



APPLICATION NOTE 3334

## Interfacing a MAX6900 RTC with an 8051-Type Microcontroller

*Abstract: This app note provides example hardware and software for interfacing the MAX6900 with an 8051-type microcontroller.*

### Description

This app note shows how to interface a MAX6900 I<sup>2</sup>C-Compatible RTC (real-time clock) to an 8051-type microcontroller ( $\mu$ C) and provides example code showing basic interface routines. The microcontroller used in this example is the DS2250, and the software is written in C.

### Operation

The program uses two general-purpose port pins on the micro to operate as a master on the I<sup>2</sup>C bus. The MAX6900 operates as a slave device on the same bus.

A schematic of the circuit is shown in **Figure 1**. The software is shown in **Figure 2**.



```

/*****
/* This program is for example only and is not supported by Dallas Maxim */
#include <stdio.h>           /* Prototypes for I/O functions */
#include <DS5000.h>         /* Register declarations for DS5000 */
#define ACK      0
#define NACK     1
#define ADD6900 0xa0      /* 2-wire addresses */
sbit scl = P0^0;          /* 2-wire pin definitions */
sbit sda = P0^1;
void I2Cstart();
void I2Cstop();
uchar I2Cwrite(uchar);
unsigned char I2Cread(int);
void writebyte6900();
void Initialize_MAX6900();
void disp_clk_regs();
void burstramread();
void burstramwrt();
/* global variables */

void I2Cstart()           /* ----- */
{
    sda = 1;             /* Initiate start condition */
    scl = 1;
    sda = 0;
}
void I2Cstop()           /* ----- */
{
    sda = 0;             /* Initiate stop condition */
    scl = 1;
    sda = 1;
}
uchar I2Cwrite(uchar d)  /* ----- */
{
    int i;

    scl = 0;
    for (i = 0; i < 8; i++)
    {
        if (d & 0x80)
            sda = 1; /* Send the msbits first */
        else
            sda = 0;
        scl = 0;
        scl = 1;
        d = d < 1;    /* add to scl high time */
        scl = 0;
    }
    sda = 1;           /* Release the sda line */
    scl = 0;
    scl = 1;
    i = sda;
    scl = 0;
    if (i)
    {
        puts("Ack missing");
    }
    return(i);
}
uchar I2Cread(int b)     /* ----- */
{

```

```

uchar i, d;

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

scl = 0;          /* toggle clock */
scl = 1;
scl = 0;

sda = 1;          /* Release the sda line */
return d;
}
void writebyte6900() /* ----- write a single byte; user enters read address ----- */
{
uchar add;
uchar dat;
    /* Get Address & Data */
    printf("\nEnter the Read Address\nADDRESS (80,82,84...FC): ");
    scanf("%bx", &add);
    printf("DATA (0-ff):");
    scanf("%bx", &dat);

    I2Cstart();
    I2Cwrite(ADD6900); /* slave address + write */
    I2Cwrite(add);
    I2Cwrite(dat);
    I2Cstop();
}
void Initialize_MAX6900() /* ----- initialize from stdio entries ----- */
/* Note: NO error checking is done on the user entries! */
{
uchar yr, mn, dt, dy, hr, min, sec, day;

    I2Cstart();
    I2Cwrite(ADD6900); /* slave address + write */
    I2Cwrite(0x8e); /* control register write address */
    I2Cwrite(0x00); /* clear write protect */
    I2Cstop();

    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);

I2Cstart();
I2Cwrite(ADD6900);      /* slave address + write */
I2Cwrite(0xbe); /* clock burst write */
I2Cwrite(sec);
I2Cwrite(min);
I2Cwrite(hr);
I2Cwrite(dt);
I2Cwrite(mn);
I2Cwrite(dy);
I2Cwrite(yr);
I2Cwrite(0);           /* control */
I2Cstart();
I2Cwrite(ADD6900);      /* slave address + write */
I2Cwrite(0x92);
I2Cwrite(0x20); /* century data */
I2Cstop();
}
void disp_clk_regs()      /* ----- display using burst mode ----- */
{
uchar Sec, prv_sec = 99, Min, Hrs, Dte, Mon, Day, Yr, cy;

while(!RI)      /* Read & Display Clock Registers */
{
    I2Cstart();
    I2Cwrite(ADD6900);      /* slave address + write */
    I2Cwrite(0xbf);      /* clock burst read */
    I2Cstart();
    I2Cwrite(ADD6900 + 1); /* slave address + read */
    Sec = I2Cread(ACK);      /* starts w/last address stored in register pointer */
    Min = I2Cread(ACK);
    Hrs = I2Cread(ACK);
    Dte = I2Cread(ACK);
    Mon = I2Cread(ACK);
    Day = I2Cread(ACK);
    Yr = I2Cread(ACK);
    cy = I2Cread(NACK);      /* dummy read of control register */
    I2Cstart();
    I2Cwrite(ADD6900);      /* slave address + write */
    I2Cwrite(0x93);      /* century byte read address */
    I2Cstart();
    I2Cwrite(ADD6900 + 1); /* slave address + read */
    cy = I2Cread(NACK);
    I2Cstop();

    if(Sec != prv_sec)      /* display every time seconds change */
    {
        printf("\n%02bX%02bX/%02bX/%02bX %01bX", cy, Yr, Mon, Dte, Day);
        printf(" %02bX:%02bX:%02bX", Hrs, Min, Sec);
    }
    prv_sec = Sec;
}
    RI = 0; /* Swallow keypress to exit loop */
}
void burstramread()      /* ----- */
{
uchar j, k;

```

```

I2Cstart();
I2Cwrite(ADD6900);      /* write slave address, write 6900 */
I2Cwrite(0xff); /* ram burst read */
I2Cstart();
I2Cwrite(ADD6900 + 1); /* slave address + read */
printf("\nRAM contents");
for (j=0; j<30; ++j)
{
    if(!(j % 8) )    printf("\n");
    printf("%2.bX ", I2Cread(ACK) );
}
printf("%2bX", I2Cread(NACK) ); /* last byte, NACK */
I2Cstop();
printf("\n");
}
void burstramwrt(uchar Data)          /* ----- */
{
uchar j, k;

I2Cstart();
I2Cwrite(ADD6900);      /* write slave address, write 6900 */
I2Cwrite(0xfe); /* ram burst write */
for (k=0; k < 31; ++k)
{
    I2Cwrite(Data);
}
I2Cstop();
}
main (void)          /* ----- */
{
uchar i, M, M1;

while (1)
{
    printf("\nMAX6900 build %s\n", __DATE__);
    printf("CI Clock Init\n");
    printf("CR Clock Read BW Byte Write\n");
    printf("RR RAM Read  RW RAM Write\n");
    printf("Enter Menu Selection: ");

    M = _getkey();

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

            switch(M1)
            {
                case 'W':
                case 'w':          writebyte6900();
                                    break;
            }          break;

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

```

```

switch(M1)
{
    case 'I':
    case 'i':        Initialize_MAX6900();
                    break;

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

case 'R':
case 'r':
printf("\rEnter Ram Routine to run:R");
M1 = _getkey();

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

    case 'W':
    case 'w':        printf("\nEnter the data to write: ");
                    scanf("%bx", &i);
                    burstramwrt(i); break;
}    break;
}
}
}

```

---

Application Note 3334: <http://www.maxim-ic.com/an3334>

### More Information

For technical questions and support: <http://www.maxim-ic.com/support>

For samples: <http://www.maxim-ic.com/samples>

Other questions and comments: <http://www.maxim-ic.com/contact>

### Related Parts

MAX6900: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

AN3334, AN 3334, APP3334, Appnote3334, Appnote 3334

Copyright © by Maxim Integrated Products

Additional legal notices: <http://www.maxim-ic.com/legal>