SIOC嵌入式軟體實驗
實驗二：Real-Time Clock(RTC)

WU-YANG

Technology Co., Ltd.

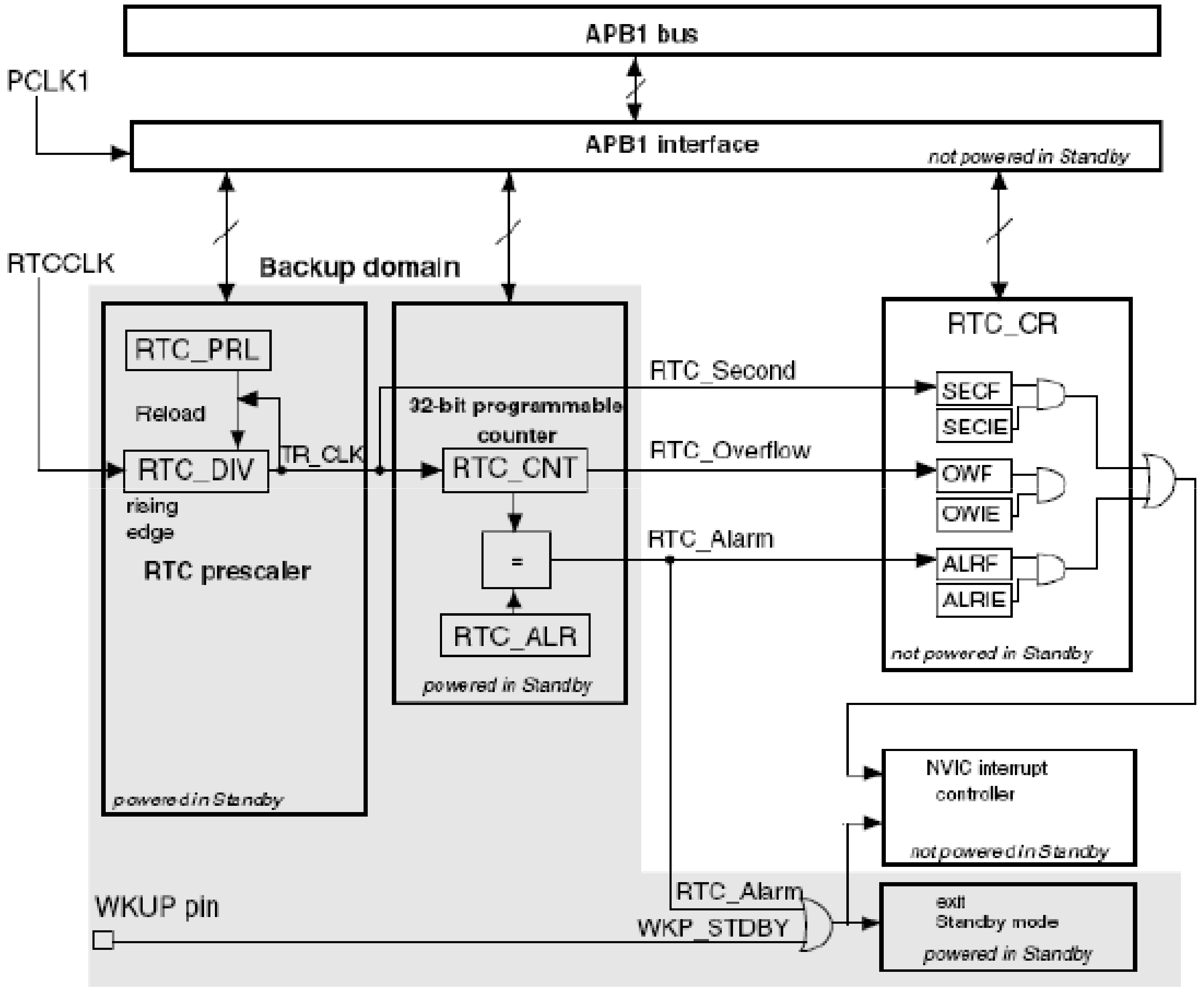
support.wuyang@gmail.com

實驗目的

- 了解Real Time Clock(RTC)硬體控制原理
- 了解STM32的RTC控制器driver設計及應用

RTC原理

- RTC是一種提供日期/時間的週邊硬體元件；
- 常用於嵌入式系統中顯示時間資訊和紀錄事件發生的時間；
- STM32的RTC硬體位於backup區，一旦系統主電源斷電，RTC仍可藉由輔助電力(例如鈕扣電池)維持運作，因而可實現不中斷的即時時鐘功能。


APB1 bus
PCLK1
APB1 interface
not powered in Standby
RTCCLK
Backup domain
RTC_PRL
Reload
RTC_DIV
TR_CLK
rising edge
RTC prescaler
powered in Standby
32-bit programmable counter
RTC_CNT
=
RTC_ALR
powered in Standby
RTC_Second
RTC_Overflow
RTC_Alarm
RTC_CR
SECF
SECIE
OWF
OWIE
ALRF
ALRIE
not powered in Standby
NVIC interrupt controller
not powered in Standby
RTC_Alarm
WKP_STDBY
WKUP pin
exit Standby mode
powered in Standby

RTC 核心

- 兩個模組：programmable prescaler + programmable counter
- prescaler division factor up to 2^{20}
- 32-bit programmable counter : 如果TR_CLK周期為1秒， 32-bit counter最多可計時至136年

```
/* Set RTC prescaler: set RTC period to 1sec */
  RTC_SetPrescaler(32767);
/* RTC period = RTCCLK/RTC_PR = (32.768 KHz)/(32767+1) */
```

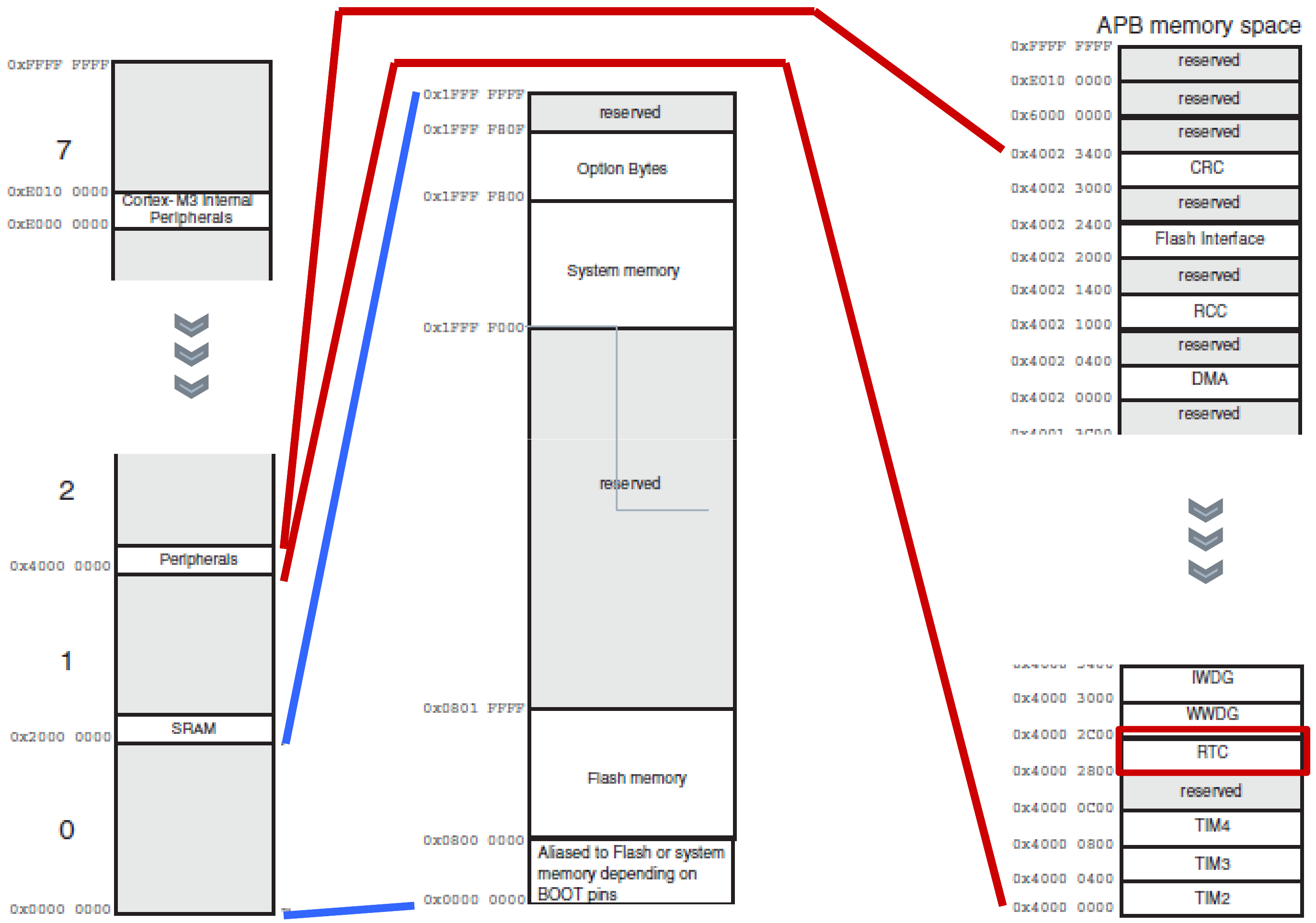
- 兩個獨立輸入的clocks: PCLK1 和 RTC

- 三個maskable interrupt:

– Alarm interrupt,產生alarm中斷
– Seconds interrupt, 產生最常1秒的週期中斷訊號
– Overflow interrupt, 偵測counter 是否overflow.

RTC Registers

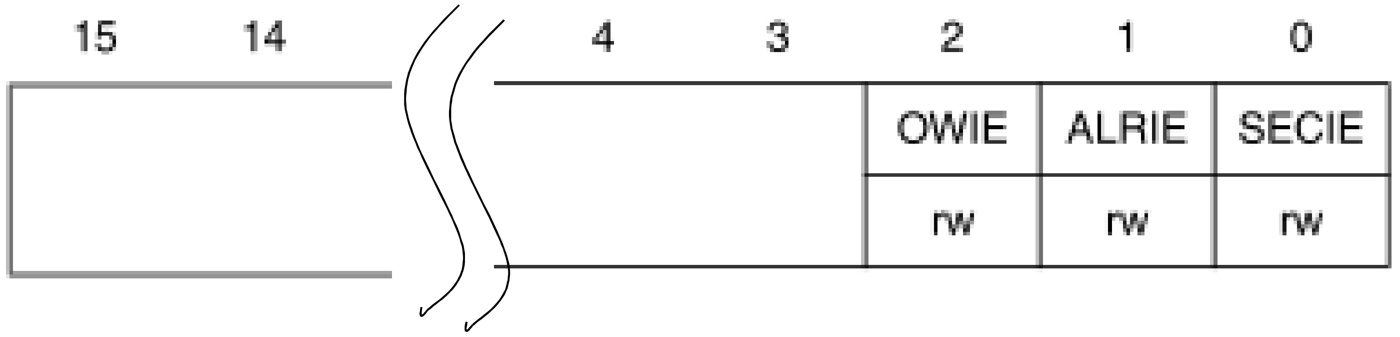
Memory Mapping Table



RTC Control Register *high*

RTC control register high (RTC_CRH)

Address offset: 0x00
Reset value: 0x0000



Bit 2 **OWIE:** Overflow interrupt enable

0: Overflow interrupt is masked.
1: Overflow interrupt is enabled.

Bit 1 **ALRIE**: Alarm interrupt enable

0: Alarm interrupt is masked.
1: Alarm interrupt is enabled.

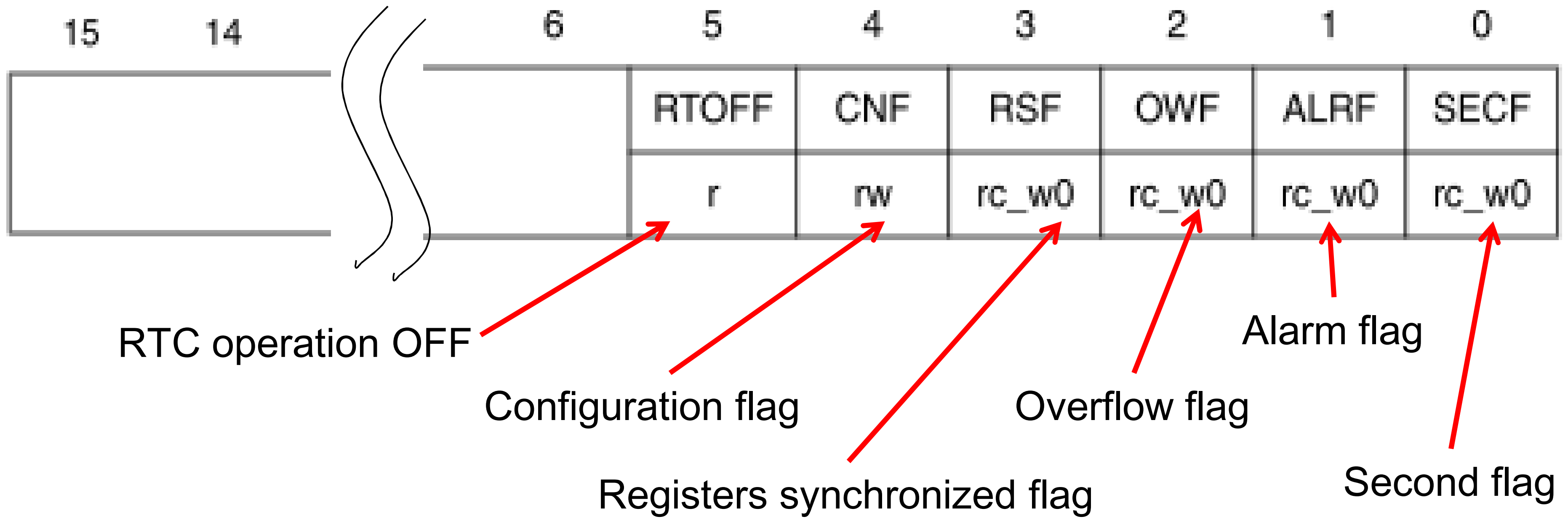
Bit 0 **SECIE:** Second interrupt enable

0: Second interrupt is masked.
1: Second interrupt is enabled.

RTC Control Register low

RTC control register low (RTC_CRL)

Address offset: 0x04
Reset value: 0x0020



RTC Counter Register

紀錄計數的秒值

RTC counter register high (RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RTC counter register low (RTC_CNTL)

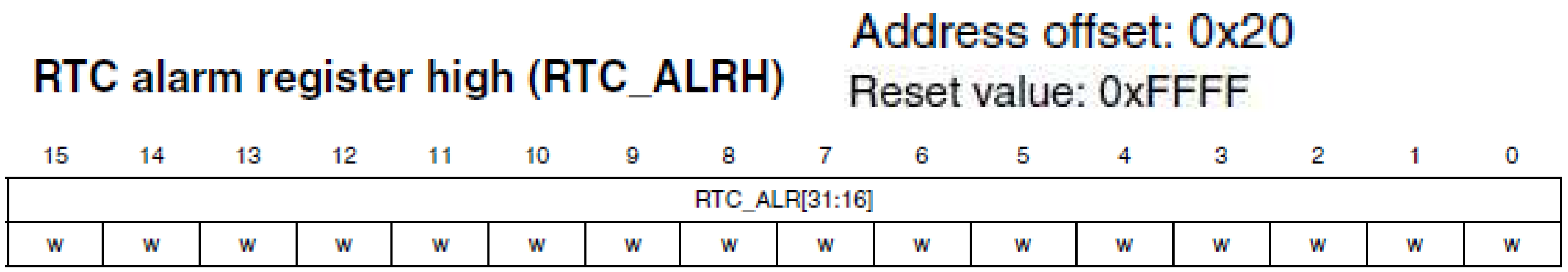
Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RTC Alarm Register

RTC_ALR暫存器存放**秒數值**，如果**RTC_CNT**的值與**RTC_ALR**的值相等，**鬧鐘發出中斷訊號**(RTC_alarmIT)

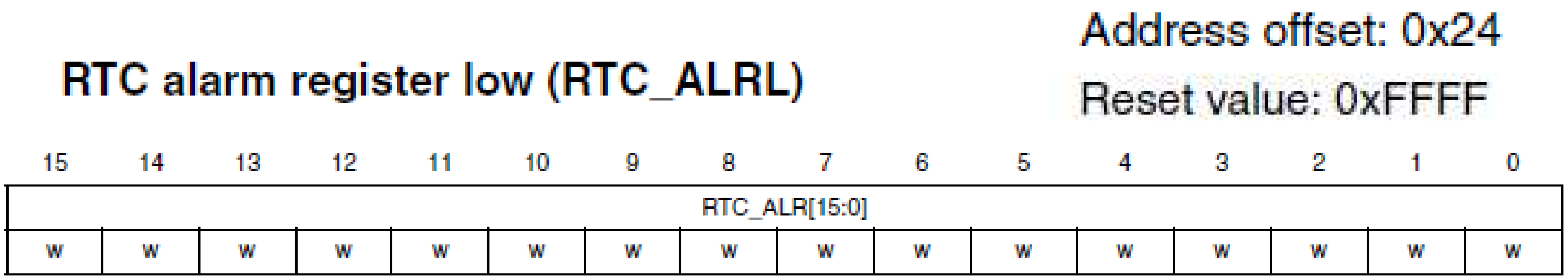


RTC alarm register high (RTC_ALRH)

Address offset: 0x20

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w



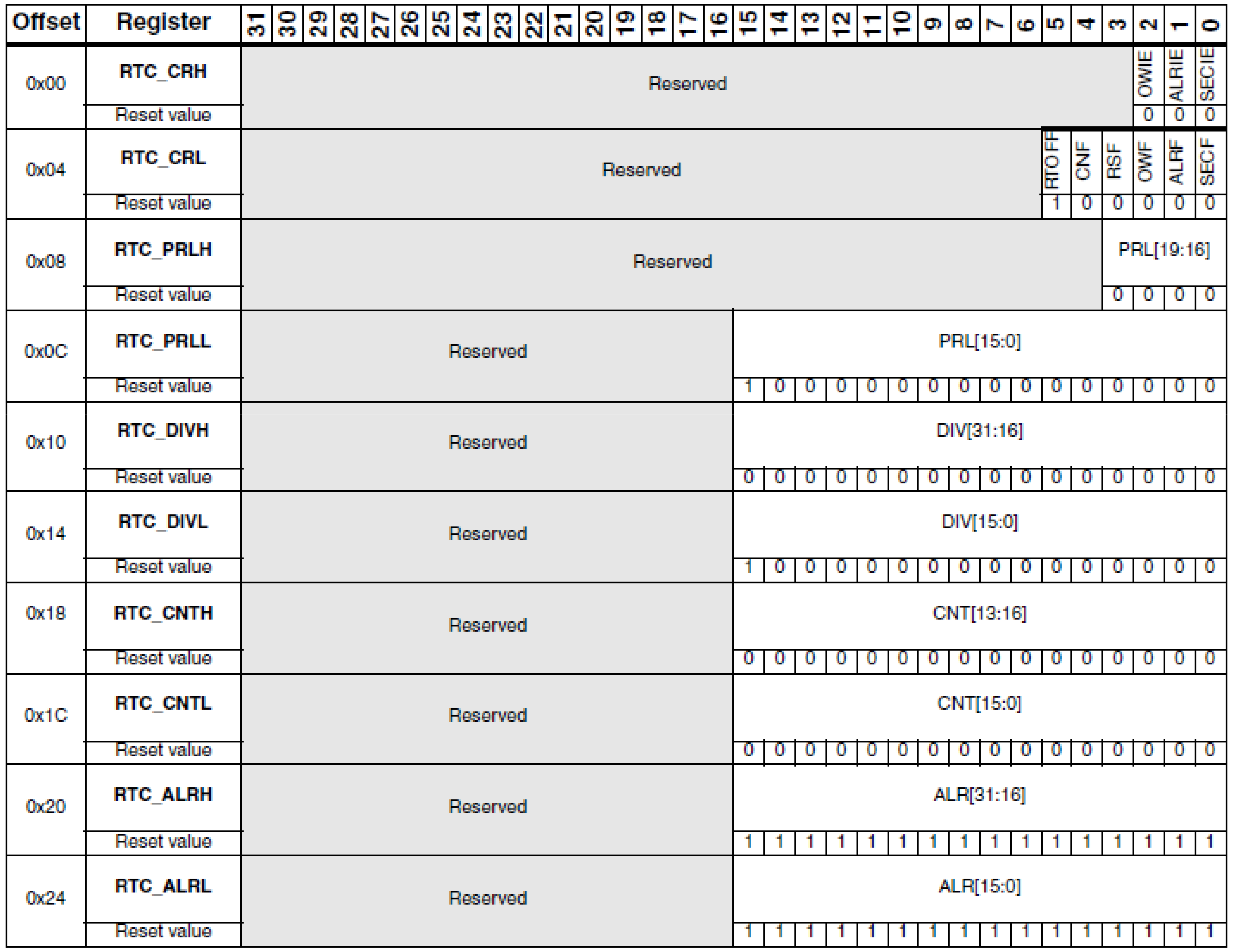
RTC alarm register low (RTC_ALRL)

Address offset: 0x24

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

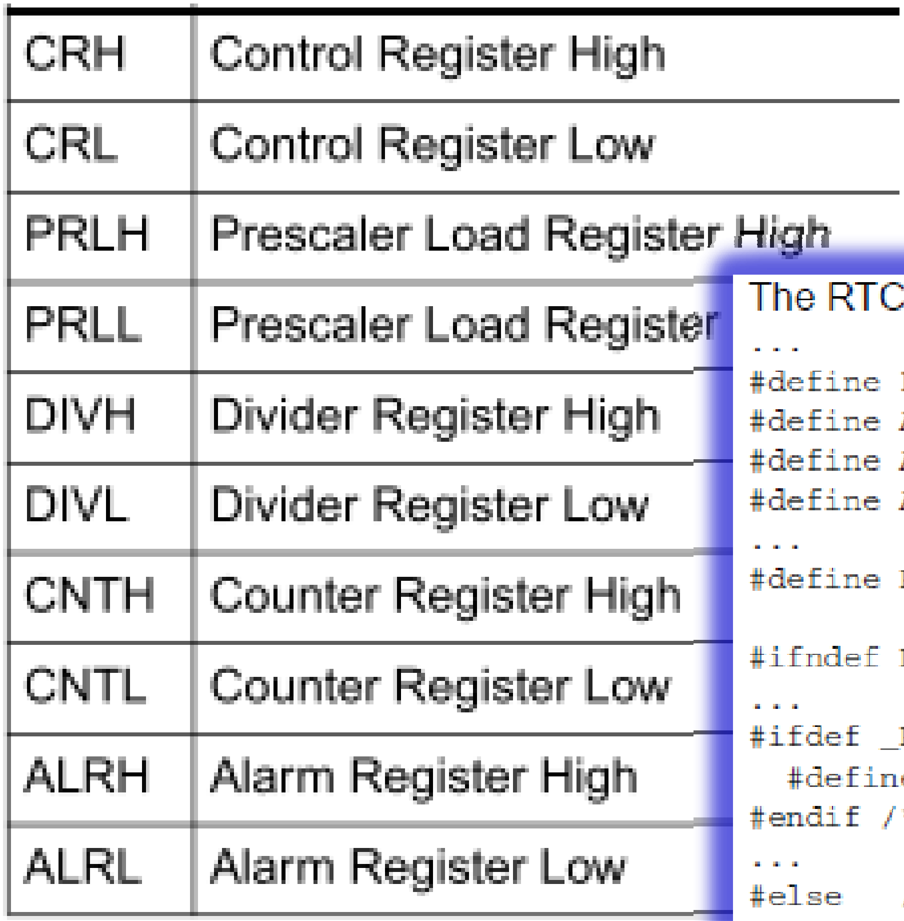
RTC Register Mapping



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_CRH	Reserved																													OWIE	ALRIE	SECIE
	Reset value																														0	0	0
0x04	RTC_CRL	Reserved																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	Reset value																											1	0	0	0	0	0
0x08	RTC_PRLH	Reserved																												PRL[19:16]			
	Reset value																													0	0	0	0
0x0C	RTC_PRLL	Reserved																PRL[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	RTC_DIVH	Reserved																DIV[31:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	RTC_DIVL	Reserved																DIV[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	RTC_CNTH	Reserved																CNT[13:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_CNTL	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RTC_ALRH	Reserved																ALR[31:16]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x24	RTC_ALRL	Reserved																ALR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

RTC Standard Driver Library

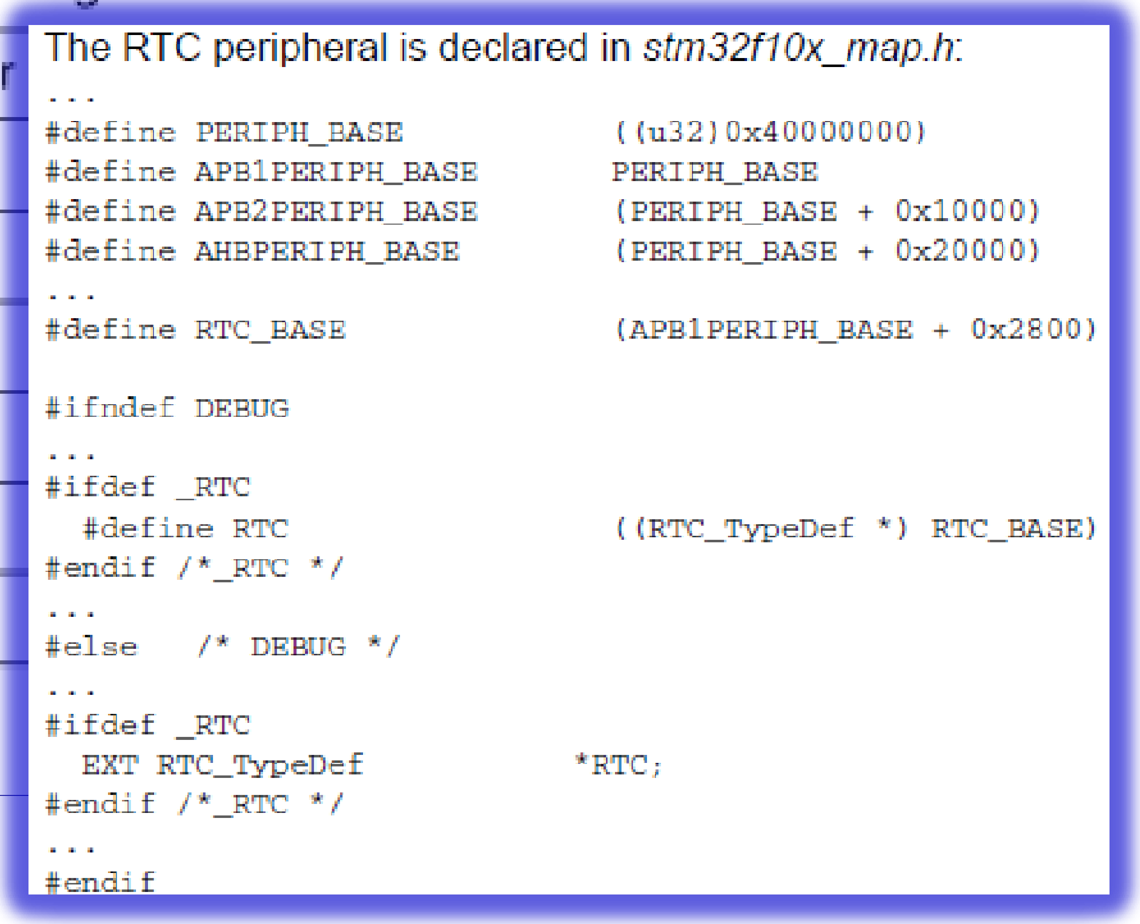
Declaration of RTC peripheral



CRH	Control Register High
CRL	Control Register Low
PRLH	Prescaler Load Register High
PRLL	Prescaler Load Register
DIVH	Divider Register High
DIVL	Divider Register Low
CNTH	Counter Register High
CNTL	Counter Register Low
ALRH	Alarm Register High
ALRL	Alarm Register Low

RTC registers

RTC declaration



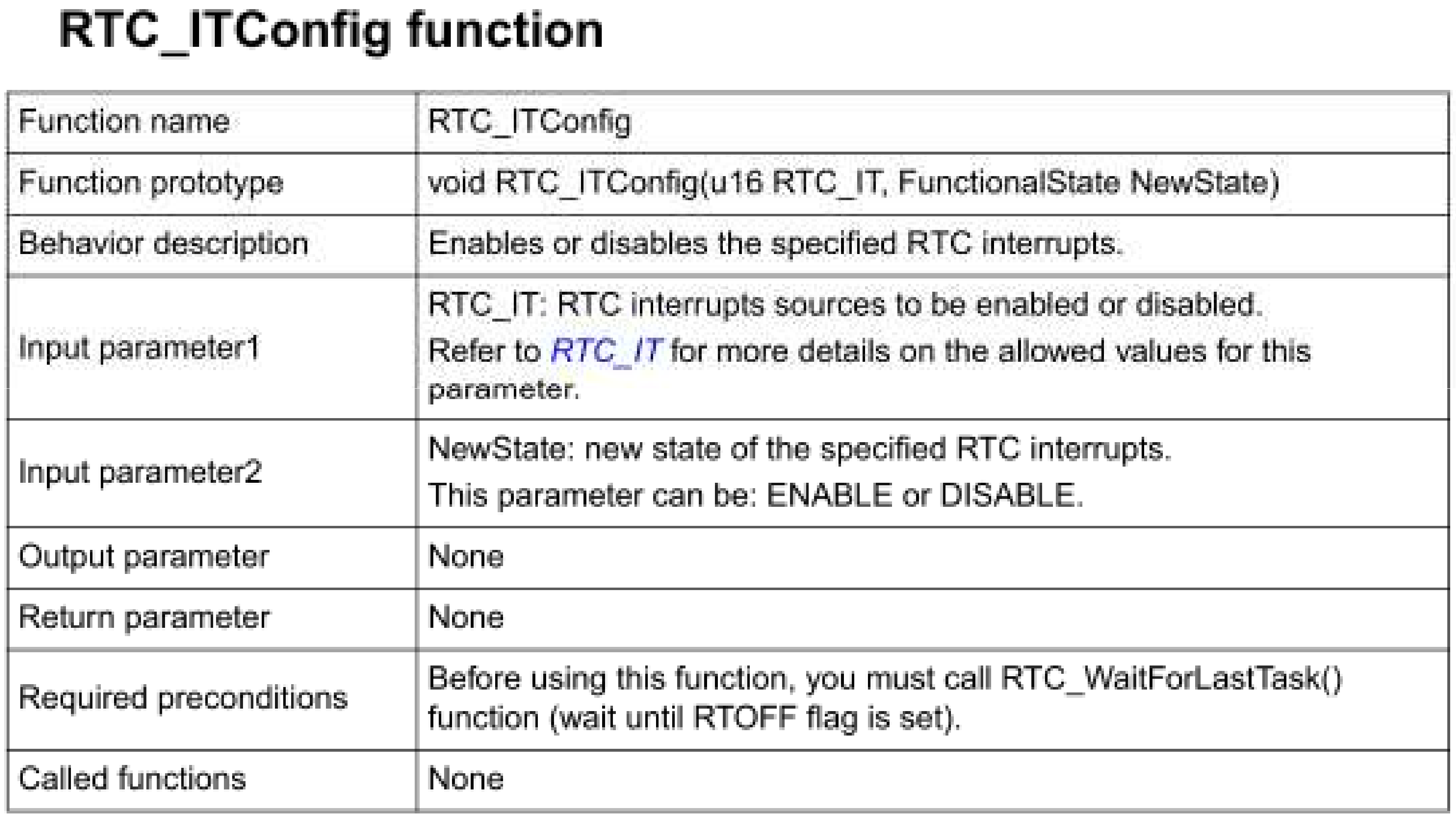
The RTC peripheral is declared in *stm32f10x_map.h*:

```
...
#define PERIPH_BASE              ((u32)0x40000000)
#define APB1PERIPH_BASE          PERIPH_BASE
#define APB2PERIPH_BASE          (PERIPH_BASE + 0x10000)
#define AHBPERIPH_BASE           (PERIPH_BASE + 0x20000)
...
#define RTC_BASE                 (APB1PERIPH_BASE + 0x2800)

#ifndef DEBUG
...
#ifdef _RTC
  #define RTC                    ((RTC_TypeDef *) RTC_BASE)
#endif /*_RTC */
...
#else   /* DEBUG */
...
#ifdef _RTC
  EXT RTC_TypeDef              *RTC;
#endif /*_RTC */
...
#endif
```

RTC firmware library function

RTC_ITConfig	Enables or disables the specified RTC interrupts.
RTC_EnterConfigMode	Enters the RTC configuration mode.
RTC_ExitConfigMode	Exits from the RTC configuration mode.
RTC_GetCounter	Gets the RTC counter value.
RTC_SetCounter	Sets the RTC counter value.
RTC_SetPrescaler	Sets the RTC prescaler value.
RTC_SetAlarm	Sets the RTC Alarm value.
RTC_GetDivider	Gets the RTC Divider value.
RTC_WaitForLastTask	Waits until last write operation on RTC registers is completed
RTC_WaitForSynchro	Waits until the RTC registers (RTC_CNT, RTC_ALR and RTC_PRL) are synchronized with RTC APB clock.
RTC_GetFlagStatus	Checks whether the specified RTC flag is set or not.
RTC_ClearFlag	Clears the RTC pending flags.
RTC_GetITStatus	Checks whether the specified RTC interrupt has occurred or not.
RTC_ClearITPendingBit	Clears the RTC interrupt pending bits.

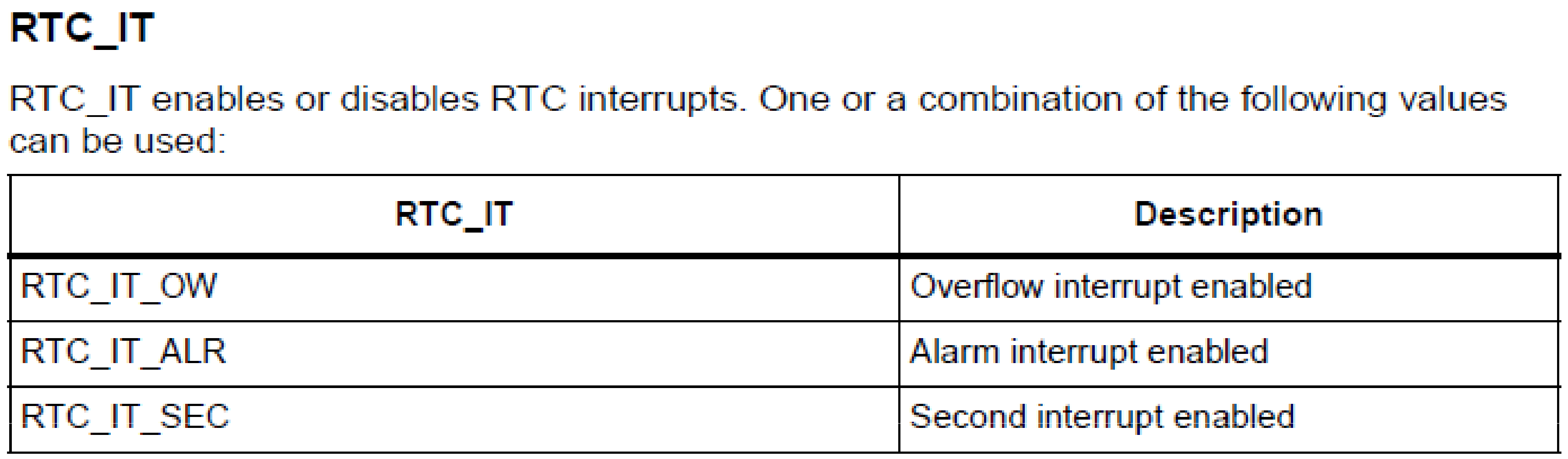
RTC Interrupt Configure Function



RTC_ITConfig function

Function name	RTC_ITConfig
Function prototype	void RTC_ITConfig(u16 RTC_IT, FunctionalState NewState)
Behavior description	Enables or disables the specified RTC interrupts.
Input parameter1	RTC_IT: RTC interrupts sources to be enabled or disabled. Refer to *RTC_IT* for more details on the allowed values for this parameter.
Input parameter2	NewState: new state of the specified RTC interrupts. This parameter can be: ENABLE or DISABLE.
Output parameter	None
Return parameter	None
Required preconditions	Before using this function, you must call RTC_WaitForLastTask() function (wait until RTOFF flag is set).
Called functions	None

RTC_IT in RTC Interrupt Configure Function



RTC_IT

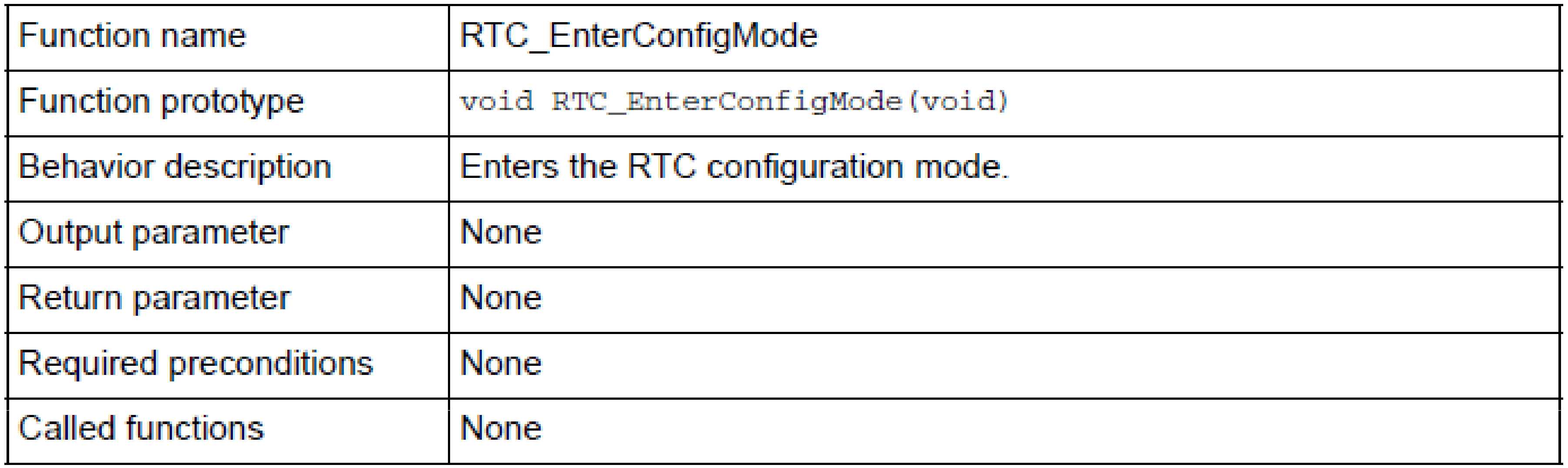
RTC_IT enables or disables RTC interrupts. One or a combination of the following values can be used:

RTC_IT	Description
RTC_IT_OW	Overflow interrupt enabled
RTC_IT_ALR	Alarm interrupt enabled
RTC_IT_SEC	Second interrupt enabled

Example:

```
/* Wait until last write operation on RTC registers is terminated */
RTC_WaitForLastTask();
/* Alarm interrupt enabled */
RTC_ITConfig(RTC_IT_ALR, ENABLE);
```

RTC EnterConfigure Mode Function



Function name	RTC_EnterConfigMode
Function prototype	`void RTC_EnterConfigMode(void)`
Behavior description	Enters the RTC configuration mode.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Enable the configuration mode */
RTC_EnterConfigMode();
```

RTC_ExitConfigureMode Function

Function name	RTC_ExitConfigMode
Function prototype	`void RTC_ExitConfigMode(void)`
Behavior description	Exits from the RTC configuration mode.
Output parameter	None
Return parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Exit the configuration mode */
RTC_ExitConfigMode();
```

RTC_GetCounter Function

Function name	RTC_GetCounter
Function prototype	`u32 RTC_GetCounter(void)`
Behavior description	Gets the RTC counter value.
Output parameter	None
Return parameter	RTC counter value
Required preconditions	None
Called functions	None

Example:

```
/* Gets the counter value */
u32 RTCCounterValue;
RTCCounterValue = RTC_GetCounter();
```

RTC_SetCounter Function

Function name	RTC_SetCounter
Function prototype	`void RTC_SetCounter(u32 CounterValue)`
Behavior description	Sets the RTC counter value.
Input parameter	CounterValue: RTC counter new value.
Output parameter	None
Return parameter	None
Required preconditions	Before issuing this function, call RTC_WaitForLastTask() function (wait until RTOFF flag is set)

Called functions	RTC_EnterConfigMode()
	RTC_ExitConfigMode()

Example:

```
/* Wait until last write operation on RTC registers is terminated */
RTC_WaitForLastTask();
RTC_SetCounter(0xFFFF5555);
```

RTC_SetPrescaler function

Function name	RTC_SetPrescaler
Function prototype	`void RTC_SetPrescaler(u32 PrescalerValue)`
Behavior description	Sets the RTC prescaler value.
Input parameter	PrescalerValue: RTC prescaler new value.
Output parameter	None
Return parameter	None
Required preconditions	Before using this function, call RTC_WaitForLastTask() function (wait until RTOFF flag is set).

Called functions	RTC_EnterConfigMode()
	RTC_ExitConfigMode()

Example:

```
/* Wait until last write operation on RTC registers is terminated */
RTC_WaitForLastTask();
/* Sets Prescaler value to 0x7A12 */
RTC_SetPrescaler(0x7A12);
```

RTC_SetAlarm Function

Function name	RTC_SetAlarm
Function prototype	`void RTC_SetAlarm(u32 AlarmValue)`
Behavior description	Sets the RTC alarm value.
Input parameter	AlarmValue: RTC alarm new value.
Output parameter	None
Return parameter	None
Required preconditions	Before using this function, call *RTC_WaitForLastTask()* function (wait until RTOFF flag is set).

Called functions	RTC_EnterConfigMode()
	RTC_ExitConfigMode()

Example:

```
/* Wait until last write operation on RTC registers is terminated */
RTC_WaitForLastTask();
RTC_SetAlarm(0xFFFFFFFA);
```

RTC_GetFlagStatus Function

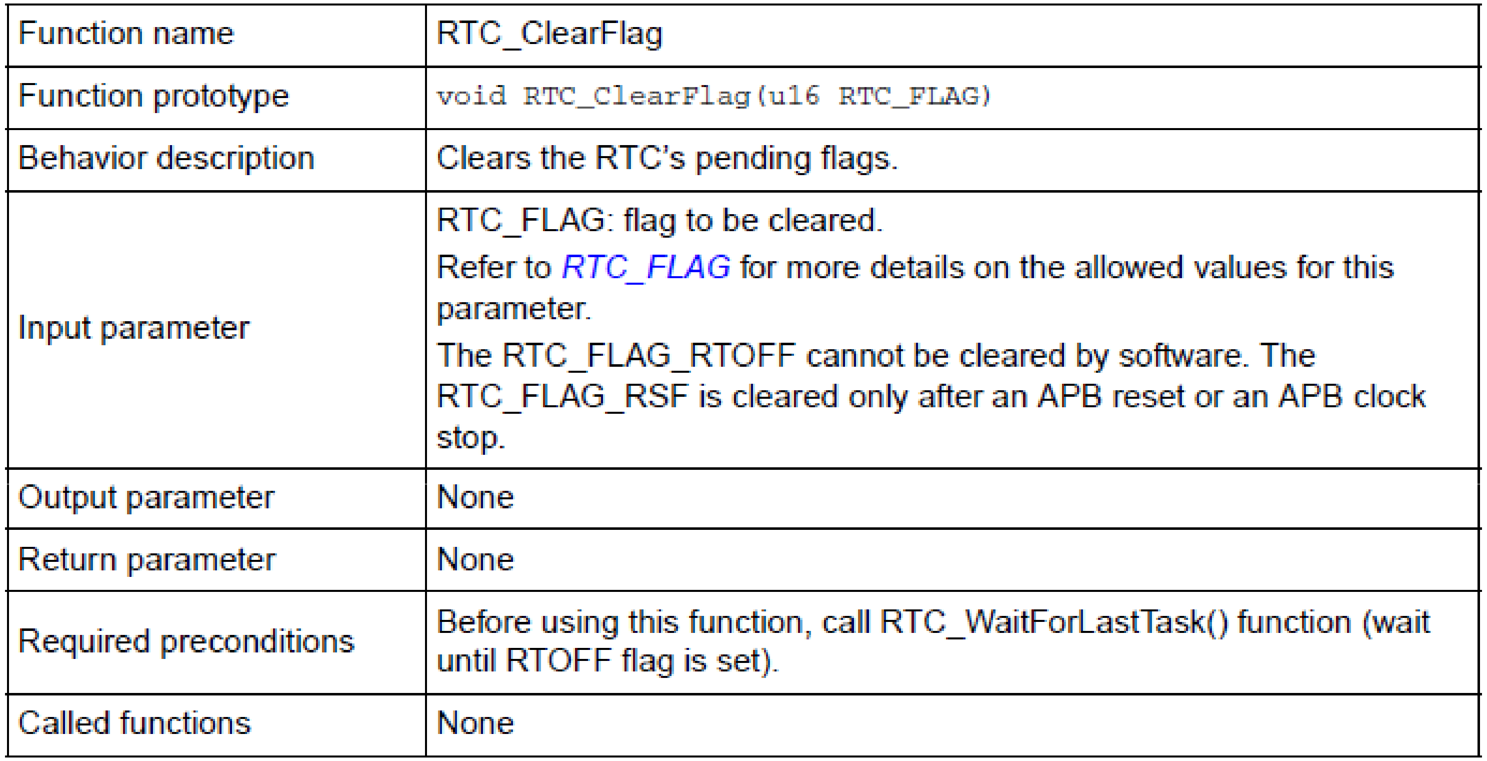
Function prototype	`FlagStatus RTC_GetFlagStatus(u16 RTC_FLAG)`
Behavior description	Checks whether the specified RTC flag is set or not.
Input parameter	RTC_FLAG: specifies the flag to check. Refer to *RTC_FLAG* for more details on the allowed values for this parameter.
Return parameter	The new state of RTC_FLAG (SET or RESET).

RTC_FLAG	Description
RTC_FLAG_RTOFF	RTC operation OFF Flag
RTC_FLAG_RSF	Registers Synchronized Flag
RTC_FLAG_OW	Overflow interrupt Flag
RTC_FLAG_ALR	Alarm interrupt Flag
RTC_FLAG_SEC	Second interrupt Flag

Example:

```
/* Gets the RTC overflow interrupt status */
FlagStatus OverrunFlagStatus;
OverrunFlagStatus = RTC_GetFlagStatus(RTC_Flag_OW);
```

RTC Clear Flag Function



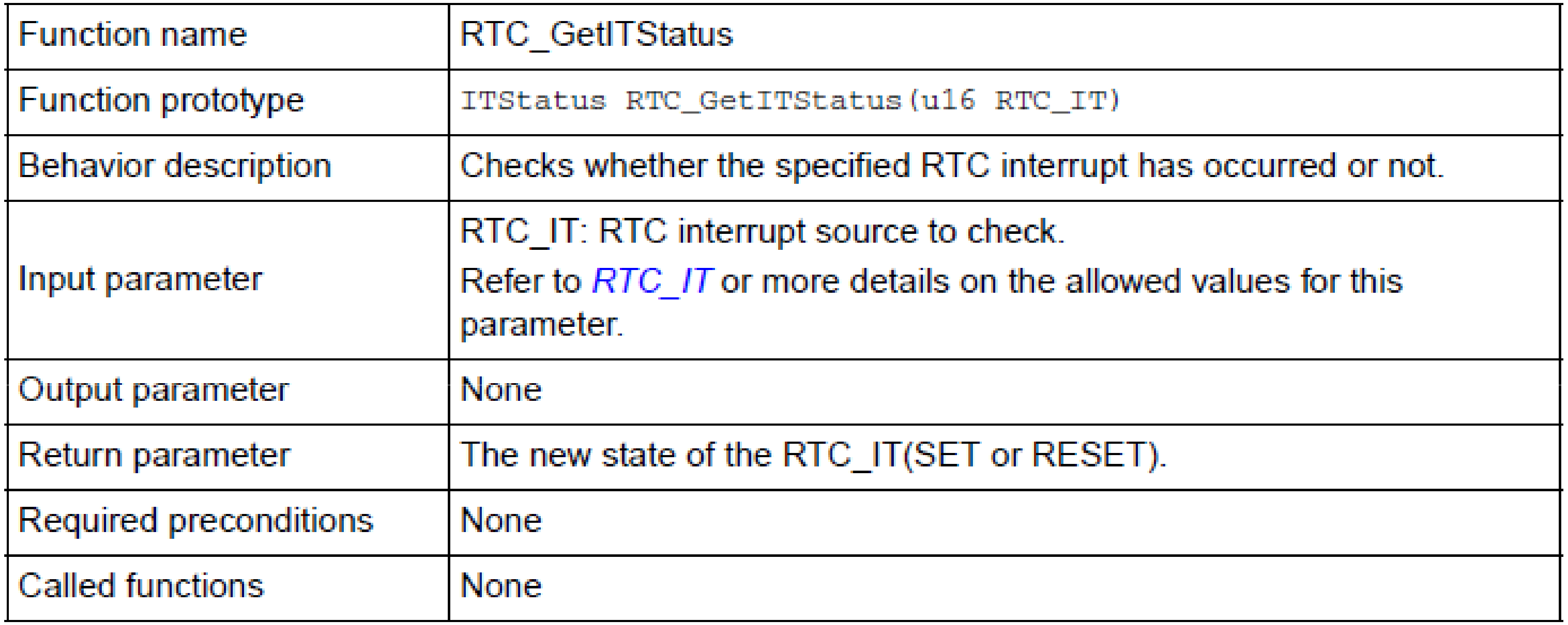
Function name	RTC_ClearFlag
Function prototype	`void RTC_ClearFlag(u16 RTC_FLAG)`
Behavior description	Clears the RTC's pending flags.
Input parameter	RTC_FLAG: flag to be cleared. Refer to *RTC_FLAG* for more details on the allowed values for this parameter. The RTC_FLAG_RTOFF cannot be cleared by software. The RTC_FLAG_RSF is cleared only after an APB reset or an APB clock stop.
Output parameter	None
Return parameter	None
Required preconditions	Before using this function, call RTC_WaitForLastTask() function (wait until RTOFF flag is set).
Called functions	None

Example:

RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers is terminated */

RTC_ClearFlag(RTC_FLAG_OW); /* Clears the RTC overflow flag */

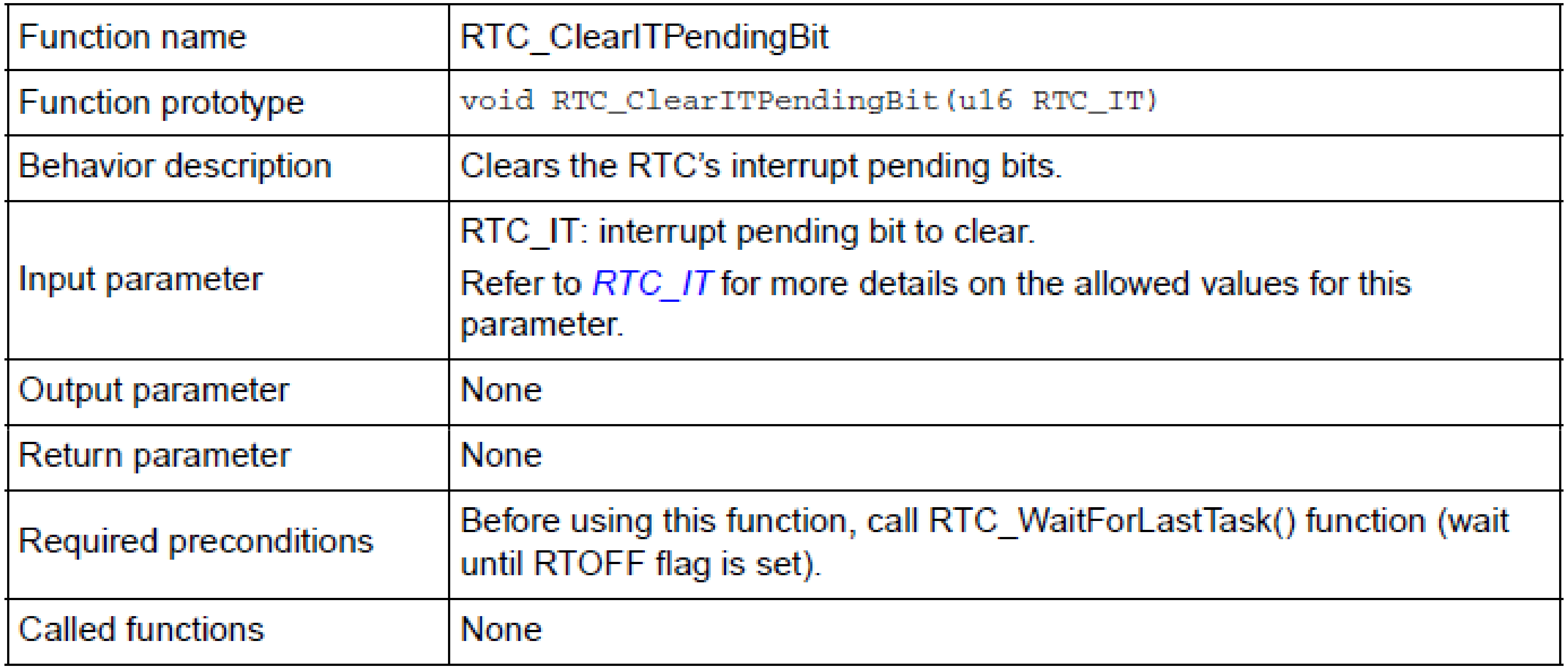
RTC Get Interrupt Status Function



Function name	RTC_GetITStatus
Function prototype	`ITStatus RTC_GetITStatus(u16 RTC_IT)`
Behavior description	Checks whether the specified RTC interrupt has occurred or not.
Input parameter	RTC_IT: RTC interrupt source to check. Refer to *RTC_IT* or more details on the allowed values for this parameter.
Output parameter	None
Return parameter	The new state of the RTC_IT(SET or RESET).
Required preconditions	None
Called functions	None

```
/* Gets the RTC Second interrupt status */
ITStatus SecondITStatus;
SecondITStatus = RTC_GetITStatus(RTC_IT_SEC);
```

RTC Clear Interrupt Pending Bit Function



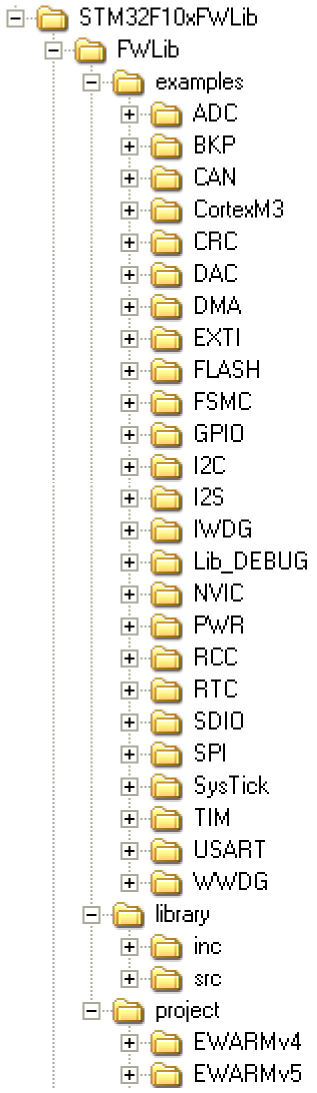
Function name	RTC_ClearITPendingBit
Function prototype	`void RTC_ClearITPendingBit(u16 RTC_IT)`
Behavior description	Clears the RTC's interrupt pending bits.
Input parameter	RTC_IT: interrupt pending bit to clear. Refer to *RTC_IT* for more details on the allowed values for this parameter.
Output parameter	None
Return parameter	None
Required preconditions	Before using this function, call RTC_WaitForLastTask() function (wait until RTOFF flag is set).
Called functions	None

Example:

```
RTC_WaitForLastTask();
/* Clears the RTC Second interrupt */
RTC_ClearITPendingBit(RTC_IT_SEC);
```

實驗

Package description



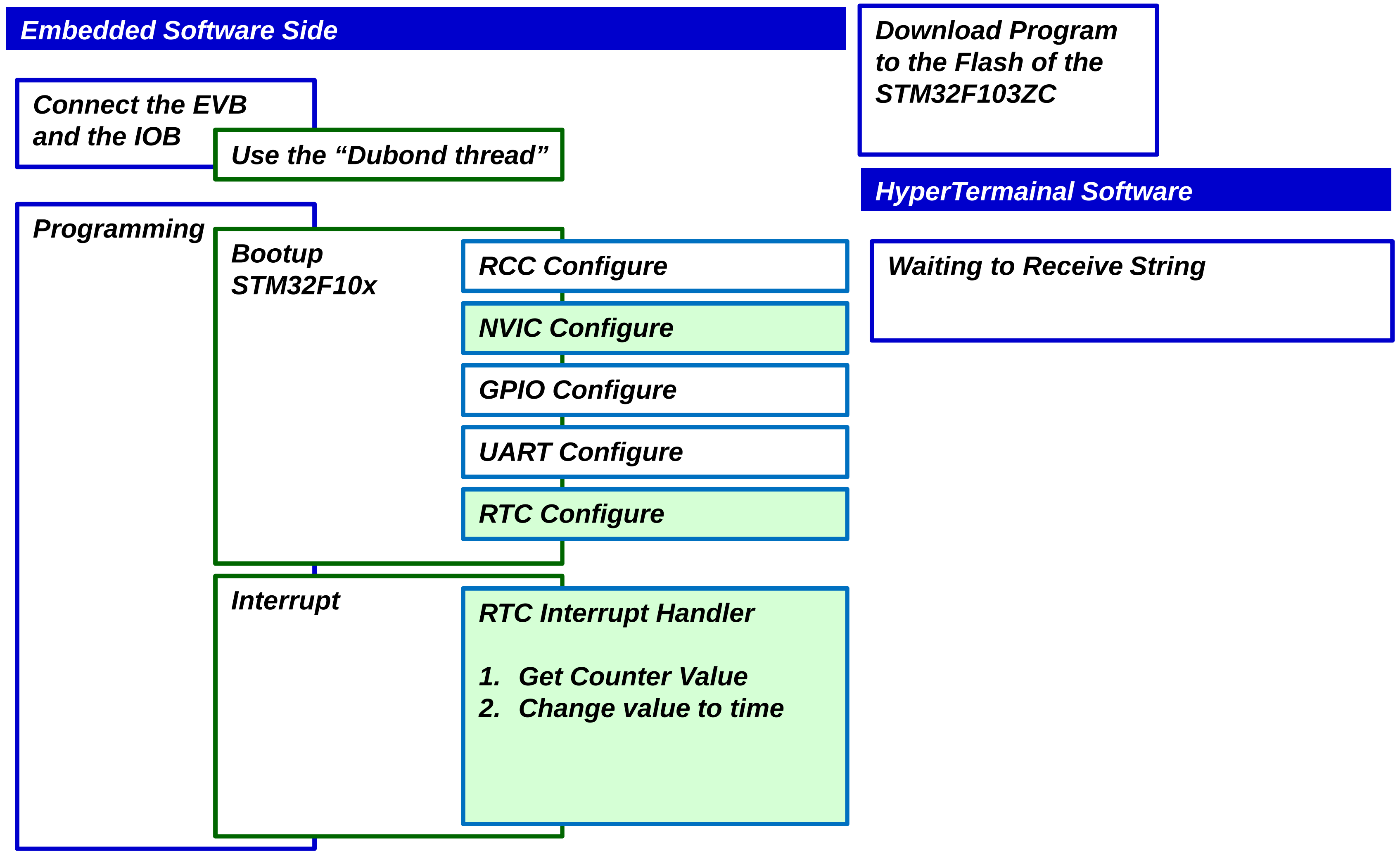
```
STM32F10xFWLib
  FWLib
    examples
      ADC
      BKP
      CAN
      CortexM3
      CRC
      DAC
      DMA
      EXTI
      FLASH
      FSMC
      GPIO
      I2C
      I2S
      IWDG
      Lib_DEBUG
      NVIC
      PWR
      RCC
      RTC
      SDIO
      SPI
      SysTick
      TIM
      USART
      WWDG
    library
      inc
      src
    project
      EWARMv4
      EWARMv5
```

路徑
C:\Keil\ARM\Examples\ST\STM32F10xFWLib

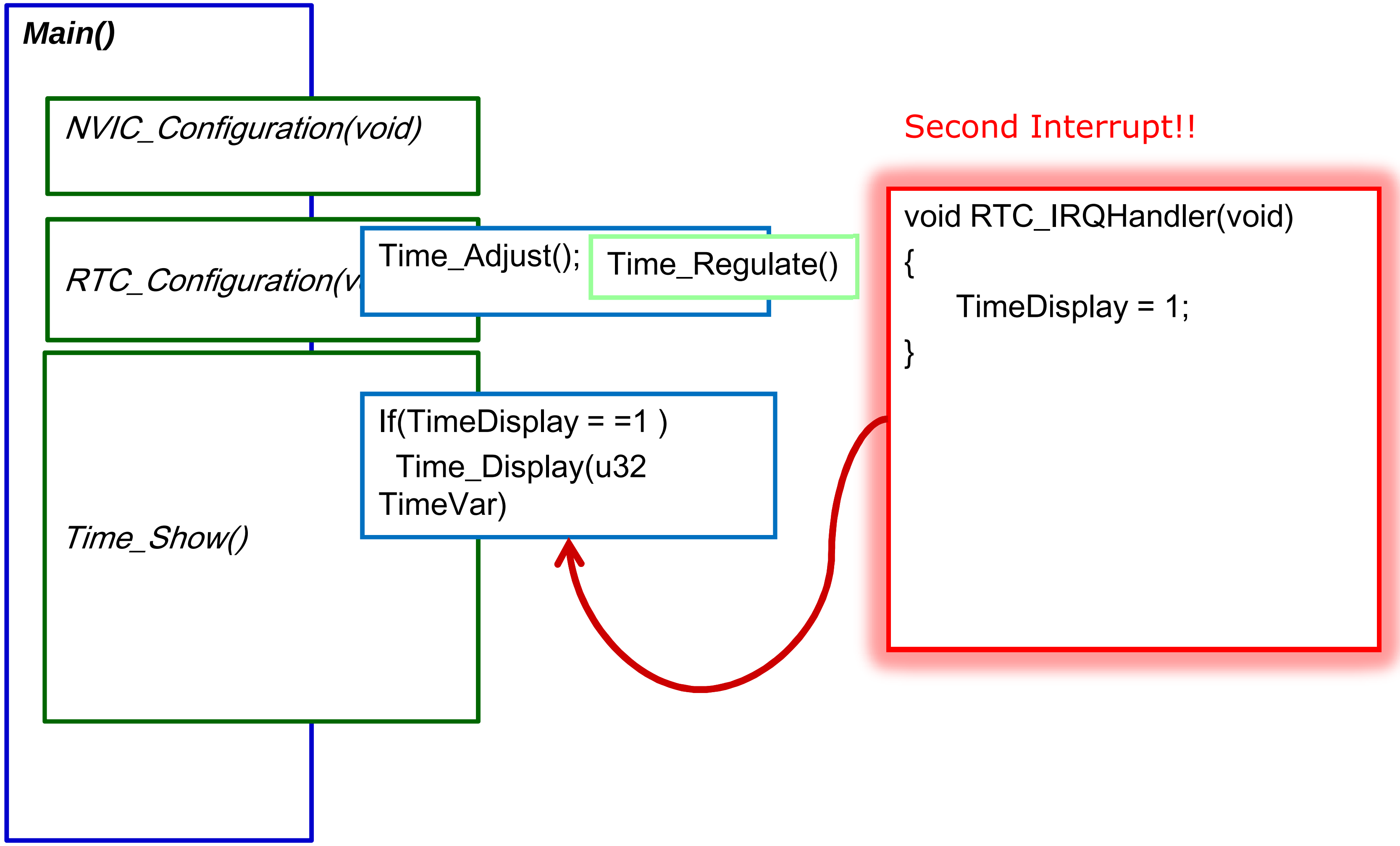
實驗1– 超級終端機顯示時間

- 使用RTC second interrupt來產生時間，並用Virtual COM Port 在超級終端機上顯示時間。

嵌入式軟體架構

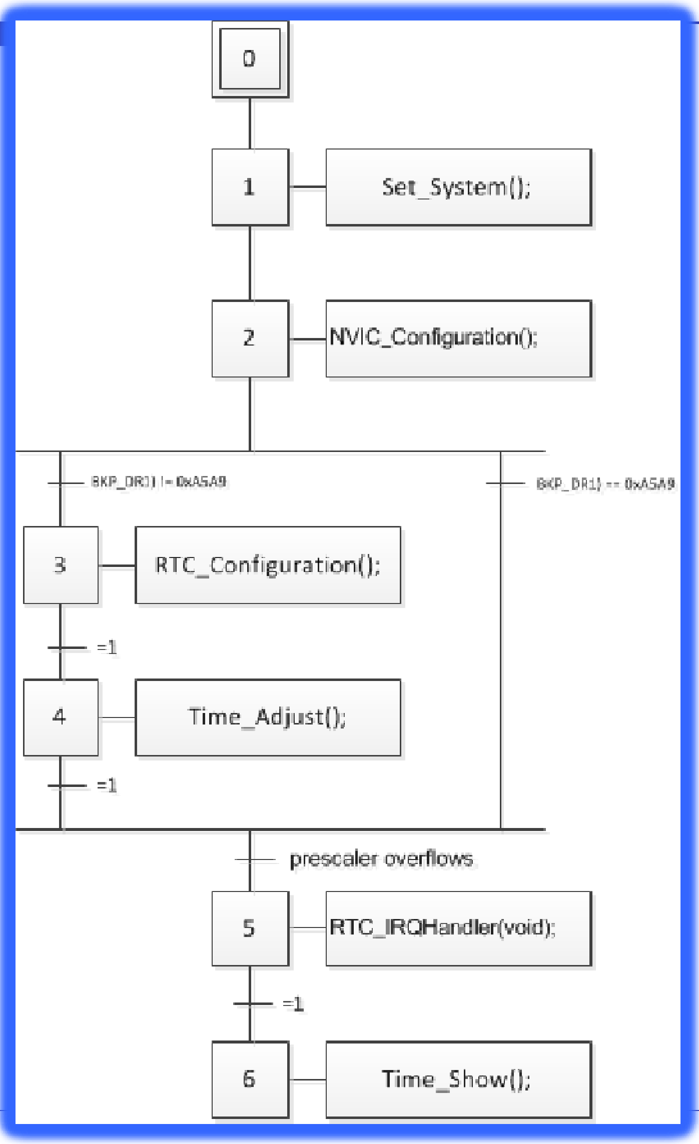

Embedded Software Side
Connect the EVB and the IOB
Use the “Dubond thread”
Programming
Bootup STM32F10x
RCC Configure
NVIC Configure
GPIO Configure
UART Configure
RTC Configure
Interrupt
RTC Interrupt Handler
1. Get Counter Value
2. Change value to time
Download Program to the Flash of the STM32F103ZC
HyperTermainal Software
Waiting to Receive String

Development Flow cont.



```
void RTC_IRQHandler(void)
{
     TimeDisplay = 1;
}
```

RTC Grafcet


0
1
Set_System();
2
NVIC_Configuration();
BKP_DR1) != 0xA5A9
BKP_DR1) == 0xA5A9
3
RTC_Configuration();
=1
4
Time_Adjust();
=1
prescaler overflows
5
RTC_IRQHandler(void);
=1
6
Time_Show();

NVIC Configure

```
/* NVIC Configure */
void NVIC_Configuration(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
#ifdef  VECT_TAB_RAM
  /* Set the Vector Table base location at 0x20000000 */
  NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
#else  /* VECT_TAB_FLASH  */
  /* Set the Vector Table base location at 0x08000000 */
  //NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
  NVIC_SetVectorTable(0x08003000, 0x0);
#endif
    /* Configure one bit for preemption priority */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
    /* Enable the RTC Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = RTC_IRQChannel;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

RTC Configure

```
/* RTC Configure */
void RTC_Configuration(void)
{
  /* Enable PWR and BKP clocks */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);
  PWR_BackupAccessCmd(ENABLE); /* Allow access to BKP Domain */
  BKP_DeInit();                /* Reset Backup Domain */
  RCC_LSEConfig(RCC_LSE_ON);   /* Enable LSE */
  while (RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET); /* Wait till LSE is ready */
  RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);       /* Select LSE as RTC Clock Source */
  RCC_RTCCLKCmd(ENABLE);                       /* Enable RTC Clock */
  RTC_WaitForLastTask();
  RTC_WaitForSynchro();   /* Wait for RTC registers synchronization */
  RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers has finished */
  RTC_ITConfig(RTC_IT_SEC, ENABLE);  /* Enable the RTC Second */
  RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers is terminated */
  /* Set RTC prescaler: set RTC period to 1sec */
  RTC_SetPrescaler(32767); /* RTC period = RTCCLK/RTC_PR = (32.768 KHz)/(32767+1) */
  RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers has finished */
}
```

Main program

```
/* 超級終端機顯示RTC時間*/
int main(void)
{
#ifdef DEBUG
  debug();
#endif
  NVIC_Configuration(); /* NVIC configuration */
  USART_Configuration(); /* Configure the USART1 */

  if (BKP_ReadBackupRegister(BKP_DR1) != 0xA5A5)
  {
    /* Backup data register value is not correct or not yet programmed (when
       the first time the program is executed) */
    printf("\r\n\n RTC not yet configured....");
    RTC_Configuration(); /* RTC Configuration */
    printf("\r\n RTC configured....");
    Time_Adjust(); /* Adjust time by values entred by the user on the hyperterminal */
    BKP_WriteBackupRegister(BKP_DR1, 0xA5A5);
  }
```

```
/* 續上頁*/
else
  {
    /* Check if the Power On Reset flag is set */
    if(RCC_GetFlagStatus(RCC_FLAG_PORRST) != RESET)
         printf("\r\n\n Power On Reset occurred....");
    else if(RCC_GetFlagStatus(RCC_FLAG_PINRST) != RESET)
          printf("\r\n\n External Reset occurred....");
     printf("\r\n No need to configure RTC....");
     RTC_WaitForSynchro(); /* Wait for RTC registers synchronization */
     RTC_ITConfig(RTC_IT_SEC, ENABLE); /* Enable the RTC Second */
    /* Wait until last write operation on RTC registers has finished */
     RTC_WaitForLastTask();
  }
  RCC_ClearFlag(); /* Clear reset flags */
  while(1)
              Time_Show();
}
```

Time_Regulate( ) & Time_Adjust( ) function

```
u32 Time_Regulate(void)
{
  u32 Tmp_HH = 0xFF, Tmp_MM = 0xFF, Tmp_SS = 0xFF;
    Tmp_HH =0;
    Tmp_MM = 0;
     Tmp_SS = 0;                    *
  /* Return the value to store in RTC counter register */
  return((Tmp_HH*3600 + Tmp_MM*60 + Tmp_SS));
}
void Time_Adjust(void)
{
   RTC_WaitForLastTask();
   RTC_SetCounter(Time_Regulate()); /* Change the current time */
   RTC_WaitForLastTask();
}
```

Time_Display( ) & Time_Show( ) function

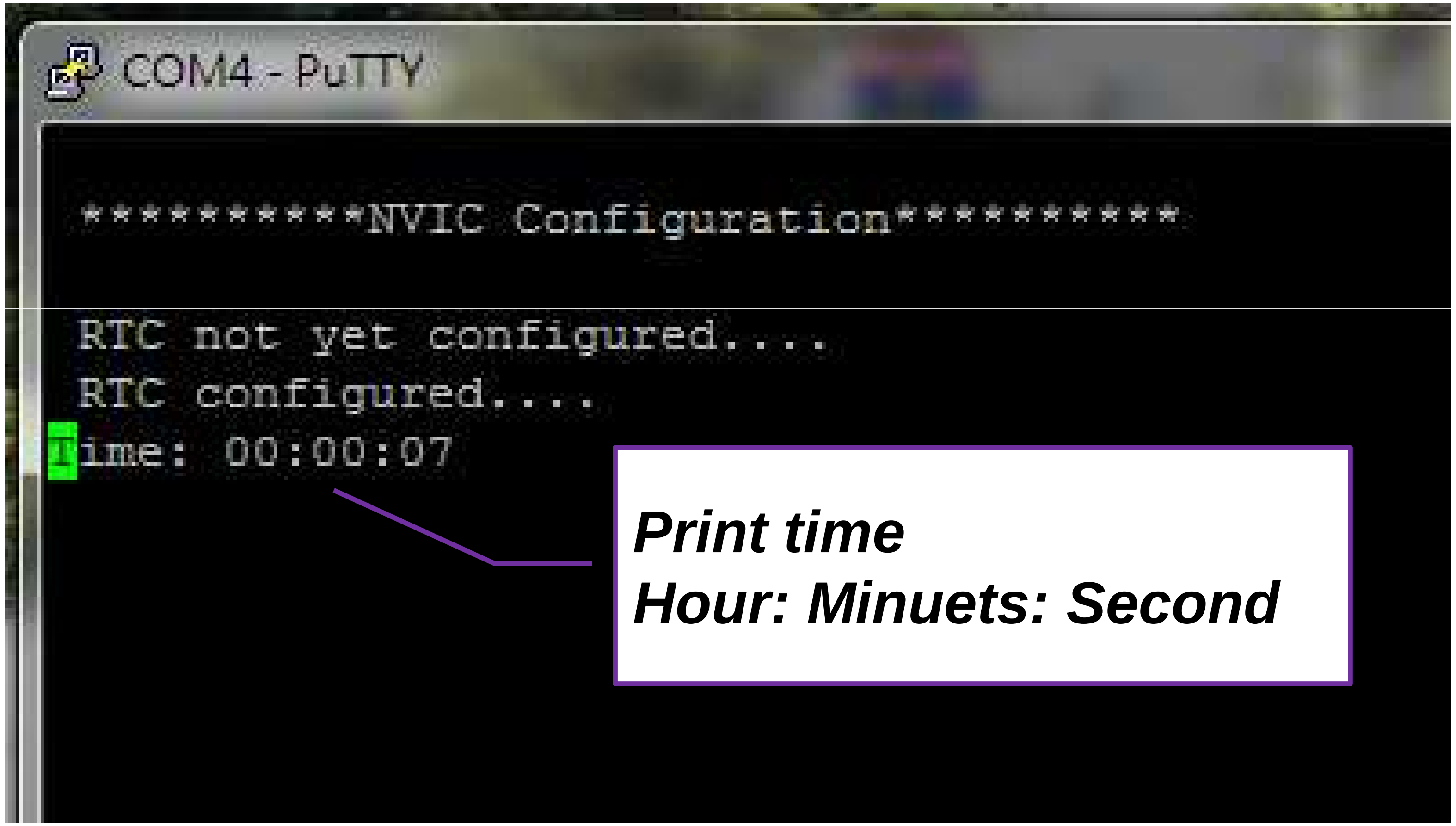
```
void Time_Display(u32 TimeVar)
{
  u32 THH = 0, TMM = 0, TSS = 0;
  THH = TimeVar / 3600; /* Compute  hours */
  TMM = (TimeVar % 3600) / 60; /* Compute minutes */
  TSS = (TimeVar % 3600) % 60; /* Compute seconds */
  printf("Time: %0.2d:%0.2d:%0.2d\r", THH, TMM, TSS);
}
void Time_Show(void)
{
  printf("\n\r");
  while (1) /* Infinite loop */ {
  if (TimeDisplay == 1) /* If 1s has paased */ {
     Time_Display(RTC_GetCounter()); /* Display current time */
     TimeDisplay = 0;
   } }
}
```

RTC Interrupt Handler

```
<stm32f10x_it.c>
/* Interrupt Handler*/
void RTC_IRQHandler(void)
{
  if (RTC_GetITStatus(RTC_IT_SEC) != RESET)
  {
   RTC_ClearITPendingBit(RTC_IT_SEC); /* Clear the RTC Second interrupt */
   TimeDisplay = 1; /* Enable time update */

    /* Wait until last write operation on RTC registers has finished */
    RTC_WaitForLastTask();
    /* Reset RTC Counter when Time is 23:59:59 */
    if (RTC_GetCounter() == 0x00015180)
    {
      RTC_SetCounter(0x0);
      /* Wait until last write operation on RTC registers has finished */
      RTC_WaitForLastTask();
    }
  }
}
```

LAB1 DEMO

HyperTermainal Output



```
**********NVIC Configuration**********

RTC not yet configured....
RTC configured....
Time: 00:00:07
```

Print time
Hour: Minuets: Second

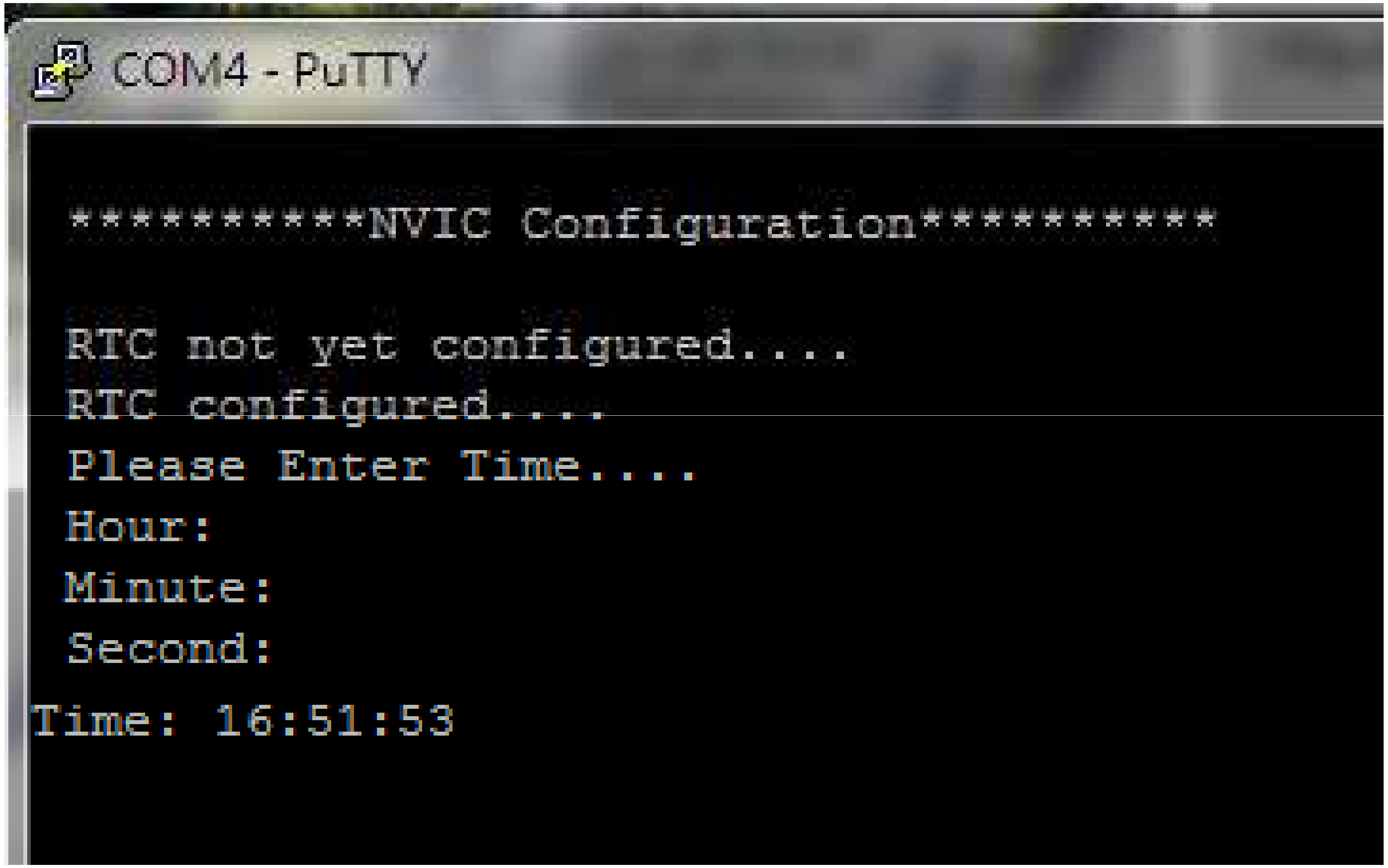
實驗2 – RTC時間調整

- 利用 RTC second interrupt來產生時間，並用Virtual COM Port 在超級終端機上顯示時間。
- 請修改u32 Time_Regulate(void) ，讓使用者自行輸入現在時間， 使用printf()與scanf() 。

Sample code

```
u32 Time_Regulate(void)
{
  u32 Tmp_HH = 0xFF, Tmp_MM = 0xFF, Tmp_SS = 0xFF;
/*    modify your code
    Tmp_HH =0;     Tmp_MM = 0;      Tmp_SS = 0;
*/                          *
  /* Return the value to store in RTC counter register */
  return((Tmp_HH*3600 + Tmp_MM*60 + Tmp_SS));
}
void Time_Adjust(void)
{
   RTC_WaitForLastTask(); /* Wait until last write operation has finished */
   RTC_SetCounter(Time_Regulate()); /* Change the current time */
   RTC_WaitForLastTask(); /* Wait until last write operationhas finished */
}
```

LAB2 DEMO



```
 ***********NVIC Configuration***********

 RTC not yet configured....
 RTC configured....
 Please Enter Time....
 Hour:
 Minute:
 Second:
Time: 16:51:53
```

實驗3 – RTC Alarm Interrupt

- 利用RTC alarm interrupt 製作鬧鐘，請使用者先輸入現在時間，再輸入鬧鐘時間。
- 請參考RTC second interrupt 的使用方式，修改

  void RTC_Configuration(void)

  void Time_Adjust(void)

  void RTC_IRQHandler(void)

RTC Configure

```
/* RTC Configure */
/* Enable the RTC Second */
  RTC_ITConfig(RTC_IT_ALR, ENABLE );
/* Wait until last write operation on RTC registers is finished */
  RTC_WaitForLastTask();
```

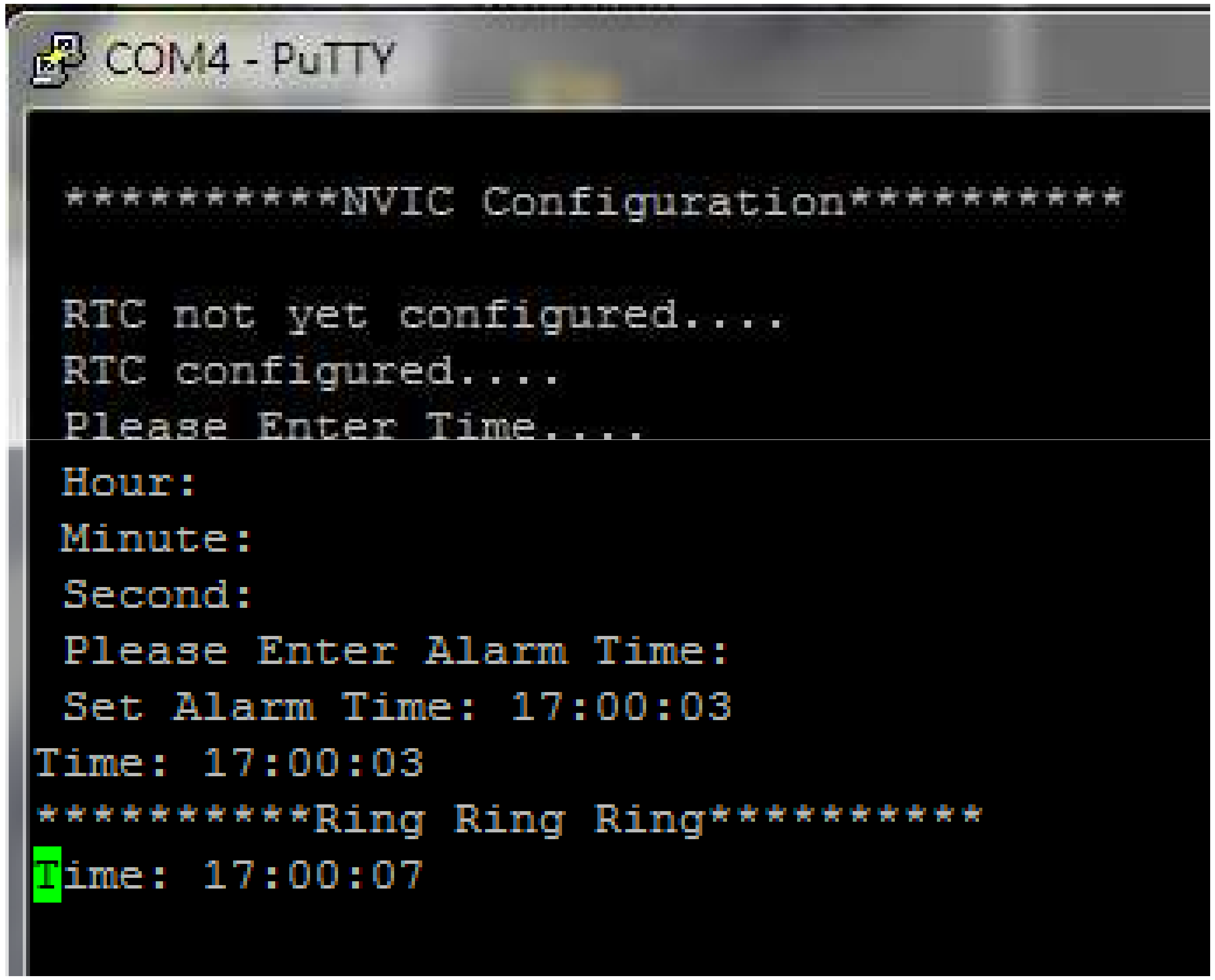
Time Adjust

```
void Time_Adjust(void)
{
  RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers has finished */
  RTC_SetCounter(Time_Regulate()); /* Change the current time */
  RTC_WaitForLastTask(); /* Wait until last write operation on RTC registers has finished */

/*   modify your code
    RTC_SetAlarm(0x10);
    RTC_WaitForLastTask();

*/
}
```

RTC Interrupt Handler

```
/* Interrupt Handler */
void RTC_IRQHandler(void)
{
   if (RTC_GetITStatus(RTC_IT_SEC) != RESET)  {
     RTC_ClearITPendingBit(RTC_IT_SEC); /* Clear RTC Second interrupt */
     TimeDisplay = 1; /* Enable time update */
     /* Wait until last write operation on RTC registers has finished */
     RTC_WaitForLastTask();
     /* Reset RTC Counter when Time is 23:59:59 */
     if (RTC_GetCounter() == 0x00015180) {
              RTC_SetCounter(0x0);
               /* Wait until last write operation on RTC registers has finished */
              RTC_WaitForLastTask();
     }
   }
/*    modify your code
              if(RTC_GetITStatus(RTC_IT_ALR) != RESET){
                 …………
              } */
}
```

LAB3 DEMO



```
 **********NVIC Configuration**********

 RTC not yet configured....
 RTC configured....
 Please Enter Time....
 Hour:
 Minute:
 Second:
 Please Enter Alarm Time:
 Set Alarm Time: 17:00:03
Time: 17:00:03
**********Ring Ring Ring**********
Time: 17:00:07
```