RTC-320

RTC series Microcontroller
using the DALLAS 80C320

Technical Manual

Release 1.0
8/3/95

Micromint, Inc.
902 Waterway Place
Longwood, FL 32750
www.micromint.com

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 both parts and labor. Any unit found to have a defect in materials or workmanship within this period will, at the discretion of Micromint Inc., be repaired or replaced.

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

3. A minimum inspection fee must be prepaid for the repair of units no longer under warranty. Call Micromint Inc. for information on current minimum charges.

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 may be obtained by phone or letter. Please retain a record of this number, because most subsequent correspondence will refer to this authorization. Under no circumstances should any product be returned to Micromint Inc. without this authorization. Micromint Inc. assumes no responsibility for returns unaccompanied by an authorization number. All returns must be shipped prepaid. Insurance is recommended because losses by a shipping carrier are not the responsibility of Micromint. Repaired units will be returned postage and insurance paid.

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

REV. 5/93

COPYRIGHT

Micromint, Inc.
902 Waterway Place
Longwood, FL 32750
www.micromint.com

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 and 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 recommended without processing by Micromint for such application.

Occasionally in this manual we refer to other manufacturers' products. Such references do 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.

Table of Contents

MICROCONTROLLER EVOLUTION

The 8031 microcontroller is probably the most widely used core for embedded processing in the industry. It's price/performance ratio as a general purpose controller helps to keep it a popular choice for even new designs. It is not surprising that manufacturers have based many of today's newer processors on the 8031 core. While adding bells and whistles may make the newer parts more desirable, many designers are looking for faster processing. Dallas Semiconductor has made the first real improvement to the 8031 core by reducing the number of clock cycles necessary per instruction cycle to four (vs. twelve).

This factor of 3 increase in speed using the same crystal allows the user's original object code to execute up to 150% faster. (Some instructions execute at less than 3 times as fast but none fewer than 1.5 times.) The maximum crystal frequency has been elevated as well. The 33MHz upper limit is triple again over 11MHz standard. Fast programs will now literally scream.

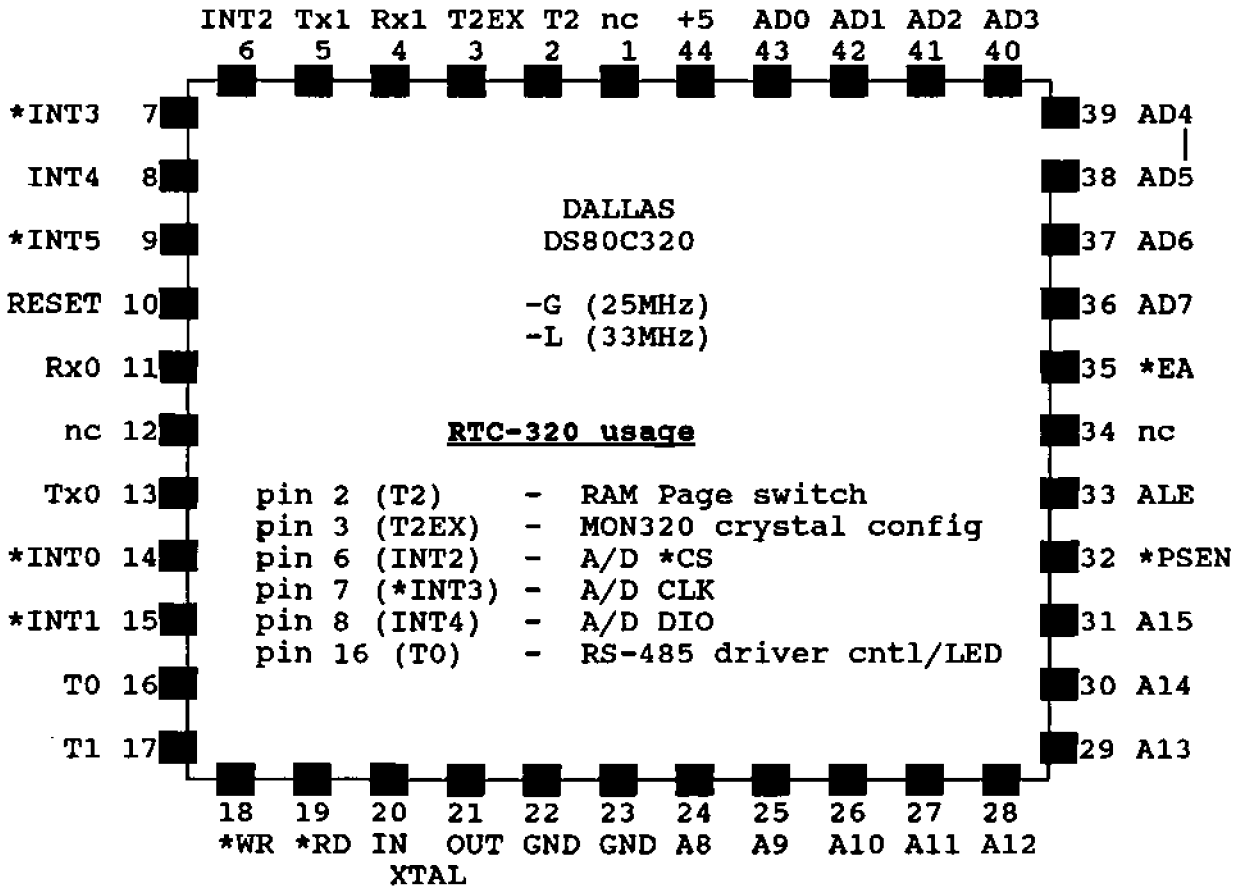
What does this do to the I/O devices connected? As far as external CODE space is concerned, installed devices (EPROMs) must comply with the timing specifications based on the crystal speed and glue logic delays. DATA space devices (SRAMs and peripheral chips) need NOT be upgraded if the user selects the appropriate number of 'Stretch Cycles' (WAIT states) which are added to the access times of the MOVX instructions. The number of stretch cycles are dynamically alterable to allowing the user complete control.

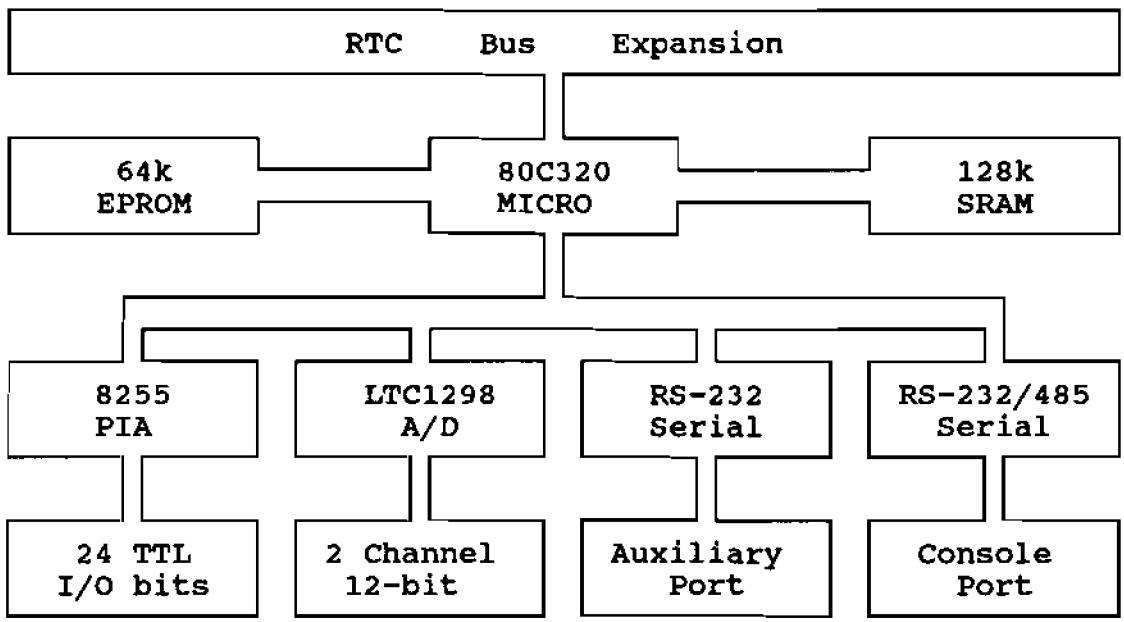
Crystal speed	CODE space	DATA space (default)
22.1184MHz	100nS	150nS
33.1776MHz	55nS	100nS

The RTC-320 is a plain-brown-wrapper 80C320 controller that is optimized with the most requested I/O on-board for minimal configuration applications. While the RTC-320 may be perfect for most 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 prototying expansion board.

The RTC system measures only 3.5 inches square and uses vertical stacking connectors for I/O expansion. The RTC-320 processor board contains the 80C320 processor, EPROM and RAM memory, address decoding and buffering, 24-bits of parallel I/O , a 2-channel 12-bit A/D convertor, and two RS-232 serial ports (one alternately available as RS-485. Each vertically stacked expansion board only increases the system height by 3/4 inch.

80C320 PIN DESCRIPTION





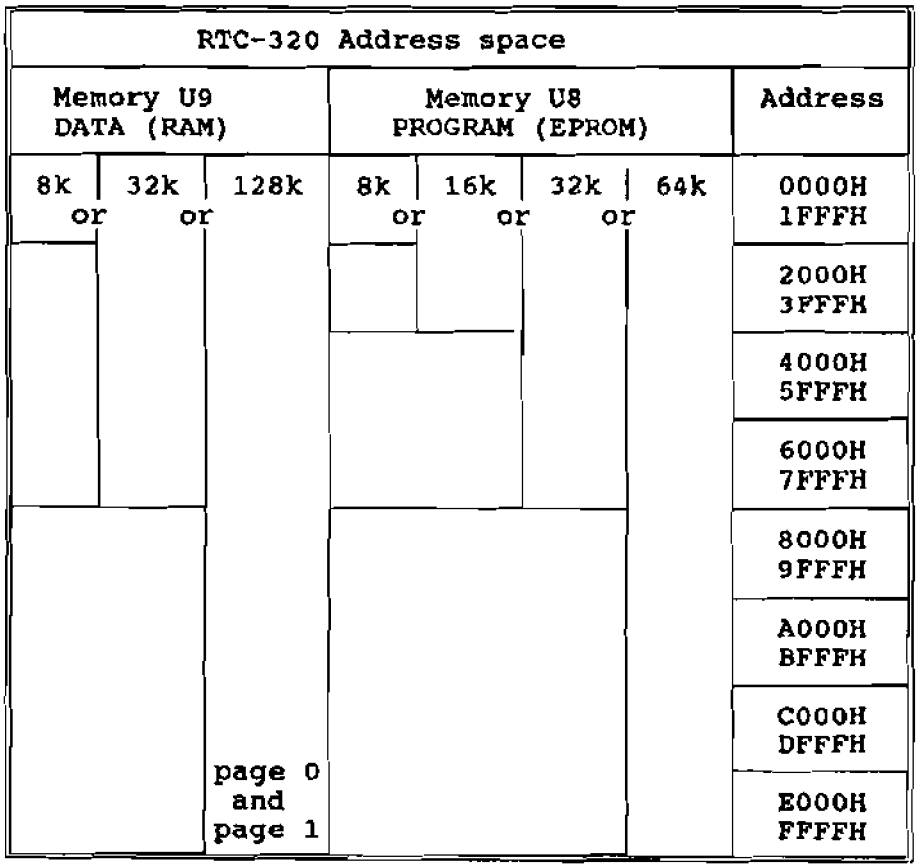
RTC-320 Block Diagram

RTC320 EXTERNAL ADDRESSING SPACE

The RTC-320 microcontroller can directly address 64k of external memory. That is, 64K of DATA memory and 64K 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-320 board. **The first 16k of 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 or combined DATA and CODE space.**

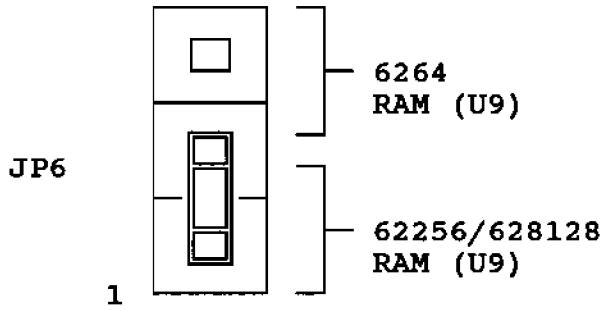
The following chart shows conventional addressing for the memories used on the RTC-320:



RTC-320 Memory Device Configuration

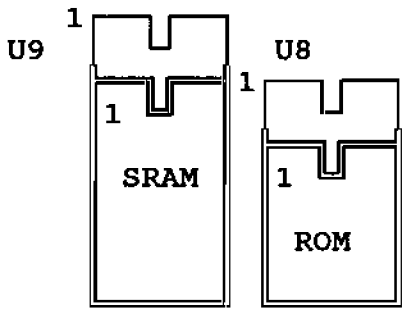
SRAM SIZE

The RTC-320 provides IC sockets for two memory devices. Socket U9 has been designated as the SRAM position and U8 as the EPROM position. SRAMs from 8k to 128k by 8 may be used in the RTC-320, 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.



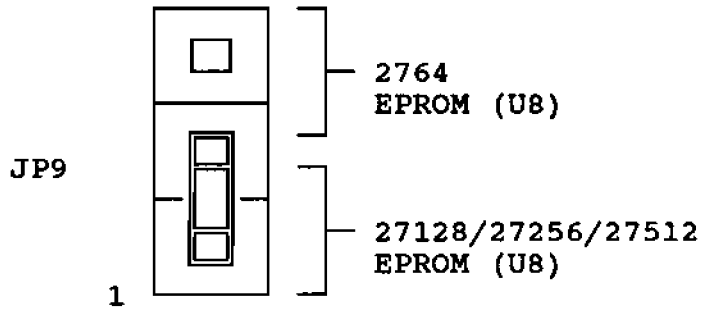
SRAM size configuration jumper JP6 is shown configured for 628128 (128kx8) SRAM.

NOTE: smaller memory devices should be lower justified in IC sockets.

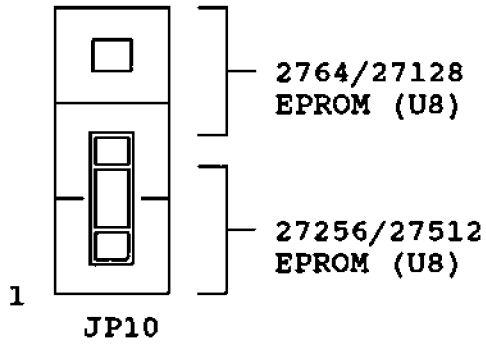


EPROM SIZE

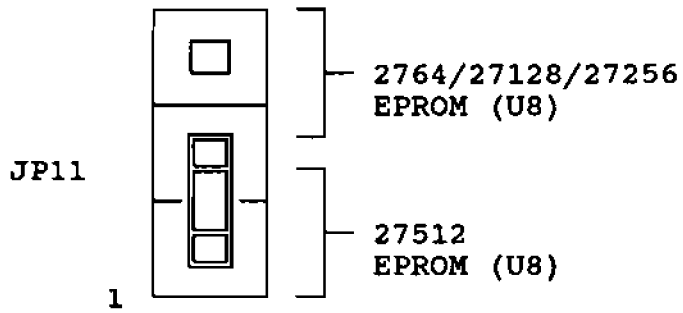
Four different size EPROMs may be used in the RTC-320 ROM socket U8. As with the RAM, EPROMs smaller than 64k will continually wrap around and be duplicated in multiple address blocks. Three jumpers need to be placed to configure the EPROM socket U8 for your EPROM chip; JP9, JP10, and JP11.



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



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



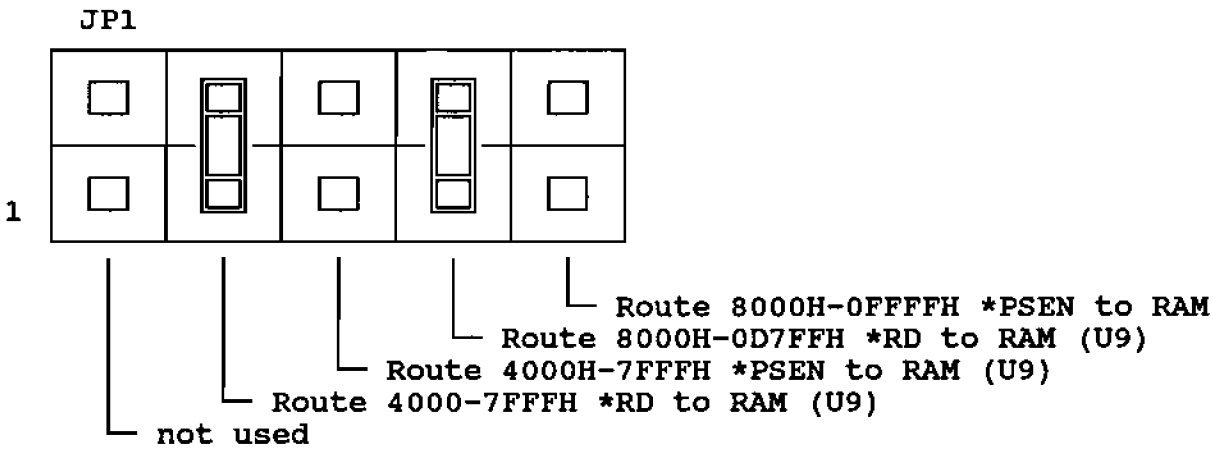
The EPROM configuration jumper JP11 is shown configured for a 27512.

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 0000H-3FFFH) goes to the U8 EPROM socket and any access to DATA space (*RD *WR 0000H-3FFFH) goes to the SRAM socket.

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).

The third area 8000H-FFFFH (third and fourth quadrants) 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 again, this may not make logical sense to you).

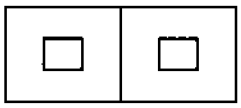


JP1 is shown configuring all *RD signals to RAM and all *PSEN signals to EPROM. This separates the DATA and CODE spaces providing non-overlapping spaces (64k [potential] DATA space and 64k [potential] CODE space).

EXTERNAL CODE SELECTION

The 80C320 microcontroller requires *EA (pin 31 on the microcontroller) to be pulled down to a logic low level. This instructs the processor to start executing machine language code starting at address 0000H (CODE space).

JP2

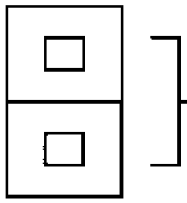


Enable External EPROM CODE at 0000H

JP2 shows the microcontroller enabled for external code execution

RESETTING THE RTC-320

Power-ON reset of the RTC-320 board 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.



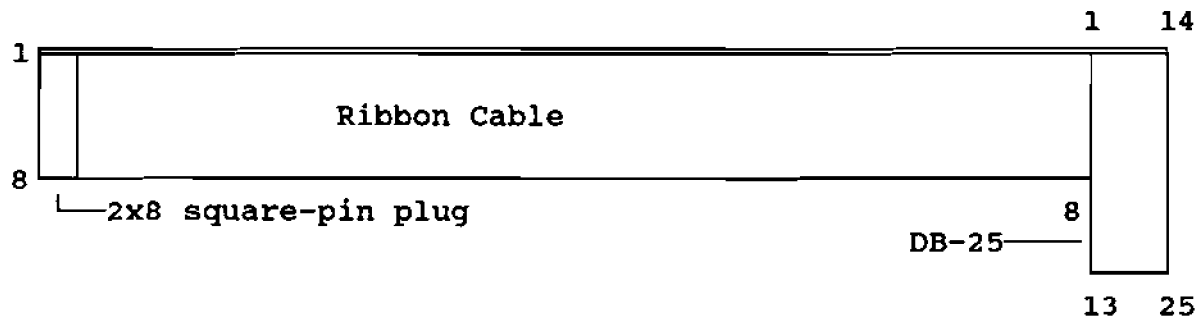
Momentarily 'short' pins to RESET board

J2

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

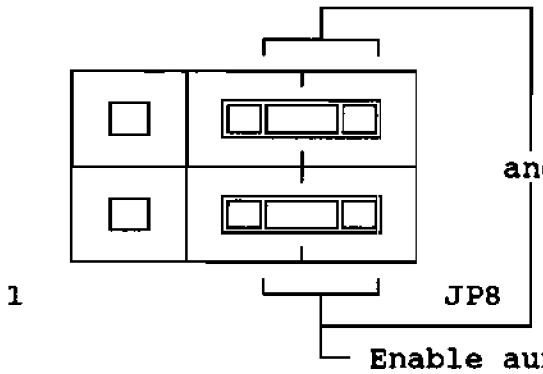
RS-232 COMMUNICATIONS

The 80C320 contains two full-duplex serial channels. TTL-level serial signals are converted to ±10-volt RS-232-compatible signals by U6, the MAX232 device. A 16-lead flat ribbon cable made with a DB-25 at one end (for connection to DTE terminal device and a 16-pin dual row plug header (connection to the RTC-320) will permit serial communication. The console port is available through JP4 and the auxiliary port is available through J3.



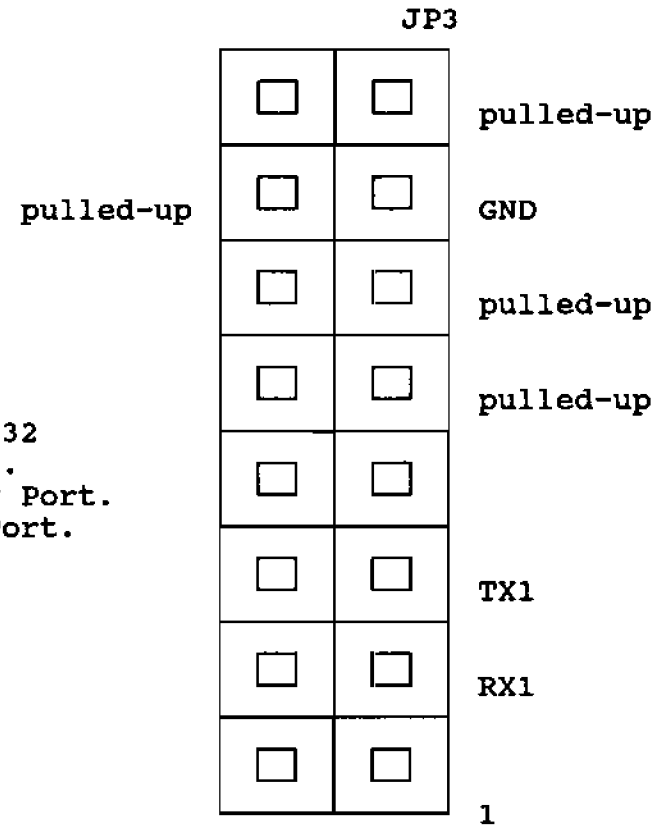
Cable required for RS-232 communications

To eliminate unwanted noise on the RX input to the processor, remove the unused 75176 line driver chip. (U4)

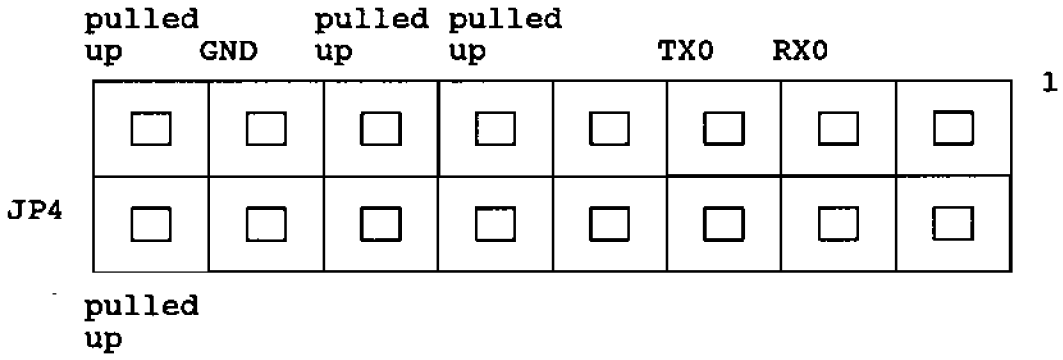


JP8 must have both jumpers installed as shown to enable the auxiliary serial port.

Enable auxiliary RS-232 for 80C320



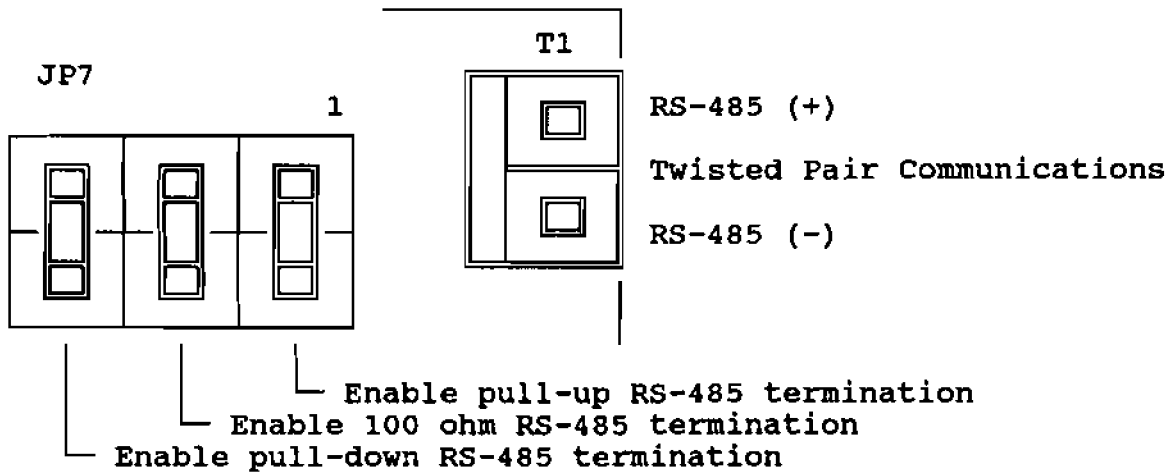
JP3 and JP4 are RS-232 interface connectors.
JP3 is the Auxiliary Port.
JP4 is the Console Port.



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 to prevent message collision. 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.)

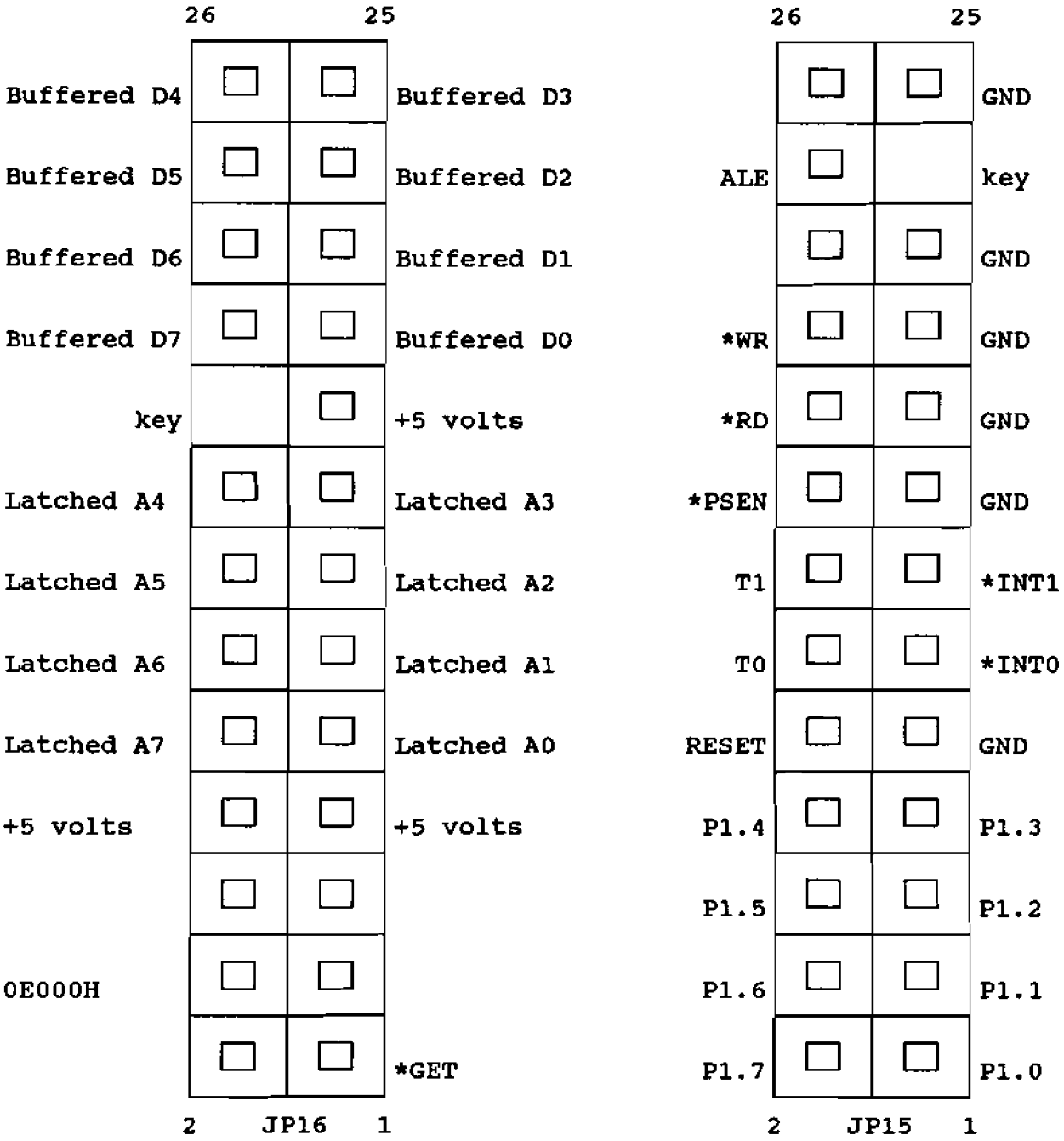
A set of screw terminal blocks are provided for RS-485 twisted pair communication.



JP7 shows termination of the RS-485 lines enabled

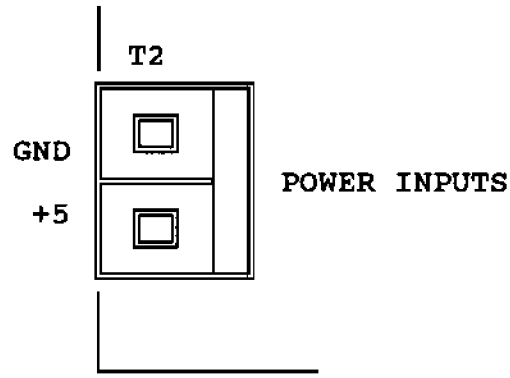
VERTICAL-STACKING EXPANSION HEADER

The small size of the RTC-320 mirocontroller board is not compromised by expanding I/O through the expansion connector. The footprint remains the same as each I/O board only adds 3/4 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 passed are through the expansion header along with control lines and power. In place of the upper address bus, the upper 8K block is decoded and passed through the expansion header as a block select.



RTC-320 POWER REQUIREMENTS

A set of screw terminal blocks are provided for +5 volt power and ground. 200mA are required for 22 MHz operation. 250mA are required for 33Mhz operation.

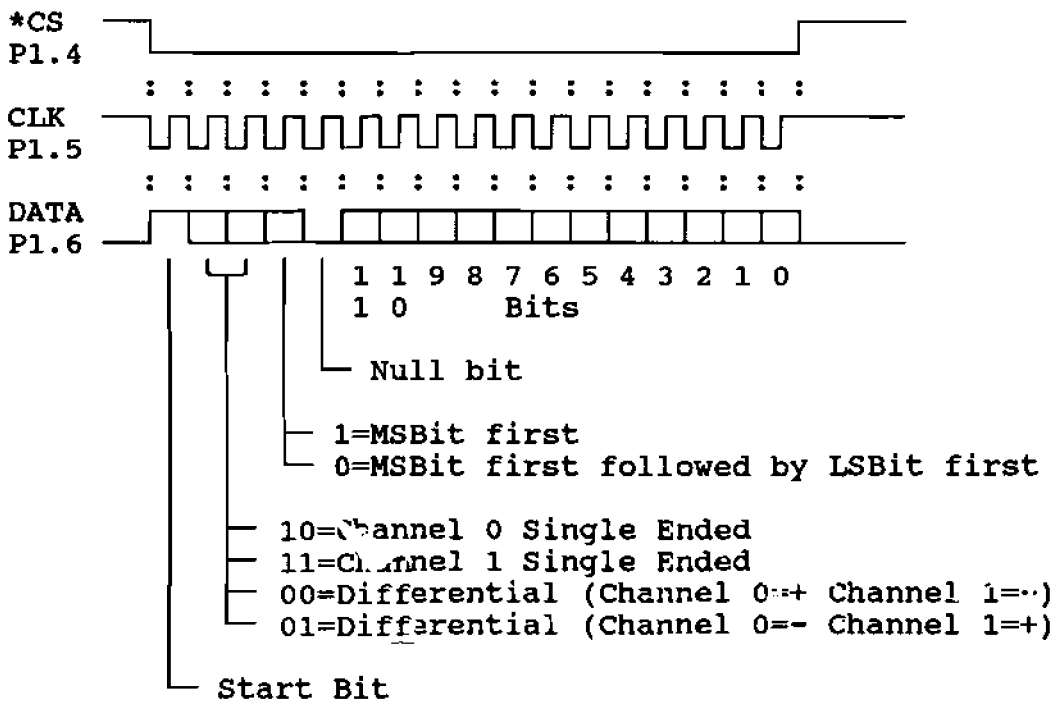


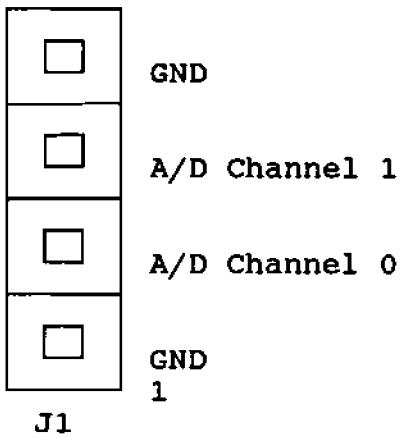
STAND-ALONE I/O CONNECTIONS

In an attempt to allow the RTC-320 to be more widely used in a standalone mode, the most popular I/O was added. Additional I/O includes 24-bits of digital I/O plus a two channel 12-bit A/D converter.

A/D Convertor

A three wire serial A/D (LTC1298) is used to provide the RTC-320 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.





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

This skeletal program suggests one possible routine for reading the A/D convertor. The user needs to set the MODE and provide two registers for the 12-bit result.

```
CLK         EQU  P1.5
DIO         EQU  P1.6
CS          EQU  P1.4
MODE        EQU  0BH          ; Single Ended Channel 0
;               0FH           ; Single Ended Channel 1
;               09H           ; Differential +=Chan0, -=Chan1
;               0DH           ; Differential +=Chan1, -=Chan0
A_D_MSB     EQU  ???          ; Register to hold upper 4 bits
A_D_LSB     EQU  ???          ; Register to hold lower 8 bits
```

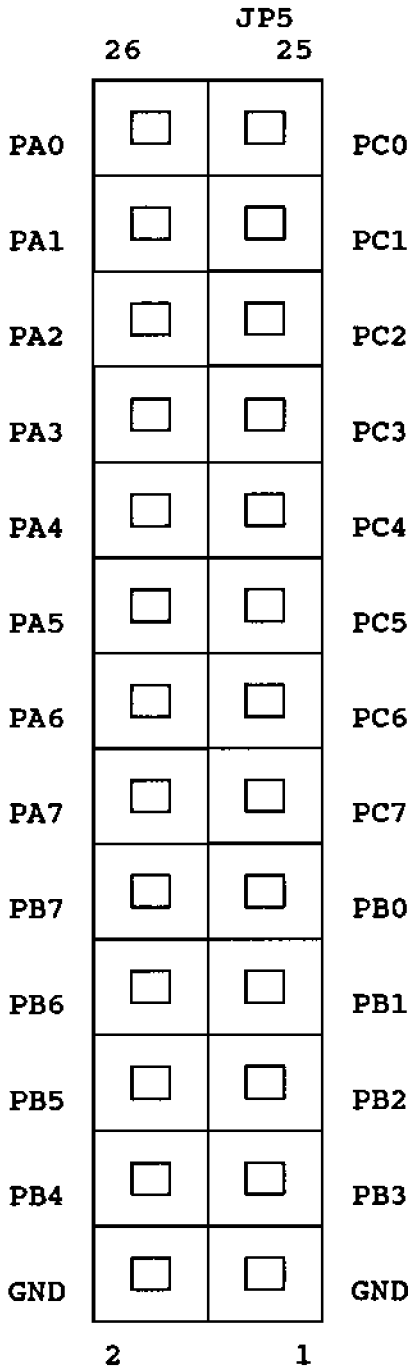
The actual routine start here:

```
LTC:        SETB CS
            SETB CLK
            MOV  B,#04H
            MOV  A,MODE
            CLR  CS

D_OUT:      CLR  CLK
            RRC  A
            MOV  DIO,C        ; 1 or more as necessary dependent
;           NOP               ; on execution speed, min 2.5µs
            SETB CLK          ; between clock edges
;           NOP
            DJNZ B,D_OUT

            CLR  A
            MOV  B,#05H
D_IN_M:     CLR  CLK
;           NOP
            MOV  C,DIO
            RLC
            SETB CLK
;           NOP
            DJNZ B,D_IN_M

            MOV  A_D_MSB,A
            MOV  B,#08H
D_IN_L:     CLR  CLK
;           NOP
            MOV  C,DIO
            RLC
            SETB CLK
;           NOP
            DJNZ B,D_IN_L

            MOV  A_D_LSB,A
            SETB CS
```

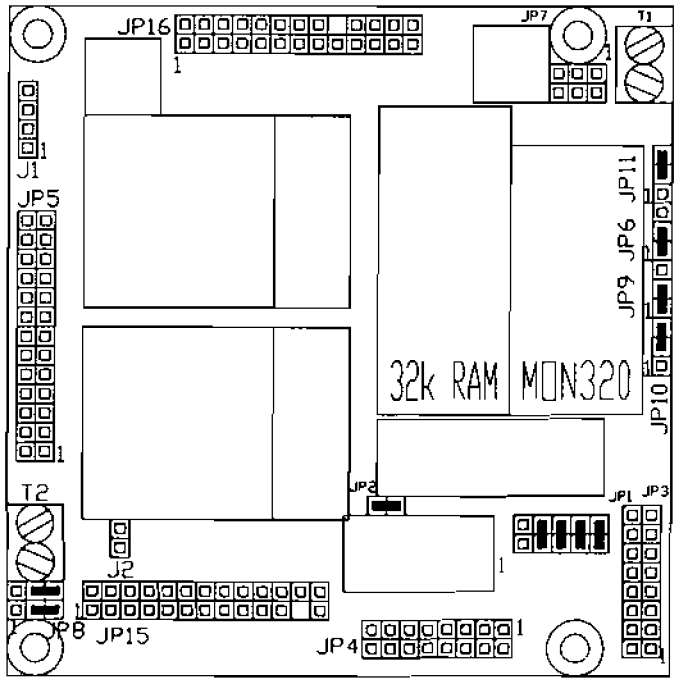
8255 PIA

The 8255 is user programable 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 PIA 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.

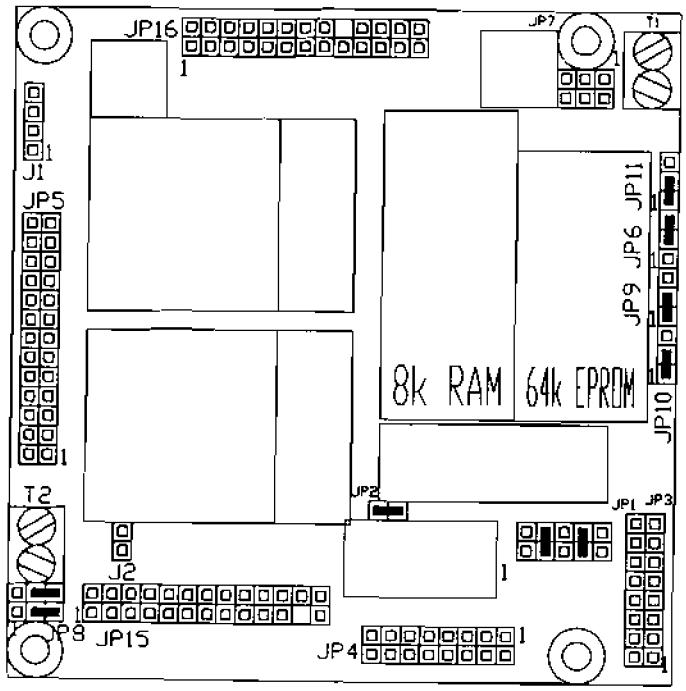
8255 Configuration				
Port A	Port B	Port C upper nybble	Port C lower nybble	MODE Value
IN	IN	IN	IN	09BH
IN	IN	IN	OUT	09AH
IN	IN	OUT	IN	093H
IN	IN	OUT	OUT	092H
IN	OUT	IN	IN	099H
IN	OUT	IN	OUT	098H
IN	OUT	OUT	IN	091H
IN	OUT	OUT	OUT	090H
OUT	IN	IN	IN	08BH
OUT	IN	IN	OUT	08AH
OUT	IN	OUT	IN	083H
OUT	IN	OUT	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 connection are available on a 2x13 sqaure pin header.



Suggested Jumper Configuration for 32k RAM (62256) and MON320 (27128).



Suggested Jumper Configuration for 8k RAM (6264) and USER EPROM (27512).

To eliminate unwanted noise on the RX input to the processor, remove the 75176 line driver chip (U4) if not using RS-485.

GETTING STARTED

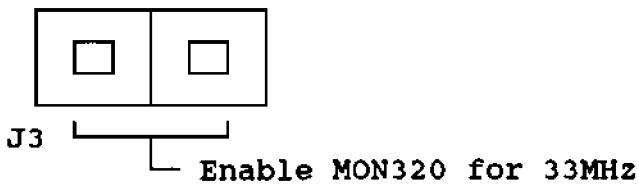
RTC-320

The RTC-320 does not have any internal program, so the user must program an EPROM with machine language code which will start execution at 0000H. The following code can be programmed into a blank EPROM and will turn the LED on and off, verifying the ability to execute code. Refer to DALLAS's High-Speed Microcontroller Data Book for the particulars on the internal registers of the 80C320.

```
0000H  15H  F0H          ;DEC  B
0002H  D5H  F0H  F5H     ;DJNZ B,0000H
0005H  14H               ;DEC  A
0006H  70H  F8H          ;JNZ  0000H
0008H  B2H  B4H          ;CPL  T0
000AH  01H  00H          ;AJMP 0000H
```

MON320

When using the MON-320 (monitor EPROM), the code looks at P1.1 on the processor to determine what crystal is connected. The user must configure J3 correctly or the sign-on BAUD rate will be incorrectly configured.



J3 is shown configured for 22MHz (or 11MHz). NOTE: You can use this jumper in your own application once MON320 has been remove.
It can be examined through P1.1 (high=no jumper low=jumper).

GETTING YOUR CODE INTO AN EPROM

Whether your writing in a high level language (like 'C') or pushing bits in assembly language, you must have some software tools (Compiler and/or Assembler) to make it come together. Check to see that your third party software includes support for the additional functions included in the 80C320. Your established 8051 code will execute as is (once you've adjusted for speed increases).

Your development software will need to know where CODE and DATA space starts. This will normally be 0000H CODE and 0000 DATA. However, you may wish to use 4000H (or other address) as the CODE start address if you wish to use MON320, the monitor ROM for the RTC-320. The monitor ROM lets you load in an Intel .HEX file and jump to your code for debugging purposes prior to actually placing your code into EPROM. Once the code has been debugged you can recompile/assemble for 0000H code space and download your new .HEX file into your favorite EPROM programmer. This EPROM goes into U8 (replacing MON320) and will autoexecute when power is applied.

Appendix A -- MON320 Command Summary

All numeric values are in hexadecimal. Parameters shown in brackets are optional; the default action depends on the command.

Command	Description
AI	Read ADC on RTC-320 board
BD [num]	Disable breakpoint num (or all BPs)
BE [num]	Enable breakpoint num (or all BPs)
BL	List breakpoints
BP [addr]	Set breakpoint at addr
D [addr [len]]	Dump XRAM at addr for len bytes
DI [addr [len]]	Dump IRAM at addr for len bytes
DX [addr len]	Dump XRAM at addr for len bytes
E addr byte	Enter byte into XRAM
EI addr byte	Enter byte into IRAM
F addr len [byte]	Fill XRAM at addr for len with byte
FI addr len [byte]	Fill IRAM at addr for len with byte
FX addr len [byte]	Fill XRAM at addr for len with byte
G [addr]	Start 80C320 program at addr, active BPs
HR [addr]	Receive hex file to data space
HS addr len	Send external data as hex file
I [addr]	Display XRAM byte at addr+I/O base addr
M addr1 len addr2	Move XRAM bytes from addr1 to addr2
O [addr byte]	Set XRAM byte at addr+I/O base addr
OP [name]	Set MON320 options
R [name [val]]	Display all regs or change register
RB	Display current working register bank
RC	Display CPU registers
RS	Display special function registers
S [addr]	Single-step instruction at address
XH	Select upper 64 KB XRAM bank for dump, fill, move
XL	Select lower 64 KB XRAM bank for dump, fill, move

Appendix B -- MON320 register and bit names

These names are used in the R (Register) command to display and change the contents of various CPU registers.

Symbol	Addr	Name
16-bit values		
DPTR0	83, 82	DPTR0 register
DPTR1	84,85	DPTR1
PC	-	Program Counter (current instruction)
8-bit values		
A	E0	Accumulator
ACC	E0	synonym for A
B	F0	B Register
CKCON	8E	Clock Configuration
DPH	83	High byte of DPTR
DPH1	85	High byte of DPTR1
DPL	82	Low byte of DPTR
DPL1	84	Low byte of DPTR1
DPS	86	DPTR Select Register
EIE	E8	Extended Interrupt Enable
EIP	F8	Extended Interrupt Priority
EXIF	91	Extended Interrupt Flags
F	D0	synonym for PSW
FLAGS	D0	synonym for PSW
IE	A8	Interrupt Enable
IP	B8	Interrupt Priority
P1	90	Port 1
PCON	87	Power Control
PSW	D0	Program Status Word (Flags)
R0 to R7	00-1F	Current bank of working registers
RCAP2H	CB	Timer 2 Reload/Capture Reg High
RCAP2L	CA	Timer 2 Reload/Capture Reg Low
SBUF	99	Serial Buffer
SBUF1	C1	Serial Buffer 1
SCON0	98	Serial Control Reg 0
SP	81	Stack Pointer
TA	C7	Timed Access (not used by MON320)
T2CON	C8	Timer 2 Control
TCON	88	Timer Control
TH0	8C	Timer 0 high byte
TH1	8D	Timer 1 high byte
TH2	CD	Timer 2 high byte
TL0	8A	Timer 0 low byte
TL1	8B	Timer 1 low byte
TL2	CC	Timer 2 low byte
TMOD	89	Timer Mode
WDCON	D8	Watchdog Configuration

1- and 2-bit values

These names are used for display only, as the current version of MON320 does not allow you to change these bits directly. Any undefined bits are displayed as "?" and should not be changed.

AC	D6	PSW.6	Auxiliary carry flag
CY	D7	PSW.7	Carry flag
ES1	AE	IE.7	Enable Serial Port 1
ET0	A9	IE.1	Enable Timer 0 interrupt
EWDI	EC	EIE.4	Enable Watchdog interrupt
EWT	D9	WDCON.1	Enable Watchdog timeout
EX0	A8	IE.0	Enable INT0 pin interrupt
EX1	AA	IE.2	Enable INT1 pin interrupt
EX2	E8	EIE.0	Enable INT2 pin interrupt
EX3	E9	EIE.1	Enable INT3 pin interrupt
EX4	EA	EIE.2	Enable INT4 pin interrupt
EX5	EB	EIE.3	Enable INT5 pin interrupt
F0	D5	PSW.5	User flag 0
F1	D1	PSW.1	User flag 1
IE0	89	TCON.1	Interrupt 0 Edge control
IE1	8B	TCON.3	Interrupt 1 Edge control
INT0	B2	P3.2	External Interrupt 0 pin
INT1	B3	P3.3	External Interrupt 1 pin
IT0	88	TCON.0	Interrupt 0 Type control
IT1	8A	TCON.2	Interrupt 1 Type control
OV	D2	PSW.2	Overflow flag
PS1	BE	IP.6	Serial Port 1 priority
PT0	B9	IP.1	Timer 0 Interrupt priority
PWDI	FC	EIP.4	Watchdog Interrupt priority
PX0	B8	IP.0	External Interrupt 0 priority
PX1	BA	IP.2	External Interrupt 1 priority
PX2	F8	EIP.0	External Interrupt 2 priority
PX3	F9	EIP.1	External Interrupt 3 priority
PX4	FA	EIP.2	External Interrupt 4 priority
PX5	FB	EIP.3	External Interrupt 5 priority
RB8	9A	SCON.2	Receiver Bit 8
REN	9C	SCON.4	Receiver Enable
RI	98	SCON.0	Receiver Interrupt Flag
RS0	D3	PSW.3	Register bank select low bit
RS1	D4	PSW.4	Register bank select high bit
RWT	D8	WDCON.0	Reset watchdog timer
SM0..2	9F..9D	SCON7..5	Serial Mode bits 0..2
TB8	9B	SCON.3	Transmitter Bit 8
TF0	8D	TCON.5	Timer 0 Overflow flag
TI	99	SCON.1	Transmitter interrupt Overflow flag
TR0	8C	TCON.4	Timer 0 Run control
T0	B4	P3.4	Timer/Counter 0 external pin
T1	B5	P3.5	Timer/Counter 1 external pin
WDIF	DB	WDCON.3	Watchdog interrupt flag
WTRF	DA	WDCON.2	Watchdog Timer Reset flag

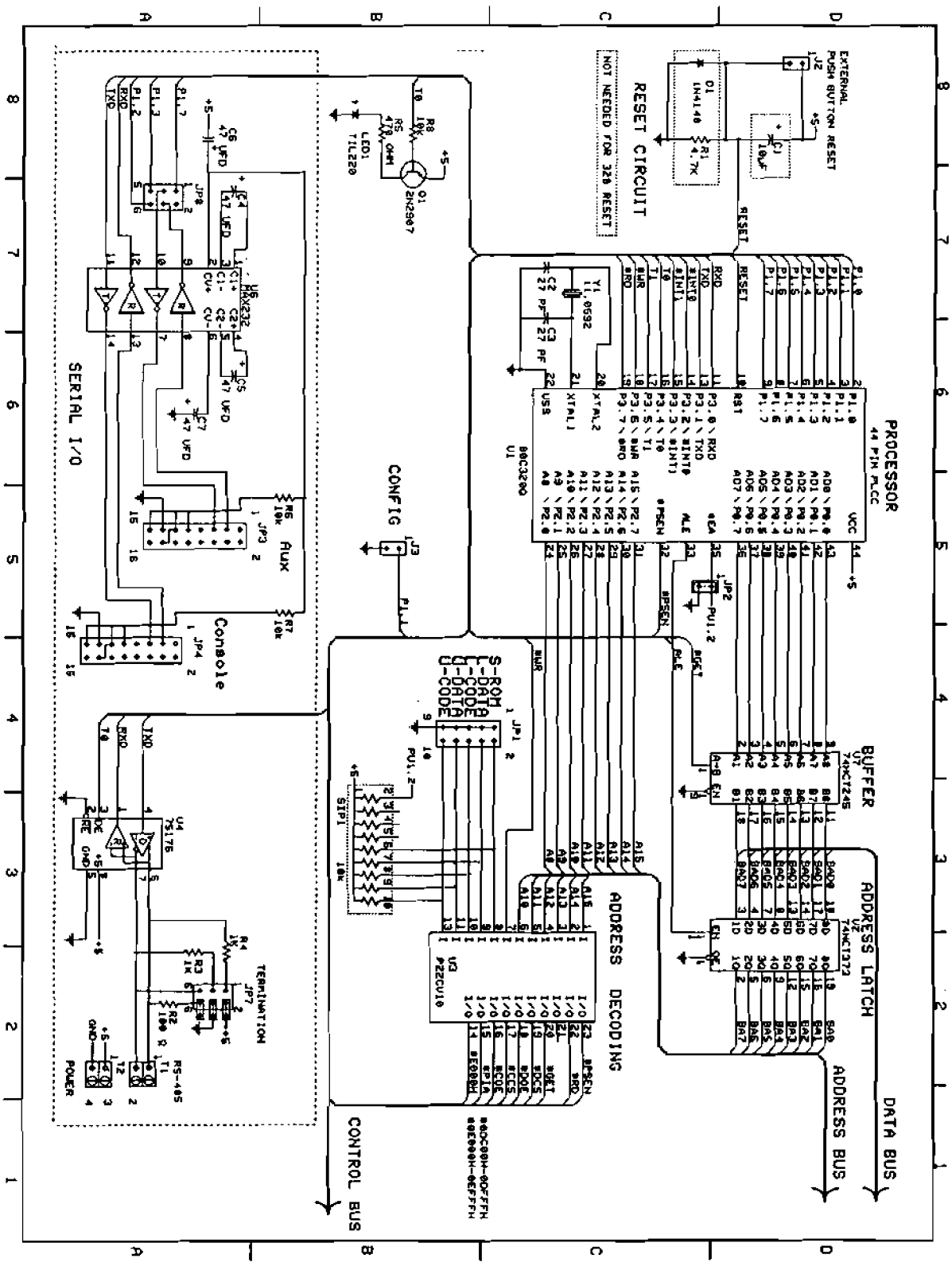
Appendix C -- BLINKY.PRN listing

```
                                1 ;-------------------------------
                                2 ; RTCMON demonstration program
                                3 ;
                                4 ; This is a simple program that
                                   blinks the RTC320 LED
                                5 ; Download this using MON320's
                                   HR command
                                6 ; -- notice the ORG statement...
                                   this avoids the debugger in
                                   EPROM
                                7
      =00B4#                    8 BLINKBIT EQU P3.4 ; visible
                                   output
                                9
      =DC00                     10   PIA_BASE EQU $DC00 ; parallel
                                   port base address
                                11
                                12   ;%INCLUDE "80320.inc"
                                13
                                14 ;-----------------------------
                                15 ; Required program origin for
                                   use with MON320
                                16
      =4000                     17 ORG $4000
                                18
                                19
                                20 ;-----------------------------
                                21 ; Set up some default values
                                22 ; You can change these using
                                   MON320 to see what
                                   happens...
                                23
4000 78 32                      24 MOV R0,#50 ; ON time
4002 79 64                      25 MOV R1,#100 ; OFF time
4004 7A FF                      26 MOV R2,#0FFh ; ripple value
                                27
                                28 ;-----------------------------
                                29 ; Set up PIA control reg and
                                   load the DPTRs
                                30
4006 75 86 00                   31 MOV DPS,#0 ; aim at DPTR0
                                32
4009 90 DC03                    33 MOV DPTR,#PIA_BASE+3
400C 74 8B                      34 MOV A,#$8B
400E F0                         35 MOVX @DPTR,A
                                36
400F 90 DC00                    37 MOV DPTR,#PIA_BASE ; set Port A
```

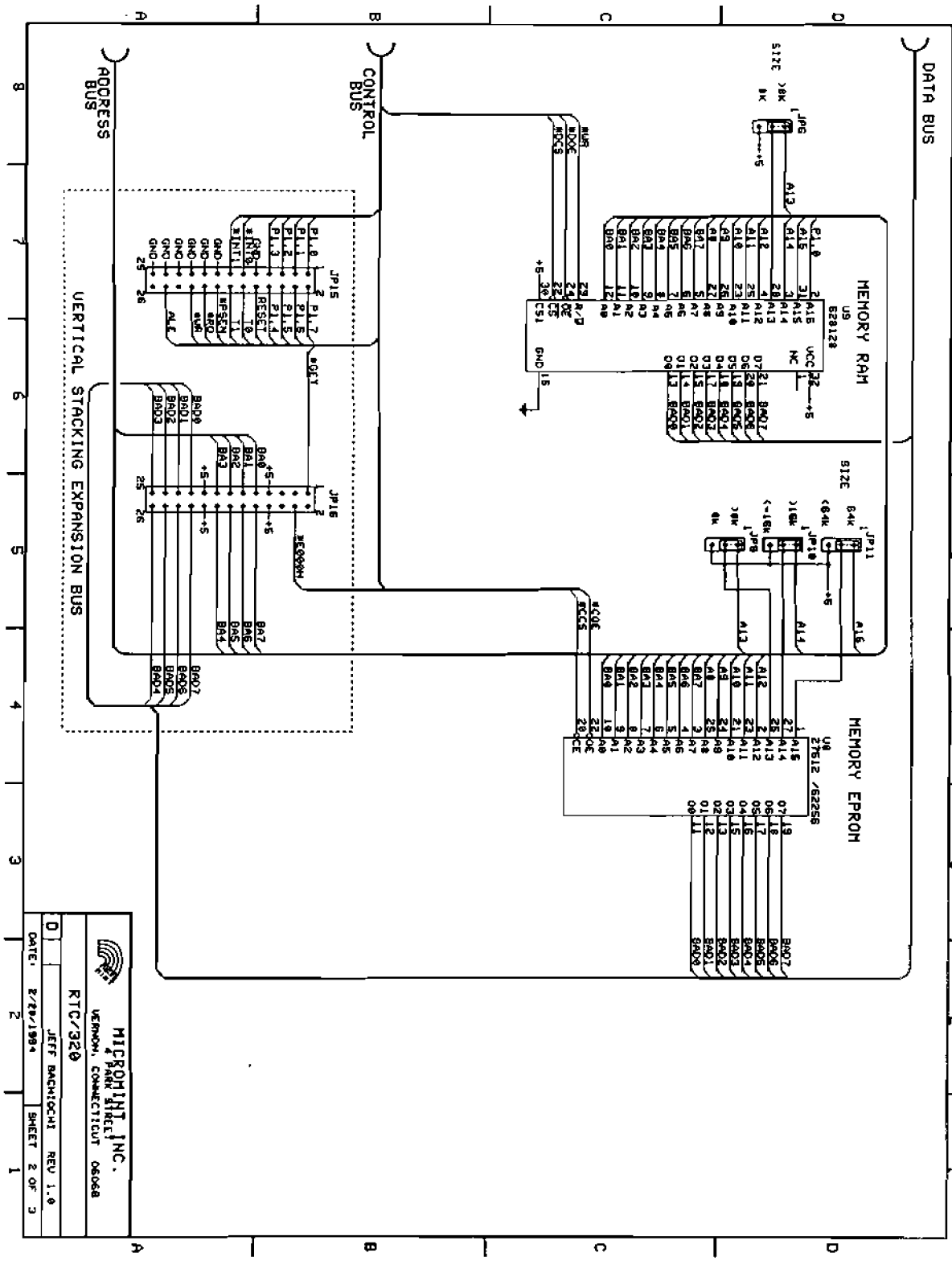
```
                                38
4012 05 86                      39 INC DPS ; aim at DPTR1
4014 90 DC01                    40 MOV DPTR,#PIA_BASE+1;set Port B
                                41
                                42 ;-----------------------------
                                43 ; The main loop picks picks the
                                     count out of Internal RAM &
                                     spins on it
                                44
     =4017'                     45 Blinker EQU $
                                46
                                47 ;---show something interesting
                                       on the parallel ports
                                48
4017 05 86                      49 INC DPS ; aim at DPTR0
4019 EA                         50 MOV A,R2 ; send bit pattern
401A F0                         51 MOVX @DPTR,A
                                52
401B 05 86                      53 INC DPS ; aim at DPTR1
401D F4                         54 CPL A ; send inverse bits
401E F0                         55 MOVX @DPTR,A
                                56
401F F4                         57 CPL A ; restore normal bits
4020 C3                         58 CLR C ; create the next pattern
4021 33                         59 RLC A
4022 70 02'                     60 JNZ PortOK ; all 1 bits gone?
4024 74 FF                      61 MOV A,#0FFh ; yes, reload
4026                            62 PortOK
4026 FA                         63 MOV R2,A ; and store R2,A ; and
                                    store for later
                                64
                                65 ;--- blink the LED
                                66
4027 E8                         67 MOV A,R0 ; pick up ON time
4028 C2 B4                      68 CLR BLINKBIT
402A 12 4040'                   69 CALL Delay
                                70
402D E9                         71 MOV A,R1 ; pick off OFF time
402E D2 B4                      72 SETB BLINKBIT
4030 12 4040'                   73 CALL Delay
                                74
                                75 ;--- a tight loop to show a
                                    single-stepping problem
                                76
4033 75 F0" 03                  77 MOV B,#3
4036 D5 F0" FD'                 78 TightLoop DJNZ B,TightLoop
                                79
4039 80 DC'                     80 SJMP Blinker
                                81
                                82 ;-----------------------------
                                83 ; Delay for about 1 ms times
                                     the contents of A
```

```
                                84 ; This depends on an 11.059 MHz
                                     crystal...
                                85 ; The ORG allows a little room
                                     at the end of the main loop
                                86 ; so a breakpoint on the SJMP
                                     doesn't collide with the
                                     start of
                                87 ; the Delay code...
                                88
     =4040                      89 ORG $4040
                                90
4040                            91 Delay PROC
                                92
4040 C0 F0                      93 PUSH B ; save bystander
                                94
4082 75 F0" 7D                  95 MOV B,#125 ; 125 loops * 8
                                    us/loop
4085                            96 inner1
4085 C0 E0                      97 PUSH ACC
4087 C0 F0                      98 PUSH B
4089 A4                         99 MUL AB ; MUL = 20 clock cycles!
408A A4                        100 MUL AB
408B D0 F0                     101 POP B
408D D0 E0                     102 POP ACC
408F D5 F0" F3'                103 DJNZ B,inner1
                               104
4092 D0 F0                     105 POP B ; restore bystander
                               106
4094 D5 E0" E9'                107 DJNZ ACC,Delay ; repeat ACC
                                    times
                               108
4097 22                        109 RET
                               110
                               111 Delay ENDPROC
                               112
                               113
                               114 ;----
                               115 ; Required by assembler
                               116
                               117 END
```

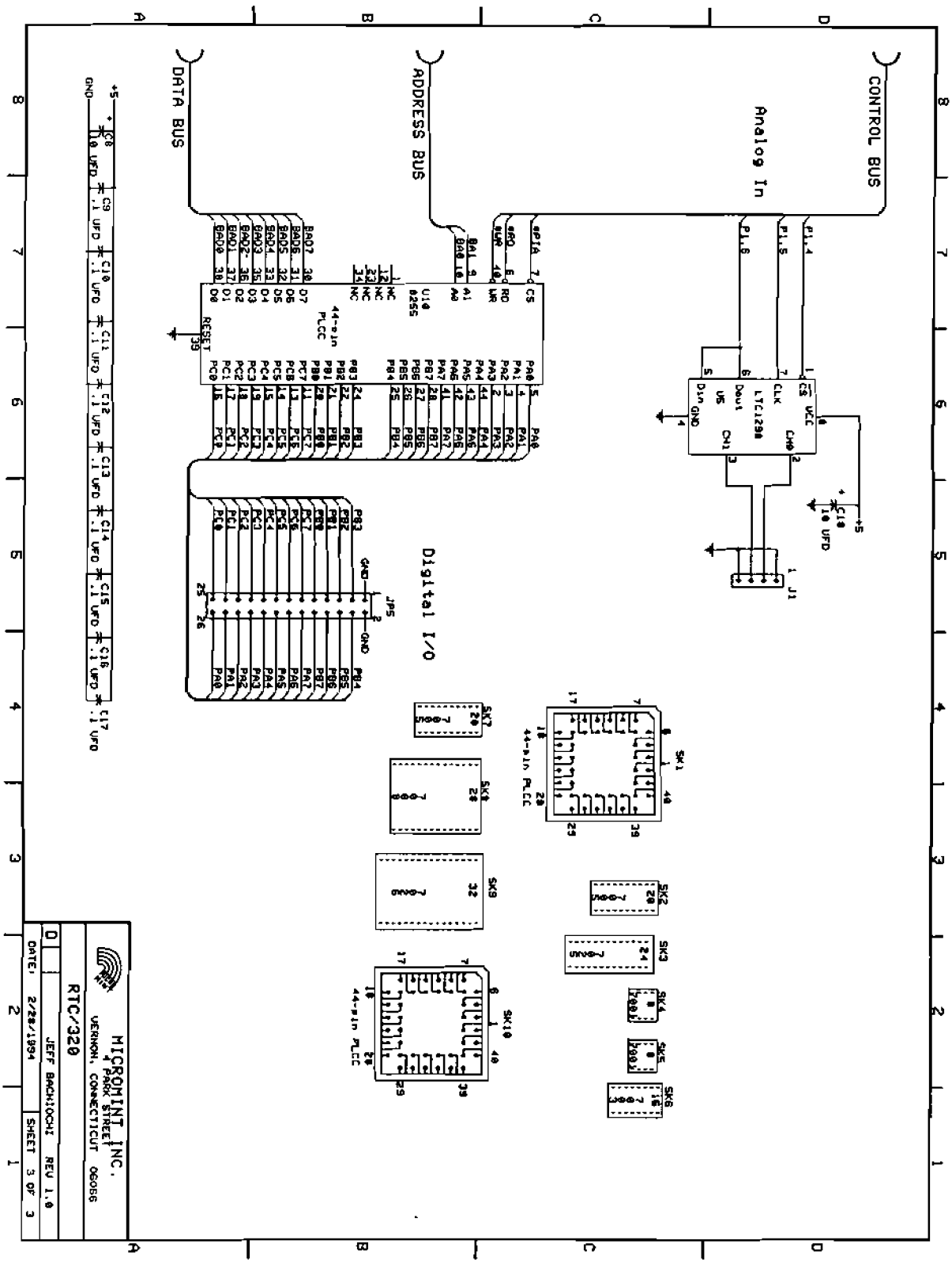
No lines contained errors. No lines contained warnings.



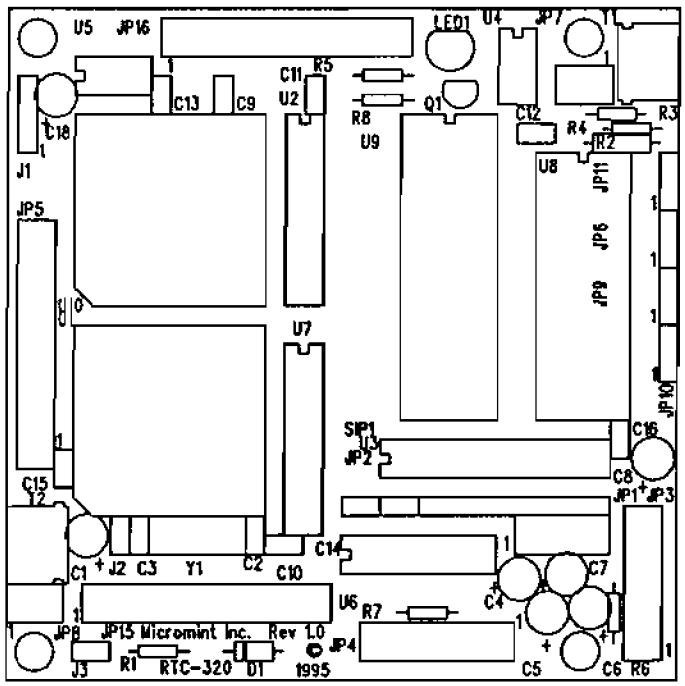
RTC-320 SCHEMATICS (1 of 3)



RTC-320 SCHEMATICS (2 of 3)



RTC-320 SCHEMATICS (3 of 3)



Silkscreen for the RTC-320

PARTS LIST for the RTC-320

DESIGNATION	PART#	DESCRIPTION
Printed Circuit Board		
PCB	RTC-320	Printed Circuit Board for the RTC-320
Integrated Circuits		
U1	80C320	PLCC 8-bit Microcontroller
U2	74F373	Octal Latch
U3	P22CV10A	Programable Logic Device
U4	75176	RS-485 Driver/Receiver
U5	LTC1298	2-channel 12-bit A/D
U6	MAX232	RS-232 Driver/Receiver
U7	74F245	Octal Bi-directional Bus Driver
U8	8k-64k (100nS)	User supplied EPROM (22MHz)
	8k-64k (55nS)	User supplied EPROM (33MHz)
U9	8k-128k (150nS)	RAM (22MHz) default stretch
	8k-128k (100nS)	RAM (22MHz) zero stretch
	8k-128k (100nS)	RAM (33MHz) default stretch
	8k-128k (55nS)	RAM (33MHz) zero stretch
U10	82C55	PLCC Peripheral I/O Adapter
Resistors		
R1 (not used)	4.7k	1/4W, 5%, (yel-vio-red)
R2	100Ω	1/2W, 5%, (brn-blk-brn)
R3 & R4	1k	1/4W, 5%, (brn-blk-red)
R5	470Ω	1/4W, 5%, (yel-vio-brn)
R6-R8	10k	1/4W, 5%, (brn-blk-org)
SIP1	10k	10-pin, 9-common
Capacitors		
C4-C8, C18, C1 (not used)	10μF	16V Tantalum
C2 & C3	22pF	Monolithic
C9-C16	0.1μF	Monolithic
Semiconductors		
D1 (not used)	1N4148	Small signal diode
LED1	TIL220	Light Emitting Diode
Q1	2N2907	PNP transistor

PARTS LIST for the RTC-320 (continued)

DESIGNATION	PART#	DESCRIPTION
Connectors		
J1	1x4	Square pin header
JP1	2x5	Square pin header
J2, J3, JP2	1x2	Square pin header
JP3 & JP4	2x8	Square pin header
JP5, JP15, JP16	2x13	Square pin header (long)
JP6, JP9-11	1x3	Square pin header
JP7 & JP8	2x3	Square pin header
T1 & T2	1x2	Screw terminal block
Sockets		
SK1 & SK10	44-pin	PLCC socket
SK2 & SK7	20-pin	DIP socket
SK3	24-pin	DIP socket
SK4 & SK5	8-pin	DIP socket
SK6	16-pin	DIP socket
SK8	28-pin	DIP socket (0.6")
SK9	32-pin	DIP socket (0.6")
Miscellaneous		
Y1	22.1184	Crystal (22MHz)
	33.1776	Crystal (33MHz)
SJ1-SJ11		Shorting jumpers