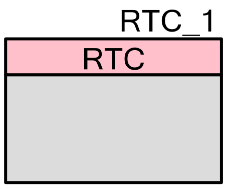
Real-Time Clock (RTC)

1.50

Features



- Multiple Alarm Options
- Multiple Overflow Options
- Daylight Saving Time (DST) Option

General Description

The Real-Time Clock (RTC) component provides accurate time and date information for the system. The time and date are updated every second based on a 1 pulse per second interrupt from a 32.768 kHz crystal. Clock accuracy is based on the crystal provided and is typically 20ppm.

The RTC keeps track of the second, minute, hour, day of the week, day of the month, day of the year, month and year. The day of the week is automatically calculated from the day, month and year. Daylight saving time may be optionally enabled and supports any start and end date, as well as a programmable saving time. The start and end dates may be absolute like 24 March or relative like the 2nd Sunday in May.

The Alarm provides match detection for a second, minute, hour, day of week, day of month, day of year, month and year. A mask selects what combination of time and date information will be used to generate the alarm. The alarm flexibility supports periodic alarms such as every 23rd minute after the hour, or a single alarm such as 4:52 AM on the 28th of September 2043.

User code stubs are provided for periodic code execution based on each of the primary time intervals. Timer intervals are provided at one second, one minute, one hour, one day, one week, one month, and one year.

When to use a RTC Component

Use the RTC component when the system requires the current time or date. The RTC may also be used when the current time and date are not required but accurate timing of events with one second resolution is required.

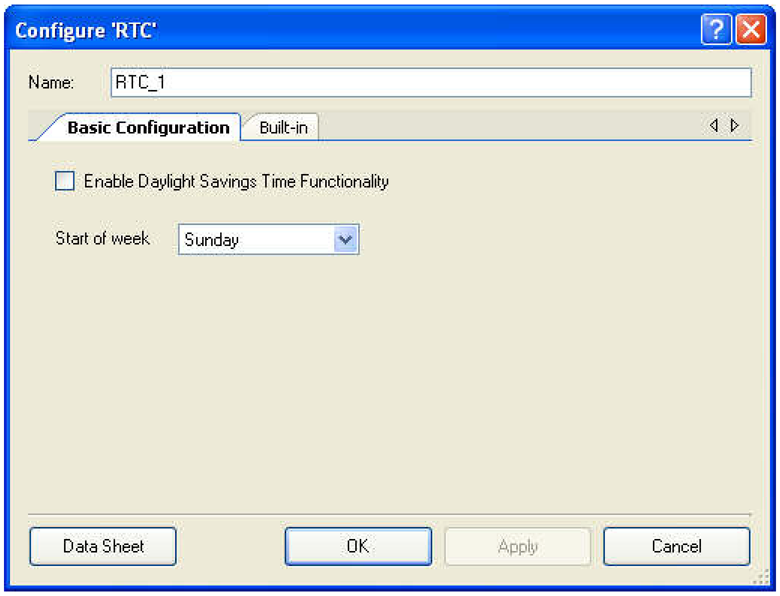
Input/Output Connections

The RTC component does not have input or output connections.

Parameters and Setup

Drag an RTC component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure RTC Dialog



Name: RTC_1

☐ Enable Daylight Savings Time Functionality

Start of week: Sunday

The RTC component contains the following options:

Enable Daylight Saving Time

This parameter allows you to choose if the daylight saving time functionality is enabled in the RTC component. The default value is unchecked (false).

Start of Week

The start of week parameter allows you to choose start day of the week. Options include:

- "Sunday" (default): Sunday is start of the week
- "Monday": Monday is start of the week
- "Tuesday": Tuesday is start of the week
- "Wednesday": Wednesday is start of the week
- "Thursday": Thursday is start of the week
- "Friday": Friday is start of the week
- "Saturday": Saturday is start of the week

Clock Selection

A 32.768 kHz clock should be provided from an external crystal oscillator. The accuracy of this component is defined by the accuracy of the connected external clock source. Refer to the PSoC Creator Help, Clock Editor section for information about how to connect and configure the built-in XTAL_32KHZ clock in your design.

Placement

Not applicable

Resources

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
RTC fixed HW *	0	0	0	0	0	?	?	?

* One Pulse per Second Interrupt from Power Management used

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "RTC_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "RTC.'

It is recommended to disable the component’s interrupts while calling functions that read/modify global variables.

Refer also to the Registers section of this data sheet for more information, as needed.

Function	Description
void RTC_Init(void)	Initializes/restores default configuration provided with the customizer.
void RTC_Enable(void)	Enables the interrupts, one pulse per second, and interrupt generation on OPPS event.

PRELIMINARY

Function	Description
void RTC_Start(void)	Enables the RTC component.
void RTC_Stop(void)	Stops RTC component operation
void RTC_EnableInt(void)	Enables interrupts of RTC component
void RTC_DisableInt(void)	Disables interrupts of the RTC component, time and date stop running
void RTC_WriteTime(RTC_TIME_DATE *timeDate)	Writes time and date values as current time and date.
RTC_TIME_DATE *timeDate RTC_ReadTime(void)	Reads the current time and date
void RTC_WriteSecond(uint8 second)	Writes Sec software register value
void RTC_WriteMinute(uint8 minute)	Writes Min software register value
void RTC_WriteHour(uint8 hour)	Writes Hour software register value
void RTC_WriteDayOfMonth(uint8 dayOfMonth)	Writes DayOfMonth software register value
void RTC_WriteMonth(uint8 month)	Writes Month software register value
void RTC_WriteYear(uint16 year)	Writes Year software register value
void RTC_WriteAlarmSecond(uint8 second)	Writes Alarm Sec software register value
void RTC_WriteAlarmMinute(uint8 minute)	Writes Alarm Min software register value
void RTC_WriteAlarmHour(uint8 hour)	Writes Alarm Hour software register value
void RTC_WriteAlarmDayOfMonth(uint8 dayOfWeek)	Writes Alarm DayOfMonth software register value
void RTC_WriteAlarmMonth (uint8 month)	Writes Alarm Month software register value
void RTC_WriteAlarmYear(uint16 year)	Writes Alarm Year software register value
void RTC_WriteAlarmDayOfWeek(uint8 dayOfWeek)	Writes Alarm DayOfWeek software register value
void RTC_WriteAlarmDayOfYear(uint16 dayOfYear)	Writes Alarm DayOfYear software register value
uint8 RTC_ReadSecond(void)	Reads Sec software register value
uint8 RTC_ReadMinute(void)	Reads Min software register value
uint8 RTC_ReadHour(void)	Reads Min software register value
uint8 RTC_ReadDayOfMonth(void)	Reads DayOfMonth software register value
uint8 RTC_ReadMonth(void)	Reads Month software register value
uint16 RTC_ReadYear(void)	Reads Year software register value
uint8 RTC_ReadAlarmSecond(void)	Reads Alarm Sec software register value
uint8 RTC_ReadAlarmMinute(void)	Reads Alarm Min software register value

Function	Description
uint8 RTC_ReadAlarmHour(void)	Reads Alarm Hour software register value
uint8 RTC_ReadAlarmDayOfMonth(void)	Reads Alarm DayOfMonth software register value
uint8 RTC_ReadAlarmMonth(void)	Reads Alarm Month software register value
uint16 RTC_ReadAlarmYear(void)	Reads Alarm Year software register value
uint8 RTC_ReadAlarmDayOfWeek(void)	Reads Alarm DayOfWeek software register value
uint16 RTC_ReadAlarmDayOfYear(void)	Reads Alarm DayOfYear software register value
void RTC_WriteAlarmMask(uint16/8 mask)	Writes the Alarm Mask software register with 1 bit per time/date entry.
void RTC_WriteIntervalMask (uint8 mask)	Writes the Interval Mask software register with 1 bit per time/date entry.
uint8 RTC_ReadStatus(void)	Reads the Status software register which has flags for DST (DST), Leap Year (LY) and AM/PM (AM_PM), Alarm active (AA)
void RTC_WriteDSTMode(uint8 mode)	Writes the DST Mode software register
void RTC_WriteDSTStartHour(uint8 hour)	Writes the DST Start Hour software register.
void RTC_WriteDSTStartDayOfMonth(uint8 dayOfMonth)	Writes the DST Start DayOfMonth software register.
void RTC_WriteDSTStartMonth(uint8 month)	Writes the DST Start Month software register.
void RTC_WriteDSTStartDayOfWeek(uint8 dayOfWeek)	Writes the DST Start DayOfWeek software register.
void RTC_WriteDSTStartWeek(uint8 week)	Writes the DST Start Week software register.
void RTC_WriteDSTStopHour(uint8 hour)	Writes the DST Stop Hour software register.
void RTC_WriteDSTStopDayOfMonth(uint8 dayOfMonth)	Writes the DST Stop DayOfMonth software register.
void RTC_WriteDSTStopMonth(uint8 month)	Writes the DST Stop Month software register.
void RTC_WriteDSTStopDayOfWeek(uint8 dayOfWeek)	Writes the DST Stop DayOfWeek software register.
void RTC_WriteDSTStopWeek(uint8 week)	Writes the DST Stop Week software register.
void RTC_WriteDSTOffset(uint8 offset)	Writes the DST Offset register.

Global Variables

Variable	Description
RTC_initVar	Indicates whether the RTC has been initialized. The variable is initialized to 0 and set to 1 the first time RTC_Start() is called. This allows the component to restart without reinitialization after the first call to the RTC_Start() routine. If reinitialization of the component is required, then the RTC_Init() function can be called before the RTC_Start() or RTC_Enable() function.
RTC_currentTimeDate	The current time and date values are stored in this variable.
RTC_statusDateTime	This variable has following flags: DST, Leap Year, AM/PM and Active Alarm statuses.
RTC_intervalCfgMask	This variable is used to define what interrupt stub should be executed.
RTC_alarmCfgTimeDate	The alarm time and date values are stored in this variable.
RTC_alarmCurStatus	This variable is used to indicate current active alarm: seconds alarm is active, minutes alarm is active and so on.
RTC_alarmCfgMask	This variable is used to mask alarm events: mask seconds alarm, mask minutes alarm and so on.
RTC_dstModeType	This variable stores DST mode type's value: '0' – fixed and '1' – relative.
RTC_dstTimeDateStart	The values for the time and date of the DST start.
RTC_dstTimeDateStop	The values for the time and date of the DST stop.
RTC_dstStartStatus	The DST start status.
RTC_dstStopStatus	The DST stop status.
RTC_dstOffsetMin	The DST offset value in minutes.

void RTC_Init (void)

Description:	Initializes/restores the default configuration provided with the customizer.
Parameters:	None.
Return Value:	None.
Side Effects:	None.

PRELIMINARY

void RTC_Enable(void)

Description: Enables the interrupts, one pulse per second, and interrupt generation on OPPS event.

Parameters: None.

Return Value: None.

Side Effects: Enables the one pulse per second and central time wheel signals to wake up device from the low power (Sleep and Alternate Active) modes and leaves them enabled.

void RTC_Start(void)

Description: Enables the RTC component. This function configures the counter, sets up interrupts, does all required calculation, and starts the counter.

Parameters: None.

Return Value: None.

Side Effects: Enables the one-pulse per-second and central time wheel signals to wake up the device from low power modes (Sleep and Alternate Active) and leaves them enabled.

void RTC_Stop(void)

Description: Stops RTC component operation.

Parameters: None.

Return Value: None.

Side Effects: Leaves the one-pulse per-second and the central time wheel signals enabled, which wake up the device from low power modes (Sleep and Alternate Active).

void RTC_EnableInt(void)

Description: Enables interrupts from RTC component.

Parameters: None.

Return Value: None.

Side Effects: None.

void RTC_DisableInt(void)

Description: Disables interrupts from RTC component, time and date stop running.

Parameters: None.

Return Value: None.

Side Effects: None.

RTC_TIME_DATE* timeDate RTC_ReadTime(void)

Description:	Reads current time and date.
Parameters:	None.
Return Value:	Pointer to the RTC_TIME_DATE..
Side Effects:	None.

void RTC_WriteTime(RTC_TIME_DATE * timeDate)

Description:	Writes time and date values as current time and date. Only passes the Second, Minute, Hour, Month, Day of Month and Year.
Parameters:	timeDate: pointer to the RTC_TIME_DATE global structure where new values of time and date are stored.
Return Value:	None.
Side Effects:	The RTC component's interrupt should be disabled before calling this function and enabled afterwards to avoid RTC counter increment in the middle of the time and date writing.

void RTC_WriteSecond(uint8 second)

Description:	Writes the Sec software register value.
Parameters:	second: Seconds value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteMinute(uint8 minute)

Description:	Writes the Min software register value.
Parameters:	minute: Minutes value
Return Value:	None.
Side Effects:	None.

void RTC_WriteHour(uint8 hour)

Description:	Writes the Hour software register value.
Parameters:	hour: Hours value.
Return Value:	None.
Side Effects:	None.

PRELIMINARY

void RTC_WriteDayOfMonth(uint8 dayOfMonth)

Description:	Writes the DayOfMonth software register value.
Parameters:	dayOfMonth: DayOfMonth value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteMonth(uint8 month)

Description:	Writes the Month software register value.
Parameters:	month: Month value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteYear(uint16 year)

Description:	Writes the Year software register value.
Parameters:	year: Years value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmSecond(uint8 second)

Description:	Writes the Alarm Sec software register value.
Parameters:	second: Alarm Seconds value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmMinute(uint8 minute)

Description:	Writes the Alarm Min software register value.
Parameters:	minute: Alarm Minutes value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmHour(uint8 hour)

Description:	Writes the Alarm Hour software register value.
Parameters:	hour: Alarm Hours value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmDayOfMonth(uint8 dayOfMonth)

Description:	Writes the Alarm DayOfMonth software register value.
Parameters:	dayOfMonth: Alarm Day Of Month value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmMonth(uint8 month)

Description:	Writes the Alarm Month software register value.
Parameters:	month: Alarm Months value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmYear(uint16 year)

Description:	Writes the Alarm Year software register value.
Parameters:	year: Alarm Years value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmDayOfWeek(uint8 dayOfWeek)

Description:	Writes the Alarm DayOfWeek software register value.
Parameters:	dayOfWeek: Alarm Day Of Week value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteAlarmDayOfYear(uint16 dayOfYear)

Description:	Writes the Alarm DayOfYear software register value.
Parameters:	dayOfYear: Alarm Day Of Year value.
Return Value:	None.
Side Effects:	None.

uint8 RTC_ReadSecond(void)

Description:	Reads the Sec software register value.
Parameters:	None.
Return Value:	None.
Side Effects:	None.

uint8 RTC_ReadMinute(void)

Description:	Reads the Min software register value.
Parameters:	None.
Return Value:	The current minute's value is returned.
Side Effects:	None.

uint8 RTC_ReadHour(void)

Description:	Reads the Min software register value.
Parameters:	None.
Return Value:	The current hour's value is returned.
Side Effects:	None.

uint8 RTC_ReadDayOfMonth(void)

Description:	Reads the DayOfMonth software register value.
Parameters:	None.
Return Value:	The current day of month's value is returned.
Side Effects:	None.

PRELIMINARY

uint8 RTC_ReadMonth(void)

Description:	This function reads the Month software register value.
Parameters:	None.
Return Value:	The current month's value is returned.
Side Effects:	None.

uint16 RTC_ReadYear(void)

Description:	Reads the Year software register value.
Parameters:	None.
Return Value:	The current year's value is returned.
Side Effects:	None.

uint8 RTC_ReadAlarmSecond(void)

Description:	Reads the Alarm Sec software register value.
Parameters:	None.
Return Value:	The current second's alarm value is returned.
Side Effects:	None.

uint8 RTC_ReadAlarmMinute(void)

Description:	Reads the Alarm Min software register value.
Parameters:	None.
Return Value:	The current minute's alarm value is returned.
Side Effects:	None.

uint8 RTC_ReadAlarmHour(void)

Description:	Reads the Alarm Hour software register value.
Parameters:	None.
Return Value:	The current hour's alarm value is returned.
Side Effects:	None.

uint8 RTC_ReadAlarmDayOfMonth(void)

Description:	Reads the Alarm DayOfMonth software register value.
Parameters:	None.
Return Value:	The current day of month’s alarm value is returned.
Side Effects:	None.

uint8 RTC_ReadAlarmMonth(void)

Description:	Reads the Alarm Month software register value.
Parameters:	None.
Return Value:	The current month’s alarm value is returned.
Side Effects:	None.

uint16 RTC_ReadAlarmYear(void)

Description:	Reads the Alarm Year software register value.
Parameters:	None.
Return Value:	The current year’s alarm value is returned..
Side Effects:	None.

uint8 RTC_ReadAlarmDayOfWeek(void)

Description:	Reads the Alarm DayOfWeek software register value.
Parameters:	None.
Return Value:	The current day of week’s alarm value is returned.
Side Effects:	None.

uint16 RTC_ReadAlarmDayOfYear(void)

Description:	Reads the Alarm DayOfYear software register value.
Parameters:	None.
Return Value:	The current day of year’s alarm value is returned.
Side Effects:	None.

void RTC_WriteAlarmMask(uint8 mask)

Description:	Writes the Alarm Mask software register with 1 bit per time/date entry. Alarm true when all masked time/date values match Alarm values.
Parameters:	mask: Alarm Mask software register value. Refer to the **Alarm Mask Register** section for more information about this parameter.
Return Value:	None.
Side Effects:	None.

void RTC_WriteIntervalMask(uint8 mask)

Description:	Writes the Interval Mask software register with 1 bit per time/date entry. 'Interrupt' true when any masked time/date overflow occur.
Parameters:	mask: Interval Mask software register value. Refer to the **Interval Mask Register** section for more information about this parameter.
Return Value:	None.
Side Effects:	None.

uint8 RTC_ReadStatus(void)

Description:	Reads the Status software register which has flags for DST (DST), Leap Year (LY) and AM/PM (AM_PM), Alarm active (AA).
Parameters:	None.
Return Value:	Current component's status with active alarm bit cleared. Refer to the **Status Register** section for more information about the return values.
Side Effects:	Alarm active (AA) flag clear after read.

void RTC_WriteDSTMode(uint8 mode)

Description:	Writes the DST Mode software register That enables or disables DST changes and sets the date mode to fixed date or relative date. Only generated if DST enabled.
Parameters:	mode: DST Mode software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStartHour(uint8 hour)

Description:	Writes the DST Start Hour software register. Used for absolute date entry. Only generated if DST enabled.
Parameters:	hour: DST Start Hour software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStartDayOfMonth(uint8 dayOfMonth)

Description:	Writes the DST Start DayOfMonth software register. Used for absolute date entry. Only generated if DST enabled.
Parameters:	dayOfMonth: DST Start DayOfMonth software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStartMonth(uint8 month)

Description:	Writes the DST Start Month software register. Used for absolute date entry. Only generated if DST enabled.
Parameters:	month: DST Start Month software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStartDayOfWeek(uint8 dayOfWeek)

Description:	Writes the DST Start DayOfWeek software register. Used for relative date entry. Only generated if DST enabled.
Parameters:	dayOfWeek: DST Start DayOfWeek software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStartWeek(uint8 week)

Description:	Writes the DST Start Week software register. Used for relative date entry. Only generated if DST enabled.
Parameters:	Week: DST Start Week software register value.
Return Value:	None.
Side Effects:	None.

void RTC_WriteDSTStopHour(uint8 hour)

Description: Writes the DST Stop Hour software register. Used for absolute date entry. Only generated if DST enabled.

Parameters: hour: DST Stop Hour software register value.

Return Value: None.

Side Effects: None.

void RTC_WriteDSTStopDayOfMonth(uint8 dayOfMonth)

Description: Writes the DST Stop DayOfMonth software register. Used for absolute date entry. Only generated if DST enabled.

Parameters: dayOfMonth: DST Stop DayOfMonth software register value.

Return Value: None.

Side Effects: None.

void RTC_WriteDSTStopMonth(uint8 month)

Description: Writes the DST Stop Month software register. Used for absolute date entry. Only generated if DST enabled.

Parameters: month: DST Stop Month software register value.

Return Value: None.

Side Effects: None.

void RTC_WriteDSTStopDayOfWeek(uint8 dayOfWeek)

Description: Writes the DST Stop DayOfWeek software register. Used for relative date entry. Only generated if DST enabled.

Parameters: dayOfWeek: DST Stop DayOfWeek software register value.

Return Value: None.

Side Effects: None.

void RTC_WriteDSTStopWeek(uint8 week)

Description: Writes the DST Stop Week software register. Used for relative date entry. Only generated if DST enabled.

Parameters: week: DST Stop Week software register value.

Return Value: None.

Side Effects: None.

void RTC_WriteDSTOffset(uint8 offset)

Description: Writes the DST Offset register. Allows a configurable increment or decrement of time between 0 and 255 minutes. Increment occurs on DST start and decrement on DST stop. Only generated if DST enabled.

Parameters: offset: DST Offset software register value.

Return Value: None.

Side Effects: None.

Data Structures

RTC_TIME_DATE

This is the data structure that is used to save the current time and date (RTC_currentTimeDate), and Alarm time and date (RTC_alarmCfgTimeDate).

```
typedef struct _RTC_TIME_DATE
{
    uint8 Sec;
    uint8 Min;
    uint8 Hour;
    uint8 DayOfWeek;
    uint8 DayOfMonth;
    uint16 DayOfYear;
    uint8 Month;
    uint16 Year;
} RTC_TIME_DATE;
```

RTC_DSTIME

This is the data structure that is used to save time and date values for Daylight Saving Time Start and Stop (RTC_dstTimeDateStart and RTC_dstTimeDateStop).

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

Constants

There are several constants that define day of week, day in month, and month. When writing code use the constants defined in the header (.h) file.

Sample Firmware Source Code PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

Interrupt Service Routines

The RTC component uses a single interrupt that triggers every second. The interrupt handler updates the internal date and time structure, and then calls specific functions at appropriate intervals. The following functions are called:

- Every Second handler – RTC_EverySecondHandler()
- Every Minute handler – RTC_EveryMinuteHandler()
- Every Hour handler – RTC_EveryHourHandler()
- Every Day handler – RTC_EveryDayHandler()
- Every Week handler – RTC_EveryWeekHandler()
- Every Month handler – RTC_EveryMonthHandler()
- Every Year handler – RTC_EveryYearHandler()

Stub routines for these functions are provided where you can add your own code. The routine stubs are generated in the *RTC_INT.c* file the first time the project is built. Your code must be added between the provided comment tags as follows:

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

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

All interrupt status bits of the Power Manager Interrupt Status register are cleared in the interrupt handler. If an interrupt gets generated at the same time as a clear, the bit will remain set (which causes another interrupt).

Functional Description

Time and date

All time and date registers are as accessible as software variables. The time and date change is based on an interrupt event from the Counter component. The following variables are provided:

- Sec – seconds 0 – 59
- Min – minutes 0 – 59
- Hour – hours (24 format only) 0 – 23
- DayOfMonth – day of month 1 – 31
- DayOfWeek – day of week 1 – 7. The number depends on StartOfWeek parameter settings. If StartOfWeek is set to Sunday then: 1 – Sunday, 2 – Monday…,7 – Saturday
- DayOfYear – day of year 1 – 366
- Month – month 1 – 12
- Year – year, 1900 – 2200 (the actual range is 1 – 65 536)

The DayOfWeek is calculated using Zeller's congruence. Zeller's congruence is a simple algorithm optimized for integer math that calculates the day of the week based on year, month and day of the month. It accounts for Leap years and leap centuries.

When you call the RTC_Start function, a StartCalculation function is called and all required flags and date calculations are executed. This includes all variables that need calculation:

- DayOfWeek
- DayOfYear
- LY
- AM_PM
- DST

Alarm function

The alarm function provides for seconds, minutes, hours, days of the month, days of the week, month, year, and day of the year. The same variable names are provided for alarm settings. The user may set any of all of these alarm settings and configure which of these settings are used in tripping the alarm.

Periodic interrupts

Interrupt stubs (locations for user code in separate functions) are provided that can run every second, minute, hour, day, week, month and year. If code is present in the stub it will be run at the appropriate interval.

Daylight Saving Time

To enable the Daylight Saving Time feature, select the check box on the Configure dialog (see Parameters section of this data sheet). Daylight Saving Time is implemented as set of API update times, dates, and durations. If the current time and date match the start of DST time and date then the DST flag is set and the time is incremented by the set duration.

The start and stop date of DST can be given as fixed or relative. The relative date converts to the fixed one and is checked against the current time as if it were an alarm function. An example of a fixed date is "24 March." An example of a relative date is "4th Sunday in May."

The conversion of a relative date to a fixed date is implemented as a separate function. It is called at the end of the first hour after the RTC_Start() function is called, and it sets a conversion flag in the RTC_Start() function itself that indicates the conversion is done. The next conversion will be in next year.

The DST variables for start and stop time and date are as follows:

- Hour – hour 0 - 23 (fixed and relative)
- DayOfWeek – day of week 1 – 7. The number depends on StartOfWeek parameter settings. If StartOfWeek is set to Sunday then: 1 – Sunday, 2 – Monday, … ,7 – Saturday (relative)
- Week – week in month 1 – 5 (relative)
- DayOfMonth – day of month 1 – 31 (fixed)
- Month – month 1 – 12 (fixed and relative)

Registers

Status Register

The status register is a read-only register that contains various RTC status bits. This value can be read using the RTC_ReadStatus() function. There are several bit-field masks defined for the status register. The #defines are available in the generated header file (.h) as follows:

- **RTC_STATUS_DST** – Status of Daylight Saving Time. This bit goes high when the current time and date match DST time and date and the time is incremented. This bit goes low after the DST interval and the time is decremented.
- **RTC_STATUS_LY** – Status of leap year. This bit goes high when the current year is a leap year.

PRELIMINARY

- **RTC_STATUS_AM_PM** – Status of current time. This bit is low from midnight to noon and high from noon to midnight.
- **RTC_STATUS_AA** – Status of alarm active (i.e., alarm bit). This bit is high when current time and date match alarm time and date. Once the status is read this bit goes low.

Alarm Mask Register

The alarm mask register is a write-only register that allows you to control the alarm bit in the status register. The alarm bit is generated by ORing the masked bit-fields within this register. This register is written with the RTC_WriteAlarmMask() function call. When writing the alarm mask register you must use the bit-field definitions as defined in the header (.h) file. The definitions for the alarm mask register are as follows:

- **RTC_ALARM_SEC_MASK** – The second alarm mask allows you to match the alarm second register with the current second register. The alarm second register is written with the RTC_WriteAlarmSecond() function call and read with RTC_ReadAlarmSecond().
- **RTC_ALARM_MIN_MASK** – The minute alarm mask allows you to match the alarm minute register with the current minute register. The alarm minute register is written with the RTC_WriteAlarmMinute() function call and read with the RTC_ReadAlarmMinute().
- **RTC_ALARM_HOUR_MASK** – The hour alarm mask allows you to match the alarm hour register with the current hour register. The alarm hour register is written with the RTC_WriteAlarmHour() function call and read with the RTC_ReadAlarmHour().
- **RTC_ALARM_DAYOFWEEK_MASK** – The day of week alarm mask allows you to match the alarm day of week register with the current day of week register. The alarm day of week register is written with the RTC_WriteAlarmDayOfWeek() function call and read with the RTC_ReadAlarmDayOfWeek().
- **RTC_ALARM_DAYOFMONTH_MASK** – The day of month alarm mask allows you to match the alarm day of month register with the current day of month register. The alarm day of month register is written with the RTC_WriteAlarmDayOfMonth() function call and read with the RTC_ReadAlarmDayOfMonth().
- **RTC_ALARM_DAYOFYEAR_MASK** – The day of year alarm mask allows you to match the alarm day of year register with the current day of year register. The alarm day of year register is written with the RTC_WriteAlarmDayOfYear() function call and read with the RTC_ReadAlarmDayOfYear().
- **RTC_ALARM_MONTH_MASK** – The month alarm mask allows you to match the alarm month register with the current month register. The alarm month register is written with the RTC_WriteAlarmMonth() function call and read with the RTC_ReadAlarmMonth().
- **RTC_ALARM_YEAR_MASK** – The year alarm mask allows you to match the alarm year register with the current year register. The alarm year register is written with the RTC_WriteAlarmYear() function call and read with the RTC_ReadAlarmYear().

Interval Mask Register

The interval mask register is a write-only register that allows you to control handling of interrupt stubs of the RTC component. The interrupt stubs are provided for every second, minute, hour, day, week, month and year. To enable interrupt stub execution, set the appropriate bit in this register. This register is written with the RTC_WriteIntervalMask() function call. When writing the interval mask register you must use the bit-field definitions as defined in the header (.h) file. The definitions for the interval mask register are as follows:

- **RTC_INTERVAL_SEC_MASK** – The second interval mask allows handling an interrupt stub every second.
- **RTC_INTERVAL_MIN_MASK** – The minute interval mask allows handling an interrupt stub every minute.
- **RTC_INTERVAL_HOUR_MASK** – The hour interval mask allows handling an interrupt stub every hour.
- **RTC_INTERVAL_DAY_MASK** – The day interval mask allows handling an interrupt stub every day.
- **RTC_INTERVAL_WEEK_MASK** – The week interval mask allows handling an interrupt stub every week.
- **RTC_INTERVAL_MONTH_MASK** – The month interval mask allows handling an interrupt stub every month.
- **RTC_INTERVAL_YEAR_MASK** – The year interval mask allows handling an interrupt stub every year.

DST Mode Register

The DST mode register is a write only register which allows you to set the Daylight Saving Time mode and enable DST operation. This register is written with the RTC_WriteDSTMode() function call. When writing the DST mode register you must use the bit-field definition as defined in the header (.h) file. The definitions for the DST mode register are as follows:

- **RTC_DST_ENABLE** – The enable bit controls enabling the daylight saving time functionality.
- **RTC_DST_FIXDATE** – Defines the fixed format of the times and dates (start and stop) for daylight saving time functionality. For example, fixed date is 24 March.
- **RTC_DST_RELDATE** – Defines the relative format of the times and dates (start and stop) for daylight saving time functionality. For example, relative date is 2nd Sunday in May.

Conditional Compilation Information

The RTC API requires one conditional compile definition to handle daylight saving time functionality. The DST functions are conditionally compiled only if this option is enabled in the Configure dialog. The software should never use this parameter directly. Instead, use the symbolic name defined.

- **RTC_DST_FUNC_ENABLE** – The daylight saving time functionality enable define is assigned to be equal to the DstEnable value (from the Configure dialog) at build time. It is used throughout the API to compile data saving time functions.

References

Not applicable

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		32	kHz	

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.50.a	The RTC version 1.50 requires cy_boot version 2.x to be used.	The cy_boot component should be updated up to version 2.x for RTC version 1.50 could be used.
	The "Sample Firmware Source Code" section was updated to note that data sheet example project has been added.	The data sheet example project was added to the PSoC Creator.
	RTC_Enable() function description was added.	Every function that is exposed to user should be described in the datasheet.
	Added reference for the parameter descriptions of the following functions: RTC_WriteAlarmMask(), RTC_WriteIntervalMask(), and RTC_ReadStatus().	Clarification comments on the functions' usage.
	The Keil reentrancy support was added in version 1.50.	PSoC 3 with the Keil compiler supports the capability for functions to be called from multiple flows of control.
1.50	The API flow has been changed - the RTC_Init() function was added.	To comply with corporate standard and provide an API to initialize/restore the component without starting it.
	Some global variables were renamed to comply with coding standard. Function headers were updated to describe function global parameters usage and returns.	Comply with coding rules. No impacts - macros were created for backward compatibility.
	The data sheet was updated.	Added "Component Changes" section. Added empty "Side Effects" section to the "Application Programming Interface" section.
	The RTCIsLeapYear(uint16 year) function was replaced by RTC_LEAP_YEAR macros. The RTC_SET_ALARM and RTC_IS_BIT_SET were created. The RTC_IS_BIT_SET macro used in a few more conditional expressions.	Fixed a bad practice to call functions within ISR, they were moved out or replaced by macros. One-line functions should be replaced by macros. The code, which is used in a few places, should be replaced by macros.