

RTC（リアルタイムクロック） DS1305

使用例解説書

第1章 RTC（リアルタイムクロック）とは

RTC は、マイコンに時計およびカレンダーの機能を付加するものです。RTC はバッテリーバックアップされており、マイコンの電源が切られた場合もバッテリーにより動作し、時を刻み続けます。したがってマイコンはいつでも現在の日付、時刻を知ることができるようになります。

RTC はマイコンに内蔵されている場合もありますが、ほとんどのマイコンは内蔵されていません。そこで外部に RTC を増設する必要があります。

第2章 RTC-IC DS1305

RTC 機能の IC は各社から出されていますが、ここでは DALLAS 社の DS1305 を紹介します。16P-DIP の小さな IC ですが、高機能で RTC に要求されるほとんどの機能を持っています。また、バッテリーバックアップされた RAM を 96 バイト持っていて、ちょっとしたデータの保存に使うこともできます。

特徴

96 バイトのバッテリーバックアップ RAM を内蔵

2 つの時刻をアラームとして設定可能、アラームにより CPU へ割り込み要求が可能

シリアルインターフェース。最小 3 本の線で CPU とインターフェースが可能

電源はバッテリー電源の他、2 系統を保有

2.1 DS1305 の型番とパッケージの関係

型番	パッケージ
DS1305	16-Pin DIP(300mil)
DS1305N	16-Pin DIP(Industrial)
DS1305E	20-Pin TSSOP(173mil)
DS1305EN	20-Pin TSSOP(Industrial)

2.2 PIN 機能

番号	名称	機能
1	Vcc2	第 2 電源入力 (出力) 必要ない場合は GND に落とす
2	Vbat	バッテリー電源入力 (3V)
3	X1	32.768KHz クリスタル接続
4	X2	32.768KHz クリスタル接続
5	N.C.	無接続
6	_INT0	アラームによる割り込み信号出力
7	_INT1	アラームによる割り込み信号出力
8	GND	
9	SERMODE	シリアルインターフェースの選択入力
10	CE	チップイネーブル入力
11	SCLK	シリアルクロック入力
12	SDI	シリアルデータ入力
13	SDO	シリアルデータ出力
14	Vccif	インターフェース電圧入力
15	_PF	パワーフェール信号出力
16	Vcc1	第 1 電源入力

解説

Vcc1 第 1 電源を入力します。

Vcc2 第 2 電源を入力します。第 2 電源は充電可能な電池などの電源にします。必要ない場合は GND に落とします。

Vbat 3V の電池電源を入力します。

Vccif CPU インターフェースの論理電圧を入力します。たとえば CPU インターフェースが 3V の場合は、3V を、5V の場合は 5V を入力します。

SERMODE SERMODE=0 のとき、シリアル送受信を 1 本の線で行います。この場合は SDI と SDO をつなぎます。SERMODE=1 のとき、シリアル送受信を SDI と SDO の 2 本の線で行います。(モトローラ SPI)

_PF パワーフェール 第 1 電源が、第 2 電源またはバッテリー電源より下がった場合に LOW 信号が出力されます。

第3章 サンプルの回路

今回のサンプルでは以下のような構成にしました

インターフェースは送受信を1本の線で行う SERMODE=0 SDI と SDO をショート

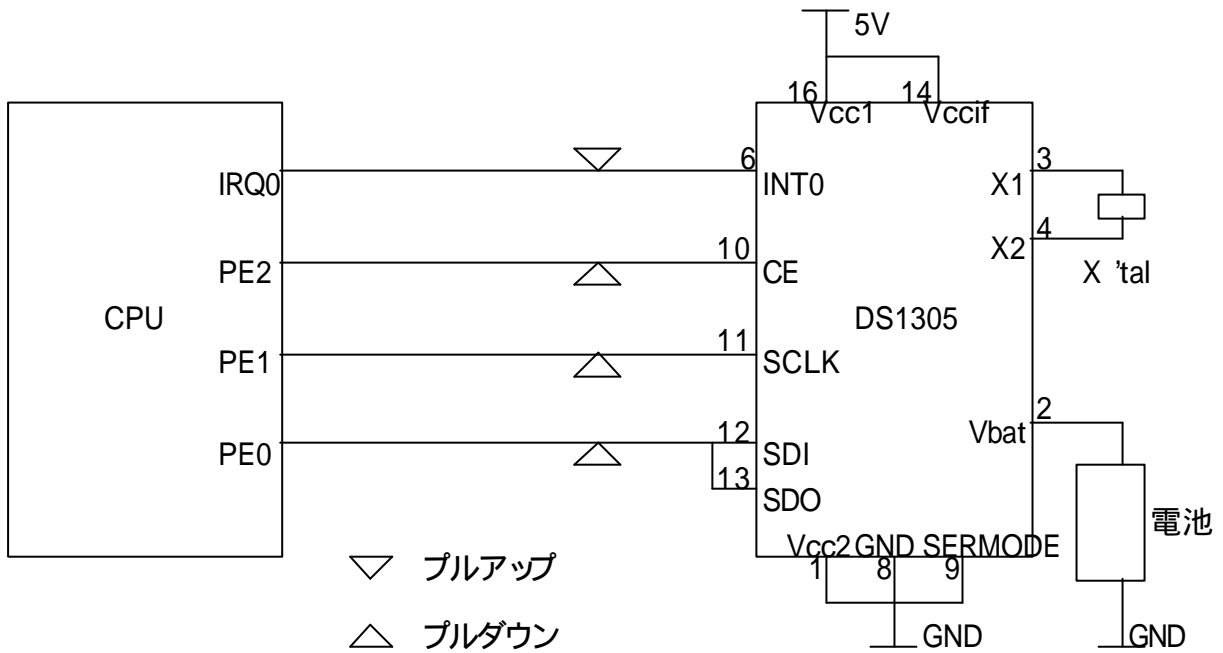
第2電源は使わない Vcc2=0

アラームによる割り込みは1本だけ _INT1は無接続

パワーフェール信号は必要なし _PFは無接続

CPUとのインターフェースにはポートEを使う

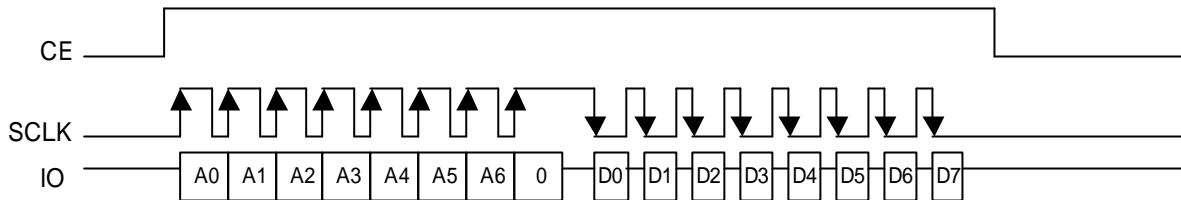
以上を考慮すると回路図は以下ようになります。



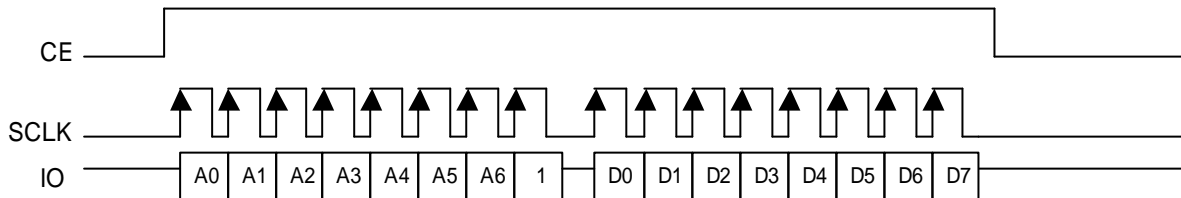
第4章 インターフェース

今回のサンプルでは、CPU とのインターフェースに 3 線式(CE、SCLK、I/O)を採用しました。その場合のロジックは以下ようになります。

1 バイト読み込み



1 バイト書き込み



注意 IO=SDI/SDO

読み込みの場合も書き込みの場合も、まず最初に 8 ビットのアドレスを書き込みます。アドレスの最上位ビットが 0 の場合は、読み込みであり、最上位ビットが 1 の場合は書き込みであることに注意して下さい。

したがって、書き込み時はアドレスの最上位ビットを 1 にセットしてから書き込みを行います。

4.1 書き込み関数

アドレスおよびデータを書き込むプログラム例を示します。

書き込む動作は、IO 信号を確定させてからクロックを立ち上げます。

```
/******  
**** 書き込みの基本関数 ****  
**** 引数 data 8ビットデータ ****  
**** 戻り値 なし ****  
*****/  
static void RtcWriteBios(unsigned char data)  
{  
    int i;  
  
    DATA_OUT;  
    for (i = 0; i < 8; i++) {  
        SDI_OFF;  
        if (data & 1) {  
            wait();  
            SDI_ON;  
        }  
        wait();  
        SCLK_OFF;  
        wait();  
        SCLK_ON;  
        wait();  
        data >>= 1;  
    }  
}
```

4.2 読み込み関数

データを読み込むプログラムを示します。

読み込み動作は、クロックを立ち下げてから、データを読み込みます。

```
/******  
**** 読み込み基本関数 ****  
**** 引数      なし ****  
**** 戻り値    読み込んだ 8 ビットデータ ****  
*****/  
static unsigned char RtcReadBios(void)  
{  
    int i;  
    unsigned char data;  
  
    data = 0;  
    DATA_IN;  
    for (i = 0; i < 8; i++) {  
        SCLK_ON;  
        wait();  
        SCLK_OFF;  
        wait();  
        data |= (SDO<<i);  
        wait();  
    }  
    return data;  
}
```

第5章 DS1305 の使い方

5.1 アドレスマップ

アドレス	B7	B6	B5	B4	B3	B2	B1	B0	範囲
H'00	0	10-SEC			SEC				00-59
H'01	0	10-MIN			MIN				00-59
H'02	0	12	P	10-HR	HOURS			01-12+P/A	
			A						
		24	10-HR	00-23					
H'03	0	0	0	0	DAY			1-7	
H'04	0	0	10-DATE		DATE			1-31	
H'05	0	0	10-MONTH		MONTH			01-12	
H'06	10-YEAR				YEAR			00-99	
ALARM0									
H'07	M	10-SEC			SEC				00-59
H'08	M	10-MIN			MIN				00-59
H'09	M	12	P	10-HR	HOURS			01-12+P/A	
			A						
		24	10-HR	00-23					
H'0A	M	0	0	0	DAY			1-7	
ALARM1									
H'0B	M	10-SEC			SEC				00-59
H'0C	M	10-MIN			MIN				00-59
H'0D	M	12	P	10-HR	HOURS			01-12+P/A	
			A						
		24	10-HR	00-23					
H'0E	M	0	0	0	DAY			1-7	
H'0F	CONTROL REGISTER								
H'10	STATUS REGISTER								
H'11	TRICKLE CHARGER REGISTER								
H'12-H'1F	予約								
H'20-H'7F	96 バイトユーザ RAM								

解説

SEC 秒です。10の桁をビット6~4で、1の桁をビット3~0で表します。範囲は00~59です。

MIN 分です。10の桁をビット6~4で、1の桁をビット3~0で表します。範囲は00~59です。

HOURS 時間です。12時間制と24時間制があります。12時間制の場合、ビット6に1をセットすると12時間制になります。PMの場合はビット5が1、AMの場合はビット5が0です。10の桁をビット4で、1の

桁をビット 3~0 で表します。24 時間制の場合、ビット 6 を 0 にすると 24 時間制になります。10 の桁をビット 5~4 で、1 の桁をビット 3~0 で表します。範囲は 12 時間制の場合、01~12、24 時間制の場合、00~23 です。

DAY 曜日です。曜日は 1~7 で表します。1 が何曜日になるかは任意です。つまりユーザが自由に決められます。

DATE 日です。10 の桁をビット 5~4 で、1 の桁をビット 3~0 で表します。範囲は 01~31 です。

MONTH 月です。10 の桁をビット 5~4 で、1 の桁をビット 3~0 で表します。範囲は 01~12 です。

YEAR 年です。西暦で表した下 2 桁の年です。10 の桁をビット 7~4 で、1 の桁をビット 3~0 で表します。範囲は 00~99 です。

5.2 コントロールレジスタ

B7	B6	B5	B4	B3	B2	B1	B0
_EOSC	WP	0	0	0	INTCN	AIE1	AIE0

_EOSC 0 にすると発振器が動作します。これにより時をカウントします。

WP ライトプロテクトです。1 にセットすると、DS1305 のすべてのレジスタ、ビットを書き換えることができなくなります。コントロールレジスタの他のビットも書き換えできません。

INTCN

INTCN=1 の場合、現在の時刻とアラーム 0 の時刻が一致した場合、_INT0 が発生します。また、現在の時刻とアラーム 1 の時刻が一致した場合、_INT1 が発生します。

INTCN=0 の場合、現在の時刻がアラーム 0 またはアラーム 1 の時刻と一致した場合に _INT0 が発生します。

AIE1 1 にセットすると INTCN1=1 の場合、_INT1 の発生を許可します。INTCN1=0 の場合、_INT0 の発生を許可します。

AIE0 1 にセットすると _INT0 の発生を許可します。

コントロールレジスタは最初、WP=1 の状態にあります。したがってコントロールレジスタを書き換えるには WP=0 にしてから、書き換える必要があります。

5.3 ドキュメントの入手先

DS1305 について、さらに詳しい情報は下記のドキュメントをご覧ください。

<http://pdfserv.maxim-ic.com/en/ds/DS1305.pdf>

第6章 サンプルプログラムについて

サンプルは H8 用と SH 用を用意しています。H8 用は H8S2633 のポート E を、SH 用は SH7047 のポート E を使った例ですが、ポートの定義を書き換えることによって、他の CPU や他のポートを使うことも可能です。

6.1 インストール

解凍後、setup.exe を実行して下さい。ファイルは YellowIDE6 のフォルダの下に次のようなフォルダが作成されコピーされます。

¥YellowIDE6

¥Application

¥RTC

¥H8S2633	H8 用プロジェクト
¥SH7047	SH 用プロジェクト

サンプルはプロジェクトとして提供していますので、YellowIDE6 以上を使って、プロジェクトファイル (*.YIP)を開いて下さい。

6.2 ファイル構成

サンプルは以下のファイルより構成されています。

TEST.C	テストプログラム
RTC-DS1305.C	DS1305 アクセス関数ライブラリ
DS1305.H	上記ヘッダファイル

6.3 DS1305 ライブラリ関数一覧

基本関数	
RtcPortInit	DS1305 と接続されたポートを初期化します
RtcWrite	DS1305 へ 1 バイトの書き込みをします
RtcRead	DS1305 から 1 バイトを読み込みます
RtcWriteProtectOff	ライトプロテクトを解除します
RtcWriteProtectOn	ライトプロテクトを有効にします
RtcOscOn	発振器を動作させます
RtcOscOff	発振器を停止させます
時刻設定、読み込み関数	
RtcSetTime12	12 時間制の時刻を設定します
RtcGetTime12	12 時間制の時刻を読み出します
RtcPrintTime12	12 時間制の時刻を表示します
RtcSetTime24	24 時間制の時刻を設定します
RtcGetTime24	24 時間制の時刻を読み出します
RtcPrintTime24	24 時間制の時刻を表示します
RtcSetTimeAnsi	ANSI 時間制の時刻を設定します
RtcGetTimeAnsi	ANSI 時間制の時刻を読み出します
アラーム設定、読み込み関数	
RtcSetAlarm12	12 時間制の時刻でアラームを設定します
RtcGetAlarm12	12 時間制の時刻で設定されたアラームを読み出します
RtcSetAlarm24	24 時間制の時刻でアラームを設定します
RtcGetAlarm24	24 時間制の時刻で設定されたアラームを読み出します
RtcSetAlarmAnsi	ANSI 時間制の時刻でアラームを設定します
RtcGetAlarmAnsi	ANSI 時間制の時刻で設定されたアラームを読み出します
RtcAlarmStart	アラームを許可します
RtcAlarmStop	アラームを停止します
デバッグ関数	
RtcPrintAllRegister	DS1305 のレジスタをすべて表示します

6.4 時刻を格納する構造体

このサンプルでは、時刻を格納する構造体として次の2つを定義しています。

```
/******  
**** 12 時間制の構造体 ****  
*****/  
typedef struct _Time12 {  
    unsigned char  Sec;    // 00-59  
    unsigned char  Min;    // 00-59  
    unsigned char  AmPm;  // AM:0 PM:1  
    unsigned char  Hour;   // 01-12  
    unsigned char  Day;    // 1-7  
    unsigned char  Date;   // 1-31  
    unsigned char  Month;  // 01-12  
    unsigned char  Year;   // 00-99  
} Time12;
```

```
/******  
**** 24 時間制の構造体 ****  
*****/  
typedef struct _Time24 {  
    unsigned char  Sec;    // 00-59  
    unsigned char  Min;    // 00-59  
    unsigned char  Hour;   // 00-23  
    unsigned char  Day;    // 1-7  
    unsigned char  Date;   // 1-31  
    unsigned char  Month;  // 01-12  
    unsigned char  Year;   // 00-99  
} Time24;
```

また、上記の構造体以外で、ANSI 規格で定義された struct tm 構造体も使用できます。この構造体については市販の参考書などをご覧ください。

6.5 ライブラリの移植

このライブラリは H8S2633 または SH7047 のポート E を使った例です。他の CPU、他のポートを使って DS1305 とインターフェースする場合は修正が必要です。

修正場所は、RTC-DS1305.C の以下の部分です。

```
/*=====
          CPU 依存部分   ここから
=====*/
//H8S2633
#define PEDDR (*(volatile unsigned char *)0xFFFE3D))
#define PEDR  (*(volatile unsigned char *)0xFFFF0D))
#define PORTE (*(volatile unsigned char *)0xFFFFBD))
#define PEPCR (*(volatile unsigned char *)0xFFFE44)) //プルアップコントロール

#define CE_ON      PEDR |= 0x04      //チップセレクト
#define CE_OFF    PEDR &= ~0x04
#define SCLK_ON   PEDR |= 0x02      //シリアルクロック
#define SCLK_OFF  PEDR &= ~0x02
#define SDI_ON    PEDR |= 0x01      //データイン
#define SDI_OFF   PEDR &= ~0x01
#define SDO       (PORTE & 0x01)    //データアウト
#define DATA_IN  PEDDR = 0x06      //入力設定
#define DATA_OUT PEDDR = 0x07      //出力設定

/*****
*****   ポートの初期化           *****
*****/
void RtcPortInit(void)
{
    PEDDR = 0x06;      //PE2:Out PE1:Out PE0:In
    PEDR  = 0x00;      //0 出力
}

//速い CPU の場合は wait を調節
static void wait() {}
/*=====
          CPU 依存部分終わり   ここまで
=====*/
```

RtcPortInit

マイコンのポートの初期化

書式

```
void RtcPortInit(void);
```

戻り値 なし

引数 なし

動作

DS1305 とインターフェースされたマイコンのポートを初期化します。この関数は実際にはユーザが作成します。ポートの入出力方向の設定や、最初の 0 出力などを行います。

RtcWritet

1 バイト書き込み

書式

```
void RtcWrite(unsigned char address, unsigned char data);
```

戻り値 なし

引数 address アドレス
 data 書き込むデータ

動作

DS1305 の番地 address にデータ data を書き込みます。address は 0 から H'7F の範囲です。DS1305 は書き込むときアドレスの最上位ビットを 1 にセットしますが、この処理はこの関数の中で行われているため、ユーザが address の最上位ビットを 1 にセットする必要はありません。

RtcRead

1 バイト読み込み

書式

unsigned char RtcRead(unsigned char address);

戻り値	読み込んだデータ	
引数	address	アドレス

動作

DS1305 の番地 address からデータを読み込みます。address は 0 から H'7F の範囲です。

RtcWriteProtectOff

ライトプロテクト解除

書式

void RtcWriteProtectOff(void);

戻り値	なし
引数	なし

動作

DS1305 のライトプロテクトを解除します。ライトプロテクトを解除しないと DS1305 のすべてのレジスタ、ビット、RAM に対して書き込みはできません。

RtcWriteProtectOn

ライトプロテクト許可

書式

void RtcWriteProtectOn(void);

戻り値	なし
引数	なし

動作

DS1305 のライトプロテクトを許可します。

RtcOscOn

発振器の動作開始

書式

```
void RtcOscOn();
```

戻り値 なし

引数 なし

動作

DS1305 の発振器を動作させます。これにより時刻がカウントされます。

RtcOscOff

発振器の動作停止

書式

```
void RtcOscOff();
```

戻り値 なし

引数 なし

動作

DS1305 の発振器を停止させます。

RtcSetTime12

12 時間制時刻の設定

書式

```
void RtcSetTime12(Time12 *t);
```

戻り値 なし

引数 t Time12 構造体へのポインタ

動作

Time12 構造体に格納された時刻を現在の時刻として DS1305 に設定します。

RtcGetTime12

12 時間制時刻の取得

書式

```
void RtcGetTime12(Time12 *t);
```

戻り値 なし
引数 t Time12 構造体へのポインタ

動作

DS1305 から現在の時刻を取得して Time12 構造体に格納します。

RtcPrintTime12

12 時間制時刻の表示

書式

```
void RtcPrintTime12(Time12 *t);
```

戻り値 なし
引数 t Time12 構造体へのポインタ

動作

Time12 構造体に格納された時刻を表示します。

RtcSetTime24

24 時間制時刻の設定

書式

```
void RtcSetTime24(Time24 *t);
```

戻り値 なし
引数 t Time24 構造体へのポインタ

動作

Time24 構造体に格納された時刻を現在の時刻として DS1305 に設定します。

RtcGetTime24

24 時間制時刻の取得

書式

```
void RtcGetTime24(Time24 *t);
```

戻り値 なし
引数 t Time24 構造体へのポインタ

動作

DS1305 から現在の時刻を取得して Time24 構造体に格納します。

RtcPrintTime24

24 時間制時刻の表示

書式

```
void RtcPrintTime24(Time24 *t);
```

戻り値 なし
引数 t Time24 構造体へのポインタ

動作

Time24 構造体に格納された時刻を表示します。

RtcSetTimeAnsi

ANSI 時間制時刻の設定

書式

```
void RtcSetTimeAnsi(struct tm *t);
```

戻り値 なし
引数 t struct tm 構造体へのポインタ

動作

struct tm 構造体に格納された時刻を現在の時刻として DS1305 に設定します。

RtcGetTimeAnsi

Ansi 時間制時刻の取得

書式

```
void RtcGetTimeAnsi(struct tm *t);
```

戻り値 なし
引数 t struct tm 構造体へのポインタ

動作

DS1305 から現在の時刻を取得して struct tm 構造体に格納します。

RtcSetAlarm12

12 時間制アラーム時刻の設定

書式

```
void RtcSetAlarm12(Time12 *t, int ch);
```

戻り値 なし
引数 t Time12 構造体へのポインタ
 ch ch=0 アラーム 0 ch=1 アラーム 1

動作

Time12 構造体に格納された時刻をアラームの時刻として DS1305 に設定します。

RtcGetAlarm12

12 時間制アラーム時刻の取得

書式

```
void RtcGetAlarm12(Time12 *t, int ch);
```

戻り値 なし
引数 t Time12 構造体へのポインタ
 ch ch=0 アラーム 0 ch=1 アラーム 1

動作

DS1305 からアラームとして設定された時刻を取得して Time12 構造体に格納します。

RtcSetAlarm24

24 時間制アラーム時刻の設定

書式

```
void RtcSetAlarm24(Time24 *t, int ch);
```

戻り値	なし
引数	t Time24 構造体へのポインタ
	ch ch=0 アラーム 0 ch=1 アラーム 1

動作

Time24 構造体に格納された時刻をアラームの時刻として DS1305 に設定します。

RtcGetAlarm24

24 時間制アラーム時刻の取得

書式

```
void RtcGetAlarm24(Time24 *t, int ch);
```

戻り値	なし
引数	t Time24 構造体へのポインタ
	ch ch=0 アラーム 0 ch=1 アラーム 1

動作

DS1305 からアラームとして設定された時刻を取得して Time24 構造体に格納します。

RtcSetAlarmAnsi

Ansi 時間制アラーム時刻の設定

書式

```
void RtcSetAlarmAnsi(struct tm *t, int ch);
```

戻り値	なし
引数	t struct tm 構造体へのポインタ
	ch ch=0 アラーム 0 ch=1 アラーム 1

動作

struct tm 構造体に格納された時刻をアラームの時刻として DS1305 に設定します。

RtcGetAlarmAnsi

Ansi 時間制アラーム時刻の取得

書式

```
void RtcGetAlarmAnsi(struct tm *t, int ch);
```

戻り値	なし
引数	t struct tm 構造体へのポインタ
	ch ch=0 アラーム 0 ch=1 アラーム 1

動作

DS1305 からアラームとして設定された時刻を取得して struct tm 構造体に格納します。

RtcAlarmStart

アラームの開始

書式

```
void RtcAlarmStart(int mode);
```

戻り値	なし
引数	mode mode=0 現在の時刻がアラーム 0 と一致した場合に INT0 を発生
	mode=1 現在の時刻がアラーム 1 と一致した場合に INT1 を発生
	mode=2 現在の時刻がアラーム 0 またはアラーム 1 と一致した場合に
	INT0 を発生
	mode=3 mode=0 と mode=1 の両方を設定

動作

アラームを開始します。つまり設定された時刻になると DS1305 から割り込み信号が発生します。

RtcAlarmStop

アラームの停止

書式

```
void RtcAlarmStop(void);
```

戻り値 なし

引数 なし

動作

アラームを停止します。

RtcPrintAllRegister

すべてのレジスタの表示

書式

```
void RtcPrintAllRegister(void);
```

戻り値 なし

引数 なし

動作

DS1305 のすべてのレジスタを表示します。

6.7 テストプログラムの解説

サンプルのテストプログラムでは次のようなことをしています。

- 1 ポートの初期化、ライトプロテクトの解除、発振器の動作開始
- 2 ユーザ RAM への書き込みテスト

アドレス 0x20 ~ 0x7F まではユーザが自由に使えるバッテリバックされた RAM です。ここに 0 から 0x5F までの数値を書き込んだ後、読み込み表示します。

- 3 DS1305 に時刻を設定します。
- 4 fgetc(stdin)でキー入力待ちになります。リターンキーの入力でプログラムを再開します。
- 5 現在の時刻を表示します。3 で設定された時刻より少し進んでいるのが分かると思います。
- 6 アラーム用の割り込み IRQ の初期設定をします。
- 7 アラーム時刻を設定した後で、アラームをスタートさせます。
- 8 ループに入ります。
- 9 アラームにより IRQ0 割り込みが発生して、AlarmOn が 1 にセットされます。
- 10 9 によりループを抜けます。
- 11 プログラム終了です。

第7章 DS1305 ICの販売について

弊社では、DS1305 チップの入手が困難な方のために、DS1305 チップと 32.768KHz のクリスタルをセットで販売をしています。概要は以下のとおりです。

内容

IC DS1305

クリスタル 32.768KHz

価格

1 セット ¥1,200 (税込) この価格には送料も含まれます。

制限

お一人様、2 セットまででお願いいたします。

お支払い方法

代金は現金か、切手でお願いいたします。

切手の場合は必ず 80 円切手×15 枚または 120 円切手×10 枚(1 セット)でお願いいたします。

お申し込み方法

下記申込書にご記入の上、現金または切手を添えて、封筒(現金の場合は現金書留)で郵送して下さい。

お申し込み書			
商品名		個数	
DS1305 & クリスタル(32.768K) セット		1セット 2セット	
合計金額		1セット ¥1,200	2セット ¥2,400
フリガナ			
ご氏名			
ご住所 (送り先)	〒		
会社名			
部署名			
TEL		FAX	