STM32CubeMX Button Debounce With Interrupt

by BurhanMuhyiddin

Hi, in this tutorial I will try to give my simple solution to prevent button bounce which is very serious issue. On internet there are many videos to offer solution to this issue, but non of them for external interrupt. In all of these videos button press is checked by polling method which is inefficient. So, let's start!

Step 1: Hardware and Software Requirements

Hardware requirements:

- STM32 ARM development board
- A computer

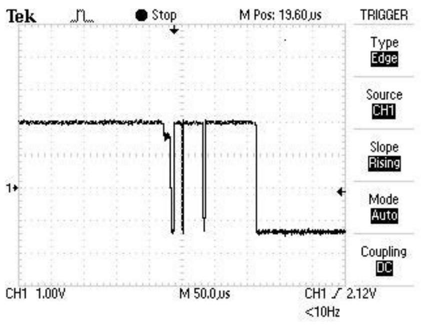
Software requirements:

- STM32CubeMX
- Keil uVision5

Step 2: Understanding the Problem

So, we try to find solution for button bouncing problem. Therefore, we have to understand the issue. So, when we press a button it should come a state which is opposite to its previous state. For example, if it was HIGH it has to be LOW and if it was LOW then it has to be HIGH. However, this is ideal state (in PROTEUS :)) In reality, when we press a button it starts to bounce between HIGH and LOW before it comes to idle state. So, pretends that it has been pressed several times which causes problems. So, what we should do?

Here I want to note that in this example, we will use external interrupt in order to detect button press. So, after we detect button press we need to wait a bit of time, like 50mS in order to reach idle state and check again whether button is idle state or not. If it's in idle state then we can continue to our task. So, let's see the code :)



Step 3: STM32CubeMX Configuration

So, we need to first enable external interrupt for our push button(I assume here that you use STM32F407VG discovery board):

- In "Pinout & Configuration" tab click on pin PA0 which is connected to the push button and choose GPIO_EXTI0 which enables external interrupt on that pin.
- Change the "user label" of the pin to "Push_Button" or anything you want.

Then, we need to configure timer in order to create 50mS time delay:

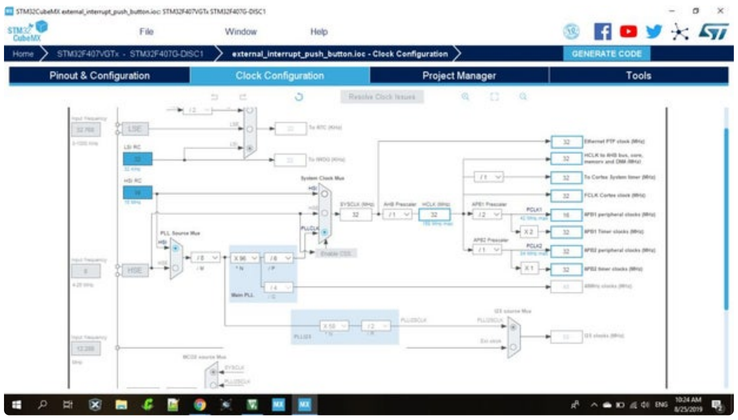
- Enter "Timers" section
- Click on TIM1
- Choose "Internal Clock" as an Clock source
- In configuration(If you want to understand this section please refer to this tutorial, highly recommended "Servo Motor Control With STM32F4 ARM MCU"):
    - Set prescaler as 32000
    - And counter period to 50

- In "NVIC Settings" tab enable all of the interrupts

Enable LED as an output:

- Click on PD12 and set as "GPIO_Output"

Then, configure clock as in image given above and generate the code.


STM32CubeMX external_interrupt_push_button.ioc: STM32F407VGTx STM32F407G-DISC1
File
Window
Help
Home
STM32F407VGTx - STM32F407G-DISC1
external_interrupt_push_button.ioc - Clock Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools

Step 4: Keil Software Development

First, we define state variable which will ensure that we don't start timer inside external interrupt when bouncing happened:

```
 /* USER CODE BEGIN PFP */
 bool state = true;
/* USER CODE END PFP */
```

Then, we write ISR for external interrupt:

```
 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
 {
 if(GPIO_Pin == Push_Button_Pin && state == true){
  HAL_TIM_Base_Start_IT(&htim1);
  state = false;
 }
 else{
  __NOP();
 }
}
```

When button pressed we check if it was our defined push button and whether the state is true. At the beginning the state will be true in order to enter the if statement. After entering we start timer and make state false in order to ensure that bouncing will not restart timer.

Then, we write ISR for timer interrupt:

```
 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
 {
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

   /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_TIM_PeriodElapsedCallback could be implemented in the user file
   */
 if(HAL_GPIO_ReadPin(Push_Button_GPIO_Port, Push_Button_Pin) == GPIO_PIN_RESET){
  HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
  state = true;
  HAL_TIM_Base_Stop_IT(&htim1);
 }
}

 /* USER CODE END 4 */
```

After 50mS we check if button yet in reset state or released, if yes then we know that button is in idle state. Then we toggle the led, make state true in order to be able to detect another button press and stop timer in order to be able to start it again.

So, this process will ensure that we prevent bouncing issue.

Step 5: Conclusion

This was code for button debounce. I want to note that this code was developed by me and I am not an expert programmer. So, there can surely be errors. If you have better solution please note it. Not forget, if you encounter any problem please write me and I will try to help.

https://www.instructabl…

Download