時計ユニット(RTC)

概要

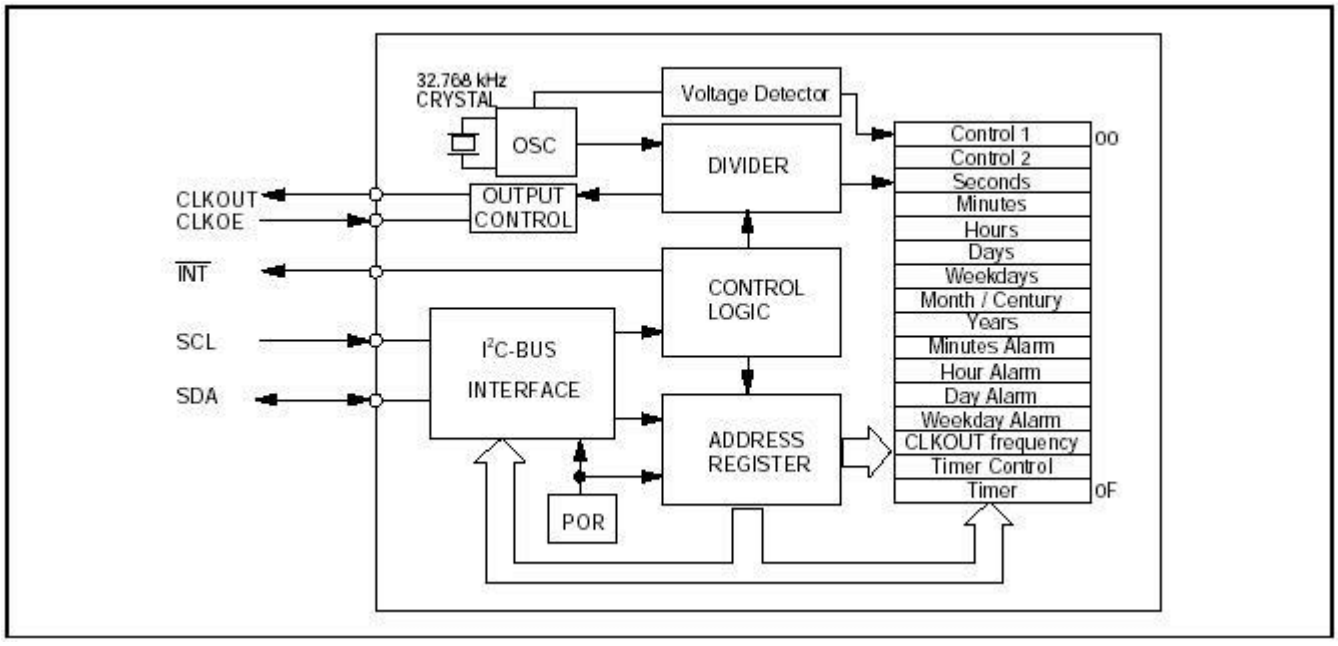
秋月電子通商で販売している「リアルタイムクロックモジュール(RTC-8564NB)」を利用した時計を作成しました。このモジュールの特長は以下のとおりです。

- セイコーのリアルタイムクロックIC(RTC-8564NB)を使ったクロックモジュールです。
- I2Cインターフェースで PICやH8などと通信可能です。
- IC内に高精度クリスタルが内蔵されています。
- パーツは全て実装済み。8ピンDIPのICとしてお使いになれます。
- INT割り込み用のLED付き
- ピンは実装済みです。
- 動作電圧:1.8V~5V
- アラーム機能、タイマー機能、周波数出力機能
- 消費電流:330nA(@5V、非アクセス時)

動作原理

RTCとPICとの接続は、I2Cインターフェースを使用します。I2CとはInter Integrated Circuitの略で、主に同一基板内などの近距離に配置されたデバイス間での高速 通信(100Kbps/400Kbps/3.4Mbps)を行うための方式です。 デバイス間はSDA(serial data)とSCL(serial clock)の2本の信号線だけをバスとして共有して通信を行います。

今回使用したモジュールのブロック図です。



このモジュールは、次のレジスタテーブルを利用して、データの書き込みや読み出しを行います。

■レジスタテーブル

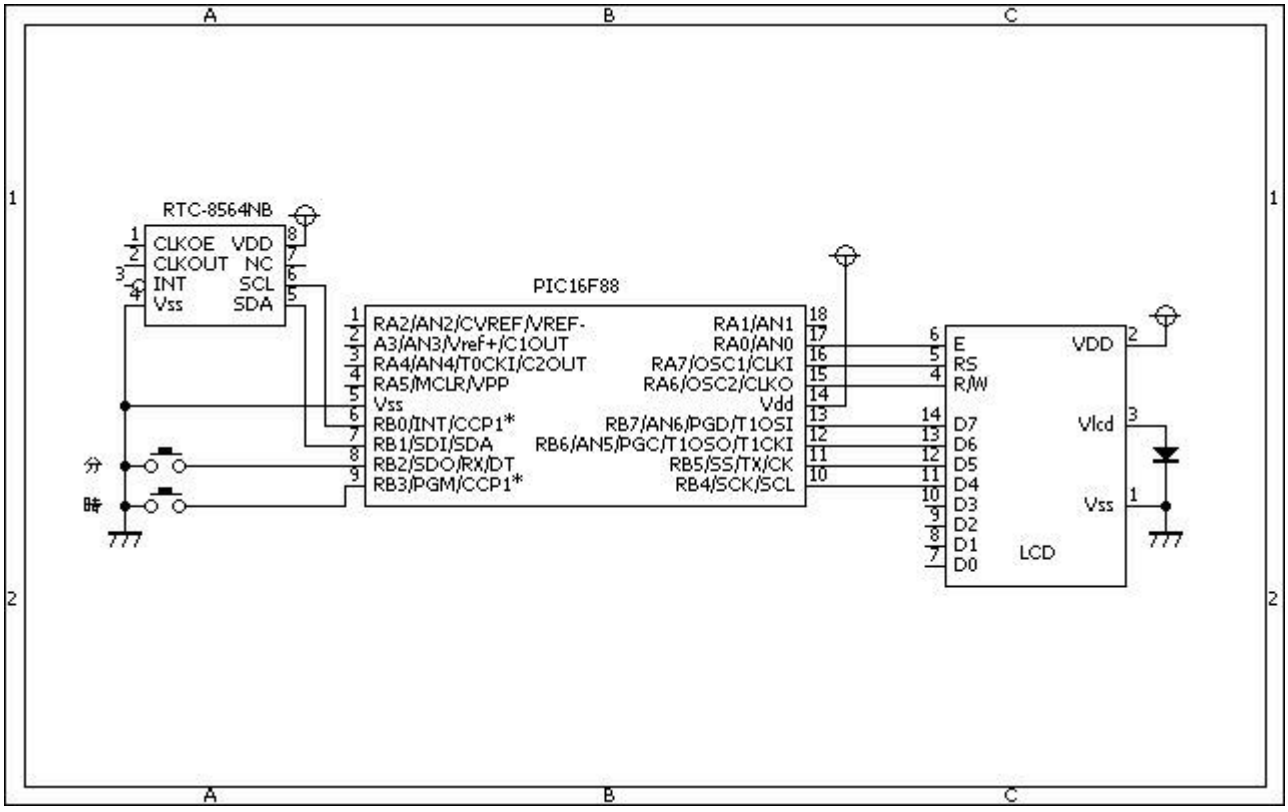
アドレス	レジスタ名	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	Control1	TEST	0	STOP	0	TEST	0	0	0
01	Control2	0	0	0	TI / TP	AF	TF	AIE	TIE
02	Seconds	VL	S40	S20	S10	S8	S4	S2	S1
03	Minutes	※	min40	min20	min10	min8	min4	min2	min1
04	Hours	※	※	h20	h10	h8	h4	h2	h1
05	Days	※	※	d20	d10	d8	d4	d2	d1
06	Day of Week	※	※	※	※	※	W4	W2	W1
07	Months / Century	C	※	※	Month10	Month8	Month4	Month2	Month1
08	Years	Year80	Year40	Year20	Year10	Year8	Year4	Year2	Year1
09	Minutes Alarm	AE	A-Min40	A-min20	A-min10	A-min8	A-min4	A-min2	A-min1
0A	Hour Alarm	AE	※	A-Hr20	A-Hr10	A-Hr8	A-Hr4	A-Hr2	A-Hr1
0B	Day Alarm	AE	※	A-d20	A-d10	A-d8	A-d4	A-d2	A-d1
0C	Weekday Alarm	AE	※	※	※	※	A-W4	A-W2	A-W1
0D	CLKOUT frequency	FE	※	※	※	※	※	FD1	FD0
0E	Timer control	TE	※	※	※	※	※	TD1	TD0
0F	Timer	128	64	32	16	8	4	2	1

今回は、機能を最小限に絞り込みました。 <時刻の設定>

- プッシュスイッチ(時)が押下されると、「時」データをインクリメント(+1)し、そのデータを、レジスタテーブル
の04に書き込みます。
- プッシュスイッチ(分)が押下されると、「分」データをインクリメント(+1)し、そのデータを、レジスタテーブル
の03に書き込みます。

<時刻の読出> 時分秒のデータを、レジスタテーブルの02,03,04より順次読み出しLCDに表示します。

回路図



ソースコード

I2Cインタフェースは、今後も使用する機会があると思いますので、汎用性を考えて以下のような関数を用意しました。

- i2c_Start()
- i2c_Stop()
- i2c_Ack()
- i2c_Nack()
- i2c_SendByte(unsigned char data)
- i2c_RecvByte()

watch.c

```
//*****************************************************************************
*

#define   i2c_SCL            PORTB.F0
#define   i2c_SDA            PORTB.F1
#define   i2c_SDA_TRIS    TRISB.F1

#define   sw_min            PORTB.F2
#define   sw_hou            PORTB.F3

//*****************************************************************************
*

void   i2c_Start()
```

```
{
    if (i2c_SDA_TRIS != 0)
        i2c_SDA_TRIS = 0;
    i2c_SDA = 1;
    i2c_SCL = 0;
    i2c_SCL = 1;
    i2c_SDA = 0;
    i2c_SCL = 0;
}

void   i2c_Stop()
{
    if (i2c_SDA_TRIS != 0)
        i2c_SDA_TRIS = 0;
    i2c_SDA = 0;
    i2c_SCL = 0;
    i2c_SCL = 1;
    i2c_SDA = 1;
    i2c_SCL = 0;
}

void   i2c_Ack()
{
    if (i2c_SDA_TRIS != 0)
        i2c_SDA_TRIS = 0;
    i2c_SDA = 0;
    i2c_SCL = 1;
    i2c_SCL = 0;
}

void   i2c_Nack()
{
    if (i2c_SDA_TRIS != 0)
        i2c_SDA_TRIS = 0;
    i2c_SDA = 1;
    i2c_SCL = 1;
    i2c_SCL = 0;
}

void   i2c_SendByte(unsigned char data)
{
    unsigned char cnt;
    if (i2c_SDA_TRIS != 0)
        i2c_SDA_TRIS = 0;
    for (cnt = 0; cnt < 8; cnt++) {
        if ((data & 0b10000000) == 0)
            i2c_SDA = 0;
        else
            i2c_SDA = 1;
        i2c_SCL = 1;
```

```
        i2c_SCL = 0;
        data = data << 1;
    }
    i2c_SDA_TRIS = 1;
    i2c_SCL = 1;
    i2c_SCL = 0;
}

unsigned   char    i2c_RecvByte()
{
    unsigned char cnt, data;
    data = 0x00;
    if (i2c_SDA_TRIS == 0)
        i2c_SDA_TRIS = 1;
    for (cnt = 0; cnt < 8; cnt++) {
        i2c_SCL = 1;
        if (i2c_SDA == 1)
            data.F0 = 1;
        else
            data.F0 = 0;
        i2c_SCL = 0;
        if (cnt < 7)
            data = data << 1;
    }
    return(data);
}

//*********************************************************************
*

void   main()
{
    unsigned    short    sec, min, hou, temp, cnt;
    char    buf[10];
    //
    OSCCON = 0b01110000;     //クロックは8Mhz
    CMCON = 0b00000111;        //コンパレータは使用しない。
    ANSEL = 0b00000000;        // A/D変換は使用しない。
    TRISA = 0b00111110;
    TRISB = 0b00001100;
    OPTION_REG.F7 = 0;
    //
    Lcd_Custom_Config(&PORTB, 7, 6, 5, 4, &PORTA, 6, 7, 0);
    TRISA = 0b00111110;
    TRISB = 0b00001100;
    Lcd_Custom_Cmd(LCD_CLEAR);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Out(2, 1, "      watch R1.0");
    //
    Delay_ms(1000);
    i2c_Start();
```

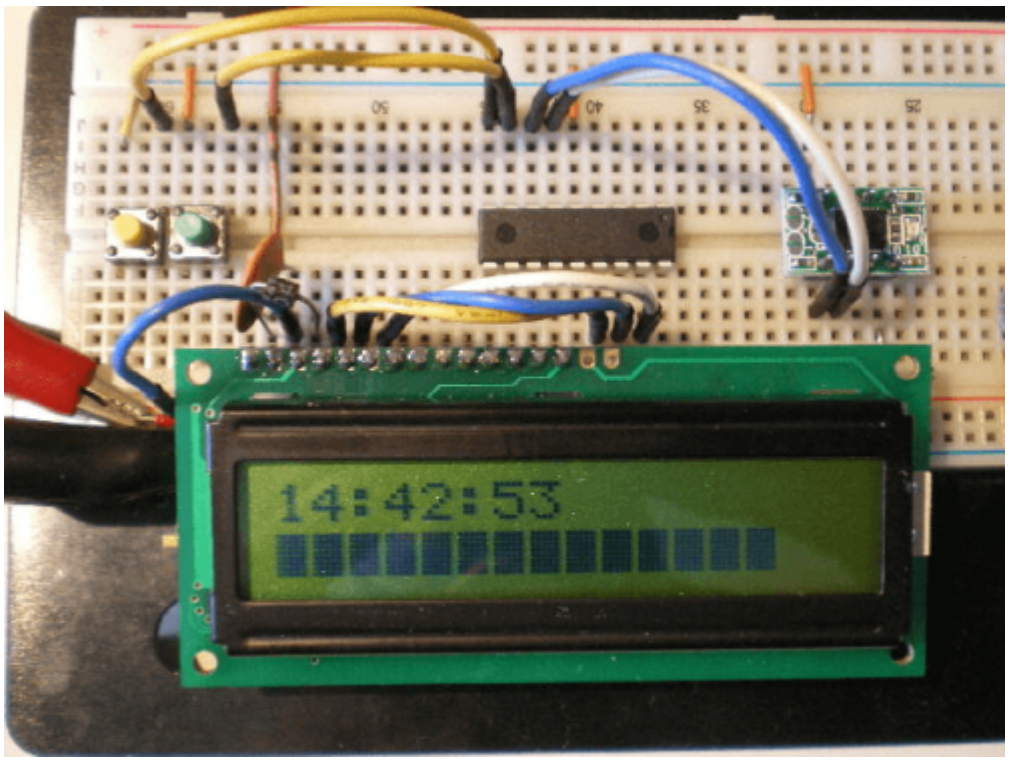
```
    i2c_SendByte(0b10100010);
    i2c_SendByte(0x02);
    i2c_SendByte(0b00000000);
    i2c_SendByte(0b00000000);
    i2c_SendByte(0b00010010);
    i2c_Stop();
    //
    while (1) {
        if (sw_min == 0)    //分の設定
        {
            while (sw_min == 0)
                Delay_ms(10);
            min &= 0x7F;
            temp = ((min >> 4) & 0x0F) * 10;
            min = temp + (min & 0x0F);
            min++;
            min = (min >= 60) ? 0 : min;
            temp = (min / 10) * 16;
            temp = temp + (min - ((min / 10) * 10));
            i2c_Start();
            i2c_SendByte(0b10100010);
            i2c_SendByte(0x03);
            i2c_SendByte(temp);
            i2c_Stop();
        }
        if (sw_hou == 0)    //時の設定
        {
            while (sw_hou == 0)
                Delay_ms(10);
            hou &= 0x3F;
            temp = ((hou >> 4) & 0x0F) * 10;
            hou = temp + (hou & 0x0F);
            hou++;
            hou = (hou >= 24) ? 0 : hou;
            temp = (hou / 10) * 16;
            temp = temp + (hou - ((hou / 10) * 10));
            i2c_Start();
            i2c_SendByte(0b10100010);
            i2c_SendByte(0x04);
            i2c_SendByte(temp);
            i2c_Stop();
        }
        //  時分秒の読み出し
        i2c_Start();
        i2c_SendByte(0b10100010);
        i2c_SendByte(0x02);
        i2c_Start();
        i2c_SendByte(0b10100011);
        sec = i2c_RecvByte();
        i2c_Ack();
```

```
        min = i2c_RecvByte();
        i2c_Ack();
        hou = i2c_RecvByte();
        i2c_Nack();
        i2c_Stop();
        //時の表示
        hou &= 0x3F;
        temp = ((hou >> 4) & 0x0F) * 10;
        temp += (hou & 0x0F);
        ByteToStr(temp, buf);
        buf[1] = (buf[1] == ' ') ? '0' : buf[1];
        Lcd_Custom_Out(1, 1, &buf[1]);
        //分の表示
        min &= 0x7F;
        temp = ((min >> 4) & 0x0F) * 10;
        temp += (min & 0x0F);
        ByteToStr(temp, buf);
        buf[0] = ':';
        buf[1] = (buf[1] == ' ') ? '0' : buf[1];
        Lcd_Custom_Out(1, 3, buf);
        //秒の表示
        sec &= 0x7F;
        temp = ((sec >> 4) & 0x0F) * 10;
        temp += (sec & 0x0F);
        ByteToStr(temp, buf);
        buf[0] = ':';
        buf[1] = (buf[1] == ' ') ? '0' : buf[1];
        Lcd_Custom_Out(1, 6, buf);
        //
        for (cnt = 0; cnt < 16; cnt++) {
            if ((temp / 4) >= cnt)
                Lcd_Custom_Chr(2, cnt + 1, 0xFF);
            else
                Lcd_Custom_Chr(2, cnt + 1, ' ');
        }
    }
}

//*********************************************************************
*
```

動作確認

いつものようにブレッドボードで確認しました。 左上の黄色と緑色のプッシュSWで時分を設定します LCD表示の下のバーは、秒の変化を表示したものです。



如何ですか、これ以上簡略化が出来ない程にシンプルにしました。