Understanding EXTI, Counters and DMA

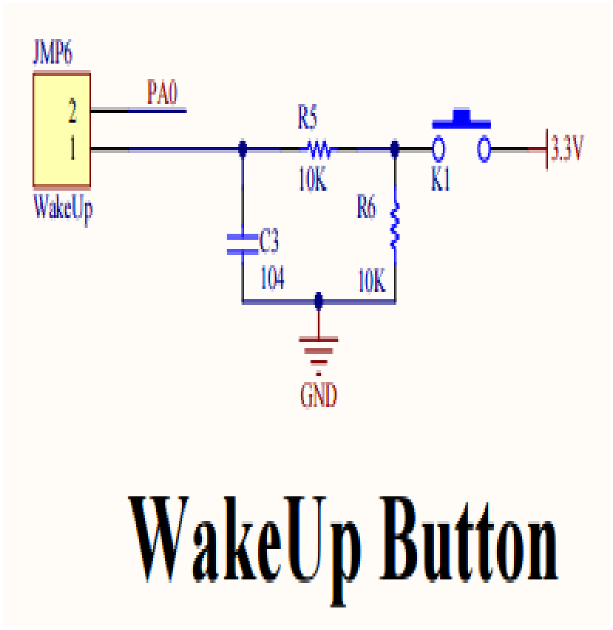
Basics of EXTI

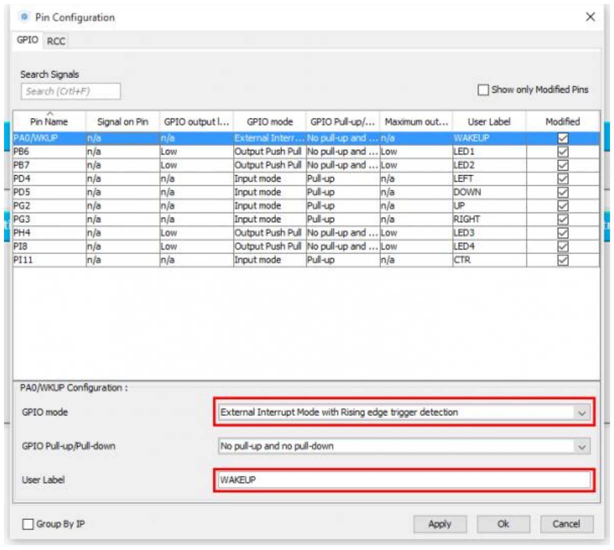
The EXTI (External Interrupt/Event) controller consists of up to 40 edge detectors for generating event/interrupt requests on STM32L47x/L48x devices. Each input line can be independently configured to select the type (interrupt or event) and the corresponding trigger event (rising, falling, or both).

All interrupts, including the core exceptions, are managed by the NVIC. The NVIC and the processor core interface are closely coupled, which ensures a low interrupt latency and enables the efficient processing of late-arriving interrupts. Access to the NVIC's control and status registers is performed through the Private Peripheral Bus (or PPB) internal to the Cortex®-M4 CPU.

The NVIC provides a fast response to interrupt requests, allowing an application to quickly serve incoming events. An interrupt is handled without waiting for the completion of a long instructions sequence. These instructions will be either restarted or resumed upon the interrupt return. The priority assigned to each interrupt request is programmable and can be dynamically changed.

By looking at the datasheet, one can realize that certain pins can be enabled with the EXTI feature by changing its configuration to GPIO_EXTI0 mode. This leads to the 'Wakeup Key' to be connected to the configured pin. In the GPIO configuration, set the pin as rising edge trigger. Select No pull-up and no pull-down in the option GPIO Pull-up/Pull-down. In the user label box, add the label WAKEUP. In NVIC, check the option EXTI Line0 interrupt to enable the interrupt. And the two options on the right are used to set the preemption priority and sub-priority. Here, we remain the default settings.





Pin Name	Signal on Pin	GPIO output l...	GPIO mode	GPIO Pull-up/...	Maximum out...	User Label	Modified
PA0/WKUP	n/a	n/a	External Interr...	No pull-up and ...	n/a	WAKEUP	✓
PB6	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED1	✓
PB7	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED2	✓
PD4	n/a	n/a	Input mode	Pull-up	n/a	LEFT	✓
PD5	n/a	n/a	Input mode	Pull-up	n/a	DOWN	✓
PG2	n/a	n/a	Input mode	Pull-up	n/a	UP	✓
PG3	n/a	n/a	Input mode	Pull-up	n/a	RIGHT	✓
PH4	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED3	✓
PI8	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED4	✓
PI11	n/a	n/a	Input mode	Pull-up	n/a	CTR	✓

PA0/WKUP Configuration :

GPIO mode	External Interrupt Mode with Rising edge trigger detection
GPIO Pull-up/Pull-down	No pull-up and no pull-down
User Label	WAKEUP

Some important parts to code:-

1.EXTI1 interrupt handler:

```
void EXTI1_IRQHandler(void)
{
   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_1);
}
```

2.GPIO Configuration to enable EXTI

```
    GPIO_InitStruct.Pin = GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

3.Enabling IRQ

```
   HAL_NVIC_SetPriority(EXTI1_IRQn, 3, 0);
   HAL_NVIC_EnableIRQ(EXTI1_IRQn);
```

4.Customized Callback

```
    void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
   {
    if(GPIO_Pin == STEP_PIN)
    {
      GlobalMotorData.step.longint++;
    }
   }
```

Understanding Counters

The basic difference between Timers and Counters is that Timers are configured to count internal clock signals while Counters are configured to count external pulses.

This means that when the timer gets clocked from an external source (input pin) and it counts the number of pulses. However, STM32 timer modules do have multiple modes for the counting mode itself. Here is a brief description for each of them, but at the end of the day, we'll be using the up-counting mode.

Here is a brief description for each of them, but at the end of the day, we'll be using the up-counting mode:-

1. **Up-counting Mode**
   In up-counting mode, the counter counts from 0 to the auto-reload value (the content of the TIMx_ARR register), then restart from 0 and generates a counter overflow event. An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

2. **Down-counting Mode**
   In down counting mode, the counter counts from the auto-reload value (the content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event. An Update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

3. **Center-Aligned Mode (Up/Down)**
   In center-aligned mode, the counter counts from 0 to the auto-reload value (the content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0. In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The counters can be enabled by going through the basic following steps:-

1. Configuring Timer Peripherals to operate in the counter mode
2. Enabling the external interrupt signal in NVIC tab
3. Configure the USART Module to operate in async mode with 9600bps
4. Set a proper RCC Clock Source
5. In Clock Configurations, set the required clock frequency

STUDY OF DMA

DMA stands for Direct Memory Access controller.

DMA is a bus master and system peripheral providing high-speed data transfers between peripherals and memory, as well as memory-to-memory. Data can be quickly moved by DMA without any CPU action, keeping CPU resources free for other operations.

Many boards are known to have two DMAs where each DMA channel is dedicated to managing memory access requests from one or more peripherals. The two DMA controllers have 14 channels in total. Each channel is dedicated to managing memory access requests from one or more peripherals. Each channel has an arbiter to handle priority between DMA requests.

Data can be quickly moved by the DMA without any CPU action. This keeps CPU resources free for other operations. The DMA channels can access any memory-mapped location, including:

• AHB peripherals, for instance the CRC generator,

• AHB memories, for instance the SRAM,

• APB peripherals, for instance the USART peripheral.

The DMA controller supports two AHB-Lite ports, one is the master port used by the DMA channels to autonomously access memory-mapped locations, memory or peripherals registers, the other is a slave port providing access to the DMA controller control and status registers.

Most APB peripherals (AHB is designed for high-bandwidth, high-clock frequency interfaces, supporting features like multiple masters, burst transfers, and split transactions. ASB is a simplified version, suited for 16-bit and 32-bit systems. Both buses can be connected to lower-performance buses like APB via bridges.) can be configured to assert DMA requests. This is particularly useful for communication peripherals and converters (ADC and DAC).

Two units oversee handling DMA transfers: the DMA request multiplexer (DMAMUX) and the DMA

controller.

The DMA controller transfers data from a source address to a destination address and manages the priority between the channels.

The DMAMUX enables the user to map requests to channels. It also handles triggers and synchronizations. The DMAMUX is described in a dedicated presentation.

The DMA controller has 7 channels in total, each dedicated to managing memory access requests from many peripherals. Each channel has flexible hardware requests and support for software triggers. The channel software priority is programmable and a hardware priority is used in case of equality. Channels are independently configurable. Each channel has its own data format, increment type and data address for both source and destination.

Independent channel interrupt flags allow triggering half transfer, transfer complete, and transfer error events. In case of a transfer error, the faulty channel is automatically disabled without any impact on the other active DMA channels.