



MSP430F5 系列 16 位超低功耗单片机模块原理

第 14 章 RTC 实时时钟 A

版本: 1.1

发布日期: 2008.9. 最后更新日期:2010.8.

原文: TI slau208.pdf (5xxfamily User's Guide)

翻译: 周欣 南京信息工程大学

编辑: DC 微控网总版主

注: 以下文章是翻译 TI slau208.pdf 文件中的部分内容。由于我们翻译水平有限, 有整理过程中难免有所不足或错误; 所以以下内容只供参考. 一切以原文为准。

文章更新详情请密切留意微控技术论坛。

第 14 章 实时时钟 A

实时时钟模块提供了具有日历模式、灵活可编程闹钟和校准的时钟计数器。这一章节介绍了实时时钟 A 模块。实时时钟模块 A 执行于 MSP430X5XX 器件中。

14.1 实时时钟介绍

实时时钟模块提供了一个具有可以配置成一般目的计数器的日历时钟。

实时时钟特点有：

- 可配置成实时时钟模式或者一般目的的计数器
- 在日历模式中提供了秒钟，分钟，小时，星期，日期，月份和年份
- 具有中断能力
- 实时时钟模式里可选择 BCD 码或者二进制格式
- 实时时钟模式里具有可编程闹钟
- 实时时钟模式里具有时间偏差的逻辑校正

实时时钟框图见图 14-1。

注意：实时时钟初始化

实时时钟模块的大多数寄存器没有初始条件。在使用这个模块之前，用户必须通过软件对寄存器进行配置。

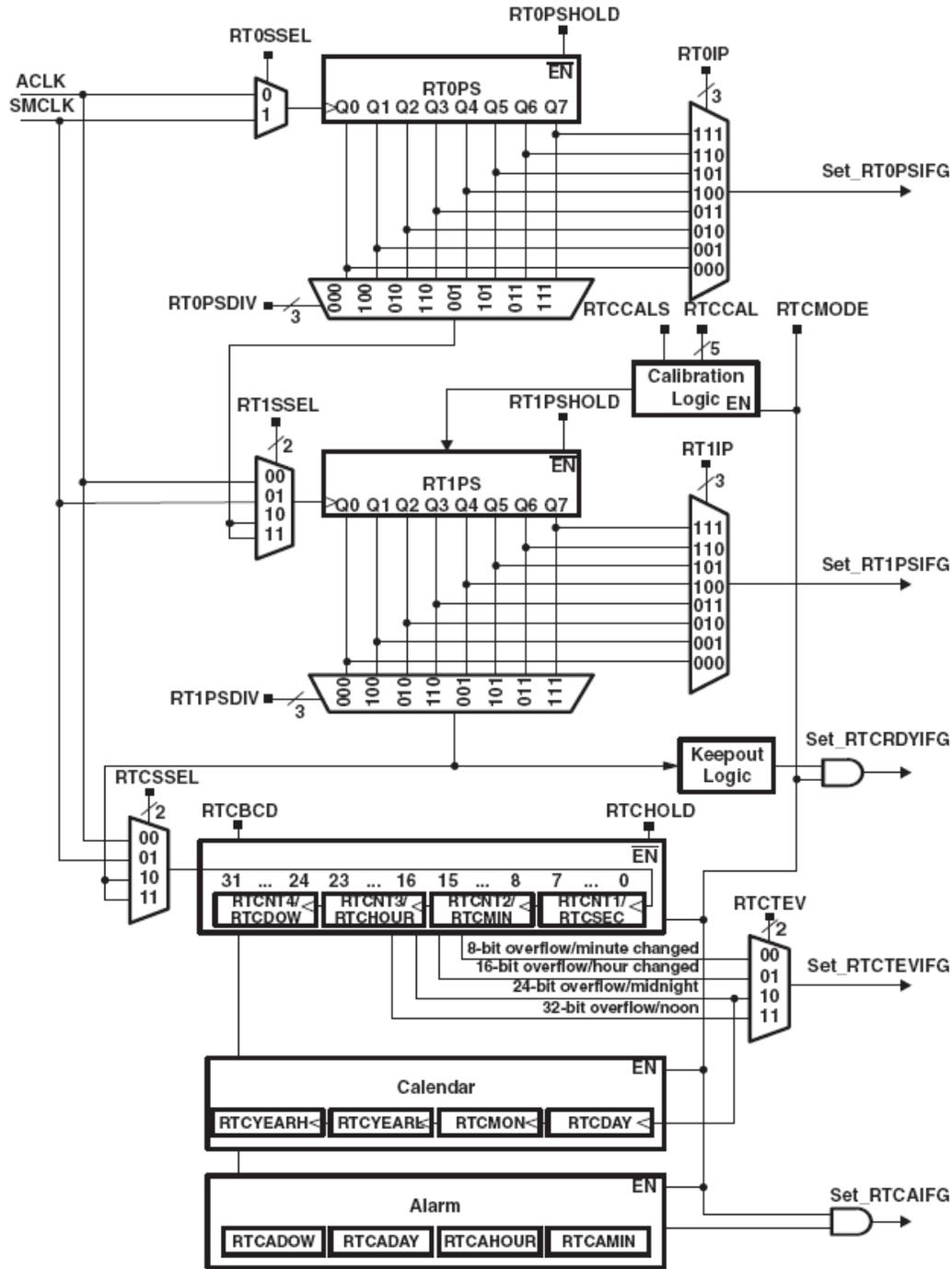


图 14-1 实时时钟

14.2 实时时钟操作

实时时钟模块可以被配置成具有日历作用的实时时钟或者是一个具有 RTCMODE 比特一般目的的 32 位计数器。

14.2.1 计数器模式

当 RTCMODE 被重置时，计数器模式被选择。在这个模式中，通过软件可以得到一个 32 位的计数器。从日历模式切换到计数器模式是通过重置计数值 (RCTNT1, RCTNT2, RCTNT3, RCTNT4)，

和预换算计数器（RT0PS，RT1PS）。

时钟的增量计数器可源于 ACLK、SMCLK 或者是分频之后的 ACLK 或 SMCLK。分频之后的 ACLK 或 SMCLK 源自分频除法器 RT0PS、RT1PS。RT0PS 和 RT1PS 分别能输出 ACLK 和 SMCLK 的 2 分频、4 分频、8 分频、16 分频、32 分频、64 分频、128 分频、256 分频。RT0PS 的输出可以与 RT1PS 进行级联。级联的输出可用作 32 位计数器的时钟源。

4 个独立的 8 位计数器级联成为 32 位的计数器。这能提供计数时钟的 8 位、16 位、24 位、32 位溢出间隔。RTCDEV 位选择各自的触发。通过设置 RTCDEVIE 位，一个 RTCDEV 发生能够触发一个中断。计数器 RTCNT1 到 RTCNT4，每一个都可以单独的访问，并可能被写入。

RT0PS和RT1PS可以被配置成两个8位的计数器，或者级联成一个16位的计数器。通过设置各自的 RT0PSHOLD和RT1PSHOLD位，RT0PS和RT1PS可以暂停功能设置成各自独立的模块。当RT0PS和RT1PS级联的时候，通过置位RT0PSHOLD可以同时停止RT0PS和RT1PS被暂停。根据不同的配置，32 位的计数器可以有不同的方法被停止。如果32位的计数器直接源于ACLK或者SMCLK，则通过置位 RTCHOLD而被停止；如果它是源于RT1PS的输出，则通过置位RT1PSHOLD或者RTCHOLD而被停止；最后，如果它源于RT0PS和RT1PS的级联，则通过置位RT0PSHOLD、RT1PSHOLD或者RTCHOLD而被停止。

注意：访问RTCNTx寄存器

当计数器的时钟与CPU时钟是异步的，当计数器不被操作的时候，对于RTCNTx、RT0PS、或者RT1PS的任一寄存器的任何读取应该发生。否则，结果将有可能是不可预知的。二者择一地，计数器在运行的时候应当被多次读取，软件通过绝大多数的结果来决定正确的读取结果。对于任一寄存器RTCNTx、RT0PS或者RT1PS的读取都会立即生效。

14.2.2 日历模式

当RTCMODE置位的时候，日历模式就被选中了。在日历模式中，实时时钟模块可选择以BCD码或者是十六进制提供秒、分、小时、星期、月份和年份。日历有能计算能否被4整除的闰年算法。这个算法可以精确到1901年到2099年。

14.2.2.1 实时时钟和预分频

分频器自动将RT0PS和RT1PS配置成为实时时钟提供一秒钟间隔的时钟。RT0PS源于ACLK。ACLK必须是32768Hz，名义上是为了实时时钟日历的运行。RT1PS与RT0PS的ACLK的256分频输出进行级联。实时时钟源于RT1PS的128分频输出，因而提供所需的间隔一秒的时钟。从日历模式切换到计数器模式时，会将秒、分、小时、星期、月份和年份全部置1。另外，RT0PS和RT1PS也会被置位。

当RTCBCD=1时，日历寄存器就会被选为BCD码格式。必须在时间设置之前选择好格式。改变RTCBCD的状态会将秒、分、小时、星期、月份和年份全部置1。另外，RT0PS和RT1PS也会被置位。在日历模式下，RT0SSEL、RT1SSEL、RT0PSDIV、RT1PSDIV、RT0PSHOLD、RT1PSHOLD和RTCSSSEL位都可以被忽略。置位RTCHOLD会停止实时计数器、分频计数器和RT0PS、RT1PS。

14.2.2.2 实时时钟的闹钟功能

实时时钟模块提供了一个灵活的闹钟系统。这个单独的、用户可编程控制的闹钟，在设置闹钟的秒、分、小时、星期、月份和年份寄存器的基础上进行编程设置。用户可编程闹钟功能只有在日历模式运行的时候才有效。

每一个闹钟寄存器都包括一个闹钟使能位，AE可用来使能每一个闹钟寄存器。通过设置各式各样闹钟寄存器的AE位，可以生成多种闹钟。

比如说，一个用户需要在每一小时的15分钟（也就是00:15:00、01:15:00、02:15:00等等时刻）进行一次闹钟。这只要将RTCAMIN设置成15即可实现上述功能要求。通过设置RTCAMIN的AE位和置位闹钟寄存器的所有其它AE位，就会使能闹钟。使能时，AF位就会在00:14:59到00:15:00、01:14:59到01:15:00、02:14:59到02:15:00等等时刻被置位。

再比如说，一个用户希望设一个闹钟在每天00:04:00时刻。将RTCAHOUR位置位成4即可实现上述功能。通过设置RTCHOUR的AE位和置位闹钟寄存器的所有其它AE位，就会使能闹钟。使能时，AF就会在03:59:59到04:00:00时刻被置位。

再举一个例子，一个用户希望设一个闹钟时刻在06:30:00，将RTCAHOUR设置成6，将RTCAMIN设置成30即可实现上述功能。通过设置RTCAHOUR和RTCAMIN的AE位，即可使能闹钟。一旦闹钟使能，AF位将会在每一个06:29:59到06:30:00的过渡时刻被置位。在种种情形下，每天的06:30:00都会闹钟发生。

比如说，一个用户希望在每一个星期二的06:30:00时刻闹钟一次。那么，RTCADOW位设置成2，RTCAHOUR设置成6，RTCAMIN将要被设置成30。通过设置RTCADOW、RTCAHOUR和RTCAMIN的AE位，闹钟即被使能。一旦使能，AF位将会在06:29:59到06:30:00的过渡时刻和RTCADOW位从1到2的过渡时刻被置位。

最后在举一例，一个用户希望在每一个月份的第五天的06:30:00时刻进行一次闹钟。那么，RTCADAY位将要设置成5，RTCAHOUR位将要被设置成6，RTCAMIN位将要被设置成30。通过设置RTCADAY位、RTCAHOUR位和RTCAMIN位的AE位，闹钟即被使能。一旦使能，AF位将要在06:29:59到06:30:00的过渡时刻和RTCADAY等于5的时刻被置位。

注意：无效的闹钟设置

无效的闹钟设置不会通过硬件的检测。用户有责任将闹钟设置正确。

注意：无效的时间和日期设置

指定的合法范围之外的无效日期、时间信息或者是日期值写入到RTCSEC、RTCMIN、RTCHOUR、RTCDAY、RTCDOW、RTCYEARH、RTCYEARL、RTCAMIN、RTCAHOUR、RTCADAY和RTCADOW寄存器将会导致不可预知的结果。

注意：设置闹钟

为了防止发生潜在的错误的闹钟情形，在设置新的时间值到实时时钟寄存器之前，应当置位RTCAIE、RTCAIFG和AE位来使闹钟失效。

14.2.2.3 在日历模式下读写实时时钟寄存器

因为系统时钟实际上是和实时时钟的时钟源是异步的，在进入实时时钟寄存器的时候要格外小心。

在日历模式下，实时时钟寄存器每秒钟更新一次。为了防止在更新的时候读取实时时钟数据而造成错误数据的读取，将会有有一个禁止进入的窗口。这个禁止进入的窗口会在更新期间左右的128/32768中心时间里。在禁止进入窗口期间和窗口外期间，只有RTCRDY复位才可以读。在RTCRDY复位的时候，对时钟寄存器的任何读取都被认为是潜在错误的，并且时间读取是被忽略的。

一个简单而安全读取实时时钟寄存器的方法是利用RTCRDYIFG中断标志位。设置RTCRDYIE使能RTCRDYIFG中断。一旦中断使能，在RTCRDY位上升沿的时候将会产生中断，致使RTCRDYIFG被置位。在这一点上，这一应用几乎有一秒钟安全地去读取任一个或者所有的实时时钟寄存器。这一同步化的进程防止在过渡时间读取时间值。当中断得到响应的时候，RTCRDYIFG会自动复位，也可以软件复位。

在计数器模式下，RTCRDY位保持复位。可以不关心RTCRDYIE位，并且RTCRDYIFG维持复位。

注意：读写实时时钟寄存器

当计数器时钟和CPU时钟异步的时候，RTCRDY位复位时，对RTCSEC、RTCMIN、RTCHOUR、RTCDOW、RTCDAY、RTCMON、RTCYEARL和RTCYEARH任一寄存器的任何读取都有可能错误的数据被读取。为了安全地读取计数器数据，多次读取RTCRDY位取平均值和同步读取操作等上述方法都可被使用。作为选择地，在运行时，可以多次读取计数器寄存器，由软件多次读取来确定正确的数据。读取RTOPS和RT1PS位不仅可以通过多次读取计数器寄存器和软件多次读取来确定正确的数据，还可以暂定计数器。

对任何计数寄存器的写操作都是瞬时有效的。当然，在写的过程中，计数器是停止的。另外，RT0PS和RT1PS寄存器复位。这有可能导致在写操作的过程中丢失近一秒钟。合法数据范围之外的写操作和不正确的时间结合将会导致不可预见的行为结果。

14.2.3 实时时钟中断

实时时钟模块有5个中断源，每一个中断源都有独立的使能位和标志位。

14.2.3.1 日历模式中的实时时钟中断

在日历模式中，有5个中断源是可用的，分别是RT0PSIFG、RT1PSIFG、RTCRDYIFG、RTCTEVIFG和RTCAIFG。这些中断标志具有优先次序，结合指向独立的中断向量。中断向量寄存器RTCIV用来决定哪一个中断标志被请求。

最高优先级使能中断在RTCIV寄存器（见寄存器描述）里产生一个数字。这个数字能够被评估或者加载到编程计数器来自动地进入相应的软件程序中去。使实时时钟无效不会影响RTCIV中的值。

任何访问、读或者写RTCIV寄存器都会自动复位最高位未决定的中断标志位。如果一个中断标志位被设置了，在相应完最初的中断之后，另一个中断会立即被生成。另外，所有的标志位都可以通过软件清除。

用户可编程闹钟发生源自实时时钟中断RTCAIFG。设置RTCAIE使能中断。用户可编程闹钟，实时时钟模块还提供了源于实时时钟中断的一个间隔闹钟RTCTEVIFG。当RTCMIN、RTCHOUR位变化或者每天午夜（00: 00: 00）、中午（12: 00: 00）时，间隔闹钟可被选中来引起闹钟的发生。事件发生是可以由RTCTEV位设定RTCTEVIE位来使能中断。

RTCRDY位源自实时时钟的中断和RTCRDYIFG，在系统时钟下同步读取时间寄存器很有用。设置RTCRDYIE位使能中断。

通过RT0IP位，可以选择地让RT0PSIFG位用来生成间接中断。在日历模式下，RT0PS的时钟源是32768Hz的ACLK，所以可以产生16384Hz、8192Hz、4096Hz、2048Hz、1024Hz、512Hz、256Hz和128Hz的时间间隔。设置RT0PSIE位可以使能中断。

通过RT1IP位，可以选择地让RT1PSIFG位用来生成间接中断。在日历模式中，RT1PS位源自RT0PS位的输出128Hz（32768/256 Hz）。所以，可以产生64 Hz, 32 Hz, 16 Hz, 8 Hz, 4 Hz, 2 Hz, 1 Hz或者0.5Hz的间隔。设置RT1PSIE位可以使能中断。

14.2.3.2 计数器模式中的实时时钟中断

在计数器模式中，三个中断源是可用的，分别是RT0PSIFG、RT1PSIFG和RTCTEVIFG。RTCAIFG位和RTCRDYIFG位被清除。RTCRDYIE和RTCAIE位可以忽略。

通过RT0IP位，可以选择地让RT0PSIFG位用来生成间接中断。在计数器模式，RT0PS位源自ACLK或者SMCLK，所以是各个可用时钟源的2分频、4分频、8分频、16分频、32分频、64分频、128分频和256分频。设置RT0PSIE位可以使能中断。

通过RT1IP位，可以选择地让RT1PSIFG位用来生成间接中断。在计数器模式下，RT1PS位源于ACLK、SMCLK或者是RT0PS位的输出，所以是各个可用时钟源的2分频、4分频、8分频、16分频、32分频、64分频、128分频和256分频。设置RT1PSIE位可以使能中断。

实时时钟模式提供了一个源于实时时钟中断的时间间隔测量器RTCTEVIFG。选择时间间隔测量器，在32位计数器里，当8位、16位、24位或者32位溢出时产生中断。当RTCTEVIE位使能中断时，可通过RTCTEV位来选择。

实时时钟模块例程

以下的例程显示显示了实时时钟典型的用法，和招收之前的学习。RTCIV值加载到PC机是为了自动跳转到相应的程序中去。

每条语句右边边缘显示了CPU的周期。不同中断程序的软件总开销包括潜在的中断和从中断状态返回的周期，但不包括任务本身处理。

```

; Interrupt handler for RTC interrupt flags.
RTC_HND          ; Interrupt latency 6
  ADD &RTCIV,PC  ; Add offset to Jump table 3
  RETI           ; Vector 0: No interrupt 5
  JMP RTCRDYIFG_HND ; Vector 2: RTCRDYIFG 2
  JMP RTCTEVIFG_HND ; Vector 4: RTCTEVIFG 2
  JMP RTCAIFG     ; Vector 6: RTCAIFG 5
  JMP RT0PSIFG   ; Vector 8: RT0PSIFG 5
  JMP RT1PSIFG   ; Vector A: RT1PSIFG 5
RETI             ; Vector C: Reserved 5

RTCRDYIFG_HND   ; Vector 2: RTCRDYIFG Flag
  to             ; Task starts here
  RETI           5

RTCTEVIFG_HND   ; Vector 4: RTCTEVIFG
  To             ; Task starts here
  RETI           ; Back to main program 5

RTCAIFG_HND     ; Vector 6: RTCAIFG
  To             ; Task starts here

RT0PSIFG_HND   ; Vector 8: RT0PSIFG
  To             ; Task starts here

RT1PSIFG_HND   ; Vector A: RT1PSIFG
  To             ; Task starts here

```

14.2.4 实时时钟校准

实时时钟具有校准逻辑，它会校准精确到标准晶体振荡的-2ppm到+4ppm。

RTCCALx位是用来校准频率的。当RTCCALS位设置好之后，每一个RTCCALx低位会导致+4ppm的校准。当RTCCALS位被清除，每一个RTCCALx低位会导致-2ppm的校准。

为了校准频率，在一个引脚上RTCCLK输出信号是有用的。RTCCALF位可以用来选择输出信号的频率比率。在校准过程中，RTCCLK位是可以被测量的。测量的结果可以应用到RTCCALS和RTCCALx位用来降低时钟的最初补偿。比如说，假使RTCCLK位的输出频率是512Hz。RTCCLK位的测量值是511.9658Hz。这里的频率误差大约是很低的67ppm。为了提高67ppm的频率误差，RTCCALS位将要被设置，并且RTCCALx位也要被设置成17（67/4）。

在计数器模式下（RTCMODE = 0），校准逻辑是失效的。

注意：校准输出频率

校准设置发生改变时，在RTCCLK引脚观察512Hz和256Hz的输出频率是不会有影响的。而校准发生改变时，1Hz的输出频率是有影响的。

14.3 实时时钟寄存器

实时时钟寄存器见表14-1。其中一些寄存器可以按字母顺序存取见表14-2。实时时钟最基本的寄存器可以在器件手册里找到。地址在14-1和14-2理都已标明。

表14-1 实时时钟寄存器

寄存器	简写	寄存器类型	相对地址	初始状态
RTC控制寄存器0	RTCCTL0	读/写	00h	00h
RTC控制寄存器1	RTCCTL1	读/写	01h	40h

RTC控制寄存器2	RTCCTL2	读/写	02h	00h
RTC控制寄存器3	RTCCTL3	读/写	03h	00h
时实预分频定时器0控制寄存器	RTCPS0CTL	读/写	08h	10h
时实预分频定时器1控制寄存器	RTCPS1CTL	读/写	0Ah	10h
时实预分频定时器0	RTCPS0	读/写	0Ch	无变化的
时实预分频定时器1	RTCPS1	读/写 ^e	0Dh	无变化的
RTC中断向量寄存器	RTCIV	读	0Eh	00h
RCT秒计数寄存器1	RTCSEC/RTCNT1	读/写	10h	无变化的
RCT分计数寄存器2	RTCMIN/RTCNT2	读/写	11h	无变化的
RCT时计数寄存器3	RTCHOUR/RTCNT3	读/写	12h	无变化的
RCT星期(日)计数寄存器4	RTCDOW/RTCNT4	读/写	13h	无变化的
RTC 日(月)	RTCDAY	读/写	14h	无变化的
RTC月	RTCMON	读/写	15h	无变化的
RTC 年(低字节)	RTCYEARL	读/写	16h	无变化的
RTC 年(高字节)	RTCYEARH	读/写	17h	无变化的
RTC分钟闹钟	RTCAMIN	读/写	18h	无变化的
RTC时钟闹钟	RTCAHOUR	读/写	19h	无变化的
RTC星期(日)闹钟	RTCADOW	读/写	1Ah	无变化的
RTC月(日)闹钟	RTCADAY	读/写	1Bh	无变化的

表14-2 计数器模式下的Word Access to Registers

字寄存器	简写	高字节寄存器	低字节寄存器	地址
实时控制寄存器0, 1	RTCCTL01	RTCCTL1	RTCCTL0	00h
实时控制寄存器2, 3	RTCCTL23	RTCCTL3	RTCCTL2	02h
时实预分频定时器0 控制	RTCPS0CTL	RTCPS0CTLH	RTCPS0CTLL	08h
时实预分频定时器1 控制	RTCPS1CTL	RTCPS1CTLH	RTCPS1CTLL	0Ah
时实预分频定时器	RTCPS	RTCPS1	RTCPS0	0Ch
RTC中断向量	RTCIV			0Eh
RTC定时器0 实时计数寄存器1, 2	RTCTIM0/RTCNT12	RTCMIN/RTCNT2	RTCSEC/RTCNT1	10h
RTC定时器0 实时计数寄存器3, 4	RTCTIM1/RTCNT34	RTCDOW/RTCNT4	RTCHOUR/RTCNT3	12h
RTC 日期	RTCDATE	RTCMON	RTCDAY	14h
RTC 年	RTCYEAR	RTCYEARH	RTCYEARL	16h
RTC分/时闹钟	RTCAMINHR	RTCAHOUR	RTCAMIN	18h
RTC日/周闹钟	RTCADOWDAY	RTCADAY	RTCADOW	1Ah

RTCCTL0、实时时钟控制寄存器0

7	6	5	4	3	2	1	0
Reserved	RTCTEVIE	RTCAIE	RTC RDYIE	Reserved	RTCTEVIFG	RTCAIFG	RTC RDYIFG
r0	rw-0	rw-0	rw-0	r0	rw-(0)	rw-(0)	rw-(0)

Reserved Bit 7 保留，读总为0

RTCTEVIE Bit 6 实时时钟-时间事件中断使能

		0 中断不被使能 1 中断被使能
RTCAIE	Bit 5	实时时钟-闹钟中断使能，在计数器模式时被清除（RTCMODE = 0） 0 中断不被使能 1 中断使能
RTCRDYIE	Bit 4	实时时钟-闹钟中断使能 0 中断不被使能 1 中断使能
Reserved	Bit 3	保留，读总为0
RTCTEVIFG	Bit 2	实时时钟-时间事件标志 0 没有时间事件发生 1 时间事件发生
RTCAIFG	Bit 1	实时时钟-闹钟标志位，在计数器模式时被清除（RTCMODE = 0） 0 没有时间事件发生 1 时间事件发生
RTCRDYIFG	Bit 0	实时时钟-闹钟标志位 0 实时时钟不能被安全读取 1 实时时钟能被安全读取

RTCCTL1、实时时钟控制寄存器1

7	6	5	4	3	2	1	0
RTCBCD	RTCHOLD	RTCMODE	RTCRDY	RTCSSEL		RTCTEV	
rw-(0)	rw-(1)	rw-(0)	r-(0)	rw-0	rw-0	rw-(0)	rw-(0)

RTCBCD Bit 7 实时时钟BCD码格式选择位，选择实时时钟的BCD格式只能应用于日历模式（RTCMODE = 1），在计数器模式中会被忽略设置。改变这个位会将秒、分、小时、星期和年清零，将日期和月份置1。实时时钟寄存器需要后来被软件设置。

- 0 选择2进制或者十六进制
- 1 选择BCD码

RTCHOLD Bit 6 实时时钟保持位
0 实时时钟（32位计数器或者是日历模式）正在运作
1 计数器模式（RTCMODE = 0）只在32位计数器停止时；在日历模式（RTCMODE = 1）日历停止，以及预分频计数器、RT0PS和RT1PS。RT0PSHOLD和RT1PSHOLD位可以忽略。

RTCMODE Bit 5 实时时钟模式
0 32位计数器模式
1 日历模式。在日历模式额计数器模式之间的切换会重置实时时钟/计数器寄存器切换到日历模式会将秒、分、小时、星期和年清零，将日期和月份置1。实时时钟寄存器需要后来被软件设置。基本定时器计数器BTOCNT和

BT1CNT 也会被清除。

RTCRDY Bit 4 实时时钟准备位
 0 实时时钟值在转换过渡（日历模式）
 1 实时时钟值可安全读取（日历模式）。这个位指示实时时钟里的事件可被安全读取（日历模式）。在计数器模式，RTCRDY保持清除。

RTCSSEL Bit 3-2 实时时钟源选择位。
 选择时钟源输入到 RTC/32 计数器。在 RTC 日历模式这两位是不考虑的, 时钟输入是设置到 RT1PS 输出。
 00 ACLK
 01 SMLK
 10 从 PT1PS 输出
 11 从 PT1PS 输出

RTCDEV Bit 1-0 RTC 定时事件

RTC Mode	RTCDEVx	中断间隔时间
计数顺模式 (RTCMODE=0)	00	8 位溢出
	01	16 位溢出
	10	24 位溢出
	11	32 位溢出
日历表模式 (RTCMODE=1)	00	分钟变换
	01	时钟变换
	10	每天在午夜(00:00)
	11	每天在中午(12:00)

RTCCTL2 RTC 控制寄存器 2

7	6	5	4	3	2	1	0
RTCCALS	Reserved	RTCCAL					
rw-(0)	rw-(1)	rw-(0)	r-(0)	rw-0	rw-0	rw-(0)	rw-(0)

RTCCALS Bit7 RTC 校准标志
 0 频率调整下降
 1 频率调整上升

Reserved Bit6 保留位, 读为 0

RTCCAL Bit5-0 RTC 校准位
 每个 LSB 位代表大约+4PPM(RTCCALS=1)或 a-2PPM(RTCCALS=0)调整频率。

RTCCTL3 RTC 控制寄存器 3

7	6	5	4	3	2	1	0
Reserved						RTCCALF	
rw-(0)	rw-(1)	rw-(0)	r-(0)	rw-0	rw-0	rw-(0)	rw-(0)

Reserved	Bit7-2	保留 读到的总是 0
RTCCALF	Bit1-0	RTC 校准频率 校准测量时选择频率输出到 RTCCLK 引脚上。相对应的端口必须配置为外围模块功能。TRCCLK 在计数模式不可用，此时保持为低且 RTCCALF 位的值不确定。 00 没有频率输出到 RTCCLK 引脚 01 512HZ 10 256HZ 11 1HZ
RTCNT1		RTC 控制寄存器 1，计时器模式
RTCNT2		RTC 控制寄存器 2，计数器模式
RTCNT3		RTC 控制寄存器 3，计数器模式
RTCNT4		RTC 控制寄存器 4，计数器模式
RTCSEC		秒寄存器，日历模式对于十六进制格式（低 5-0 位）
RTCSEC		秒寄存器，日历模式对于 BCD 格式（5-4 位为秒十位[0-5]，3-0 位为秒个位[0-9]）
RTCMIN		分寄存器，日历模式对于十六进制格式（低 5-0 位）
RTCMIN		分寄存器，日历模式对于 BCD 格式（5-4 位为秒十位[0-5]，3-0 位为秒个位[0-9]）
RTCHOUR		时寄存器，日历模式对于十六进制格式（低 3-0 位[0-24]）
RTCHOUR		时寄存器，日历模式对于 BCD 格式（5-4 位为秒十位[0-2]，3-0 位为秒个位[0-9]）
RTCDOW		星期日数寄存器，日历模式（低 2-0 位[0-6]）
RTCDAY		日(月)寄存器，日历模式对于十六进制格式（低 3-0 位[0-28,29,30,21]）
RTCDAY		日(月)寄存器，日历模式对于 BCD 格式（5-4 位为秒十位[0-3]，3-0 位为秒个位[0-9]）
RTCMON		月寄存器，日历模式对于十六进制格式（低 3-0 位[0-12]）
RTCMON		月寄存器，日历模式对于 BCD 格式（位 4 为秒十位[0-3]，3-0 位为秒个位[0-9]）
RTCYEARL		年低字节寄存器，日历模式对于十六进制格式（低 7-0 位[0-4095 的低字节]）
RTCYEARL		年低字节寄存器，日历模式对于 BCD 格式（7-4 位为十年位[0-9]，3-0 位为最低数字[0-9]）
RTCYEARH		年高字节寄存器，日历模式对于十六进制格式（低 7-0 位[0-4095 的高字节]）
RTCYEARH		年高字节寄存器，日历模式对于 BCD 格式（6-4 位为 100 年位[0-9]，3-0 位为最低数字[0-9]）

RTCAMIN 分闹铃寄存器，日历模式对于十六进制格式 (5-0 位[0-59])

RTCAMIN 分闹铃寄存器，日历模式对于 BCD 格式 (6-4 位分高数位[0-59]，3-0 位分低数位[0-9])

RTCAHOUR 时闹铃寄存器，日历模式对于十六进制格式 (4-0 位[0-24])

RTCAHOUR 时闹铃寄存器，日历模式对于 BCD 格式 (4-5 位分高数位[0-2]，3-0 位分低数位[0-9])

RTCADOW 星期闹铃寄存器，日历模式(2-0 位 [0-6])

RTCADAY 日闹铃寄存器，日历模式对于十六进制格式。(2-0 位[0-28, 29, 30, 31])

RTCADAY 日闹铃寄存器，日历模式对于 BCD 格式。(5-4 位高数字[0-3]，3-0 位低数位[0-9])

RTCPS0CTL 预分频定时器 0 控制寄存器

15	14	13	12	11	10	9	8
Reserved	RT0SSEL	RT0PSDIV			Reserved	Reserved	RT0PSHOLD
r-(0)	rw-(0)	rw-(0)	r-(0)	rw-0	r-0	r-(0)	rw-(1)
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	RT0IP		RT0PSIE	RT0PSIFG	
r-(0)	r-(1)	r-(0)	rw-(0)	rw-0	rw-0	rw-(0)	rw-(0)

Reserved Bit15 读总是为 0

RT0SSEL Bib14 预分频定时器 0 时钟源选择位。选择时钟源输入到 PT0PS 计数器。在 RTC 日历模式这些位是不确定的。RT0PS 时钟输入自动设置到 ACLK。RT1PS 时钟输入自动设置到 RT0PS。
 0 ACLK
 1 SMCLK

RT0PSDIV Bit13-11 预分频定时器 0 分频，这些位控制 RT0PS 计数器的分频比率。在 RTC 日历模式对于 RT0PS 和 RT1PS 这些位是不确认的。PT0PS 时钟输出自动设置到/256。RT1PS 时钟输出自动设置到/128

000	/2
001	/4
010	/8
011	/16
100	/32
101	/64
110	/128
111	/256

Reserved Bit10-9 读总是为 0

RT0PSHOLD Bit8 预分频定时器 0 保持。在 RTC 时钟日历模式这些位是不确定的。RT0PS 停止是由 RTCHOLD 位控制。

- Reserved** Bit7-5 读总是为 0

- RT0IP** Bit4-2 预分频定时器 0 中断间隔
 - 000 /2
 - 001 /4
 - 010 /8
 - 011 /16
 - 100 /32
 - 101 /64
 - 110 /128
 - 111 /256

- RT0IE** Bit1 预分频定时器 0 中断允许
 - 0 中断不允许
 - 1 中断允许

- RT0IFG** Bit0 预分频定时器 0 中断标志
 - 0 没有定时事件发生
 - 1 有定时事件发生

RTCPS1CTL 预分频定时器 1 控制寄存器

	15	14	13	12	11	10	9	8
	RT1SSEL		RT1PSDIV			Reserved	Reserved	RT1PSHOLD
	r-(0)	rw-(0)	rw-(0)	r-(0)	rw-0	r-0	r-(0)	rw-(1)
	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	RT1IP		RT1PSIE	RT1PSIFG	
	r-(0)	r-(1)	r-(0)	rw-(0)	rw-0	rw-0	rw-(0)	rw-(0)

RT1SSEL Bit15-14 预分频定时器 1 时钟源选择。选择时钟源到 RT1PS 计数器。在 RTC 日历模式这些是不确定的。RT1PS 时钟输入自动设置到 RT0PS 上的输出。

- 00 ACLK
- 01 SMCLK
- 10 从 RT0PS 输出
- 11 从 RT0PS 输出

RT1PSDIV Bit13-11 预分频定时器 1 时钟分频器，这位控制 PT0PS 计数器的分频比率。在 RTC 日历模式对于 RT0PS 和 PT1PS 这些位是不确认的。RT0PS 时钟输出自动设置到/256。RT1PS 时钟输出是自动设置到/128。

- 000 /2
- 001 /4
- 010 /8
- 011 /16
- 100 /32
- 101 /64
- 110 /128
- 111 /256

Reserved Bit10-9 读总是为 0

RT1PSHOLD Bit8 预分频定时器 1 保持。在 RTC 时钟日历模式这些位是不确定的。RT1PS 停止是由 RTCHOLD 位控制。

Reserved Bit7-5
RT1IP Bit4-2 预分频定时器 1 中断间隔

000	/2
001	/4
010	/8
011	/16
100	/32
101	/64
110	/128
111	/256

RT1IE Bit1 预分频定时器 1 中断允许
 0 中断不允许
 1 中断允许

RT1IFG Bit0 预分频定时器 1 中断标志
 0 没有定时事件发生
 1 有定时事件发生

RTCPS0 预分频定时器 0 7-0 位预分频定时器 0 计数值

RTCPS1 预分频定时器 1 7-0 位预分频定时器 1 计数值

RTCIV Bits15-0 RTC 中断向量值寄存器

RTCIV 内容	中断源	中断标志	中断优先级
00H	没中断发生		
02H	RTC 读	RTCRDYIFG	高
04H	RTC 间隔定时器	RTCTEVIFG	
06H	RTC 用户闹钟	RTCAIFG	
08H	RTC 预分频 0	RT0PSIFG	
0AH	RTC 预分频 1	RT1PSIFG	
0CH	保留		
0EH	保留		
10H	保留		低