Power Control (PWR)

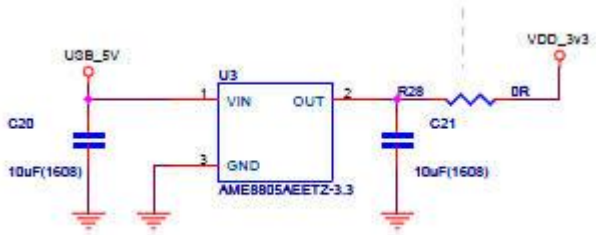
2010.02.08

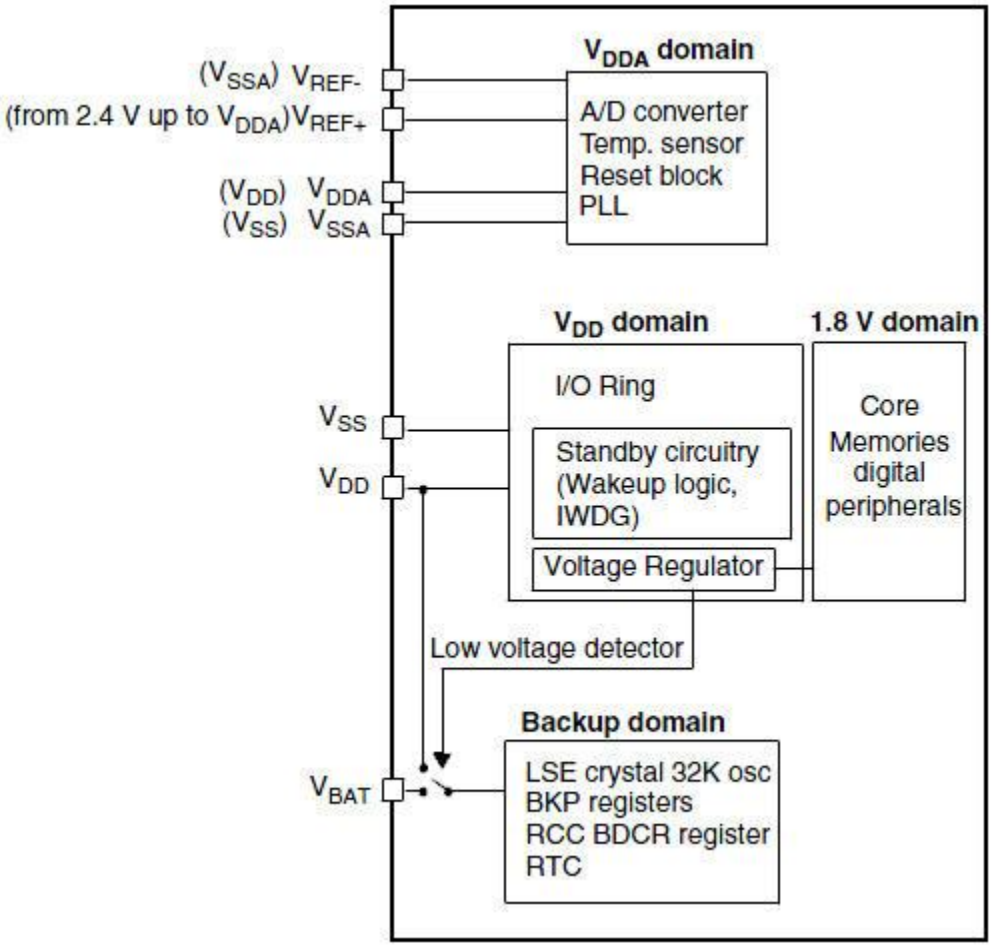
Low power mode

- Low Power로 어떻게 동작시킬 것인가 하는 부분과 깨어나는 상황 등에 대해 살펴본다.

- 예제 코드는 3가지
  - PWR1_STOP
  - PWR2_STOP_RTCAlarm
  - PWR3_STANDBY

Power domain





- VDD에 공급되는 전원은 2.0에서 3.6V까지 지원
- 망고보드는 3.3V를 사용하고 있다.
- Voltage Regulator가 내부 1.8V의 전원을 공급

Voltage Regulator

- Voltage Regulator는 리셋 후에 항상 활성화 되어 있다.
- low power mode를 Voltage Regulator 관점에서 3가지로 구분
  - Run mode: 1.8 V domain (core, memory, digital peripheral)에 최대 전력을 공급
  - Stop mode: 1.8 V domain에 레지스터와 SRAM의 내용을 보존해 줄 수 있는 정도의 적은 전력만을 공급
  - Standby mode: regulator의 전력 공급 중단. 만약 Standby circuitry와 Backup Domain에 전력 공급이 없다면 레지스터와 SRAM의 내용은 보존되지 않음.

- RTC (real-time clock)와 BKP (backup) 레지스터를 위해서는 따로 전원을 공급해 줄 수 있다. 이 부분이 VBAT 부분이고, VDD 공급이 차단되었을 경우에도 VBAT을 통해 전원 공급이 가능하다.
  - 망고보드는 이 부분을 따로 전원을 공급하는 것이 아니라 똑같이 3.3V를 연결해서 사용하고 있다.

Sleep mode와 Deep Sleep mode

Table 26. Low-power mode wakeup timings

Symbol	Parameter	Conditions	Typ	Unit
$t_{WUSLEEP}$(1)	Wakeup from Sleep mode	Wakeup on HSI RC clock	1.8	µs
t_{WUSTOP}(1)	Wakeup from Stop mode (regulator in run mode)	HSI RC wakeup time = 2 µs	3.6	µs
	Wakeup from Stop mode (regulator in low power mode)	HSI RC wakeup time = 2 µs, Regulator wakeup from LP mode time = 5 µs	5.4	
$t_{WUSTDBY}$(1)	Wakeup from Standby mode	HSI RC wakeup time = 2 µs, Regulator wakeup from power down time = 38 µs	50	µs

- 3가지 low-power mode를 크게 나누면 **Sleep mode와 Deep Sleep** mode로 나눌 수 있다
  - Stop mode, Standby mode는 Deep Sleep mode의 2가지 종류
- 전력소모가 많고 적음과 관련이 있는 요소로서 다시 동작을 하게 되기까지의 시간(**startup time**)과 동작을 하게 만드는 원인이 되는 부분(**wakeup sources**)을 들 수 있다. 전력소모가 적을수록 다시 동작이 되기까지의 시간이 오래 걸리고, 깨어나게 할수 있는 방법 또한 적어진다.
- 위 표의 각각의 시간은 wakeup event가 발생한 시점에서부터 사용자의 application code의 첫 번째 명령을 읽는 순간까지를 측정한 것
  - Standby mode와 같이 초 저전력 모드로 동작 중이다가 깨어나는 것은 시간이 10배 이상 많이 소모

Sleep mode

Table 14. Maximum current consumption in Run mode, code with dat running from RAM

Symbol	Parameter	Conditions	f_{HCLK}	Max(1)	
				T_A = 85 °C	T_A = 1
I_{DD}	Supply current in Run mode	External clock(2), all peripherals enabled	72 MHz	48	5(
			48 MHz	31.5	3:
			36 MHz	24	25
			24 MHz	17.5	1(
			16 MHz	12.5	1:
			8 MHz	7.5	8
		External clock(2), all peripherals disabled	72 MHz	29	29
			48 MHz	20.5	2:
			36 MHz	16	16
			24 MHz	11.5	1:
			16 MHz	8.5	9
			8 MHz	5.5	6

Table 15. Maximum current consumption in Sleep mode, code running from Flash or RAM

Symbol	Parameter	Conditions	f_{HCLK}	Max(1)		Unit
				T_A = 85 °C	T_A = 105 °C	
I_{DD}	Supply current in Sleep mode	External clock(2), all peripherals enabled	72 MHz	30	32	mA
			48 MHz	20	20.5	
			36 MHz	15.5	16	
			24 MHz	11.5	12	
			16 MHz	8.5	9	
			8 MHz	5.5	6	
		External clock(2), all peripherals disabled	72 MHz	7.5	8	
			48 MHz	6	6.5	
			36 MHz	5	5.5	
			24 MHz	4.5	5	
			16 MHz	4	4.5	
			8 MHz	3	4	

- Sleep mode에서는 오직 CPU core만 멈춘다. 모든 외부의 peripheral 들은 그 동작을 정상적으로 수행한다.
- 깨나는 동작 또한 모든 interrupt나 event가 발생하면 바로 깨어난다
  - ➢ 사용자의 입장에서 Sleep mode에 들어갔는지 아닌지를 거의 느낄 수 없다
- 8 MHz에서 7.5 mA -> 5.5 mA. 약 25% 정도의 전력 소모 감소
- Stop mode나 Standby mode에서 전력 소모의 단위는 uA.
- 동작 주파수가 늘어나는 것에 비례해서 소모 전류 또한 늘어난다

Deep sleep mode

Symbol	Parameter	Conditions	Typ(1)		Max		Unit
			V_{DD}/V_{BAT} = 2.4 V	V_{DD}/V_{BAT} = 3.3 V	T_A = 85 °C	T_A = 105 °C	
I_{DD}	Supply current in Stop mode	Regulator in Run mode, low-speed and high-speed internal RC oscillators and high-speed oscillator OFF (no independent watchdog)	23.5	24	200	370	µA
		Regulator in Low Power mode, low-speed and high-speed internal RC oscillators and high-speed oscillator OFF (no independent watchdog)	13.5	14	180	340	
	Supply current in Standby mode	Low-speed internal RC oscillator and independent watchdog ON	2.6	3.4	-	-	
		Low-speed internal RC oscillator ON, independent watchdog OFF	2.4	3.2	-	-	
		Low-speed internal RC oscillator and independent watchdog OFF, low-speed oscillator and RTC OFF	1.7	2	4	5	
I_{DD_VBAT}	Backup domain supply current	Low-speed oscillator and RTC ON	1.1	1.4	1.9(2)	2.2	

- Stop mode: SRAM, 레지스터들 상태 유지하면서 매우 낮은 전력 소모
  - Stop mode에서 깨나기 위해서 EXTI line 중의 하나를 받아야만 한다.
  - 16개의 external line 중 하나, PVD output, RTC alarm, USB wakeup
- Standby mode는 가장 낮은 전력 소모를 얻을 수 있는 모드
  - SRAM과 레지스터들의 내용은 모두 잃어버리게 된다. 다만 Backup domain 이나 Standby 관련 부분에 일부 내용만 남아있게 된다.
- Standby mode에서 깨어나는 방법은 아래의 4가지
  - external reset (NRST pin), IWDG reset
  - rising edge on the WKUP pin, RTC alarm

PWR 예제 1 - STOP mode 설정

```
void EXTI0_IRQHandler(void) {
    if(EXTI_GetITStatus(GPIO_EXTI_Line_KEY1) != RESET) {
        LED_On_Yellow();
..... ..... ..... ..... ..... .....
}
void EXTI1_IRQHandler(void) {
    if(EXTI_GetITStatus(GPIO_EXTI_Line_KEY2) != RESET) {
        LED_On_Red();
..... ..... ..... ..... ..... .....
}
```

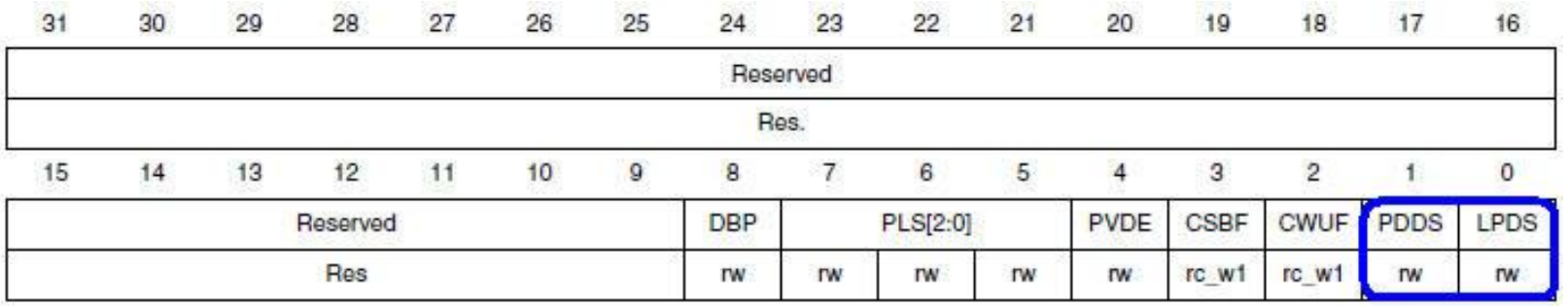
- KEY1이 눌렸을 경우에는 Yellow LED를 켜고
- KEY2가 눌렸을 경우는 Red LED를 켜도록 변경

PWR_EnterSTOPMode

```
#define PWR_Regulator_LowPower     ((uint32_t)0x00000001)
#define PWR_STOPEntry_WFI          ((uint8_t)0x01)
#define PWR_STOPEntry_WFE          ((uint8_t)0x02)
/* Request to enter STOP mode with regulator in low power m
ode*/
PWR_EnterSTOPMode
   (PWR_Regulator_LowPower, PWR_STOPEntry_WFI);
```

- Power control register (PWR_CR)
  - Stop mode로 동작하면서,
  - 이 때 Voltage regulator를 low-power mode로 동작하게 한다.
- System control register (SCB_SCR)도 함께 설정

Power control register (PWR_CR)

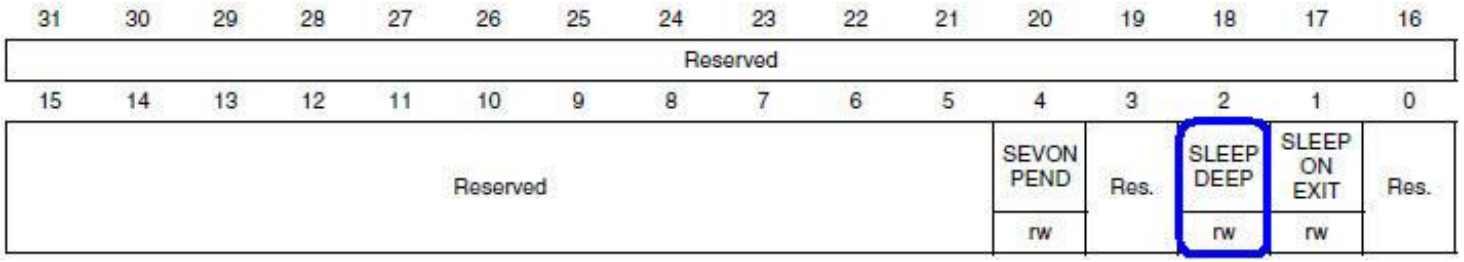


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Res.															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
Res							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

- PDDS, LPDS 두 비트는 함께 설정이 되어야 하고 같이 동작하는 비트
- DS가 의미하는 것이 deep sleep을 의미
- **Bit 1 PDDS: Power down deep sleep**
  - 1이면 CPU가 Deep sleep 상태가 될 때 Standby mode
  - 0이면 CPU가 Deep sleep 상태가 될 때 Stop mode
- **Bit 0 LPDS: Low-power deep sleep**
  - 이 비트 값은 Stop mode에서만 작동하는 비트
  - 값이 0이면 Voltage regulator가 On 상태
  - 값이 1이면 Stop mode 동안 Voltage regulator는 low-power mode로 동작한다.

System control register (SCB_SCR)



- SCB_SysCtrl은 ((uint32_t)0xE000ED10)으로 정의되어 있고, 이것은 System control register (SCB_SCR)를 의미한다.
- CPU를 deep sleep으로 보낼 수 있도록 하는 비트가 2번 비트이다.
- WFI나 WFE를 호출해서 low power mode로 진입할때 이 2번 비트가 1로 되어 있으면 CPU는 deep sleep mode로 들어가게 된다.

SYSCLKConfig_STOP

```
void SYSCLKConfig_STOP(void) {
    RCC_HSEConfig(RCC_HSE_ON);  /* Enable HSE */
    if(RCC_WaitForHSEStartUp() == SUCCESS) {  /* Wait till HSE is ready */
        RCC_PLLCmd(ENABLE);   /* Enable PLL */
        /* Wait till PLL is ready */
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) { }
        /* Select PLL as system clock source */
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
        /* Wait till PLL is used as system clock source */
        while(RCC_GetSYSCLKSource() != 0x08) { }
}}
```

- Stop mode에서 깨어날 때 Stop mode 상에서 꺼져있던 클럭 초기화
  - ➢ 처음 리셋 되었을 때 클럭을 초기화하는 과정과 거의 동일한 작업
  - ➢ 이미 설정된 레지스터들 값을 설정하는 작업은 불필요. 그 부분들은 Stop mode에서도 그대로 저장이 되어 있는 상태이므로 다시 설정할 필요가 없다.
- 3가지 작업을 수행한다.
  1) HSE를 On하고 Ready 비트가 설정될 때까지 기다리는 작업. 외부 클럭 12MHz를 사용하고 있기 때문에 그 부분에 대한 설정을 가장 먼저 수행
  2) PLL을 On하고 PLL이 Lock될 때까지 기다리는 부분
  3) 마지막으로 System clock에 대한 소스로 PLL을 설정하고 기다리는 부분

수행 결과

- 5번 메뉴에 Stop mode를 시험하는 코드 추가
- 처음 모든 LED를 끄고 난 다음 Stop mode에 진입
- 프로그램의 수행은 바로 그 시점에서 멈춘다.
- 어떠한 인터럽트를 발생하기 전에는
  Stop mode에서 빠져나올 수 없다.

- 만약 Key0나 Key1을 눌러서 EXTI interrupt를 발생시키게 되면 바로 그 순간에 Stop mode에서 빠져 나와서 SYSCLKConfig_STOP()를 수행해서 클럭을 정상적으로 동작하게 만든 이후에
  <<Exit Stop mode>>를 출력
- Stop mode를 빠져 나오는 인터럽트로 사용하려는 것이 key interrupt 이다. Key1이나 Key2를 누르면 Stop mode에서 빠져나온다.
- Key1을 눌렀을 경우는 Yellow LED가 켜지고, Key2를 누르면 Red LED가 켜지면서 화면에는 <<Exit Stop mode>>를 출력하면서 Stop mode에서 빠져 나와서 다시 메뉴를 출력하게 된다.

PWR 예제 2 - RTC Alarm 설정

- Stop mode에 진입했다가 깨어나는 용도로 RTC Alarm을 사용

```
void NVIC_Configuration(void) {
..... ..... ..... ..... ..... .....
    /* Enable the RTCAlarm Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = RTCAlarm_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

```
void EXTI_Configuration(void) {
..... ..... ..... ..... ..... .....
    /* EXTI Line17(RTC Alarm) to generate an interrupt on rising edge */
    EXTI_ClearITPendingBit(EXTI_Line17);
    EXTI_InitStructure.EXTI_Line     = EXTI_Line17;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);
}
```

RTC_Configuration

```
void RTC_Configuration(void) {
    /* RTC clock source configuration */ /* Allow access to BKP Domain */
    PWR_BackupAccessCmd(ENABLE);

    BKP_DeInit(); /* Reset Backup Domain */
    RCC_LSICmd(ENABLE);
    while(RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET) { }

    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI); /* Select RTC Clock Source */
    RCC_RTCCLKCmd(ENABLE); /* Enable the RTC Clock */
    RTC_WaitForSynchro(); /* Wait for RTC APB registers synchronisation */
    RTC_WaitForLastTask(); /* Wait until last write on RTC registers finished */

    /* Set the RTC time base to 1s */
    RTC_SetPrescaler(39999);
            /* RTC period = RTCCLK/RTC_PR = (40 KHz)/(39999+1) */
    RTC_WaitForLastTask(); /* Wait until last write on RTC registers finished */

    RTC_ITConfig(RTC_IT_ALR, ENABLE); /* Enable the RTC Alarm interrupt */
    RTC_WaitForLastTask(); /* Wait until last write on RTC registers finished */
}
```

main - RTC_SetAlarm

```
int main(void) {
..... ..... ..... ..... ..... .....
    /* Enable PWR and BKP clock */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);
    /* Configure RTC clock source and prescaler */
    RTC_Configuration();
..... ..... ..... ..... ..... .....
        case '5':
            RTC_ClearFlag(RTC_FLAG_SEC); /* Wait till RTC Second event occurs */
            while(RTC_GetFlagStatus(RTC_FLAG_SEC) == RESET);
            RTC_SetAlarm(RTC_GetCounter() + 3); /* Alarm in 3 second */
            RTC_WaitForLastTask(); /* Wait until last write on RTC registers has finished */
            LED_Off_All();
            /* Request to enter STOP mode with regulator in low power mode*/
            PWR_EnterSTOPMode(PWR_Regulator_LowPower, PWR_STOPEntry_WFI);
            /* Configures system clock after wake-up from STOP: enable HSE, PLL and select
            PLL as system clock source (HSE and PLL are disabled in STOP mode) */
            SYSCLKConfig_STOP();
            printf("<<Exit Stop mode>>₩n");
            break;
        }
..... ..... ..... ..... ..... .....
}
```

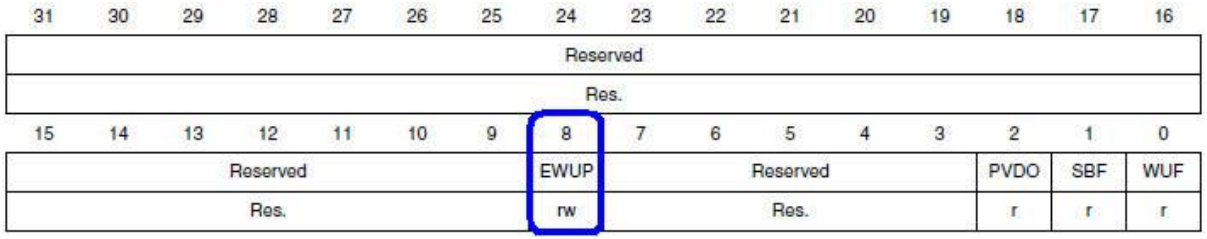
PWR 예제 3 - STANDBY mode

```
void EXTI1_IRQHandler(void)
{
    if(EXTI_GetITStatus(GPIO_EXTI_Line_KEY2) != RESET) {
        EXTI_ClearITPendingBit(GPIO_EXTI_Line_KEY2);
                                /* Clear EXTI line pending bit */
        LED_On_Yellow(); /* Turn on Yellow LED */

        RTC_ClearFlag(RTC_FLAG_SEC); /* Wait till RTC Second event occurs */
        while(RTC_GetFlagStatus(RTC_FLAG_SEC) == RESET);
        RTC_SetAlarm(RTC_GetCounter()+ 3); /* Set the RTC Alarm after 3s */
        RTC_WaitForLastTask();
                /* Wait until last write operation on RTC registers has finished */
        /* Request to enter STANDBY mode
          (Wake Up flag is cleared in PWR_EnterSTANDBYMode function) */
        PWR_EnterSTANDBYMode();
        printf("Right-USER Button Press₩n");
..... ..... ..... ..... ..... .....
}
```

PWR_WakeUpPinCmd

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);
PWR_WakeUpPinCmd(ENABLE); /* Enable WKUP pin */
PWR_BackupAccessCmd(ENABLE); /* Allow access to BKP Domain */
RTC_Configuration(); /* Configure RTC clock source and prescaler */
```



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
Res.															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EWUP	Reserved					PVDO	SBF	WUF
Res.							rw	Res.					r	r	r

- 위 내용은 Power control/status register (PWR_CSR)의 8번 비트를 제어하는 함수 입니다.
- 이 값을 0으로 만들면 WKUP pin (GPIO A 0, 우리의 경우는 Key1이 할당되어 있다)을 일반 용도의 GPIO로 사용하게 된다.
- 이 때는 더 이상 이 핀으로 Standby mode에서 깨나게 만들 수 없다.
- 이 값을 1로 설정해야만 Standby mode에 대한 wakeup 용도로 사용할 수 있는 것이다.
- WKUP pin의 rising edge가 Standby mode로부터 wakeup을 시킨다
- 이 비트는 시스템이 리셋 되면 0으로 바뀐다.

RTC_Configuration

```
void RTC_Configuration(void)
{
    /* Check if the StandBy flag is set */
    if(PWR_GetFlagStatus(PWR_FLAG_SB) != RESET)
    { /* System resumed from STANDBY mode */
        LED_On_Red(); /* Turn on RED LED */
        PWR_ClearFlag(PWR_FLAG_SB); /* Clear StandBy flag */
        RTC_WaitForSynchro(); /* Wait for RTC APB registers synchronisation */
        /* No need to configure the RTC as the RTC configuration(clock source, enable,
        prescaler,...) is kept after wake-up from STANDBY */
    } else { /* StandBy flag is not set */
        이 부분은 PWR 예제 2 - RTC Alarm 설정 부분의 RTC_Configuration()과 동일
    }
}
```

- PWR_GetFlagStatus()를 통해서 이 함수가 시작되는 상태가 처음 power를 켜서 수행이 된 것인지 아니면 standby mode에서 깨나서 수행이 되는 것인지를 구분해서 동작하고 있다.
- 만약 처음 power를 켜서 수행이 된 것이라면 이전 절의 예제에서 수행하던 것과 동일하게 수행하게 하면 된다.

실행 결과

- Key2 User button을 누르면 시스템은 Standby mode로 진입 한다.
- 이때 3초를 기다리거나 혹은 Key1 Wake up button을 누르면 깨어나게 되는데 밑줄 친 부분을 보면 Mango Z1 test start ... 가 찍혀있다.
  - ➢ 프로그램이 처음부터 다시 시작하고 있다.
- stop mode에서는 모든 메모리 공간이 살아 있기 때문에 멈추었던 바로 그 부분에서 다시 시작할 수 있었지만 standby mode에서는 모든 것들이 사라지기 때문에 프로그램은 처음부터 다시 시작하는 것이다.

- 최초 프로그램이 시작하면 Blue LED가 주기적으로 점등 된다. 이때 Key2를 누르면 Standby mode로 진입한다. Yellow LED가 잠시 켜지는 것을 볼 수 있지만 이내 모든 LED가 꺼진다.
- 다시 깨어나서 동작하게 될 때는 Blue LED가 주기적으로 점등되는 것은 동일하지만 Red LED가 켜져 있는 것이 다르다. Power on reset을 통해서 시작한 것인지 Standby mode에서 깨어난 것인지를 확인할 수 있다.