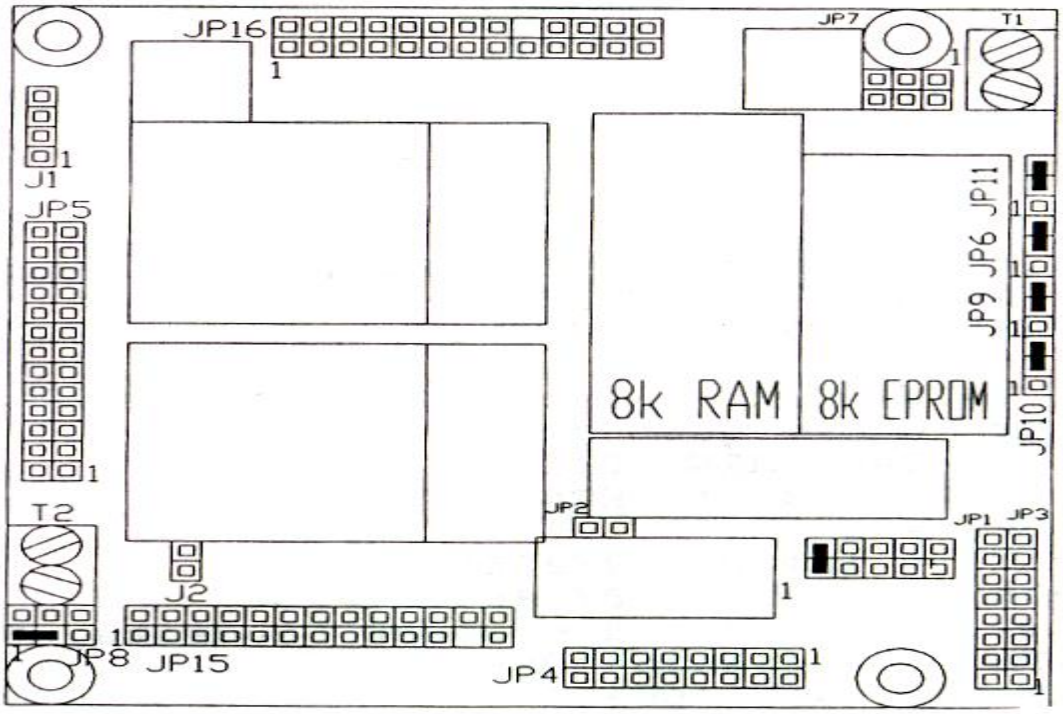
The HOST-52 development software will create an Intel HEX file of your program and auto-start characteristics. This file can be loaded into your favorite EPROM programmer. The programmed EPROM goes into U8 and will be read by the 80C52 BASIC interpreter when power is applied.

If you wish to use a non-volatile SRAM module (DS1645/AB RAM or DS1646 RAM/RTC), append the following program to the end of your and then type in ‘GOTO 60000’. Answer the questions and your program will be saved into the appropriate location. You will probably want to use the PROG2 or PROG4 command.

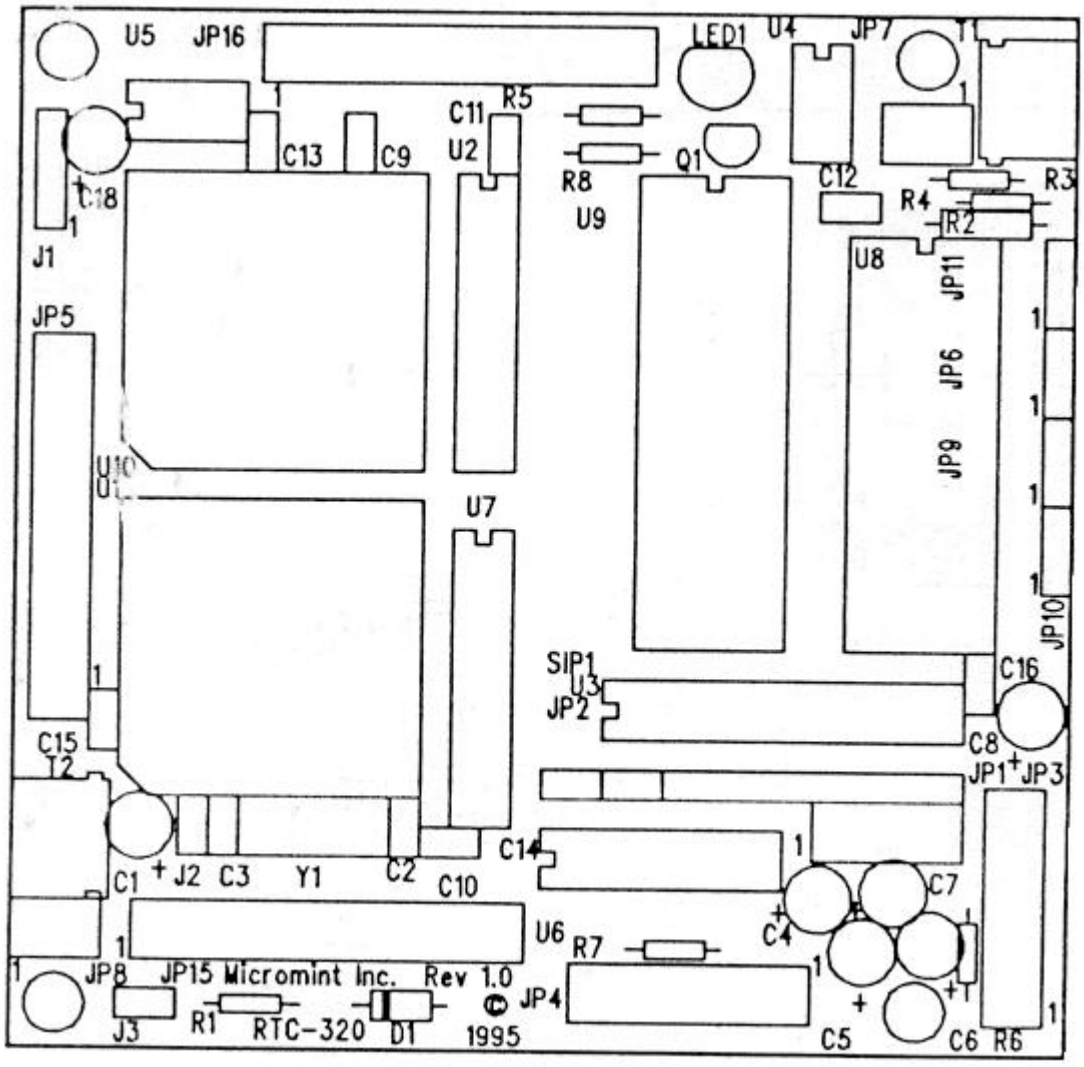
```
60000 PRINT "Hit 'P' to simulate a PROGx command"
60010 PRINT " or 'Q' to quit now."
60020 G=GET
60030 G=GET : IF G=0 THEN 60030
60040 IF (G<>50H.AND.G<>70H) THEN GOTO 60220
60050 XBY(8010H) = 55H : PRINT "Moving BASIC Program"
60060 FOR X = 200H TO (200H+LEN)
60070 XBY(X+7E11H) = XBY(X) : PRINT ".",
60080 NEXT X
60090 PRINT : PRINT "Hit '1-6' to do a PROGx"
60100 PRINT " OR 'Q' to quit."
60110 G=GET : IF G=0 THEN 60110
60120 IF (G<31H.OR.G>36H) THEN GOTO 60220
60130 XBY(8000H)= G
60140 XBY(8001H) = INT(RCAP2/256)
60150 XBY(8002H) = RCAP2-(XBY(8001H)*256)
60160 PRINT "Saving ",G," and the present baud rate, ",
60170 PRINT XTAL/(32*(65536-RCAP2))
60180 IF G<32H THEN GOTO 60220
60190 XBY(8003H) = INT(MTOP/256)
60200 XBY(8004H) = MTOP-(XBY(8003H)*256)
60210 PH0." plus present MTOP, ",MTOP
60220 PRINT "DONE"
```



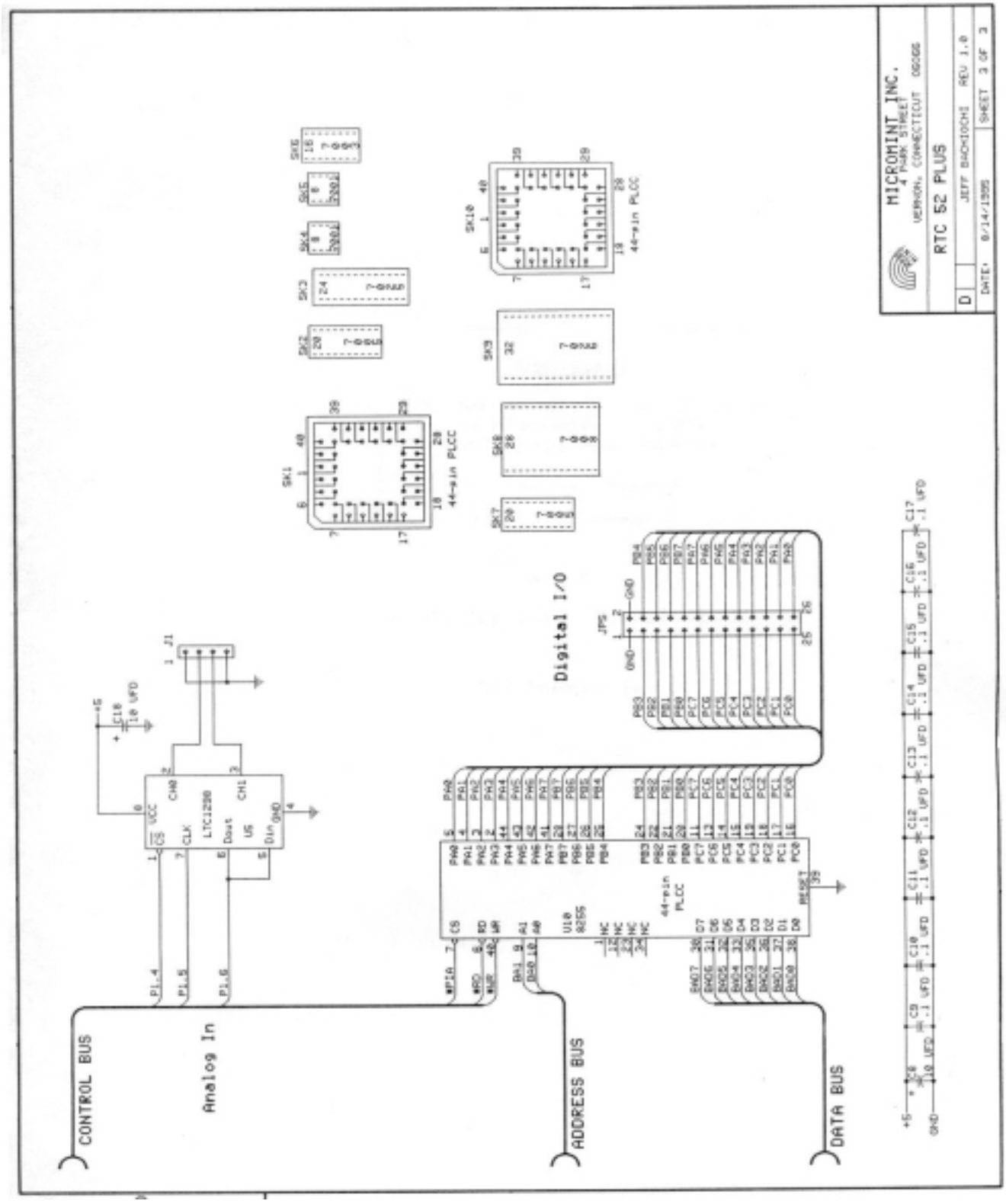
This shows the suggested jumper configurations for an RTC-52 PLUS with 128K battery backed RAM using the DALLAS DS1645/AB (RAM only) or the DALLAS DS1646 (RAM/RTC). This configuration does not require an EPROM.



This shows the suggested jumper configuration for an RTC-52 PLUS with an 8K RAM (6264) and an 8K EPROM (27C64).



Silkscreen for the RTC-52 PLUS



RTC-52 PLUS Schematic (3 of 3)