Real Time Clock Datasheet RTC V 1.10

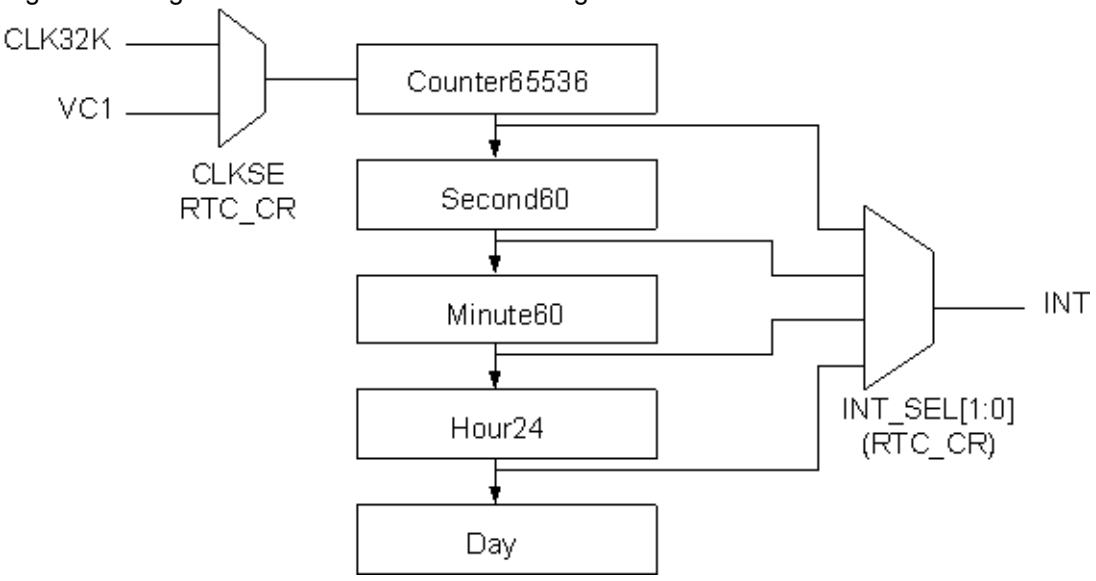
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C21x45, CY8C22x45, CY8C28x45, CY8C28xxx						
RTC	0	0	0	166	0	0

Features and Overview

- Real time clock that keeps time with an external 32K crystal oscillator
- Flexible interrupt sources between second, minute, hour, and day
- Hour, Minute and Second time read and write in BCD format
- Normal timer if using VC1 as clock source
- Sleep mode with internal or external 32K clock source
- Reset by PPOR, IPOR, and watchdog reset

The Real Time Clock User Module provides a timer without firmware maintenance. This user module supports the Hour:Minute:Second format. You can get a time display by reading data from the related registers. Interrupts may be generated based on the value of configurable parameters. The Real Time Clock User Module supports two modes depending on the clock source, a general timer or a real time clock.

Figure 1. Figure 1. Real Time Clock Block Diagram



Functional Description

The Real Time Clock User Module employs the hardware RTC. There are three counters in the RTC block to count the hour, minute, and second in BCD format. The second counter increases by 1 for every 32768 clock periods. The minute counter increases by 1 for every 60 seconds. The hour counter increases by 1 for every 60 minutes. Every 24 hours there is an optional day interrupt. The Real Time Block has two choices for clock source; VC1 or CLK32K. When VC1 is selected as the clock input, the Real Time Clock block can be used as a fixed period timer based on the VC1 period.

There are three registers in the Real Time Clock block that correspond to the hour, minute and second: RTC_H, RTC_M, and RTC_S. They are all readable and writable. Firmware can write a BCD value for hour, minute and second.The legal range for RTC_H is 0 to 23, and for RTC_M and RTC_S is 0 to 59.

You can read time from the registers in synchronous or nonsynchronous mode.

- Sync read ensures that the HH:MM:SS data is synchronized to the time of RTC_H read operation. When sync read is enabled, the read of RTC_H latches the current minute and second value into a buffer to ensure that the MM:SS are synchronized to the time of RTC_H read operation.
- If sync read is disabled, the read value of HH:MM:SS is aligned to each register's read time independently. In other words, if you read the hours register at 23:59:59, the minutes register may be 00 when you read it with the next instruction.

Placement

The Real Time Clock User Module occupies the RTC Block. Multiple placement of the Real Time Clock User Module is not possible.

Parameters and Resources

Clock

The Clock parameter sets the current clock source.This parameter has the following selections:

Clock	Description
VC1	The Real Time Clock User Module is used as a general timer.
32KHz	The Real Time Clock User Module is used as a real time clock module.

InterruptType

The InterruptType parameter sets the interrupt period.This parameter has the following selections:

InterruptType	Description
1 second	An interrupt occurs each second.
1 minute	An interrupt occurs each minute.
1 hour	An interrupt occurs each hour
1 day	An interrupt occurs each day

SynchronousRead

The SynchronousRead parameter sets the current methods of reading.This parameter has the following selections:

SynchronousRead	Description
Disable	The read value of HH:MM:SS is aligned to each register's read time independently.
Enable	The read of RTCH_REG latches the current minute and second value into a buffer, therefore ensure that the MM:SS are synchronized to the time of RTCH_REG read operation.

Application Programming Interface

The Application Programming Interface (API) functions are provided as part of the user module to allow you to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the include files.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns RTC_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable, and constant symbol. In the following descriptions, the instance name has been shortened to RTC for simplicity.

Note

** In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This "registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must also ensure their code observes the policy. Though some user module API functions may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

RTC_Start

Description:

Enables the RTC Block and starts the real time clock.

C Prototype:

```
void  RTC_Start(void);
```

Assembly:

```
lcall  RTC_Start
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_Stop

Description:

Disables the RTC Block and stops the real time clock.

C Prototype:

```
void  RTC_Stop(void);
```

Assembly:

```
call  RTC_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_EnableInt

Description:

Enables the RTC Block interrupt.

C Prototype:

```
void  RTC_EnableInt(void);
```

Assembly:

```
lcall  RTC_EnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_DisableInt

Description:

Disables the RTC Block interrupt.

C Prototype:

```
void  RTC_DisableInt(void);
```

Assembly:

```
lcall  RTC_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_ClearInt

Description:

Clears the posted RTC interrupt.

C Prototype:

```
void  RTC_ClearInt(void);
```

Assembly:

```
lcall  RTC_ClearInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_SetIntPeriod

Description:

Configures the interrupt period of the Real Time Clock module.

C Prototype:

```
void  RTC_SetIntPeriod(BYTE bConfiguration);
```

Assembly:

```
mov   A,bConfiguration
lcall  RTC_SetIntPeriod
```

Parameters:

bConfiguration is the interrupt period. Symbolic names are provided in C and assembly. Their associated values are given in the following table:

Symbolic Name	Value	Description
RTC_INT_SEC	0x00	Sets the interrupt period to 1 second
RTC_INT_MIN	0x04	Sets the interrupt period to 1 minute
RTC_INT_HOUR	0x08	Sets the interrupt period to 1 hour
RTC_INT_DAY	0x0C	Sets the interrupt period to 1 day

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

RTC_bReadSecond

Description:

Reads the second value in BCD format.

C Prototype:

```
BYTE  RTC_bReadSecond(void);
```

Assembly:

```
lcall  RTC_bReadSecond
```

Parameters:

None

Return Value:

Returns the second data in BCD format.

Side Effects:

See Note ** at the beginning of the API section.

RTC_bReadMinute

Description:

Reads the minute value in BCD format.

C Prototype:

```
BYTE  RTC_bReadMinute(void);
```

Assembly:

```
lcall  RTC_bReadMinute
```

Parameters:

None

Return Value:

Returns the minute data in BCD format.

Side Effects:

See Note ** at the beginning of the API section.

RTC_bReadHour

Description:

Reads the hour value in BCD format.

C Prototype:

```
BYTE  RTC_bReadHour(void);
```

Assembly:

```
lcall  RTC_bReadHour
```

Parameters:

None

Return Value:

Returns the hour data in BCD format.

Side Effects:

See Note ** at the beginning of the API section.

RTC_SetSecond

Description:

Sets the second value.

C Prototype:

```
void  RTC_SetSecond(BYTE bSecond);
```

Assembly:

```
mov A,bSecond
lcall  RTC_SetSecond
```

Parameters:

bSecond is the second data. The legal range for writing the second value is 0 to 59. This data must be in BCD format

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

RTC_SetMinute

Description:

Sets the minute value.

C Prototype:

```
void  RTC_SetMinute(BYTE bMinute);
```

Assembly:

```
mov A,bMinute
lcall  RTC_SetMinute
```

Parameters:

bMinute is the minute data. The legal range for writing the minute value is 0 to 59. This data must be in BCD format.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

RTC_SetHour

Description:

Set the hour value.

C Prototype:

```
void  RTC_SetHour(BYTE bHour);
```

Assembly:

```
mov A,bHour
lcall  RTC_SetHour
```

Parameters:

bHour is the hour data. The legal range for writing the hour value is 0 to 23. This data must be in BCD format.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

Sample Firmware Source Code

The C code illustrated here shows you how to use the RTC User Module.

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

BYTE abRTCData[3];
void main(void)
{

RTC_SetHour(0x11); // Set 11 hour as initial hour value
RTC_SetMinute(0x20); // Set 20 minutes as initial minute value
RTC_SetSecond(0x0); // Set 0 seconds as initial second value
RTC_SetIntPeriod(RTC_INT_SEC); // Set 1 sec as interrupt period
RTC_EnableInt(); // Enable RTC interrupt
RTC_Start(); // Start RTC block
```

```
M8C_EnableGInt; // Enable global interrupt

while(1)
{
abRTCData[0] = RTC_bReadHour(); // Read current hour value in BCD
abRTCData[1] = RTC_bReadMinute(); //Read current minute value in BCD
abRTCData[2] = RTC_bReadSecond(); //Read current second value in BCD
}
}
```

The same code in assembly is:

```
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

_abRTCData:
abRTCData:               BLK   3

export  main
export _main

_main:
main:
mov A, 0x11
call RTC_SetHour ;Set 11 hour as initial hour value
mov A, 0x20
call RTC_SetMinute ;Set 20 minutes as initial minute value
mov A,0x0
call RTC_SetSecond ;Set 0 seconds as initial second value
mov A, RTC_INT_SEC
call RTC_SetIntPeriod ;Set 1 sec as interrupt period
call RTC_EnableInt ;Enable RTC interrupt
call RTC_Start ;Start RTC block

M8C_EnableGInt ;Enable global interrupt
.ReadRTCData:
call RTC_bReadHour ;Read current hour value in BCD
RAM_SETPAGE_CUR >abRTCData
mov [abRTCData],A
call RTC_bReadMinute ;Read current minute value in BCD
RAM_SETPAGE_CUR >abRTCData
mov [abRTCData+1],A
call RTC_bReadSecond ;Read current second value in BCD
RAM_SETPAGE_CUR >abRTCData
mov [abRTCData+2],A
jmp .ReadRTCData
```

Configuration Registers

This register is used to read and write the current hour value in BCD format. HR_DEC bits represent the decal number of hour value. HR_UNIT bits represent the units number of hour value.The register value can be changed by calling the RTC_SetHour API.

Table 1. RTCM_REG

Bit	7	6	5	4	3	2	1	0
Value	0	MIN_DEC			MIN_UNIT			

This register is used to read and write the current minute value in BCD format. MIN_DEC bits represent the decal number of minute value. MIN_UNIT bits represent the units number of minute value. The register value can be changed by calling the RTC_SetMinute API.

Table 2. RTCS_REG

Bit	7	6	5	4	3	2	1	0
Value	0	SEC_DEC			SEC_UNIT			

This register is used to read and write the current second value in BCD format. SEC_DEC bits represent the decal number of second value. SEC_UNIT bits represent the units number of second value. The register value can be changed by calling the RTC_SetSecond API.

Table 3. RTCCR_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	InterruptEn	ClkSelect	InterruptSelect		Sync	Enable

The Enable bit is modified by calling the RTC_Start or RTC_Stop API routine.

The Sync bit determines the reading method. The value of this bit is determined by the choice made for the "SynchronousRead" parameter under user module parameters in the Device Editor.

The InterruptSelect bits determine the interrupt period. The value of these bits are determined by the choice made for the "InterruptType" parameter under user module parameters in the Device Editor.The value can also be changed by calling the RTC_SetIntPeriod API.

The ClkSelect bit determine the clock source. The value of this bit is determined by the choice made for the "Clock" parameter under user module parameters in the Device Editor.

The InterruptEn bit is modified by calling the RTC_EnableInt or RTC_DisableInt API routine.

Version History

Version	Originator	Description
1.0	DHA	Initial version
1.10	HPHA	Corrected method of clearing posted interrupts.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.