Application Note 2004
Calibrating the DS1340 Real-Time Clock

www.maxim-ic.com

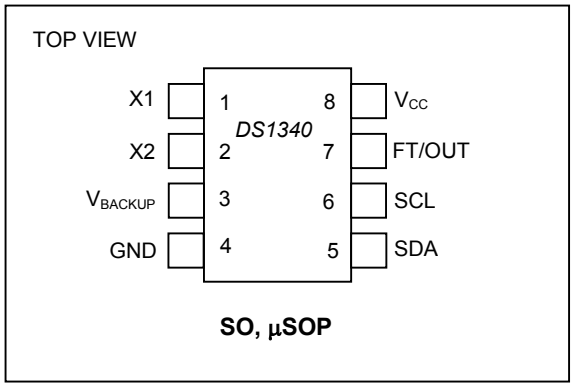
INTRODUCTION

The DS1340 2-wire real-time clock (RTC) with a trickle charger offers the ability to calibrate the clock using software. This application note shows an example of how to calibrate the clock. In this example, the DS1340 is interfaced to a 8051-compatible microcontroller. A frequency or timing counter is attached to the FT/OUT pin, and writing a 40h to the control register enables the 512Hz output. After measuring the actual frequency of the 512Hz output, the measured value is entered at the appropriate prompt of the calibration subroutine. The calibration subroutine calculates the closest calibration value and writes the value to the control register.

Figure 1 shows a flowchart of the DS1340 calibration procedure, followed by full code and a circuit schematic.

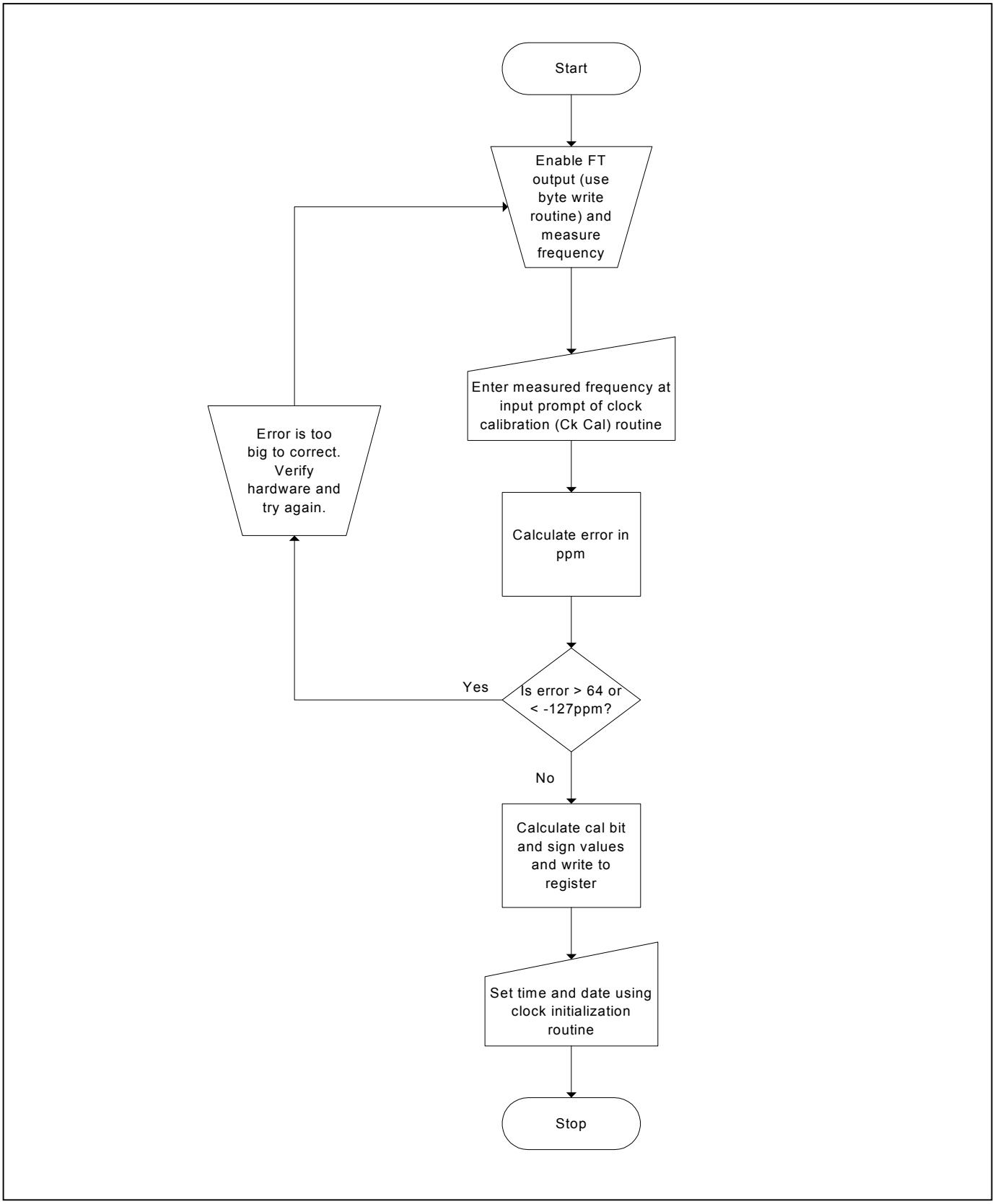
Download the full DS1340 data sheet from our website at ***www.maxim-ic.com/DS1340***.

PIN CONFIGURATION



SO, µSOP

Figure 1. Flowchart for Calibrating the DS1340



DS1340 Code

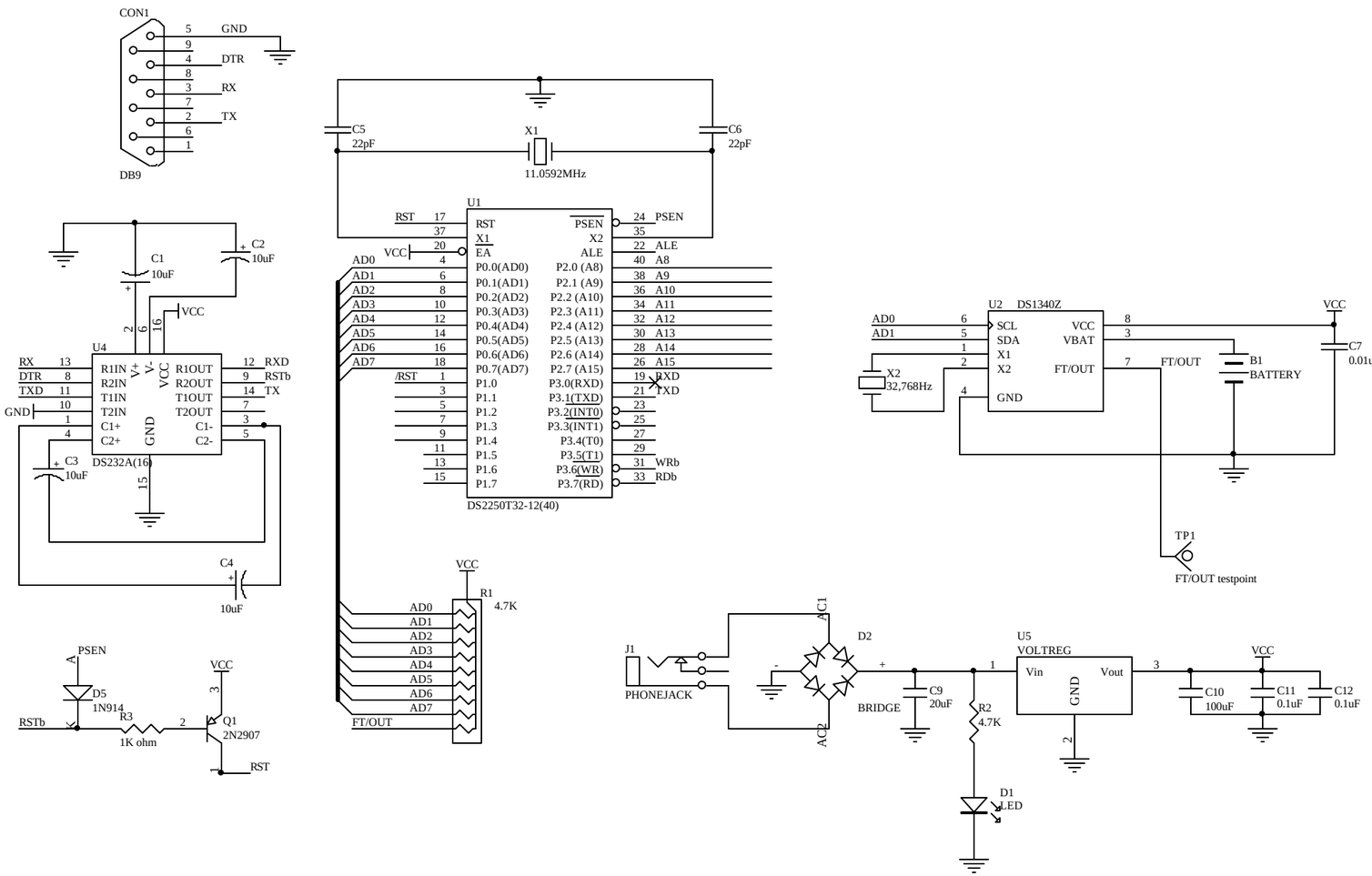
```
/*********************************************************************/
/* DD1340an.c  This program is for example only and is not supported */
/*             by Dallas Semiconductor Maxim                         */
/*********************************************************************/
/* #pragma code symbols debug */
#include <stdio.h>            /* Prototypes for I/O functions        */
#include <DS5000.h>           /* Register declarations for DS5000    */
/***************************** Defines ****************************/
/************************* bit definitions ************************/
#define     ACK   0
#define     NACK  1
#define     ADDRTC      0xd0  /* 2-wire addresses */
sbit scl = P0^0;              /* 2-wire pin definitions */
sbit sda = P0^1;
sbit sqw = P3^2;
void  start2w();
void  stop2w();
void  writebyte2w(unsigned char d);
uchar readbyte2w(uchar);
void  readbyte();
void  writebyte();
void  initialize();
void  disp_clk_regs(uchar);
/* global variables */
uchar sec, min, hr, dy, dt, mn, yr;
void start2w()          /* ---------------------------------------------- */
{
      sda = 1;  scl = 1;      /* Initiate start condition */
      sda = 0;
}
void stop2w()           /* ---------------------------------------------- */
{
      sda = 0; sda = 0; sda = 0; sda = 0; /* Initiate stop condition */
      scl = 1; scl = 1; sda = 1;
}
void writebyte2w(uchar d)           /* ----------------------------- */
{
char i;

      scl = 0;
      for (i = 1; i <= 8; i++)
      {
/*          if(d & 0x80)
                sda = 1; /* Send the msbits first */
/*          else
                sda = 0; */
            sda = (d >> 7);
            scl = 1;
            d = d << 1; /* increase scl high time */
            scl = 0;
      }
      sda = 1;    /* Release the sda line */
      scl = 0;
      scl = 1;
      if(sda) printf("Ack bit missing  %02X\n",(unsigned int)d);
      scl = 0;
}
```

```
uchar readbyte2w(uchar b)     /* ---------------------------------- */
{
char i;
uchar d;

      sda = 1;             /* Let go of sda line */
      scl = 0;
      for (i = 1; i <= 8; i++)      /* read the msb first */
      {
            scl = 1;
            d = d << 1;
            d = d | (unsigned char)sda;
            scl = 0;
      }
      sda = b;          /* Hold sda low for acknowledge */

      scl = 0;
      scl = 1;
      if(b == NACK)     sda = 1;    /* sda = 1 if next cycle is reset */
      scl = 0;

      sda = 1;          /* Release the sda line */
      return d;
}
void writebyte()  /* ------------------------------------------------ */
{
uchar Add;
uchar Data;
      printf("\nAddress: ");  /* Get Address */
      scanf("%bx", &Add);
      printf("DATA: ");
      scanf("%bx", &Data);    /* and data */
      start2w();
      writebyte2w(ADDRTC);
      writebyte2w(Add);
      writebyte2w(Data);
      stop2w();
}
void readbyte()   /* ------------------------------------------------ */
{
uchar Add;
      printf("\nADDRESS: ");  /* Get Address */
      scanf("%bx", &Add);
      start2w();
      writebyte2w(ADDRTC);
      writebyte2w(Add);
      start2w();
      writebyte2w(ADDRTC | 1);
      printf("%2.bx", readbyte2w(NACK) );
      stop2w();
}
void initialize() /* ---------------------------------------- */
/* Note: NO error checking is done on the user entries! */
{
uchar yr, mn, dt, dy, hr, min, sec, day;

      start2w();        /* The following Enables the Oscillator */
      writebyte2w(ADDRTC);    /* address the part to write */
      writebyte2w(0x00);      /* position the address pointer to 0 */
```

```
      writebyte2w(0x00);      /* write 0 to the seconds register, clear the CH bit
*/
      stop2w();

      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);
      hr = hr & 0x3f;   /* force clock to 24 hour mode */
      printf("Enter the minute (0-59): ");
      scanf("%bx", &min);
      printf("Enter the second (0-59): ");
      scanf("%bx", &sec);

      start2w();
      writebyte2w(ADDRTC);    /* write slave address, write 1340 */
      writebyte2w(0x00);      /* write register address, 1st clock register */
      writebyte2w(sec);
      writebyte2w(min);
      writebyte2w(hr);
      writebyte2w(dy);
      writebyte2w(dt);
      writebyte2w(mn);
      writebyte2w(yr);
      stop2w();

}
void  disp_clk_regs(uchar prv_sec)  /* ---------------------------------------- */
{
uchar Sec, Min, Hrs, Dte, Mon, Day, Yr, mil, pm;

      printf("\n    Yr Mn Dt Dy Hr:Mn:Sc");

      while(!RI)  /* Read & Display Clock Registers */
      {
            /* burstramwrite(0x55); /* try to corrupt data in clock by writing to
the RAM */

            start2w();
            writebyte2w(ADDRTC);    /* write slave address, write 1340 */
            writebyte2w(0x00);      /* write register address, 1st clock register */
            start2w();
            writebyte2w(ADDRTC | 1);      /* write slave address, read 1340 */
            Sec = readbyte2w(ACK);  /*  starts  w/last  address  stored  in  register
pointer */
            Min = readbyte2w(ACK);
            Hrs = readbyte2w(ACK);
            Day = readbyte2w(ACK);
            Dte = readbyte2w(ACK);
            Mon = readbyte2w(ACK);
            Yr  = readbyte2w(NACK);
            stop2w();
            if(Hrs & 0x40)
```

```
                  mil = 0;
            else
                  mil = 1;

            if(Sec != prv_sec)      /* display every time seconds change */
            {
                  if(mil)
                  {
                        printf("\n      %02.bX  %02.bX  %02.bX  %2.bX",  Yr,  Mon,  Dte,
Day);
                        printf(" %02.bX:%02.bX:%02.bX", Hrs, Min, Sec);
                  }
                  else
                  {
                        if(Hrs & 0x20)
                              pm = 'A';
                        else
                              pm = 'P';
                        Hrs &= 0x1f;      /* strip mode and am/pm bits */
                        printf("\nDUT  %02.bx  %02.bx  %02.bx  %02.bx",  Yr,  (Mon  &
0x1f), Dte, Day);
                        printf(" %02.bx:%02.bx:%02.bx %cM", Hrs, Min, Sec, pm);
                  }
            }
            if(prv_sec == 0xfe)     return;
            prv_sec = Sec;
      }
      RI = 0;  /* Swallow keypress to exit loop */
}
void  clk_cal()   /* ----- calculate correction value & write to cal bits ----- */
{
float val, err;
char  cal;  /* signed 8 bit int value */

      do
      {
            printf("\nEnter FT freq (ie 512.0037): ");
            scanf("%f", &val);
            err = ( (val/512)-1) * 1e6;         /* calculate error in ppm */
      }     while(err > 64 || err < -127);      /* must be within correctable range
*/

      if(err < 0)       /* osc is running slow */
      {
            cal = -(err + 2.034)/4.068;   /* # of adjustment increments (+ppm) */
            printf("%f %bx ", err, cal);
            cal |= 0x20;      /* set sign bit for positive calibration */
            printf("%bx", cal);
      }
      if(err >= 0)            /* osc is running fast */
      {
            cal = (err + 1.018)/2.034;    /* # of adjustment increments (-ppm) */
            printf("%f %bx ", err, cal);
      }
      start2w();
      writebyte2w(ADDRTC);    /* write slave address, write 1340 */
      writebyte2w(0x07);      /* set pointer to control register address */
      writebyte2w(cal); /* set cal */
      stop2w();
```

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

      sqw = 1;    /* set up for read, weak pull-up */
      while (1)
      {
            printf("\nDS1340 build 100\n");
            printf("CR Read Time CI Clk Init\n");
            printf("CC Clk Cal\n");
            printf("BR Byte Read BW Write Byte\n");
            printf("Enter Menu Selection:");

            M = _getkey();

            switch(M)
            {
                  case 'B':
                  case 'b':
                  printf("\nRead or Write: ");
                  M1 = _getkey();

                  switch(M1)
                  {
                        case 'R':
                        case 'r':   readbyte();       break;

                        case 'W':
                        case 'w':   writebyte();      break;
                  }
                  break;

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

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

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

                        case 'R':
                        case 'r':   disp_clk_regs(0x99);    break;
                  }
                  break;
            }
      }
}
```



Title	Dallas Semiconductor Maxim DS1340AN demo board	
Size B	Number	Revision A
Date: 29-May-2003		Sheet bf 1
File: C:\Data\Designlab\DS1340AN\DS1340AN.ddb		Drawn By: