

Application Note 520

Using a Watchdog Timekeeping RAM with a Microcontroller

www.maxim-ic.com

INTRODUCTION

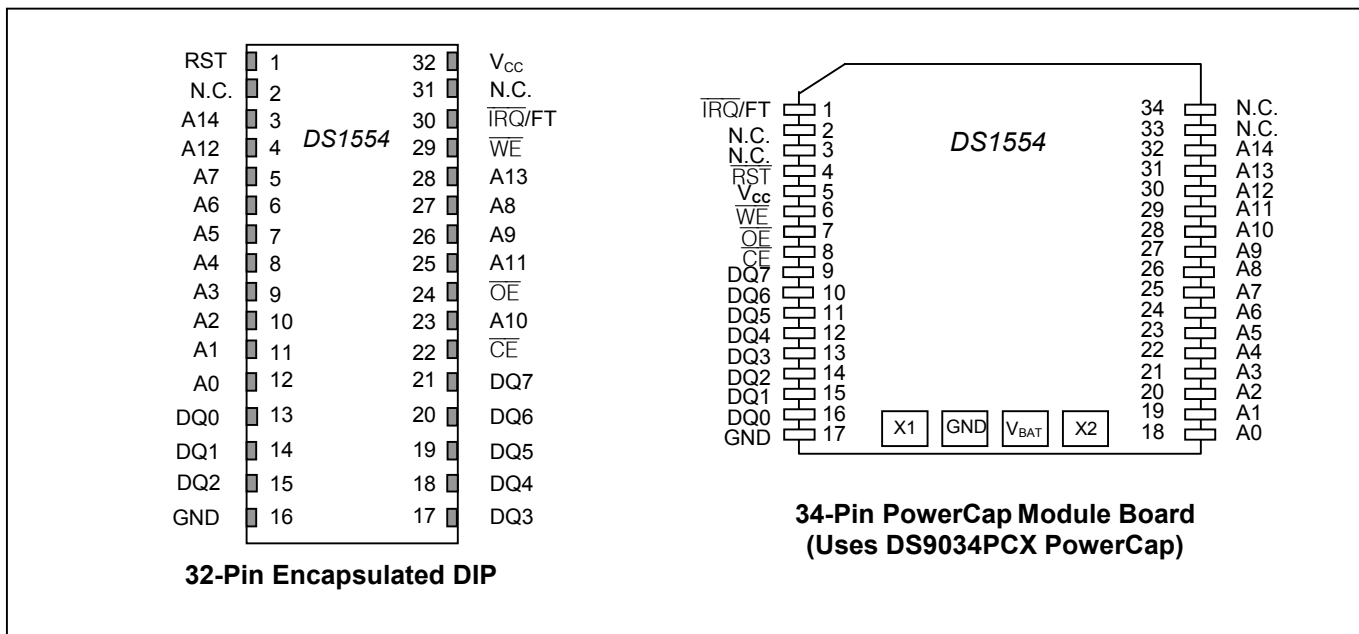
The nonvolatile (NV) Y2K-compliant timekeeping family of real-time clocks (RTCs) provides battery-backed NV static RAM (SRAM), time, and date as well as a watchdog timer. The clock registers are accessed identically to the SRAM. The DS155_ family is available in an encapsulated DIP or surface-mountable PowerCap® package. Table 1 shows a list of products in the nonvolatile timekeeping family.

In this example, a DS1553P, DS1554P, or DS1556P (PowerCap) is connected to a DS87C320. The DS87C320 high-speed microcontroller is compatible with the industry-standard 8051 architecture. For related application notes, please see the QuickView data sheets for the DS87C320 and DS1554. Other NV RAM densities could be supported with appropriate changes to the circuit. The DS155_ PowerCap is placed in the socket labeled DS1646 PWR_CAP. The PowerCap footprint is common to all PowerCap products. The software in this example does not support the DS1557.

Table 1. Nonvolatile Timekeeping Products

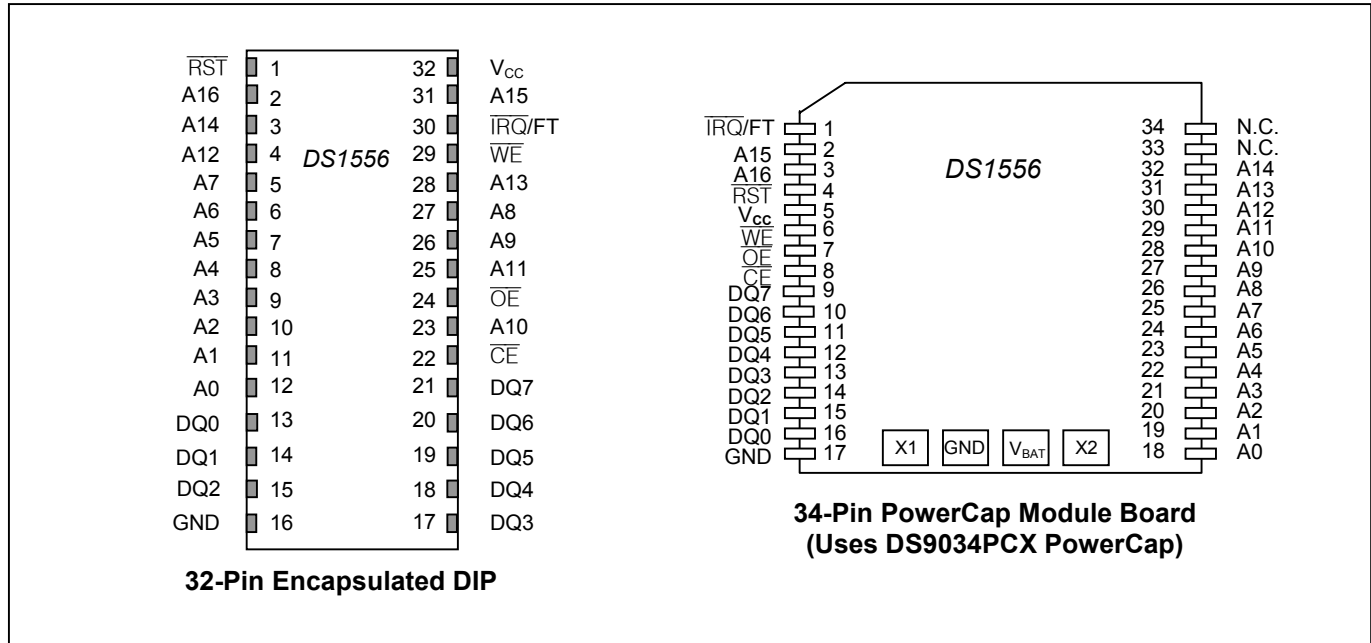
PART	RAM DENSITY	V _{CC} (V)
DS1553	8k x 8	3.3 or 5
DS1554	32k x 8	3.3 or 5
DS1556	128k x 8	3.3 or 5
DS1557	512k x 8	3.3 or 5

PIN CONFIGURATION



PowerCap is a registered trademark of Dallas Semiconductor.

PIN CONFIGURATION (CONTINUED)



EXAMPLE SOFTWARE

```

/*****
/* DS1556.c - access DS1553/4/6/7 using Ref RTC circuit          */
/* This program is for example only and is not supported by Dallas */
/* Semiconductor Maxim                                           */
/*****
#pragma code symbols debug
#include <stdio.h>          /* Prototypes for I/O functions          */
#include <DS5000.h>        /* Register declarations for DS5000      */
#include <absacc.h>        /* needed to define xdata addresses     */
#include <delay.c>
/***** bit definitions *****/
/***** Defines *****/
/***** Global Variables *****/
uchar int_flg = 0;
unsigned MAXADD;

sbit  IRQb  = P3^2;
sbit  AD16  = P1^5;
sbit  AD17  = P1^6;
sbit  AD18  = P1^7;
/***** Function Prototypes *****/
void  writereg();
void  init_rtc();
void  Disp_Clk_Regs();
void  ramread();
void  ramwrite();
void  wdlshot();
void  toggle_ft();
void  checkint();

void init_rtc() /* ----- set the time and date ----- */
/* Note: NO error checking is done on the user entries! */

```

```

{
uchar yr, mn, dt, dy, hr, min, sec, day;

    XBYTE[0xffff7] = 0;      /* disable watchdog */
    XBYTE[0xffff6] = 0;      /* disable alarms */

    printf("\nEnter the year (0-99): ");
    scanf("%bx", &yr);
    printf("Enter the month (1-12): ");
    scanf("%bx", &mn);
    printf("Enter the date (1-31): ");
    scanf("%bx", &dt);
    printf("Enter the day (1-7): ");
    scanf("%bx", &dy);
    printf("Enter the hour (1-23): ");
    scanf("%bx", &hr);
    printf("Enter the minute (0-59): ");
    scanf("%bx", &min);
    printf("Enter the second (0-59): ");
    scanf("%bx", &sec);

    AD16 = AD17 = AD18 = 1;
    XBYTE[0xffff8] = 0x80;   /* set write bit for write */
    XBYTE[0xffff9] = sec;
    XBYTE[0xffffa] = min;
    XBYTE[0xffffb] = hr;
    XBYTE[0xffffc] = dy;
    XBYTE[0xffffd] = dt;
    XBYTE[0xffffe] = mn;
    XBYTE[0xfffff] = yr;
    XBYTE[0xffff8] = 0;      /* clear write bit to allow update */
}
void Disp_Clk_Regs()        /* ----- */
{
uchar msec, Sec, prv_sec = 99, Min, Hrs, Dte, Mon, Day, Yr;

    AD16 = AD17 = AD18 = 1;
    printf("\n    Yr Mn Dt Dy Hr:Mn:Sec AD AH:AM:AS");
    while(!RI) /* Read & Display Clock Registers */
    {
        XBYTE[0xffff8] = 0x40; /* set read bit to stop updates */
        Sec = XBYTE[0xffff9];
        Min = XBYTE[0xffffa];
        Hrs = XBYTE[0xffffb];
        Day = XBYTE[0xffffc];
        Dte = XBYTE[0xffffd];
        Mon = XBYTE[0xffffe];
        Yr = XBYTE[0xfffff];
        XBYTE[0xffff8] = 0;      /* clear read bit to resume updates */
        delay(3);                /* must allow time for transfer to occur */

        if(Sec != prv_sec) /* display every time seconds change */
        {
            printf("\nDUT %02.bX %02.bX %02.bX %02.bX", Yr, Mon, Dte, Day);
            printf(" %02.bX:%02.bX:%02.bX", Hrs, Min, Sec);

            printf(" %02.bX %02.bX:%02.bX.%02.bX", XBYTE[0xffff5],
                XBYTE[0xffff4], XBYTE[0xffff3], XBYTE[0xffff2]);
        }
    }
}

```

```

        prv_sec = Sec;
    }
    RI = 0; /* Swallow keypress to exit loop */
}
void disp_clk_regs_int() /* ----- display time using irq output ----- */
{
    uchar msec, Sec, Min, Hrs, Dte, Mon, Day, Yr;

    XBYTE[0xffff5] = 0x80; /* mask alarm date */
    XBYTE[0xffff4] = 0x80; /* mask alarm hours */
    XBYTE[0xffff3] = 0x80; /* mask alarm minutes */
    XBYTE[0xffff2] = 0x80; /* mask alarm seconds */
    XBYTE[0xffff6] = 0x80; /* enable alarm, configured for 1/sec */
    XBYTE[0xffff8] = 0; /* clear read bit to enable updates */

    EX0 = 1; /* enable interrupt 0 */
    IT0 = 1; /* edge activated */
    PX0 = 0; /* low priority */
    EX1 = 0; /* disable interrupt 1 */
    EA = 1; /* enable all interrupts */

    AD16 = AD17 = AD18 = 1;
    printf("\n Yr Mn Dt Dy Hr:Mn:Sc WA AD AH:AM:AS FL");
    while(!RI) /* Read & Display Clock Registers */
    {
        while(!int_flg); /* wait for interrupt */

        /* in a real application, we'd service the interrupt once per second,
        and let the micro do something else the rest of the time. We
        don't need to use the 'R' bit to disable transfers as long as
        we take less than 1 second to service the interrupt */

        if(int_flg & 0x40) /* only display if an alarm caused the interrupt
*/
        {
            EX0 = EX1 = 0; /* disable interrupts while updating the display
*/

            Sec = XBYTE[0xffff9];
            Min = XBYTE[0xffffa];
            Hrs = XBYTE[0xffffb];
            Day = XBYTE[0xffffc];
            Dte = XBYTE[0xffffd];
            Mon = XBYTE[0xffffe];
            Yr = XBYTE[0xfffff];

            printf("\nDUT %02.bX %02.bX %02.bX %02.bX", Yr, Mon, Dte, Day);
            printf(" %02.bX:%02.bX:%02.bX", Hrs, Min, Sec);

            printf(" %02.bX %02.bX %02.bX:%02.bX.%02.bX %02.bX",
XBYTE[0xffff7],
                XBYTE[0xffff5], XBYTE[0xffff4], XBYTE[0xffff3], XBYTE[0xffff2],
                XBYTE[0xffff0]);

            int_flg = 0; /* for this example, we'll just clear all
interrupt sources */
            EX0 = EX1 = 1; /* re-enable interrupts */
        }
    }
}

```

```

    EX0 = 0;          /* disable interrupt */
    XBYTE[0xffff] = 0; /* disable alarm */
    RI = 0;          /* Swallow keypress to exit loop */
}
void ramread()      /* ----- */
{
    unsigned int j;

    AD16 = AD17 = AD18 = 0;

    for (j = 0; j <= MAXADD; j++)
    {
        if(j != 0 && !(j % 256) )
        {
            printf("\nPress a key or Esc to exit ");
            if( _getkey() == 0x1b)
                return;
        }
        if(!(j % 16)) printf("\n%bx%04x ", (uchar) AD16, j);
        printf("%02.bx ", (uchar) XBYTE[j]);

        if (j == 0xffff) break;
    }
    if(MAXADD == 0xffff)
    {
        printf("\nPress a key or Esc to exit ");
        if( _getkey() == 0x1b)
            return;

        AD16 = 1;
        for (j = 0; j <= MAXADD; j++)
        {
            if(j != 0 && !(j % 256) )
            {
                printf("\nPress a key or Esc to exit ");
                if( _getkey() == 0x1b)
                    break;
            }
            if(!(j % 16)) printf("\n%bx%04x ", (uchar) AD16, j);
            printf("%02.bx ", (uchar) XBYTE[j]);

            if (j == 0xffff) break;
        }
    }
}
void ramwrite()    /* ----- write with incrementing data ----- */
{
    unsigned int j;
    uchar offset = 0;

    if(MAXADD == 0xFFFF)
    {
        AD16 = AD17 = AD18 = 0;
        for (j = 0; j <= MAXADD; j++)
        {
            XBYTE[j] = (uchar) ( j & 0xff) + offset);
            if( (j & 0xff) == 0xff) offset++;
            if(j == 0xffff) break;
        }
    }
}

```

```

}

AD16 = 1;
offset = 0;
for (j = 0; j < (MAXADD - 16); j++)
{
    XBYTE[j] = (uchar) ( (j & 0xff) + offset);
    if( (j & 0xff) == 0xff) offset++;
}
}

void ramfill(uchar dat) /* ----- fill with fixed data ----- */
{
    unsigned int j;

    if(MAXADD == 0xFFFF)
    {
        AD16 = AD17 = AD18 = 0;
        for (j = 0; j <= MAXADD; j++)
        {
            if(j & 0x100)
                XBYTE[j] = dat;
            else
                XBYTE[j] = dat ^ 0xff;
            if(j == 0xffff) break;
        }
    }
    AD16 = 1;
    for (j = 0; j < (MAXADD - 16); j++)
    {
        if(j & 0x100)
            XBYTE[j] = dat;
        else
            XBYTE[j] = dat ^ 0xff;
    }
}

void toggle_ft() /* ----- toggle the Frequency Test bit ----- */
{
    uchar creg;

    AD16=1;
    creg = XBYTE[0xffffc];
    if( (creg & 0x40) ) /* check FT bit */
    {
        XBYTE[0xffff8] = 0x80; /* set write bit for write */
        XBYTE[0xffffc] = (creg & 0xbf); /* clear FT bit */
        XBYTE[0xffff8] = 0; /* clear write bit */
    }
    else
    {
        XBYTE[0xffff8] = 0x80; /* set write bit for write */
        XBYTE[0xffffc] = (creg | 0x40); /* set FT bit */
        XBYTE[0xffff8] = 0; /* clear write bit */
    }
}

void checkint() /* ----- */
{
    AD16=1;
    XBYTE[0xffff7] = 0x80; /* set watchdog register */
}

```

```

XBYTE[0xffff6] = 0x80; /* set the interrupt register */

XBYTE[0xffff5] = 0x80; /* mask date alarm */
XBYTE[0xffff4] = 0x80; /* mask hour alarm */
XBYTE[0xffff3] = 0x80; /* mask minute alarm */
XBYTE[0xffff2] = 0; /* minute alarm */

while(1)
{
    if( XBYTE[0xffff9] == 0x01 )
        XBYTE[0xffff0]; /* clear the alarm */
}

void wd_irq_tst() /* ---- test wd to IRQb function (repeat) ---- */
{
    uchar tmp;

    AD16=1;

    IRQb = 1; /* drive /IRQ high using port i/o */
    XBYTE[0xffff0] = 0; /* clear interrupt flags */
    XBYTE[0xffff7] = 0x08; /* enable watchdog, timeout = 2 * 1/16 = 125ms */
    XBYTE[0xffff6] = 0; /* disable alarm */
    printf("\nPress a key to exit ");
    while(!RI)
    {
        while(IRQb); /* wait for IRQb to go active from watchdog
timeout */
        tmp = XBYTE[0xffff0]; /* clear IRQ */
        tmp = XBYTE[0xffff7]; /* restart timer */
    }
    RI = 0;
}

void wd_rst_tst() /* ---- test wd to RSTb function (one shot) ---- */
{
    uchar tmp;

    AD16=1;

    XBYTE[0xffff6] = 0; /* disable alarm */
    XBYTE[0xffff0] = 0; /* clear interrupt flags */
    XBYTE[0xffff7] = 0x90; /* enable watchdog, timeout = 4 * 1/16 = 250ms */
    XBYTE[0xffff6] = 0; /* disable alarm */
}

void external0_int(void) interrupt 0 /* --- display time/date on interrupt from
RTC --- */
{
    EX0 = EX1 = 0; /* disable interrupt */
    int_flg = XBYTE[0xffff0]; /* clear the interrupt source, save the info */
    EX0 = EX1 = 1; /* re-enable interrupt */
}

main (void) /* ----- */
{
    uchar i, M, M1;

    IRQb = 1;

    printf("\n1) DS1553 2) DS1554 or 3)DS1556? ");
    i = _getkey();

```

```

MAXADD = 0xffff; /* default */
if (i == '1')     MAXADD = 0x1fff;
if (i == '2')     MAXADD = 0x7fff;
printf("%x", MAXADD);
while (1)
{
printf("\nDS1553/4/6 build 106\n");
printf("CI Init clock CC Check Int\n");
printf("CR Read Time  CN Rd CLK w/int\n");
printf("RW Write RAM  RR Read RAM\n");
printf("W  Wtchdg tst T  Toggle FT\n");
printf("Enter Menu Selection:");

M = _getkey();

switch(M)
{
    case 'C':
    case 'c':
printf("\rEnter Clock Routine to run: C");
M1 = _getkey();

    switch(M1)
    {
        case 'C':
        case 'c':    checkint(); break;

        case 'I':
        case 'i':    init_rtc(); break;

        case 'N':
        case 'n':    disp_clk_regs_int();    break;

        case 'R':
        case 'r':    Disp_Clk_Regs();    break;
    }
break;

    case 'R':
    case 'r':
printf("\rFill RAM A)a, 5)5, I)nc data or R)ead: ");
M1 = _getkey();
switch(M1)
{
    case '5':    ramfill(0x55);    break;

    case 'A':
    case 'a':    ramfill(0xaa);    break;

    case 'R':
    case 'r':    ramread();    break;

    case 'I':
    case 'i':    ramwrite();    break;
}
break;

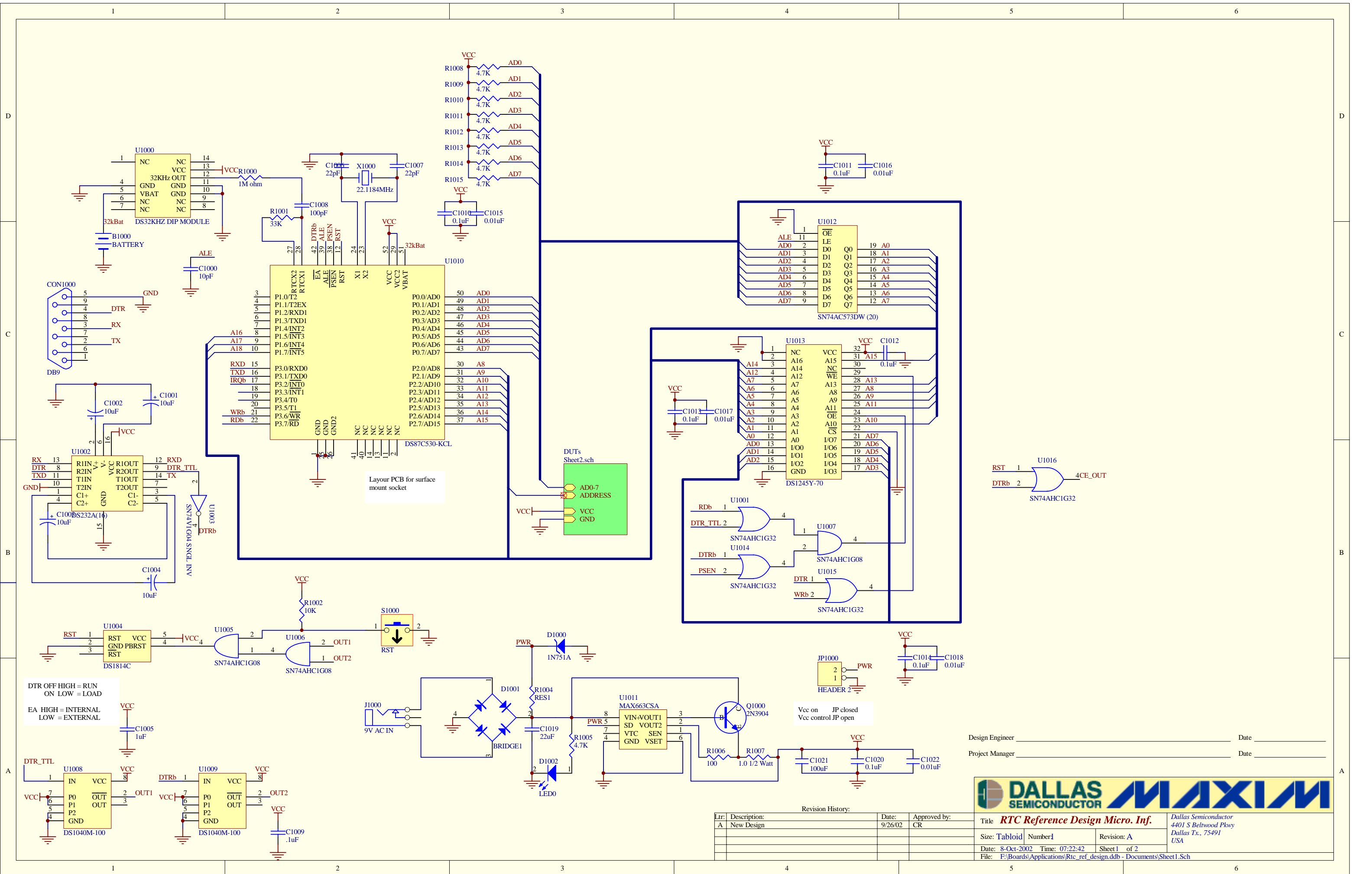
    case 'T':
    case 't':    toggle_ft();    break;

```



```
case 'W':
case 'w':
printf("\rEnter wd test I)rq: R)st");
M1 = _getkey();
switch(M1)
{
    case 'I':
    case 'i':    wd_irq_tst();    break;

    case 'R':
    case 'r':    wd_rst_tst();    break;
}
break;
}
}
}
```



DTR OFF HIGH = RUN
ON LOW = LOAD

EA HIGH = INTERNAL
LOW = EXTERNAL

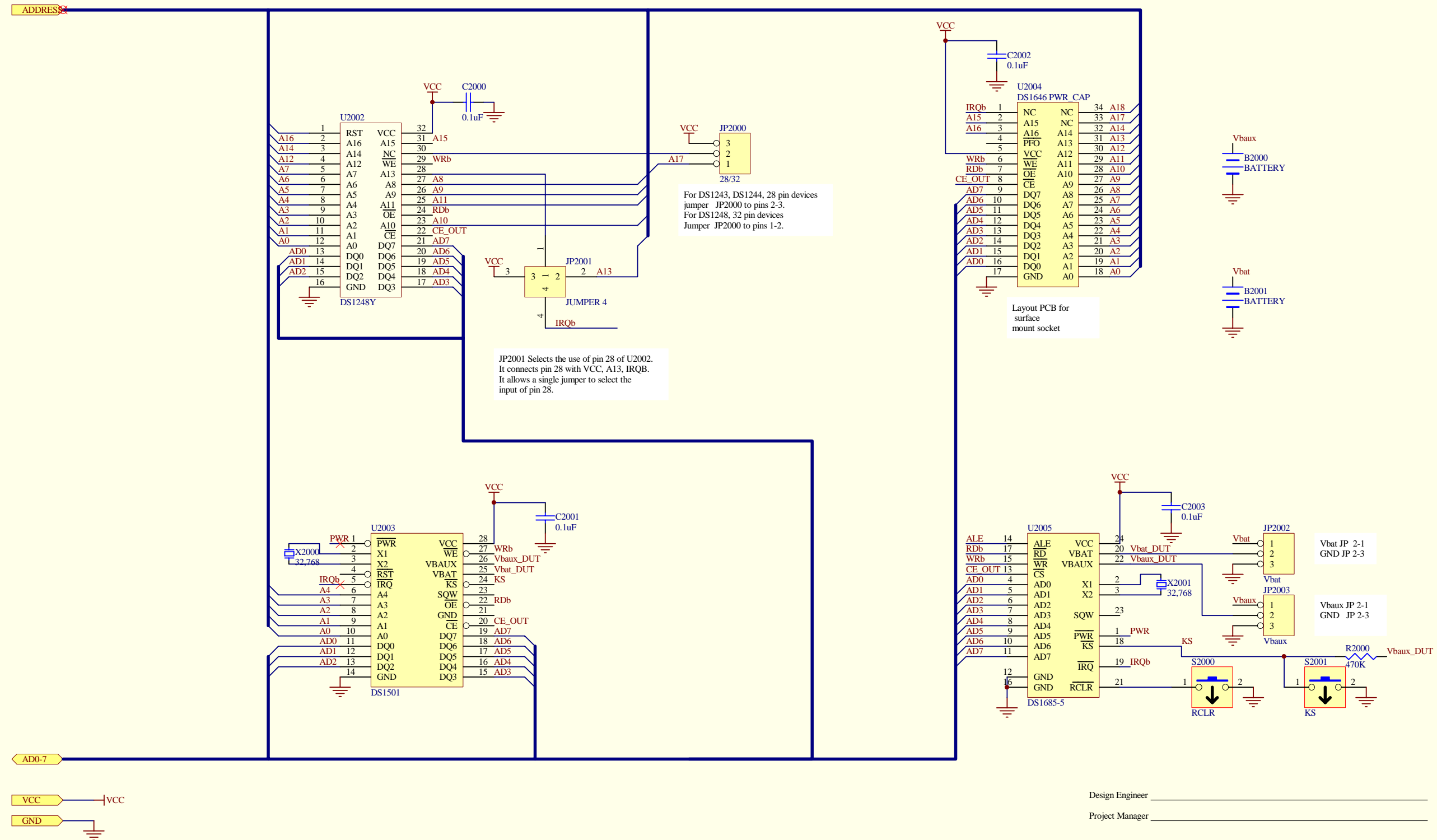
Revision History:

Ltr:	Description:	Date:	Approved by:
A	New Design	9/26/02	CR

Design Engineer _____ Date _____
 Project Manager _____ Date _____



Note: Populate U2002, U2003, U2004 or U2005 as needed.



Design Engineer _____ Date _____
 Project Manager _____ Date _____

Revision History:		Date:	Approved by:	Title	
Lr:	Description:	9/26/02	CR	RTC Reference Design - R.T.C. Inf.	
A	New Design			Dallas Semiconductor 4401 S Belwood Pkwy Dallas Tx., 75491 USA	
				Size: Tabloid	Number: 1
				Date: 8-Oct-2002	Time: 07:22:44
				Sheet 2	of 2
				File: F:\Boards\Applications\Rtc_ref_design.ddb - Documents\Sheet2.sch	

