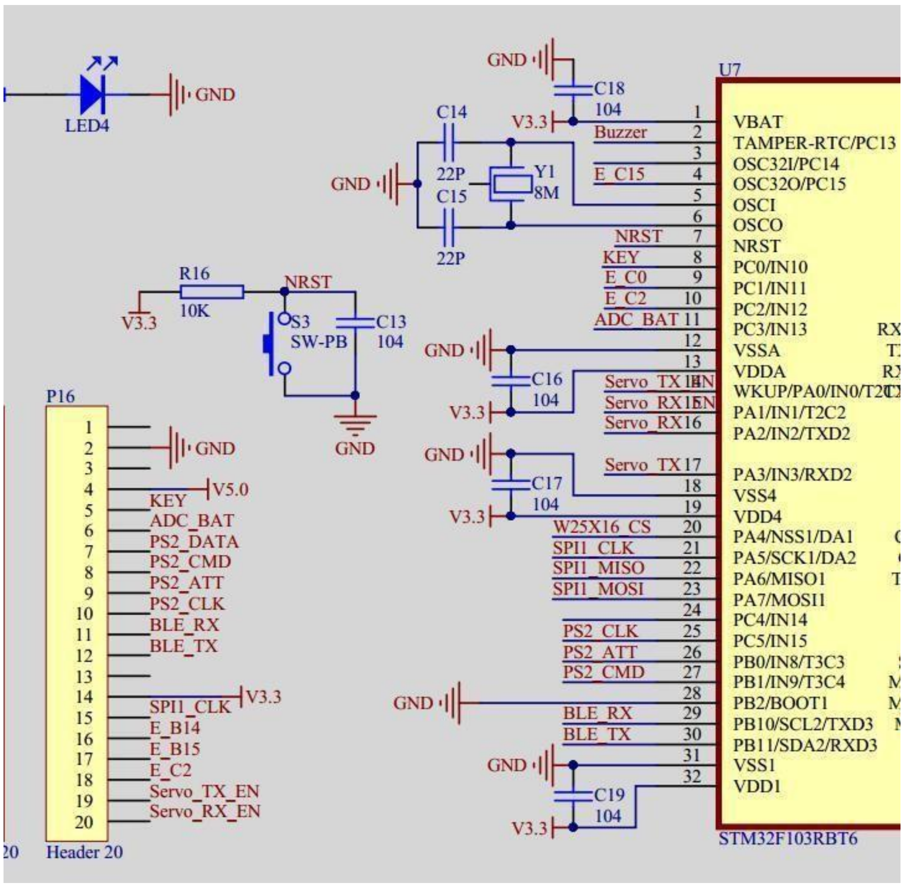
Lesson 2 Button Detection

1. Project Purpose

Use the button on controller to control the on and off of the LED light.

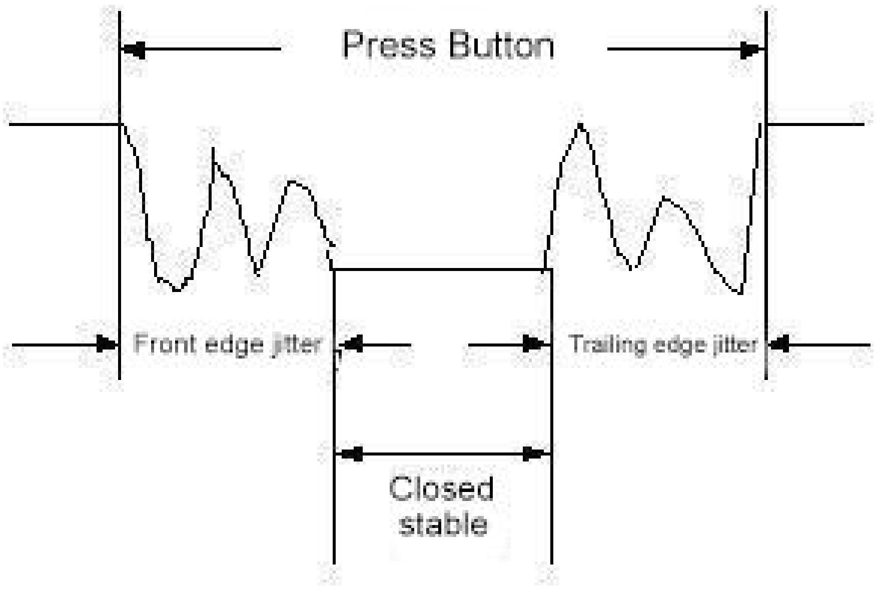
2. Project Principle

Through checking the schematic diagram of the controller, the button is connected to the PC0I/O port of the STM32 and the other end is GND. Therefore, if want to use the buttons, PC14 needs to be configured to pull-ups input mode, and then judge the status of the buttons by detecting the status of the PC14.



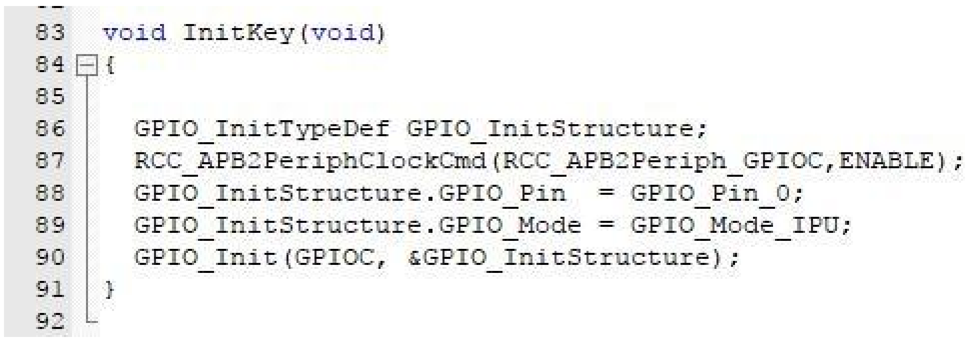
When the mechanical buttons is pressed or released, due to the effect of mechanical elasticity, it is usually accompanied by a certain period of the mechanical jitter of the contacts. The vibration time is generally 5-10ms. In addition, the connection status of the button can be detected during the contact vibration.

The mechanical jitter of the button can be eliminated by using hardware circuits or software de-jittering method. In this section, software method is used to eliminate jitter. The principle of software de-jittering is to execute a delay program about 10ms first when a button pressed is detected, and then re-detect whather the button is still pressed to confirm that the button pressed is not caused by jitter. Similarly, when a button released is detected, the method of deferring and then judging is used to eliminate the effect of jitter.



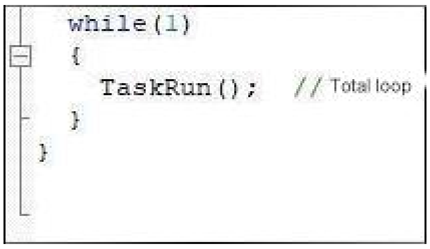
3. Program Analyst

1) In this section, buttons are added to control the LED light. Similar to controlling LED lights, initialize corresponding I/O port first.



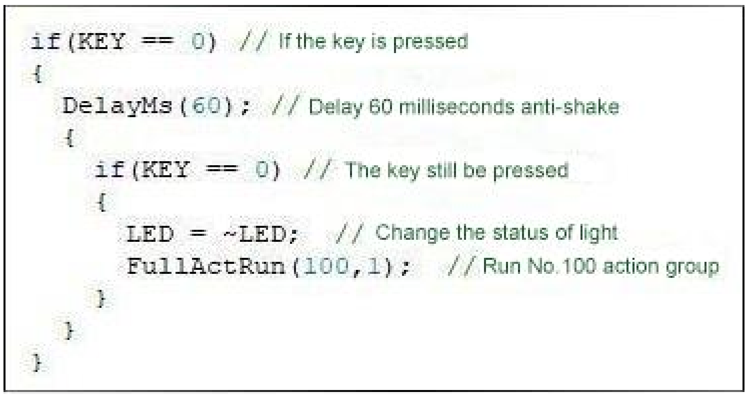
```
void InitKey(void)
{

    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
    GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

2) The biggest difference from the initialization of I/O port of LED is to configure I/O port to pull-up input GPI/O_Mode_IPU. After finishing the configuration of the I/O port, read the status of I/O port to judge whether the button is pressed. We ca see that TaskRun is executed cyclically in the main function.



```
  while(1)
  {
    TaskRun();    // Total loop
  }
}
```

3) The following code can be found in TaskRun, which is used to detect the button state as shown in the following figure:



```
if(KEY == 0)  // If the key is pressed
{
  DelayMs(60); // Delay 60 milliseconds anti-shake
  {
    if(KEY == 0)  // The key still be pressed
    {
      LED = ~LED;   // Change the status of light
      FullActRun(100,1);   // Run No.100 action group
    }
  }
}
```

4) Detect whether the I/O port where the button is located is low. If it is low,it will be pressed, and then delay 60 milliseconds. If it is still low, then confirm that the button is ot pressed while being disturbed.
5) After confirming the button is pressed, the corresponding operation can be executed. FullActRun function is called in the code, which is used to execute the action group. We will introduce it in the following chapters.