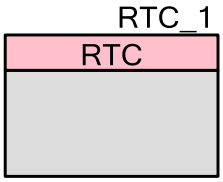
Real-Time Clock (RTC)
1.60



特長

- 様々なアラーム オプション
- 様々なオーバーフロー オプション
- 夏時間（DST：Daylight Savings Time) オプション

概要説明

Real-Time Clock (RTC) のコンポーネントは、システムに正確な時間と日付の情報を提供します。時間と日付は、32.768-kHz の水晶からの1秒に1回の割込みに基づいて毎秒毎に更新されます。クロックの精度は、提供されているクリスタルに準拠し、通常 20 ppm です。

RTC は、秒、分、時間、曜日、月の特定日、年の特定日、月、年を常時追跡します。曜日は日、月、年から自動的に計算されます。夏時間（Daylight savings time）を必要に応じて有効にして、開始日、終了日、プログラム可能なセービング・タイムに対応することができます。開始日および終了日を 3 月 24 日のような絶対値にしたり、5 月の第 2 日曜日といった、相対的な日にちにすることができます。

アラームは、秒、分、時、曜日、月の特定日、年の特定日、月、年に一致した検出を提供します。マスクは、アラームを生成するのに使用される時間と日付情報の組合せを選択します。アラームは、23 分ごとなどのような定期的アラームや、2043 年 9 月 28 日の午前 4 時 52 分などの単一アラームに柔軟に対応します。

ユーザコードスタブは、各主要周期に基づいた定期的なコード実行のために提供されています。タイマーの刻み間隔は、1 秒、1 分、1 時間、1 日、1 週間、1 か月、1 年で提供されます。

RTC コンポーネントの用途

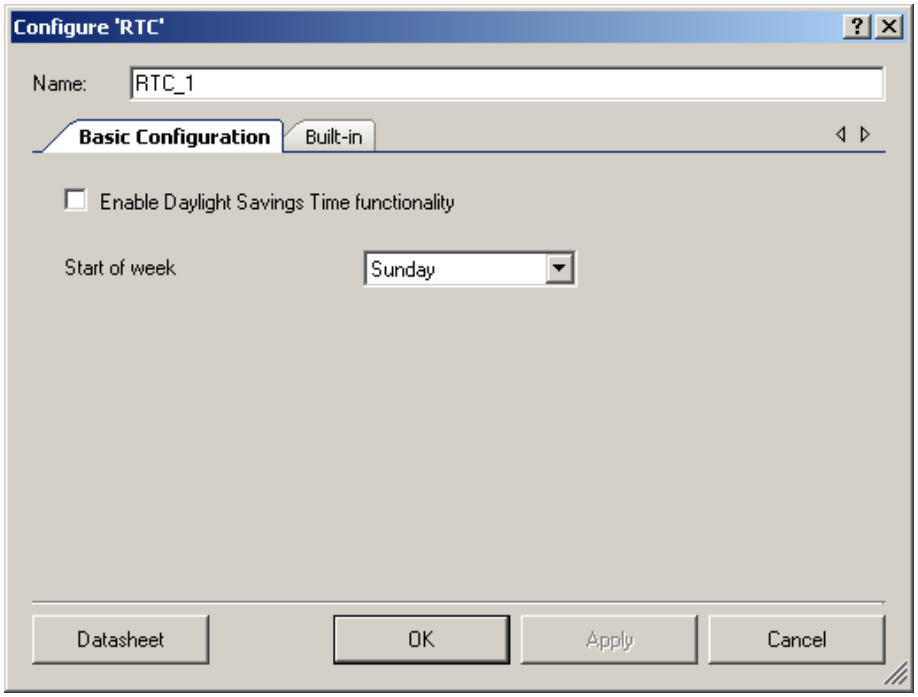
システムが現在の時間や日付を要求する場合に RTC コンポーネントを使用します。現在の時間と日付は必要ないが、1 秒単位での正確なイベントのタイミングが必要な場合にも、RTC を使用することができます。

入出力接続

RTC コンポーネントには、入力または出力接続がありません。

コンポーネント・パラメータ

RTC コンポーネントをデザイン画面上にドラッグして、ダブルクリックして **Configure**(設定) ダイアログを開きます。



RTC コンポーネントには、次のパラメータがあります。

Enable Daylight Savings Time functionality

このパラメータは、RTC コンポーネントで夏時間関数を有効にするかどうかを選択します。デフォルトではクリア(false)されています。

Start of week

Start of week パラメータは、開始曜日を選択します。オプション:**Sunday** (デフォルト)、**Monday**、**Tuesday**、**Wednesday**、**Thursday**、**Friday**、**Saturday**。

クロック選択

外部クリスタル・オシレータから 32.768-kHz クロックを提供する必要があります。このコンポーネントの精度は、接続したクロックソースの精度によって決定されます。設計に組み込み XTAL_32KHZ クロックを接続および設定する方法については、PSoC Creator ヘルプのクロックエディタセクションを参照してください。

リソース

Resolution	デジタルブロック					API メモリ(バイト)		ピン(外部入出力ごと)
	データパス	Macro セル	状態 Registers	制御レジスタ	Counter7	フラッシュ	RAM	
RTC 固定ハードウェア *	該当せず	該当せず	該当せず	該当せず	該当せず	2738	25	該当せず

* Power Management の 1 パルス/1 秒間隔を使用

アプリケーション プログラミング インタフェース

アプリケーション・プログラミング・インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。以下のセクションでは、各関数について詳しく説明します。

デフォルトでは、PSoC Creator は、デザイン画面の最初のコンポーネントのインスタンスに「RTC_1」インスタンス名を割り当てます。コンポーネントのインスタンス名称は、識別子の文法ルールに従って独自の名前に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。読みやすいように、下表では「RTC」というインスタンス名を使用しています。

グローバル変数を読み取りまたは変更する関数の呼び出し中は、コンポーネントの割り込みを無効にする必要があります。

詳しくは、このデータシートの レジスタ セクションを参照してください。

関数	説明
RTC_Start()	RTC コンポーネントを有効にします
RTC_Stop()	RTC コンポーネントの動作を停止します
RTC_EnableInt()	RTC コンポーネントの割り込みを有効にします
RTC_DisableInt()	RTC コンポーネントの割り込みを無効にします。時間と日付の更新は停止します
RTC_WriteTime()	現在の時間と日付として時間と日付の値を書き込みます
RTC_ReadTime()	現在の時間と日付を読み込みます
RTC_WriteSecond()	Sec ソフトウェアレジスタ値を書き込みます
RTC_WriteMinute()	Min(分) ソフトウェアレジスタ値を書き込みます
RTC_WriteHour()	Hour(時) ソフトウェアレジスタ値を書き込みます
RTC_WriteDayOfMonth()	DayOfMonth(日) ソフトウェアレジスタ値を書き込みます
RTC_WriteMonth()	Month(月)フトウェアレジスタ値を書き込みます

関数	説明
RTC_WriteYear()	Year（年）ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmSecond()	秒アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmMinute()	分アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmHour()	時アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmDayOfMonth()	日アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmMonth()	月アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmYear()	年アラーム ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmDayOfWeek()	曜日アラームk ソフトウェアレジスタ値を書き込みます
RTC_WriteAlarmDayOfYear()	Alarm DayOfYear（1/1からの経過日数）ソフトウェアレジスタ値を書き込みます
RTC_ReadSecond()	秒ソフトウェアレジスタ値を読み込みます
RTC_ReadMinute()	分ソフトウェアレジスタ値を読み込みます
RTC_ReadHour()	時ソフトウェアレジスタ値を読み込みます
RTC_ReadDayOfMonth()	日ソフトウェアレジスタ値を読み込みます
RTC_ReadMonth()	月 ソフトウェアレジスタ値を読み込みます
RTC_ReadYear()	年ソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmSecond()	秒アラーム ソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmMinute()	分アラームソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmHour()	時アラームソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmDayOfMonth()	日アラームソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmMonth()	月アラーム ソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmYear()	年アラーム ソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmDayOfWeek()	曜日アラームソフトウェアレジスタ値を読み込みます
RTC_ReadAlarmDayOfYear()	Alarm DayOfYear(1/1からの経過日数）ソフトウェアレジスタ値を読み込みます
RTC_WriteAlarmMask()	アラームマスクソフトウェアレジスタ値を時間/日付エントリにつき1ビットで書き込みます
RTC_WriteIntervalMask()	RTC ISR からどの割り込みハンドラが呼び出されるのかを設定します
RTC_ReadStatus()	ソフトウェアレジスタを読み込みます。DST (DST)、うるう年 (LY)、午前/午後 (AM_PM)、アラーム動作 (AA) のフラグがあります
RTC_WriteDSTMode()	DST Mode（夏時間モード）ソフトウェアレジスタを書き込みます
RTC_WriteDSTStartHour()	DST Start Hour(夏時間開始時刻）ソフトウェアレジスタを書き込みます

関数	説明
RTC_WriteDSTStartDayOfMonth()	DST Start DayOfMonth(夏時間開始日) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStartMonth()	DST Start Month(夏時間開始月) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStartDayOfWeek()	DST Start DayOfWeek(夏時間曜日) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStartWeek()	DST Start Week(夏時間開始週) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStopHour()	DST Stop Hour(夏時間終了時) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStopDayOfMonth()	DST Stop DayOfMonth(夏時間終了日) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStopMonth()	DST Stop Month(夏時間終了月) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStopDayOfWeek()	DST Stop DayOfWeek(夏時間終了日) ソフトウェアレジスタを書き込みます
RTC_WriteDSTStopWeek()	DST Stop Week(夏時間終了週) ソフトウェアレジスタを書き込みます
RTC_WriteDSTOffset()	DST Offset(夏時間オフセット) レジスタを書き込みます
RTC_Init()	初期化を行い、カスタマイザで提供されるデフォルト設定値に再設定します
RTC_Enable()	1秒毎のパルス(OPPS)割込みをイネーブルにし、OPPSイベントでの割込みを生成します

グローバル変数

変数	説明
RTC_initVar	RTC が初期化されたかどうかを示します。変数は、0 に初期化され、最初に RTC_Start() が呼び出されると 1 に設定されます。これにより、RTC_Start() ルーチンの最初の呼び出し後に再初期化しないでコンポーネントを起動できます。 コンポーネントの再初期化が必要な場合は、RTC_Init() 関数を RTC_Start() または RTC_Enable() 関数の前に呼び出します。
RTC_currentTimeDate	現在の時間と日付の値がこの変数に格納されます。
RTC_statusDateTime	この変数には以下のフラグがあります。DST(夏時間)、Leap Year(うるう年)r、AM/PM、Active Alarm のステータス。
RTC_intervalCfgMask	この変数は実行する割り込みスタブを定義するのに使用されます。
RTC_alarmCfgTimeDate	アラーム時間と日付の値がこの変数に格納されます。
RTC_alarmCurStatus	この変数は現在のアクティブ アラーム(秒アラームや分アラーム) を示すのに使用されます。
RTC_alarmCfgMask	この変数はアラームイベントをマスクするのに利用されます。(秒アラームマスク、分アラームマスクなど)。
RTC_dstModeType	この変数は、DST(夏時間) モードタイプの値を格納します。「0」– 絶対、「1」– 相対。
RTC_dstTimeDateStart	DST(夏時間) 開始の時間と日付の値。

変数	説明
RTC_dstTimeDateStop	DST（夏時間）終了の時間と日付の値。
RTC_dstStartStatus	DST（夏時間）開始ステータス。
RTC_dstStopStatus	DST（夏時間）停止ステータス。
RTC_dstOffsetMin	DST（夏時間）オフセット値（分単位）。

void RTC_Start(void)

説明: RTC コンポーネントを有効にします。この関数はカウンタを設定し、割り込みをセットアップし、必要なすべてを計算して、カウントを開始します。

パラメータ: なし

戻り値: なし

副作用: 低消費電力モード (スリープやオリタネートアクティブ) からデバイスを起動しするため、1秒周期のパルス信号（OPPS)と定周期 CTW（訳注：1kHzのILOで動く13ビットカウンタ）信号が有効になり、イネーブルのままになります。

void RTC_Stop(void)

説明: RTC コンポーネントの動作を停止します。

パラメータ: なし

戻り値: なし

副作用: 低消費電力モード (スリープやオリタネートアクティブ)からデバイスをウェイクアップするOPPS と CTW 信号は有効状態のままになります。

void RTC_EnableInt(void)

説明: RTC コンポーネントの割り込みを有効にします。

パラメータ: なし

戻り値: なし

副作用: なし

void RTC_DisableInt(void)

説明: RTC コンポーネントの割り込みを無効にします。時間と日付の更新は停止します。

パラメータ: なし

戻り値: なし

副作用: なし

RTC_TIME_DATE* timeDate RTC_ReadTime(void)

説明: 現在の時間と日付を読み込みます。

パラメータ: なし

戻り値: RTC_TIME_DATE のポインタ..

副作用: なし

void RTC_WriteTime(RTC_TIME_DATE * timeDate)

説明: 現在の時間と日付として時間と日付の値を書き込みます。秒、分、時間、月、月の特定日、年のみを渡します。

パラメータ: timeDate: 新しい時間と日付が保管される RTC_TIME_DATE グローバル構造体へのポインタ

戻り値: なし

副作用: RTC コンポーネントの割り込みは、時間と日付の書き込み中に RTC カウンタがインクリメントしないよう、この関数を呼び出す前にディセーブルし、後でイネーブルします。

void RTC_WriteSecond(uint8 second)

説明: Sec（秒）ソフトウェアレジスタ値を書き込みます。

パラメータ: second: 秒値

戻り値: なし

副作用: なし

void RTC_WriteMinute(uint8 minute)

説明: Min（分）ソフトウェアレジスタ値を書き込みます。

パラメータ: minute: 分値

戻り値: なし

副作用: なし

void RTC_WriteHour(uint8 hour)

説明: Hour（時）ソフトウェアレジスタ値を書き込みます。

パラメータ: hour: 時間値

戻り値: なし

副作用: なし

void RTC_WriteDayOfMonth(uint8 dayOfMonth)

説明: DayOfMonth（日）ソフトウェアレジスタ値を書き込みます。

パラメータ: dayOfMonth: DayOfMonth（日）値

戻り値: なし

副作用: なし

void RTC_WriteMonth(uint8 month)

説明: Month（月）ソフトウェアレジスタ値を書き込みます。

パラメータ: month: 月値

戻り値: なし

副作用: なし

void RTC_WriteYear(uint16 year)

説明: Year（年）ソフトウェアレジスタ値を書き込みます。

パラメータ: year: 年値

戻り値: なし

副作用: なし

void RTC_WriteAlarmSecond(uint8 second)

説明: Alarm Sec（秒アラーム）ソフトウェアレジスタ値を書き込みます。

パラメータ: second: アラーム秒値

戻り値: なし

副作用: なし

void RTC_WriteAlarmMinute(uint8 minute)

説明: Alarm Min（分アラーム）ソフトウェアレジスタ値を書き込みます。

パラメータ: minute: アラーム分値

戻り値: なし

副作用: なし

void RTC_WriteAlarmHour(uint8 hour)

説明: Alarm Hour（時アラーム）ソフトウェアレジスタ値を書き込みます。

パラメータ: hour: アラーム時間値

戻り値: なし

副作用: なし

void RTC_WriteAlarmDayOfMonth(uint8 dayOfMonth)

説明: Alarm DayOfMonth（日アラーム） ソフトウェアレジスタ値を書き込みます。

パラメータ: dayOfMonth: アラーム日値

戻り値: なし

副作用: なし

void RTC_WriteAlarmMonth(uint8 month)

説明: Alarm Month（月アラーム） ソフトウェアのレジスタ値を書き込みます。

パラメータ: month: アラーム月値

戻り値: なし

副作用: なし

void RTC_WriteAlarmYear(uint16 year)

説明: Alarm Year（年アラーム）ソフトウェアレジスタ値を書き込みます。

パラメータ: year: アラーム年値

戻り値: なし

副作用: なし

void RTC_WriteAlarmDayOfWeek(uint8 dayOfWeek)

説明: Alarm DayOfWeek（曜日アラーム）ソフトウェアレジスタ値を書き込みます。

パラメータ: dayOfWeek: アラーム曜日値

戻り値: なし

副作用: なし

void RTC_WriteAlarmDayOfYear(uint16 dayOfYear)

説明: Alarm DayOfYear（1/1からの経過日数） ソフトウェアレジスタ値を書き込みます。

パラメータ: dayOfYear: 1/1からの経過日数アラーム値

戻り値: なし

副作用: なし

uint8 RTC_ReadSecond(void)

説明: Sec（秒）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: なし

副作用: なし

uint8 RTC_ReadMinute(void)

説明: Min（分）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の分の値を返します。

副作用: なし

uint8 RTC_ReadHour(void)

説明: Hour（時）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の時間の値を返します。

副作用: なし

uint8 RTC_ReadDayOfMonth(void)

説明: DayOfMonth（日）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の日にち値を返します。

副作用: なし

uint8 RTC_ReadMonth(void)

説明: Month（月）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の月の値を返します。

副作用: なし

uint16 RTC_ReadYear(void)

説明: Year（年）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の年の値を返します。

副作用: なし

uint8 RTC_ReadAlarmSecond(void)

説明: Alarm Sec（秒アラーム）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の秒のアラーム値を返します。

副作用: なし

uint8 RTC_ReadAlarmMinute(void)

説明: Alarm Min（分アラーム）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の分のアラーム値を返します。

副作用: なし

uint8 RTC_ReadAlarmHour(void)

説明: Alarm Hour（時）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の時間のアラーム値が返された状態を返します。

副作用: なし

uint8 RTC_ReadAlarmDayOfMonth(void)

説明: Alarm DayOfMonth（日アラーム）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の日のアラーム値を返します。

副作用: なし

uint8 RTC_ReadAlarmMonth(void)

説明: Alarm Month（月アラーム）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の月のアラーム値を返します。

副作用: なし

uint16 RTC_ReadAlarmYear(void)

説明: Alarm Year（年アラーム）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の年のアラーム値を返します。

副作用: なし

uint8 RTC_ReadAlarmDayOfWeek(void)

説明: Alarm DayOfWeek（曜日）ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の曜日のアラーム値を返します。

副作用: なし

uint16 RTC_ReadAlarmDayOfYear(void)

説明: Alarm DayOfYear（1/1からの経過日数アラーム） ソフトウェアレジスタ値を読み込みます。

パラメータ: なし

戻り値: 現在の1/1からの経過日数のアラーム値を返します。

副作用: なし

void RTC_WriteAlarmMask(uint8 mask)

説明: Alarm Mask（アラームマスク）ソフトウェアレジスタ値を時間/日付エントリにつき1ビットで書き込みます。すべてのマスクされた時間/日付がアラーム値に一致した場合に、「True」をアラームします。

パラメータ: mask: Alarm Mask（アラームマスク）ソフトウェアレジスタ値。このパラメータについての詳細は、「アラーム マスク レジスタ」セクションを参照してください。

戻り値: なし

副作用: なし

void RTC_WriteIntervalMask(uint8 mask)

説明: RTC ISRからどの定周期ハンドラが呼び出されるのかを設定します。この関数の使用方法については、「割り込みサービスルーチン 」セクションを参照してください。

パラメータ: mask: Interval Mask ソフトウェアレジスタ値。このパラメータについての詳細は、「インターバル マスク レジスタ 」セクションを参照してください。

戻り値: なし

副作用: なし

uint8 RTC_ReadStatus(void)

説明: ソフトウェアレジスタを読み込みます。夏時間 (DST)、うるう年 (LY)、午前/午後 (AM_PM)、アラームアクティブ (AA) のフラグがあります。

パラメータ: なし

戻り値: アクティブなアラームビットクリアと共に、現在のコンポーネントステータス。この戻り値については、「ステータスレジスタ」セクションを参照してください。

副作用: 読み込み後アラーム動作 (AA) フラグはクリアされます。

void RTC_WriteDSTMode(uint8 mode)

説明: DST Mode レジスタは、DST の変更を有効または無効にして、日付モードを絶対または相対日付に設定します。DST 有効な場合にのみ生成されます。

パラメータ: mode: DST Mode ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStartHour(uint8 hour)

説明: DST Start Hour ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST 有効な場合にのみ生成されます。

パラメータ: hour: DST Start Hour ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStartDayOfMonth(uint8 dayOfMonth)

説明: DST Start DayOfMonth（夏時間開始日）ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: dayOfMonth: DST Start DayOfMonth ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStartMonth(uint8 month)

説明: DST Start Month（夏時間開始月）ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST（夏時間）有効な場合にのみ生成されます。

パラメータ: month: DST Start Month ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStartDayOfWeek(uint8 dayOfWeek)

説明: DST Start DayOfWeek（夏時間開始曜日）ソフトウェアレジスタを書き込みます。相対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: dayOfWeek: DST Start DayOfWeek ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStartWeek(uint8 week)

説明: DST Start Week（夏時間開始週）ソフトウェアレジスタを書き込みます。相対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: Week: DST Start Week ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStopHour(uint8 hour)

説明: DST Stop Hour（夏時間終了時）ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: hour: DST Stop Hour ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStopDayOfMonth(uint8 dayOfMonth)

説明: DST Stop DayOfMonth（夏時間終了日）ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST 有効な場合にのみ生成されます。

パラメータ: dayOfMonth: DST Stop DayOfMonth（夏時間終了日）ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStopMonth(uint8 month)

説明: DST Stop Month（夏時間終了月）ソフトウェアレジスタを書き込みます。絶対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: month: DST Stop Month ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStopDayOfWeek(uint8 dayOfWeek)

説明: DST Stop DayOfWeek（夏時間終了曜日）ソフトウェアレジスタを書き込みます。相対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: dayOfWeek: DST Stop DayOfWeek ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTStopWeek(uint8 week)

説明: DST Stop Week（夏時間終了週）ソフトウェアレジスタを書き込みます。相対日付エントリに使用されます。DST（夏時間）が有効な場合にのみ生成されます。

パラメータ: week: DST Stop Week ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_WriteDSTOffset(uint8 offset)

説明: DST Offset(夏時間オフセット)レジスタを書き込みます。時間を 0 から 255 分の範囲内で増加または減少することができます。DST(夏時間)開始で増加し、DST 停止で減少します。DST(夏時間)が有効な場合にのみ生成されます。

パラメータ: offset: DST Offset ソフトウェアレジスタ値

戻り値: なし

副作用: なし

void RTC_Init (void)

説明: カスタマイザの [Configure] (設定) ダイアログの設定に従って、コンポーネントを初期化または復元します。RTC_Start() API が RTC_Init() 関数を呼び出すので、この関数を呼び出す必要はありません。これはコンポーネントの動作を開始する際に推奨される方法です。

パラメータ: なし

戻り値: なし

副作用: 全てのレジスタには、カスタマイザの [Configure] (設定) ダイアログの設定に対応する値が設定されます。

void RTC_Enable(void)

説明: 割り込み、OPPS(1秒周期パルス)、OPPS イベントでの割り込み生成を有効にします。

パラメータ: なし

戻り値: なし

副作用: 低消費電力モード (スリープとオリタネートアクティブ) からデバイスをウェイクアップするためのOPPS 信号とCTW(1ms毎にカウントする13ビットレジスタ) 信号を有効にして、それらをイネーブルのままにしておきます。

データ構造

RTC_TIME_DATE

これは、現在の時間と日付 (RTC_currentTimeDate) とアラームの時間と日付 (RTC_alarmCfgTimeDate) を保存するのに使用されるデータ構造体です。

```
typedef struct _RTC_TIME_DATE
{
    uint8 Sec;
    uint8 Min;
    uint8 Hour;
    uint8 DayOfWeek;
    uint8 DayOfMonth;
    uint16 DayOfYear;
```

```
    uint8 Month;
    uint16 Year;
} volatile RTC_TIME_DATE;
```

RTC_DSTIME

これは、夏時間の開始と終了(RTC_dstTimeDateStart と RTC_dstTimeDateStop) の時間と日付の値 を保存するのに使用されるデータ構造体です。

```
typedef struct _RTC_DSTIME
{
    uint8 Hour;
    uint8 DayOfWeek;
    uint8 Week;
    uint8 DayOfMonth;
    uint8 Month;
} volatile RTC_DSTIME;
```

定数

曜日、日、月を定義するいくつかの定数があります。プログラムを書く場合、ヘッダー (.h) ファイルで定義されている定数を使用します。

サンプルファームウェアのソースコード

サンプル ファームウェア ソース コード PSoC Creator は、Find Example Project(例題プロジェクトの検索) ダイアログに数多くのサンプルプロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、[Component Catalog](コンポーネント・カタログ) または回路図に置いたコンポーネント インスタンスからダイアログを開きます。一般例については、[Start Page] (開始ページ) または **[File]**(ファイル) メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options**(フィルタオプション) を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator ヘルプの「Find Example Project（サンプルプロジェクトを検索）」を参照してください。

割り込みサービスルーチン

RTC コンポーネントは、秒ごとにトリガーする単一の割り込みを使用します。割り込みハンドラは内部時間と日付構造体を更新し、RTC_WriteIntervalMask() 関数で行った設定に基づいた適切な周期で特定の関数を呼び出します。対応するビットが割り込みマスクレジスタに設定されている場合は、以下の関数が呼び出されます。

- 1 秒間隔ハンドラ – RTC_EverySecondHandler()

- 1 分間隔ハンドラ – RTC_EveryMinuteHandler()
- 1 時間間隔ハンドラ – RTC_EveryHourHandler()
- 1 日間隔ハンドラ – RTC_EveryDayHandler()
- 1 週間間隔ハンドラ – RTC_EveryWeekHandler()
- 1 か月間隔ハンドラ – RTC_EveryMonthHandler()
- 1 年間隔ハンドラ – RTC_EveryYearHandler()

これら関数のスタブ ルーチンは、ユーザ独自のコードを追加できるようにするために提供されています。スタブルーチンは、プロジェクトが初めてビルトされた際に *RTC_INT.c* ファイル中に生成されます。以下のように提供されたコメントタグの間にコードを追加する必要があります。

```
void RTC_EverySecondHandler( void )
{
    /*  Place your every second handler code here. */
    /* `#START EVERY_SECOND_HANDLER_CODE` */

    /* `#END` */
}
```

パワーマネージャ割り込み状態レジスタのすべての割り込みステータスビットは、割り込みハンドラでクリアされます。レジスタのクリアと同時に割り込みが生成された場合、このビットはセットされたままになります（それによって別の割り込みが発生します）。

関数の説明

時間と日付

すべての時間と日付のレジスタは、ソフトウェア変数同様に利用することができます。時間と日付は、Counter コンポーネントからの割り込みイベントに基づいて更新されます。以下の変数が提供されています。

- Sec – 秒: 0〜59
- Min – 分: 0〜59
- Hour – 時 (24 時間形式のみ): 0〜23
- DayOfMonth – 日: 1 ~ 31
- DayOfWeek – 曜日: 1 〜 7。数は StartOfWeek パラメータ設定に依存します。**週の開始** が **日曜日** に設定されている場合: 1 – 日曜日、2 – 月曜日…、7 – 土曜日

- DayOfYear – 1/1 からの経過日数: 1 ~ 366
- Month – 月: 1 ~ 12
- Year – 年: 1900 ～ 2200 (実際の範囲は 1 ～ 65536)

曜日はツェラーの公式を使用して計算されます。ツェラーの公式は、年、月、日に基づいて曜日を計算する整数演算用に最適化された簡単なアルゴリズムです。うるう年およびうるう世紀を含みます。

RTC_Start() 関数を呼び出すと、RTC_SetInitValues() 関数が呼び出され、必要なすべてののフラグと日付計算が実行されます。これには、計算を必要とする以下のすべての変数が含まれています。

- DayOfWeek
- DayOfYear
- LY
- AM_PM
- DST

アラーム関数

アラーム関数は、秒、分、日、曜日、月、年、年の特定日に用意されています。アラーム設定では、同じ変数名が提供されています。これらアラーム設定のいずれか、またはすべてを設定し、これらの設定のうちどれが、アラーム作動に使われるのか、コンフィギュレーションすることができます。

周期割り込み

割り込みスタブ (別の関数内のユーザコードの場所) は、1 秒、1 分、1 時間、1 日、1 週間、1 か月、1 年ごとに実行することができます。コードがスタブにある場合、それは適切な間隔で実行されます。

Daylight Savings Time（夏時間)

Daylight Savings Time 機能を有効にするには、[Configure](設定) ダイアログのチェックボックスを選択します (このデータシートの「コンポーネント・パラメータ」セクションを参照)。Daylight Savings Time は、時間、日付、間隔を更新する API 一式として実装されます。現在の時間と日付が DST 時間と日付に一致している場合、DST フラグが設定され、設定された持続時間によって時間が増加されます。

DST の開始日と終了日を絶対的または相対的に指定することができます。相対的日付は絶対日に変換され、アラーム関数のように現在の時間とチェックされます。絶対的日付は「3 月 24 日」のようになります。相対的日付は「5 月の第 4 日曜日」のようになります。

相対的日付の絶対的日付への変換は、別の関数として実行されます。RTC_Start() 関数が呼び出された後最初の時間の終了時に呼び出され、RTC_Start() 関数自身に変換フラグを設定して、変換が完了していることを示します。次の変換は翌年になります。

以下は開始および終了の時間と日付の DST 変数です。

- Hour – 時: 0 ～ 23 (絶対および相対)
- DayOfWeek – 曜日 1 ～ 7。数は StartOfWeek パラメータ設定によって異なります。**週の開始** が **日曜日** に設定されている場合: 1 – 日曜日、2 – 月曜日、… 、7 – 土曜日 (相対)
- Week – 月の週目: 1 ～ 5 (相対)
- DayOfMonth – 日: 1 ～ 31 (絶対)
- Month – 月: 0 ～ 12 (絶対および相対)

レジスタ

ステータスレジスタ

ステータスレジスタは、さまざまな PTC ステータスビットを含んでいる読み取り専用レジスタです。この変数は、RTC_ReadStatus() 関数を使用して読み取ることができます。ステータスレジスタ用に定義されているビット フィールド マスクは様々なものがあります。以下のように、#defines（定義番号）は作成されたヘッダファイル（.h）の中で利用可能です。

- **RTC_STATUS_DST** – Daylight Saving Time の状態。現在の時間と日付が DST 時間と日付に一致している場合、このビットが HIGH になり、時間が増加されます。DST 割り込み後このビットは LOW になり、時間が減少します。
- **RTC_STATUS_LY** – うるう年ステータス。現在の年がうるう年の場合、このビットは HIGH になります。
- **RTC_STATUS_AM_PM** – 現在の時間のステータス。このビットは深夜から昼まで LOW になり、昼から深夜まで HIGH になります。
- **RTC_STATUS_AA** – アラーム動作ステータス (つまりアラームビット)。現在の時間と日付がアラームの時間と日付に一致している場合、このビットが HIGH になります。状態の読み取り後、ビットが LOW になります。

アラーム マスク レジスタ

アラーム マスク レジスタは書き込み専用で、状態レジスタのアラーム ビットを制御することができます。アラーム ビットは、このレジスタ内のマスクされたビットフィールドの論理和（OR）で生成されます。このレジスタは、

RTC_WriteAlarmMask() 関数を呼び出すことで書き込まれます。アラーム マスク レジスタの書き込みを行う場合、ヘッダ（.h）ファイルに定義されているビットフィールド定義を使用しなければなりません。以下はアラーム マスク レジスタの定義です。

- **RTC_ALARM_SEC_MASK** – 秒アラーム マスクでは、アラーム秒レジスタと現在の秒レジスタを一致させることができます。アラーム秒レジスタは RTC_WriteAlarmSecond() 関数で書き込まれ、RTC_ReadAlarmSecond() 関数で読み取られます。
- **RTC_ALARM_MIN_MASK** – 分アラーム マスクでは、アラーム分レジスタと現在の分レジスタを一致させることができます。アラーム分レジスタは RTC_WriteAlarmMinute() 関数で書き込まれ、RTC_ReadAlarmMinute() 関数で読み取られます。
- **RTC_ALARM_HOUR_MASK** – 時アラーム マスクでは、アラーム時マスクと現在の時レジスタを一致させることができます。アラーム時レジスタは RTC_WriteAlarmHour() 関数で書き込まれ、RTC_ReadAlarmHour() 関数で読み取られます。
- **RTC_ALARM_DAYOFWEEK_MASK** – 曜日アラーム マスクでは、アラーム曜日レジスタと現在の曜日を一致させることができます。アラーム曜日レジスタは RTC_WriteAlarmDayOfWeek() 関数で書き込まれ、RTC_ReadAlarmDayOfWeek() 関数で読み取られます。
- **RTC_ALARM_DAYOFMONTH_MASK** – 日アラーム レジスタでは、アラーム日レジスタを現在の日付に一致させることができます。アラーム月の特定日レジスタは RTC_WriteAlarmDayOfMonth() 関数で書き込まれ、RTC_ReadAlarmDayOfMonth() 関数で読み取られます。
- **RTC_ALARM_DAYOFYEAR_MASK** – 年始からの経過日数アラーム マスクでは、年始からの経過日数アラームレジスタを現在の年始からの経過日数レジスタに一致させることができます。年始からの経過日数アラームレジスタは RTC_WriteAlarmDayOfYear() 関数で書き込まれ、RTC_ReadAlarmDayOfYear() 関数で読み取られます。
- **RTC_ALARM_MONTH_MASK** – 月マスク レジスタでは、アラーム月レジスタを現在のアラーム月レジスタに一致させることができます。アラーム月レジスタは RTC_WriteAlarmMonth() 関数で書き込まれ、RTC_ReadAlarmMonth() 関数で読み取られます。
- **RTC_ALARM_YEAR_MASK** – 年アラーム マスク レジスタは、アラーム年レジスタを現在の年レジスタに一致させるます。アラーム年レジスタは RTC_WriteAlarmYear() 関数で書き込まれ、RTC_ReadAlarmYear() 関数で読み取られます。

インターバル マスク レジスタ

インターバル マスク レジスタは書き込み専用で、RTC コンポーネントの割り込みスタブのハンドリングを制御します。割り込みスタブは、秒、分、時、日、週、年それぞれに提供されます。割り込みスタブの実行を有効にするには、このレジスタの対応するビットを設定します。このレジスタは、RTC_WriteIntervalMask() 関数を呼び出すことで書き込まれます。インターバル マスク レジスタの書き込みを行う場合、ヘッダ(.h)ファイルに定義されているビットフィールド定義を使用しなければなりません。以下はインターバル マスク レジスタの定義です。

- **RTC_INTERVAL_SEC_MASK** – 秒インターバル マスクは、秒ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_MIN_MASK** – 分インターバル マスクは、分ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_HOUR_MASK** – 時インターバル マスクは、時ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_DAY_MASK** – 日インターバル マスクは、日ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_WEEK_MASK** – 週インターバル マスクは、週ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_MONTH_MASK** – 月インターベル マスクは、月ごとに割り込みスタブをハンドリングします。
- **RTC_INTERVAL_YEAR_MASK** – 年インターバル マスクは、年ごとに割り込みスタブをハンドリングします。

DST モード レジスタ

DST モード レジスタは書き込み専用のレジスタで、Daylight Savings Time モードを設定して、DST 動作を有効にします。このレジスタは、RTC_WriteDSTMode() 関数を呼び出すことで書き込まれます。DST モード レジスタの書き込みを行う場合、ヘッダ(.h)ファイルに定義されているビットフィールド定義を使用しなければなりません。以下は DST モード レジスタの定義です。

- **RTC_DST_ENABLE** – イネーブル ビットは Daylight savings time 関数の有効を制御します。
- **RTC_DST_FIXDATE** – Daylight savings time 関数の時間と日付の絶対的形式を定義します (開始と終了)。例えば、3 月 24 日。
- **RTC_DST_RELDATE** – Daylight savings time 関数の時間と日付の相対的形式を定義します (開始と終了)。例えば、5 月の第 2 日曜日。

条件付きコンパイル情報

RTC API は、Daylight savings time 関数を操作する条件付きコンパイル定義が必要です。DST 関数は、このオプションが [Configure](設定) ダイアログで有効にされている場合にのみ条件に従ってコンパイルされます。ソフトウェアがこのパラメータを直接使用することはありません。代わりに、定義されたシンボル名を使用します。

- **RTC_DST_FUNC_ENABLE** – Daylight savings time 関数のイネーブル定義は、ビルド時の DstEnable 値 (Configure(設定) ダイアログから) と同じに割り当てられます。API 全体で使用され、Data saving time 関数をコンパイルします。

コンポーネントの変更

ここでは、過去のバージョンからコンポーネントに加えられた主な変更を示します。

バージョン	変更の説明	変更の理由 / 影響
1.60.a	RTC_WriteIntervalMask() 関数の説明をアップデートしました。「割り込みサービス ルーチン」セクションをアップデートして、さらに割り込みハンドラの設定方法をわかりやすくしました。	割り込みハンドラ使用説明をアップデートしました。
1.60	コードと ISR の両方に使用されるグローバル変数がコンパイラで最適化で除外される可能性がある場合の問題を修正しました。	予期しない結果をもたらす可能性がある最適化問題を防止します。
	結合領域が ISR に追加されました (RTC_INT.c ファイル)。	RTC コンポーネントの柔軟性を改善しました。
	コード コメントを改善してさらに明確にしました。入力ミスを修正しました。	コンポーネントに関する情報を補正しました。
	データシートのマイナーな編集と更新	
1.50.a	RTC バージョン 1.50 を使用するには、cy_boot バージョン 2.x が必要です。	RTC バージョン 1.50 を使用する場合は、cy_boot コンポーネントをバージョン 2.x にアップデートしてください。
	データシート例題プロジェクトが追加され、「サンプル ファームウェア ソース コード」セクションがアップデートされました。	データシート例題プロジェクトが PSoC Creator に追加されました。
	RTC_Enable() 関数の説明が追加されました。	ユーザに提供される各関数はデータシートで説明する必要があります。
	以下の関数のパラメータ説明の参照を追加しました。RTC_WriteAlarmMask()、RTC_WriteIntervalMask()、RTC_ReadStatus()。	関数の使用方法に関するコメントで明確にしました。
	バージョン 1.50 に Keil 社の再入可能サポートを追加。	Keil 社のコンパイラを用いたPSoC 3は、複数のコントロールフローから呼び出される関数の機能に対応。

バージョン	変更の説明	変更の理由 / 影響
1.50	API フローの変更 - RTC_Init() 関数の追加。	企業の標準に準拠し、コンポーネントを起動せずに初期化・復元できるAPIを提供するため。
	一部のグローバル変数の名前を変更してコーディング標準に準拠。関数のグローバルパラメータの使用と戻りを説明する関数ヘッダのアップデート。	コーディング規則に準拠。影響なし - 下位互換性のためのマクロを作成。
	データベースの更新。	「コンポーネントの変更」セクションを追加。空の「副作用」セクションから「アプリケーション プログラミング インターフェイス」セクションを追加。
	RTCIsLeapYear(uint16 year) 関数を RTC_LEAP_YEAR マクロに置き換え。 RTC_SET_ALARM および RTC_IS_BIT_SET を作成。RTC_IS_BIT_SET マクロが使用される条件式の追加。	ISR 内で関数を呼び出すための悪い例を修正。移動またはマクロでの置き換え。オンライン関数はマクロに置き換える必要があります。いくつかの場所で使用されるコードは、マクロに置き換える必要があります。

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporationは、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及びProgrammable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)はCypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。