实时时钟数据表 RTC V 1.0

Copyright © 2009-2010 Cypress Semiconductor Corporation. All Rights Reserved.

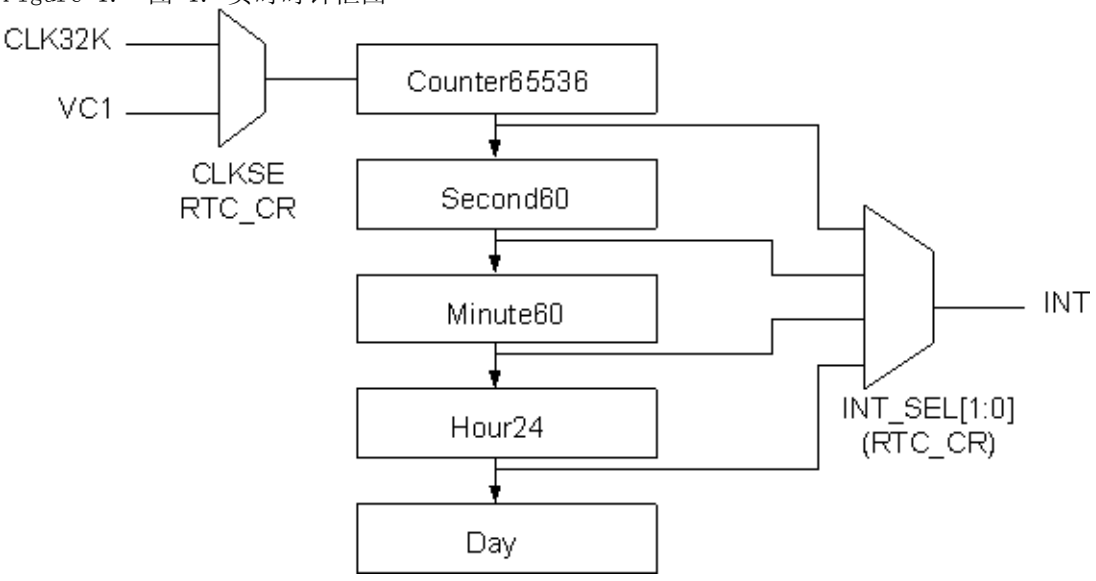
资源	PSoC® 模块			API 存储器 （字节）		引脚 （每个外部 I/O）
	数字	模拟 CT	模拟 SC	闪存	RAM	
CY8C21x45, CY8C22x45, CY8C28x45, CY8C28xxx						
RTC	0	0	0	166	0	0

功能和概述

- 实时时钟使用外部 32K 晶体振荡器记录时间
- 在秒、分钟、小时及日之间提供灵活的中断源
- 以 BCD 格式进行小时、分钟、秒时间读写
- 使用 VC1 作为时钟源时为常规定时器
- 睡眠模式采用内部或外部 32K 时钟源
- 通过 PPOR、IPOR 和看门狗进行复位

实时时钟用户模块提供无需固件维护的定时器。 此用户模块支持 小时：分钟：秒 时间格式。 您可以通过从相关寄存器中读取数据获得时间。 可以基于可配置参数值生成中断。 实时时钟用户模块根据不同的时钟源支持两种模式：普通定时器或实时时钟。

Figure 1. 图 1. 实时时钟框图



功能说明

实时时钟用户模块使用硬件 RTC。 RTC 模块中有三个计数器以 BCD 格式分别计算小时、分钟、秒。 秒计数器每 65536 次时钟周期增加 1 秒。 分钟计数器每 60 秒增加 1 分钟。 小时计数器每 60 分钟增加 1 小时。 每 24 小时为一个可选的日中断。 实时时钟模块可以在两个时钟源中进行选择，即 VC1 或 CLK32K。 以 VC1 作为时钟源输入时，实时时钟模块可以根据 VC1 期间用作固定期间的定时器。

实时时钟模块中有三个寄存器，分别对应小时、分钟和秒： RTC_H，RTC_M 和 RTC_S.。这些寄存器都可以读写。 固件可以以 BCD 格式设置小时、分钟和秒。RTC_H 的合法范围是 0 到 23，RTC_M 和 RTC_S 的是 0 到 59。

您可以以同步或非同步模式从寄存器中读取时间。

- 同步读取可以确保 HH:MM:SS 数据与 RTC_H 读操作的时间保持同步。 启用同步读取时，RTC_H 读取会将当前分、秒值锁存到缓冲区中，以确保 MM:SS 数据与 RTC_H 读操作的时间保持同步。
- 如果禁用同步读取，HH:MM:SS 的读取值将分别与每个寄存器的读取时间保持一致。 也就是说，如果在 23:59:59 读小时寄存器，那么在下一个指令读取分钟寄存器时，分钟寄存器的值就可能为 00。

放置

实时时钟用户模块会占用 RTC 模块。 无法放置多个实时时钟用户模块。

参数和资源

时钟

“ 时钟 ” 参数设置当前时钟源。此参数包含以下选择：

时钟	说明
VC1	实时时钟用户模块用作普通定时器。
32KHz	实时时钟用户模块用作实时时钟模块。

中断类型

“ 中断类型 ” 参数设置中断期间。此参数包含以下选择：

中断类型	说明
1 秒	每秒发生一次中断。
1 分钟	每分钟发生一次中断。
1 小时	每小时发生一次中断。
1 天	每天发生一次中断。

同步读取

“ 同步读取 ” 参数设置当前的读取方法。此参数包含以下选择：

[+] Feedback

同步读取	说明
禁用	HH:MM:SS 读取值分别与每个寄存器的读取时间保持一致。
启用	RTCH_REG 读取将当前分钟、秒值锁存到缓冲区中，以确保 MM:SS 数据与 RTCH_REG 读操作的时间保持同步。

应用程序编程接口

应用程序编程接口 (API) 函数作为用户模块的一部分提供，从而能够采用更高级的方式处理模块。 本节指定每个函数的接口，以及头文件所提供的相关常量。

每次放置用户模块时，都会为其分配一个实例名称。 默认情况下，PSoC Designer 会为指定项目中此用户模块的第一个实例分配 RTC_1。 可将该值更改为符合标识符语法规则的任意唯一值。 分配的实例名称成为每个全局函数名称、变量和常量符号的前缀。 为简便起见，在以下说明中将实例名称缩写为 RTC。

Note ** 在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能因调用 API 函数被更改。 如果在调用后需要 A 和 X 的值，则调用函数负责在调用前保存 A 和 X 的值。 此 “ 寄存器易失 ” 策略是针对提高效率的目的选择，自 PSoC Designer 1.0 版起已强制使用此策略。 C 编译器自动遵循此要求。 汇编语言编程人员也必须确保他们的代码符合此策略。 虽然一些用户模块 API 函数可以保留 A 和 X 不变，但是无法保证它们将来也会如此。

RTC_Start

说明：

启用 RTC 模块与启动实时时钟。

C 原型：

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

汇编：

```
lcall  RTC_Start
```

参数：

无

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

RTC_Stop

说明：

禁用 RTC 模块与停止实时时钟。

C 原型：

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

汇编：

```
call  RTC_Stop
```


[+] Feedback

参数：

无

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

RTC_EnableInt

说明：

启用 RTC 模块中断。

C 原型：

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

汇编：

```
lcall  RTC_EnableInt
```

参数：

无

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

RTC_DisableInt

说明：

禁用 RTC 模块中断。

C 原型：

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

汇编：

```
lcall  RTC_DisableInt
```

参数：

无

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

RTC_ClearInt

说明：

清除已发布的 RTC 中断。

C 原型：

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

汇编：

```
lcall  RTC_ClearInt
```

参数：

无

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

RTC_SetIntPeriod

说明：

配置实时时钟模块的中断周期。

C 原型：

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

汇编：

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

参数：

“bConfiguration” 代表中断周期。 符号名以 C 和汇编语言形式提供。 相关值在以下表格中给出：

符号名	值	说明
RTC_INT_SEC	0x00	将中断周期设置为 1 秒钟
RTC_INT_MIN	0x04	将中断周期设置为 1 分钟
RTC_INT_HOUR	0x08	将中断周期设置为 1 小时
RTC_INT_DAY	0x0C	将中断周期设置为 1 天

返回值：

无

副作用：

请参阅 API 一节开头的注意 **。

[+] Feedback

RTC_bReadSecond

说明：

以 BCD 格式读取秒值。

C 原型：

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

汇编：

```
lcall  RTC_bReadSecond
```

参数：

无

返回值：

以 BCD 格式返回秒数据。

副作用：

请参阅 API 一节开头的注意 **。

RTC_bReadMinute

说明：

以 BCD 格式读取分钟值。

C 原型：

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

汇编：

```
lcall  RTC_bReadMinute
```

参数：

无

返回值：

以 BCD 格式返回分钟数据。

副作用：

请参阅 API 一节开头的注意 **。

RTC_bReadHour

说明：

以 BCD 格式读取小时值。

C 原型：

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

汇编：

```
lcall  RTC_bReadHour
```

参数：

无

返回值：

以 BCD 格式返回小时数据。

副作用：

请参阅 API 一节开头的注意 **。

RTC_SetSecond

说明：

设置秒值。

C 原型：

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

汇编：

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

参数：

“bSecond”代表秒数据。秒写入值的合法范围是 0 到 59。此数据必须为 BCD 格式。

返回值：

无。

副作用：

请参阅 API 一节开头的注意 **。

RTC_SetMinute

说明：

设置分钟值。

C 原型：

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

汇编：

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

参数：

“bMinute”代表分钟数据。分钟写入值的合法范围是 0 到 59。此数据必须为 BCD 格式。

返回值：

无。

副作用：

请参阅 API 一节开头的注意 **。

RTC_SetHour

说明：

设置小时值。

[+] Feedback

C 原型：

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

汇编：

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

参数：

“bHour” 代表小时数据。 小时写入值的合法范围是 0 到 23。 此数据必须为 BCD 格式。

返回值：

无。

副作用：

请参阅 API 一节开头的注意 **。

固件源代码示例

此处所示 C 代码向您说明如何使用 RTC 用户模块。

```
#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
}
}
```


[+] Feedback

同一代码用汇编语言表示为：

```
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
```

配置寄存器

此寄存器用来以 BCD 格式读写当前小时值。HR_DEC 位代表小时值的十位数。HR_UNIT 位代表小时值的个位数。寄存器值可通过调用 RTC_SetHour API 进行更改。

Table 1. RTCM_REG

位	7	6	5	4	3	2	1	0
值	0	MIN_DEC			MIN_UNIT			

此寄存器用来以 BCD 格式读写当前分钟值。MIN_DEC 位代表分钟值的十位数。MIN_UNIT 位代表分钟值的个位数。 寄存器值可通过调用 RTC_SetMinute API 进行更改。

Table 2. RTCS_REG

位	7	6	5	4	3	2	1	0
值	0	SEC_DEC			SEC_UNIT			

此寄存器用来以 BCD 格式读写当前秒值。SEC_DEC 位代表秒值的十位数。SEC_UNIT 位代表秒值的个位数。寄存器值可通过调用 RTC_SetSecond API 进行更改。

Table 3. RTCCR_REG

位	7	6	5	4	3	2	1	0
值	0	0	中断启用	时钟选择	中断选择		同步	启用

“ 启用 ” 位通过调用 RTC_Start 或 RTC_Stop API 函数进行更改。

“ 同步 ” 位决定读方法。 此位值取决于在器件编辑器中的用户模块参数选择了哪个 “ 同步读取 ” 参数。

“ 中断选择 ” 位决定中断期间。 这些位值取决于在器件编辑器中的用户模块参数选择了哪个 “ 中断类型 ” 参数。该值也可以通过调用 RTC_SetIntPeriod API 进行更改。

“ 时钟选择 ” 位决定时钟源。 此位值取决于在器件编辑器中的用户模块参数选择了哪个 “ 时钟 ” 参数。

“ 中断启用 ” 位通过调用 RTC_EnableInt 或 RTC_DisableInt API 函数进行修改。

[+] Feedback

版本历史记录

版本	创作者	说明
1.0	DHA	初始版本

Note PSoC Designer 5.1 在所有的用户模块数据表中提供版本历史记录。 本数据表详细介绍了当前和先前用户模块版本之间的区别。

Copyright © 2009-2010 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.