

INTRODUCTION

The DS1620 Digital Thermometer and Thermostat provides 9-bit temperature readings. It has three alarm outputs, so the device can also act as a thermostat. The DS1620, which incorporates a three-wire interface can be controlled using an 8051-compatible DS5000 Secure Microcontroller. The DS1620 is connected directly to the I/O port on the DS5000 microcontroller, and the three-wire handshaking and temperature readings are handled by low-level software drivers as shown in this document.

TEMPERATURE CONTROL OF THE DS1620

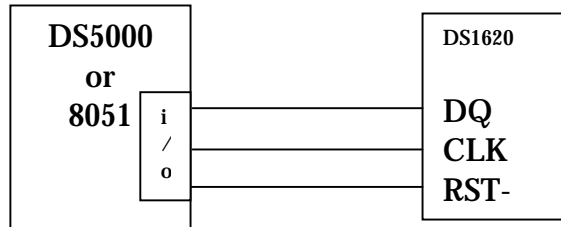
The thermostat outputs of the DS1620 allow it to directly control heating and cooling devices. T_{HIGH} is driven high if the device exceeds a predefined limit set within the TH Register. The output T_{HIGH} can be used to indicate that a high temperature tolerance boundary has been met or exceeded, or it can be used as part of a closed loop system to activate a cooling system and deactivate it when the system temperature returns to tolerance. T_{LOW} is driven high when the temperature of the device falls below the limit set in the TL Register. T_{LOW} remains active until the DS1620's temperature becomes greater than the value stored in the low temperature register, TL. T_{COM} is driven high when the temperature exceeds the limit set in the TH Register and remains high until the device temperature falls below that set in the TL Register. In this way, any amount of user-defined temperature hysteresis can be obtained.

For typical thermostat operation, the DS1620 will operate in continuous mode. However, for applications where only one reading is needed at certain times or to conserve power, the one-shot mode may be used. Note that the thermostat outputs (T_{HIGH} , T_{LOW} , T_{COM}) will remain in the state they were in after the last valid temperature conversion cycle when operating in one-shot mode.

HARDWARE CONFIGURATION

The 3-Wire bus is comprised of three signals. These are the RST- (reset) signal, the CLK (clock) signal, and the DQ (data) signal. All data transfers are initiated by driving the RST- input high. Driving the RST- input low terminates the communication. A clock cycle is a sequence of a falling edge followed by a rising edge. For data inputs, the data must be valid during the rising edge of the clock cycle. Data bits are output on the falling edge of the clock and remain valid through the rising edge. When reading data from the DS1620, the DQ pin goes to a high-impedance state while the clock is high. Taking RST- low during a communication cycle will cause DQ to go to a high-impedance state, thus ending the communication. Data over the 3-Wire interface is sent LSB first. Figure 1.0 illustrates the device connection to the microcontroller programmable input/output port.

Figure 1.0: Hardware Block Diagram



The actual hardware used to simulate the microcontroller environment is provided in Appendix B. Note that the DS5000T is run at a frequency of 11.05949 MHz. The DS232A is used to handle the PC to micro interface. As shown in Appendix B, the 3-Wire connection is made via the I/O port P2. Port I/O P1 can be used to report status or to power a peripheral reporting device such as an LCD

SOFTWARE CONTROL

The C source code used to test the hardware and perform the temperature readings is provided in Appendix A. Note that the header file "reg5000.h" is also provided. Development software tools can be downloaded from the Dallas Semiconductor website: ftp://ftp.dalsemi.com/pub/auto_id/softdev/softdev.html

Appendix A -- C Source Microcontroller Software

```

/*----- DS1620 Microcontroller C Source code -----*/
#pragma CODE SMALL OPTIMIZE(3) /* command line directives */

#include <absacc.h> /* absolute addressing modes */
#include <ctype.h> /* character types */
#include <math.h> /* standard math */
#include <stdio.h> /* standard I/O */
#include <string.h> /* string functions */
#include "reg520.h" /* DS5000 series 8052 registers */
/*-----*/
/* Configuration parameters */
/*-----*/

#define XtalFreq (11059490) /* main crystal frequency */
#define CntrFreq (XtalFreq/12) /* main counter frequency */
#define BaudRate (9600) /* baud rate */
#define CntrTime (8) /* number of cycles for counter */
#define Ft (32768.0) /* target crystal frequency */
/*-----*/
/* External signals */
/*-----*/
sbit RST = 0xA2; /* DS1620 reset line */
sbit CLK = 0xA3; /* DS1620 clock line */
sbit DQ = 0xA4; /* DS1620 data line */
/*-----*/
/* LCD registers */
/*-----*/
#define CONTROL XBYTE[0x0000] /* address for sending commands */
#define STATUS XBYTE[0x0100] /* address for receiving commands */
#define DATABYTE XBYTE[0x0200] /* address for sending data */
#define DQRD XBYTE[0x0004] /* address for receiving commands */
/*-----*/
/* Delay macro */
/*-----*/
#define DELAY(time) \
{ \
    unsigned int k; \
    unsigned int j; \
    j=(XtalFreq/12/65536)*time; \
    for (k=0; k<j; k++) { \
        TF0=TR0=0; \
        TH0=TL0=0; \
        TR0=1; \
    }

```

```

    while (!TF0) {}; \
}; \
}
/*-----*/
/* Control macros */
/*-----*/
#define WrtCtrl(x) {while (STATUS & 0x80) {}; CONTROL=x; }
#define WrtData(x) {while (STATUS & 0x80) {}; DATABYTE=x; }
/*-----*/
/* Function prototypes */
/*-----*/
unsigned char Get1620byte( void );
unsigned char Read1620byte( void );
void Put1620byte(unsigned char m);
void Write1620byte(unsigned char m);
/*-----*/
/* M M AAA III N N */
/* M MA A I N N */
/* MM MM A A I NN N */
/* MM MM AAAAA I N N N */
/* M M MA A I N NN */
/* M MA A I N N */
/* M MA A III N N */
/*-----*/
void main (void) { /* main program */
/*-----*/
/* Local variables */
/*-----*/
unsigned char c; /* dummy variable */
unsigned char Select_Type; /* Function variable */
unsigned char k; /* counter */
unsigned intq; /* used for getting 9 bits */
unsigned char Config_Data; /* Config. Reg. Data */
unsigned int Temp_High; /* Temp. High Data */
unsigned int Temp_Low; /* Temp_Low Data */
unsigned char Read_Temp; /* Temp Data */
unsigned char temp_and_half_bit; /* temp byte and half bit */
unsigned char sign_bit; /* sign bit */
float temp_c; /* temperature in deg. C */
float temp_f; /* temperature in deg. F */
float temp_read; /* */
unsigned intcount_remain; /* used for getting .1 deg. */
unsigned intcount_per_c; /* used for getting .1 deg. */
unsigned char temp_string[10]; /* */

```

```

/*-----*/
/*
/* Start of program execution
/*
/*-----*/
/* Inhibit the watchdog timer and set up memory
/*-----*/
TA = 0xAA; /* timed access
TA = 0x55;
PCON = 0x00; /* inhibit watchdog timer
Select_Type = 0; /*initialize selection
/*-----*/
/* Set up the serial port
/*-----*/
SCON = 0x50; /* SCON: mode 1, 8-bit UART, enable rcvr
TMOD = 0x21; /* TMOD: timer 1, mode 2, 8-bit reload
/* TMOD: timer 0, mode 1, 16-bit

PCON |= 0x80; /* SMOD = 1 Double Baud Rate for TH1 load
TH0=TL0 = 0;
TH1=TL0 = (unsigned int)(256 - ( XtalFreq / BaudRate) / 192));
TR0 = 1; /* TR0: timer 0 run
TR1 = 1; /* TR1: timer 1 run
TI = 1; /* TI: set TI to send first char of UART
/*-----*/
/* Display banner
/*-----*/
printf("\n");
printf(" Dallas Semiconductor - Systems Extension\n");
printf(" Source for DS1620 Temperature Reading only.\n");
printf(" Updated Code September 12, 2000 \n");
printf(" [C Program for DS500x or 8051 Compatible Microcontroller]");
printf("\n\n");
printf("\n*****\n");
printf(" Select Menu Option\n");
printf(" 1. Read Temperature\n");
printf(" 2. Read Temperature High (TH)\n");
printf(" 3. Read Temperature Low (TL) \n");
printf(" 4. Read Configuration Register\n");
printf(" 5. Write Configuration Register = 03h, Clear Flags\n");
printf(" 6. Set Temperature High (TH) = 50h\n");
printf(" 7. Set Temperature High (TH) = 20h\n");
printf(" 8. Set Temperature Low (TL) = 0\n");
printf(" 9. Set Temperature Low (TL) = 19h\n");

```

```

printf ("      A. Start Conversion\n");

printf ("      Note: Temperature Settings are in degrees Centigrade\n");

/*-----*/
/*  Setup to get temp                               */
/*-----*/
RST=0;
RST=1;
Put1620byte(0xAC);      /* Read Config Register   */
k = Get1620byte();      /* get Config Register Data */
RST=0;

if ( (k & 0x18) != 0x08 )      /* bad or no part          */
{
    printf("Error talking to part!\n");

    printf("before Config. Data loop = %02X\n", k); /*What is Config. Setting? */
    c = getchar();
}

if ( (k & 0x03) != 0x03 )      /* check mode settings     */
{
    RST=1;
    Put1620byte(0x0C);
    Put1620byte(0x03);      /* set to CPU & 1SHOT mode */
    RST=0;
}

RST=1;
Put1620byte(0x22);          /* stop convert (ie reset) */
RST=0;

/*-----*/
/*  Setup to get time  C5 3A A3 5C C5 3A A3 5C      */
/*-----*/
do {
/*-----*/
/*  Enable CE2                                       */
/*-----*/
EA      = 0;          /* Inhibit interrupts      */
TA      = 0xAA;      /* timed access           */
TA      = 0x55;
MCON    = MCON | 0x04;      /* Enable topside CE      */
}

```

```

/*-----*/
/*  Disable CE2                               */
/*-----*/
TA  = 0xAA;          /* timed access      */
TA  = 0x55;
MCON = 0xC8;          /* Disable topside CE */
EA   = 1;             /* Enable interrupts   */
/*-----*/
/*  Start convert and wait to finish          */
/*-----*/
DELAY(1);
RST=1;
Put1620byte(0xEE);    /* start temp convert */
RST=0;

do
    {
        RST=1;
        Put1620byte(0xAC);    /* open status register */
        k = Get1620byte();    /* get status byte      */
        RST=0;
        /*  printf("Waiting : %02X\n", k); Debug print line*/
        } while ((k & 0x80) != 0x80); /*changed 0x80 to 0x00*/

Select_Type = getchar();    /* get variable to start */
switch(Select_Type)
{

    case '1': printf ("    1. Read Temperature\n");

/*-----*/
/*  Read temp and sign bit                               */
/*-----*/
RST=1;
Put1620byte(0xAA);          /* read temp command    */
temp_and_half_bit = Get1620byte(); /* read 1st byte of temp */
sign_bit = Get1620byte();    /* read 2nd byte of temp */
RST=0;
/*-----*/
/*  Get count remain & count per C for .1 resolution    */
/*-----*/
RST=1;
Put1620byte(0xA0);          /* read count remaining */
count_remain = Get1620byte(); /* read 1st byte        */

```

```

count_remain += Get1620byte() * 256;      /* read 2nd byte      */
RST=0;

RST=1;
Put1620byte(0x41);      /* load slope into counter */
RST=0;

RST=1;
Put1620byte(0xA0);      /* read slope as count/c */
count_per_c = Get1620byte();      /* read 1st byte      */
count_per_c += Get1620byte() * 256;      /* read 2nd byte      */
RST=0;
/*-----*/
/* Calculate  $\theta C$  and  $\theta F$  */
/*-----*/
if ( count_per_c == 0 ) count_per_c = 1;
    temp_read = ( temp_and_half_bit >> 1 );
if ( sign_bit != 0 ) temp_read = (temp_read - 128);
    temp_c = (float)temp_read-0.25
    + (count_per_c-count_remain)/(float) count_per_c;
    temp_f = temp_c * 9/5 + 32;

/*-----*/
/* Display temp to CRT */
/*-----*/
printf( "\n\sign=%2d T&HB=%3d\n ", (int)sign_bit,(int)temp_and_half_bit);
printf ( "T/2=%5.1f \n", temp_read );
printf( "c remain=%3d c per c=%3d\n ",count_remain, count_per_c);
printf( "TempC=%5.1f \n", temp_c);
printf( "TempF=%5.1f \n", temp_f);
break;

    case '2': printf ( "      2. Read Temperature High (TH)\n");
        RST=1;
        Put1620byte(0xA1);      /* open Temp. High register */
        DELAY(1);
        Temp_High = Read1620byte();      /* get status byte      */
        RST=0;
        printf( "Data : %02X\n", Temp_High); /*Debug print line*/
        break;

    case '3': printf ( "      3. Read Temperature Low (TL) \n");
        RST=1;
        Put1620byte(0xA2);      /* open Temp. Low register */

```



```

    DELAY(.1);
    Temp_Low = Get1620byte();           /* get status byte */
    RST=0;
    printf("Data : %02X\n", Temp_Low);/*Debug print line*/
    break;
case '4': printf("    4. Read Configuration Register\n");
    RST=0;
    DELAY(.1);
    RST=1;
    Put1620byte(0xAC);                 /* open Config. register */
    DELAY(.1);
    Config_Data = Get1620byte();       /* get status byte */
    RST=0;
    printf("Data : %02X\n", Config_Data);/*Debug print line*/
    break;

case '5': printf("    5. Write Configuration Register = 03h, Clear Flags\n");
    RST=1;
    DELAY(.1);
    Put1620byte(0x0C); /* Write to Config. Register Clear Flags*/
    DELAY(.1);
    Put1620byte(0x03); /* set to 02h */
    DELAY(.1);
    Put1620byte(0xAC);                 /* open Config. register */
    DELAY(.1);
    Config_Data = Get1620byte();       /* get status byte */
    printf("Data : %02X\n", Config_Data);/*Debug print line*/
    RST=0;
    break;

case '6': printf("    5. Set Temperature High (TH) = 50\n");
    RST=1;
    Put1620byte(0x01); /* Write to Temp. High Register */
    Write1620byte(0x50); /* set to 30h */
    RST=0;
    break;

case '7': printf("    6. Set Temperature High (TH) = 20\n");
    RST=1;
    Put1620byte(0x01); /* Write to Temp. High Register */
    Write1620byte(0x20); /* set to 20h */
    RST=0;
    break;

```

```

        case '8': printf ("      7. Set Temperature Low (TL) = 0\n");
                RST=1;
                Put1620byte(0x02); /* Write to Temp. Low Register */
                Write1620byte(0x0); /* set to 00h */
                RST=0;
                break;

        case '9': printf ("      8. Set Temperature Low (TL) = 19\n");
                RST=1;
                Put1620byte(0x02); /* Write to Temp. Low Register */
                Write1620byte(0x19); /* set to 19h */
                RST=0;
                break;

        case 'a': printf ("      Start Conversion \n");
                DELAY(.1);
                RST=1;
                Put1620byte(0xEE); /* start temp conversion */
                RST=0;
                break;
        default: printf ("      Typo: Select Another Menu Option\n");
                break;
}; /* end switch*/

} while (1); /* Loop forever */
/*-----*/
/* End of program */
/*-----*/
}
/*----- Begin Function Definitions -----*/
/*-----*/
/* Get temp from DS1620 */
/*-----*/
unsigned char Get1620byte(void)

{

unsigned char j,k=0,b=1;
k=0;
b=1;
for (j=0; j<8; j++)
{
CLK=0;
if (DO) k|=b; /* Read bit and or if = 1 */

```

```

        CLK=1;
        b=(b<<1);          /* Setup for next read and or */
    }
return k;
}

/*-----*/
/* Put temp from DS1620 */
/*-----*/
void Put1620byte(unsigned char m)

{
unsigned char k,b=1;
    RST=1;
    for (k=0; k<8; k++)
    {
        CLK=0;
        DQ = (m & b);      /* Send bit to 1620 */
        CLK=1;
        b=(b<<1);        /* Setup to send next bit */
    }
return;
}

/*-----*/
/* read temp from DS1620 */
/*-----*/

unsigned char Read1620byte( void )
{
unsigned char j,k=0,b=1;
k=0;
b=1;
    for (j=0; j<10; j++)
    {
        CLK=0;
        if (DQ) k|=b;      /* Read bit and or if = 1 */
        CLK=1;
        b=(b<<1);        /* Setup for next read and or */
    }
return k;
}

/*-----*/
/* write temp from DS1620 */
/*-----*/

```

```

/*-----*/
void Write1620byte(unsigned char m)

{
unsigned char k,b=1;
RST=1;
for (k=0; k<10; k++)
{
CLK=0;
DQ = (m & b);          /* Send bit to 1620 */
CLK=1;
b=(b<<1);             /* Setup to send next bit */
}
return;
}

```

```

*-----*/
/* Register Declarations for DS5000/DS80C320/DS80C520 Processor */
/*-----*/
/* DS5000 series special registers filename: reg5000.h */
/*-----*/
sfr STATUS = 0xDA;
sfr RPCTL = 0xD8;
sfr RNR = 0xCF;
sfr TA = 0xC7;
sfr MCON = 0xC6;
sfr CRCHIGH = 0xC3;
sfr CRCLOW = 0xC2;
sfr CRC = 0xC1;
/*-----*/
/* BYTE register */
/*-----*/
sfr P0 = 0x80;
sfr P1 = 0x90;
sfr P2 = 0xA0;
sfr P3 = 0xB0;
sfr PSW = 0xD0;
sfr ACC = 0xE0;
sfr B = 0xF0;
sfr SP = 0x81;
sfr DPL = 0x82;
sfr DPH = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;

```

```

sfr TMOD = 0x89;
sfr TL0 = 0x8A;
sfr TL1 = 0x8B;
sfr TH0 = 0x8C;
sfr TH1 = 0x8D;
sfr IE = 0xA8;
sfr IP = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

sfr T2CON = 0xC8;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;
sfr TL2 = 0xCC;
sfr TH2 = 0xCD;
/*-----*/
/* BIT registers */
/*-----*/
/* PSW */
/*-----*/
sbit CY = 0xD7;
sbit AC = 0xD6;
sbit F0 = 0xD5;
sbit RS1 = 0xD4;
sbit RS0 = 0xD3;
sbit OV = 0xD2;
sbit P = 0xD0;
/*-----*/
/* TCON */
/*-----*/
sbit TF1 = 0x8F;
sbit TR1 = 0x8E;
sbit TF0 = 0x8D;
sbit TR0 = 0x8C;
sbit IE1 = 0x8B;
sbit IT1 = 0x8A;
sbit IE0 = 0x89;
sbit IT0 = 0x88;
/*-----*/
/* IE */
/*-----*/
sbit EA = 0xAF;
sbit ES = 0xAC;
sbit ET1 = 0xAB;

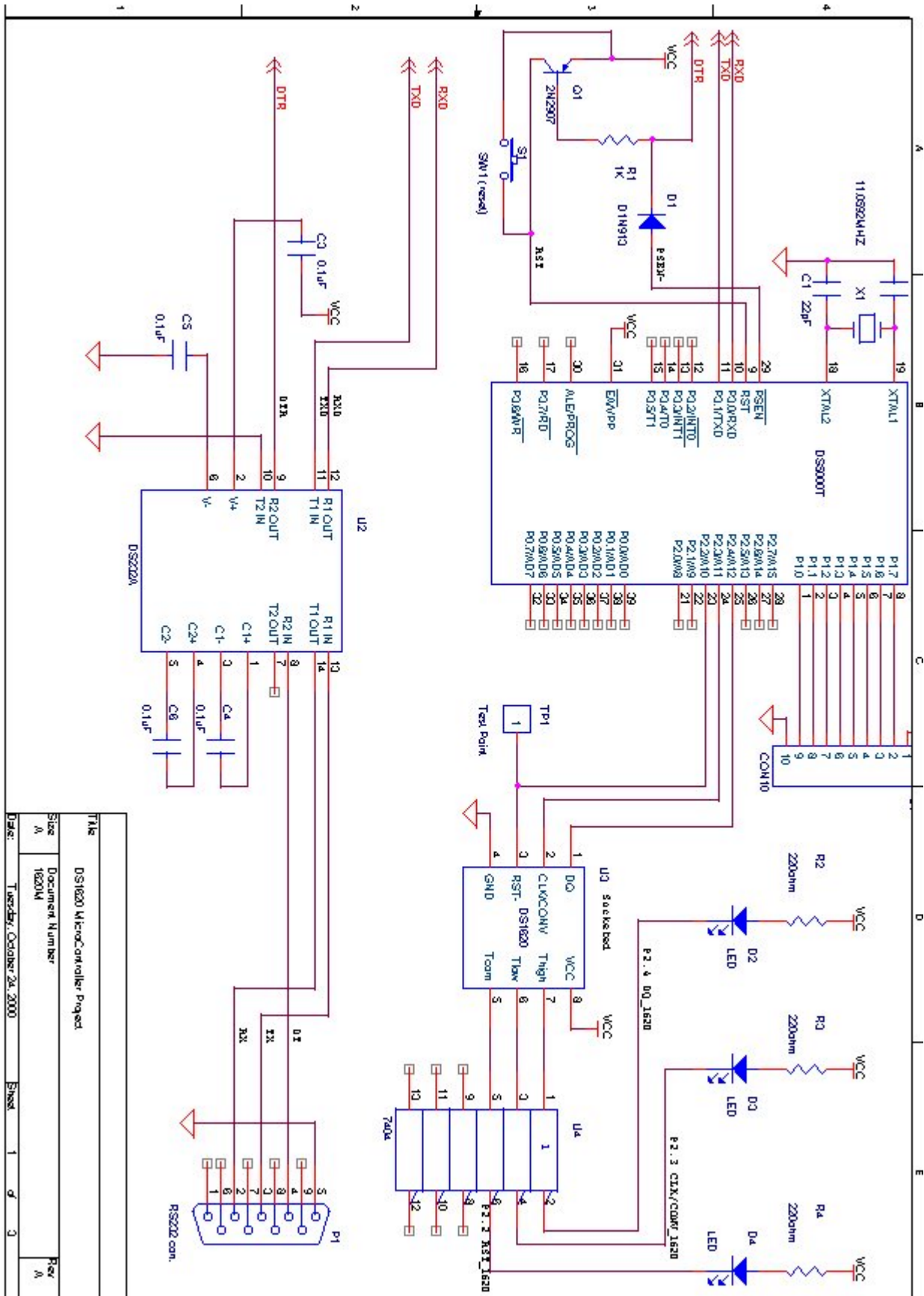
```

```

sbit  EX1  = 0xAA;
sbit  ET0  = 0xA9;
sbit  EX0  = 0xA8;
/*-----*/
/*  IP          */
/*-----*/
sbit  PS  = 0xBC;
sbit  PT1 = 0xBB;
sbit  PX1 = 0xBA;
sbit  PT0 = 0xB9;
sbit  PX0 = 0xB8;
/*-----*/
/*  P3          */
/*-----*/
sbit  RD  = 0xB7;
sbit  WR  = 0xB6;
sbit  T1  = 0xB5;
sbit  T0  = 0xB4;
sbit  INT1 = 0xB3;
sbit  INT0 = 0xB2;
sbit  TXD = 0xB1;
sbit  RXD = 0xB0;
/*-----*/
/*  SCON          */
/*-----*/
sbit  SM0 = 0x9F;
sbit  SM1 = 0x9E;
sbit  SM2 = 0x9D;
sbit  REN = 0x9C;
sbit  TB8 = 0x9B;
sbit  RB8 = 0x9A;
sbit  TI  = 0x99;
sbit  RI  = 0x98;
/*-----*/
/*  T2CON          */
/*-----*/
sbit  TF2 = 0xCF;
sbit  T2IP = 0xCE;
sbit  T2IE = 0xCD;
sbit  T2RSE = 0xCC;
sbit  BGEN = 0xCB;
sbit  TR2 = 0xCA;
sbit  C_T2 = 0xC9;
sbit  CP_RL2 = 0xC8;
/*-----*/
/*  End of DS definitions          */
/*-----*/

```

Appendix B DS5000/8051 Microcontroller Hardware



FILE	DS1620 MicroController Project		
Size	Document Number	Rev	A
A	16201M		
Date:	Tuesday, October 24, 2000	Sheet	1 of 3