## **GPIO\_INPUT**

Main.c

|  |
| --- |
|  |

BML\_GPIO.c

(Double click to open)

|  |
| --- |
|  |

BML\_GPIO.h

(Double click to open)

|  |
| --- |
|  |

***GPIO Functions:-***

***GPIO Interrupt Functions***

* gpio\_IT\_config(port, pinNumber, edge)
* gpio\_IT\_EN(pinNumber, irqNumber)
* gpio\_IT\_DI(pinNumber, irqNumber)
* gpio\_IT\_CLR(pinNumber)

***How to use Functions:-***

* First, configure the GPIO’s port, pin, output, speed and pupd.
* Next, configure the pin for interrupt using ‘gpio\_IT\_config’.
* In that function, we have to put PORT, PIN and a third parameter which is edge.
* The edge is for the interrupt to happen either at RISING\_EDGE, when the button is pressed making the pin ‘0’ to ‘1’ or FALLING\_EDGE, when Pin is reading ‘1’ to ‘0’ or RISING\_FALLING\_EDGE, it will interrupt the pin at both ‘0’ to ‘1’ and ‘1’ to ‘0’.
* After configuring the interrupt, we have to enable it using ‘gpio\_IT\_EN’.
* pinNumber parameter is known but the irqNumber is not. There is an EXTI controller in our controller which connects the Pin interrupt to NVIC and then NVIC to MCU.
* In the reference manual, there is a vector table in Interrupts section. In that we can see the EXTI lines. The numbers in EXTI lines corresponds to the pin position. It shows where our irqNumber lies.
* *In the project folder, Go to -> Project Name\Drivers\CMSIS\Device\ST\STM32F0xx\Include and in that include there is a file named ‘stm32f070xb.h’*
* Search for IRQn\_type, it is a typedef enum and inside look for **EXTI4\_15\_IRQn**. This is our irqNumber for lines from ‘4’ to ‘15’ and our pin in 13 for the blue button.
* Use this in the ‘gpio\_IT\_EN’ function to enable it.
* Next step, is to make a handler of it. Handler function name is nothing but the irqNumber and ‘n’ replaced by ‘Handler’ i.e. EXTI4\_15\_IRQn to EXTI4\_15\_IRQHandler.
* Write any code which to happen when interrupt triggers, at the end of interrupt clear the Interrupt flag using ‘gpio\_IT\_CLR’.

***How the function works?***

1. **GPIO\_IT\_CONFIG function -** The EXTI controller is what takes the trigger to NVIC line and then from NVIC line to MCU. But there is another register which connects the GPIO to EXTI lines. That register is **System configuration controller (SYSCFG).** We have to enable it according to the Pin on which we want the interrupt.

* First we have to enable its clock to get it starting.
* There are four EXTI registers in the SYSCONFIG, and in each register there are 4 sets of EXTI. In first EXTI control register 0, the four sets are EXTI0, EXTI1, EXTI2 and EXTI3 and next 4 to 15 are in other registers and need to be set according to the Pin number like for Pin 13 we have to set the EXTI13.
* In that EXTI13, there are 4 bits but we need 3 bits to select the port of the pin we are using. That is what is happening in our long function.
* Going back to Interrupt register in EXTI, we will select the ‘edge’ of interrupt.
* Rising edge is set by shifting ‘1’ to the pin positon in the RTSR register.
* For falling edge, similar process but in FTSR register.
* For both, set both bit in the RTSR and FTSR register.

1. **GPIO\_IT\_EN function –** To enable the interrupt, we have to use the **‘Interrupt Mask register**’ and set the bit according to the Pin position and this will enable the interrupt. This function also have the NVIC\_EnableIRQ() function which takes the IRQNumber to enable the interrupt at NVIC line. This is specific to ARM Core architecture.
2. **GPIO\_IT\_DI function –** Just opposite of enable fumction. It shifts ‘0’ to the IMR register according to pin position.
3. **GPIO\_IT\_CLR function –** After our interrupt is triggered, a bit in a register will be set to ‘1’ on triggering and that register is **EXTI\_Pending\_register.** When this register is set, the interrupt handler function will process and will keep processing until PR register in ‘1’ even when interrupt is not happening. So, we need to clear it and make in zero. This is what our function is doing. According to data sheet this bit is cleared by writing ‘1’ to it and not ‘0’.