Hi everyone,

I'm working on a project and i'm a beginner on STM32F0. I have the discovery board. I have the code at the bottom of the page but i don't understand many parts of it...

In this code i have the following question :

What's the difference between this instruction BUTTON_MODE_GPIO and this instruction BUTTON_MODE_EXTI. There is someting with the interruptions... I don't really understand. When the button mode is exti, when i press the user button, I generate an interruption ? And when the button mode is GPIO what happens ? It's a bit difficult for me ...

Normally, it works like that :

"After reset, the program waits for User button connected to the PA.00 to be pressed to enter the selected low power mode. " In the below code, it's the SLEEP MODE.
"- When the RTC is used, the wakeup from low power mode is automatically generated by the RTC (after 5s). " useffull for the STOP MODE
"- In Sleep mode and Standby mode, press again the User button to exit the low power mode."

I think the BUTTON MODE EXTI is used to wake up the device from the sleep mode. But i don't know how it works... I attached two extract extracted from the datasheet of the STM32. It describes what is the sleep mode and how do we enter in and how can we go out from it.

6.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex®-M0 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

In the Sleep mode, all I/O pins keep the same state as in the Run mode.

Refer to *Table 15* and *Table 16* for details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex®-M0 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Refer to *Table 15* and *Table 16* for more details on how to exit Sleep mode.

Table 75. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

I don't understand also these instructions :

```
/* Enable PWR APB1 Clock */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);

/* Allow access to Backup */
PWR_BackupAccessCmd(ENABLE);

/* Reset RTC Domain */
RCC_BackupResetCmd(ENABLE);
RCC_BackupResetCmd(DISABLE);?????????
```

I don't know what is a backup Register. I look for in the datasheet but i didn't find them ... and i don't know what is the RCC domain ... If someone can explain me what is it ? And what is the function of these instructions in the above lines of code ?

Thank you very much for your help. It's very difficult for me. I hope that I will be better in few weeks...

Have a nice day,

```
int main(void)
{
 /*!< At this stage the microcontroller clock setting is already configured,
 this is done through SystemInit() function which is called from startup
 file (startup_stm32f0xx.s) before to branch to application main.
 To reconfigure the default setting of SystemInit() function, refer to
 system_stm32f0xx.c file
 */
 /* Configure User Button */
 STM_EVAL_PBInit(BUTTON_USER,BUTTON_MODE_GPIO);

 /* Loop while User button is maintained pressed */
 while(STM_EVAL_PBGetState(BUTTON_USER) != RESET)
 {
 }

 /* Enable PWR APB1 Clock */
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);

 /* Allow access to Backup */
 PWR_BackupAccessCmd(ENABLE);

 /* Reset RTC Domain */
 RCC_BackupResetCmd(ENABLE);
 RCC_BackupResetCmd(DISABLE);

 /* Loop while User button is maintained pressed */
 while(STM_EVAL_PBGetState(BUTTON_USER) == RESET)
 {
```

```
  }
  /* Loop while User button is maintained pressed */
  while(STM_EVAL_PBGetState(BUTTON_USER) != RESET)
  {
  }

 #if defined (SLEEP_MODE)
  /* Sleep Mode Entry
  - System Running at PLL (48MHz)
  - Flash 3 wait state
  - Prefetch and Cache enabled
  - Code running from Internal FLASH
  - All peripherals disabled.
  - Wakeup using EXTI Line (User Button PA.00)
  */
  SleepMode_Measure();

 }??????????????????????????????????????????????

 ------------------------------------------------------------------------------------------------------------------------
 -------------------------

 void SleepMode_Measure(void)
 {
  __IO uint32_t index = 0;
  GPIO_InitTypeDef GPIO_InitStructure;
 /* Configure all GPIO as analog to reduce current consumption on non used IOs */
  /* Enable GPIOs clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA | RCC_AHBPeriph_GPIOB | RCC_AHBPeriph_GPIOC |
  RCC_AHBPeriph_GPIOD | RCC_AHBPeriph_GPIOF , ENABLE);

  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  GPIO_Init(GPIOF, &GPIO_InitStructure);
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  GPIO_Init(GPIOB, &GPIO_InitStructure);
 /* Disable GPIOs clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA | RCC_AHBPeriph_GPIOB | RCC_AHBPeriph_GPIOC |
  RCC_AHBPeriph_GPIOD | RCC_AHBPeriph_GPIOF, DISABLE);
 /* Configure User Button */
  STM_EVAL_PBInit(BUTTON_USER, BUTTON_MODE_EXTI);
 /* Request to enter SLEEP mode */
  __WFI();
 /* Initialize LED4 on STM32F0-Discovery board */
  STM_EVAL_LEDInit(LED4);
 /* Infinite loop */
  while (1)
  {
  /* Toggle The LED4 */
  STM_EVAL_LEDToggle(LED4);
 /* Inserted Delay */
  for(index = 0; index < 0x7FFFF; index++);
  }
 }??????????????????????????????????????
```