按鍵消抖常用的軟硬件方法

原創 濟南行遠智能科技有限公司 玩轉單片機與嵌入式 2022-10-16 09:28 發表於山東

收錄於合集
#經驗分享

13個

▼點擊下方名片，關注公眾號，獲取更多精彩內容▼

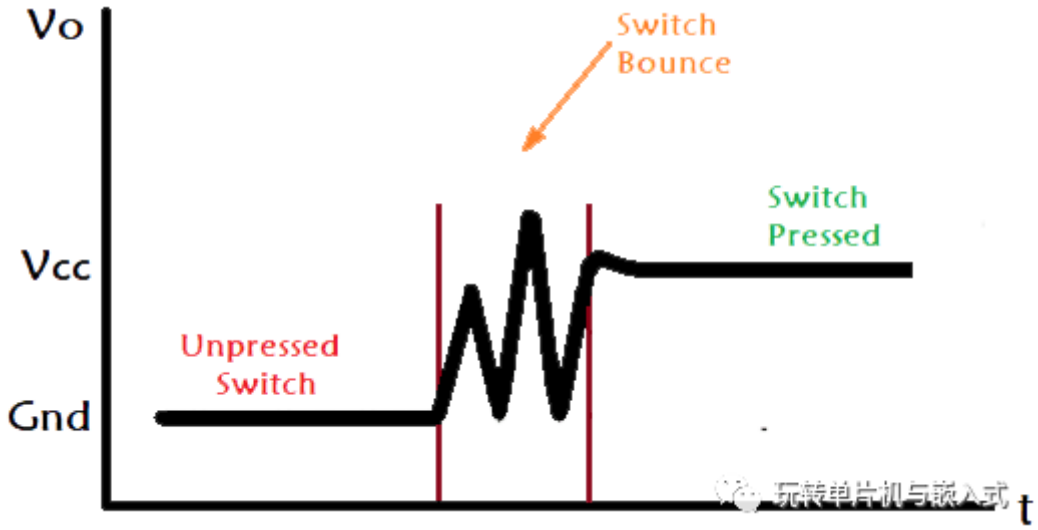
玩轉單片機與嵌入式
有乾貨，有資料，有方案，有設計……一個想要提高您技術水平的嵌入式公眾號，一起...
131篇原創內容

公眾號

歡迎關注【玩轉單片機與嵌入式】公眾號，回復關鍵字獲取更多免費視頻和資料
回复【加群】，【單片機】、【STM32 】、【硬件知識】、【硬件設計】、【經典電路】、【論文】、【畢業設計】、【3D封裝庫】、【PCB 】、【電容】、【 TVS 】、【阻抗匹配】、【資料】、【終端電阻】、【Keil 】、【485 】、【CAN 】、【振盪器】、| USBCAN |、【PCB 】、【智能手環】、【智能家居】、【智能小車】、【555 】、【I2C 】、【華為】、【中興】，等……

一：什麼是開關抖動？

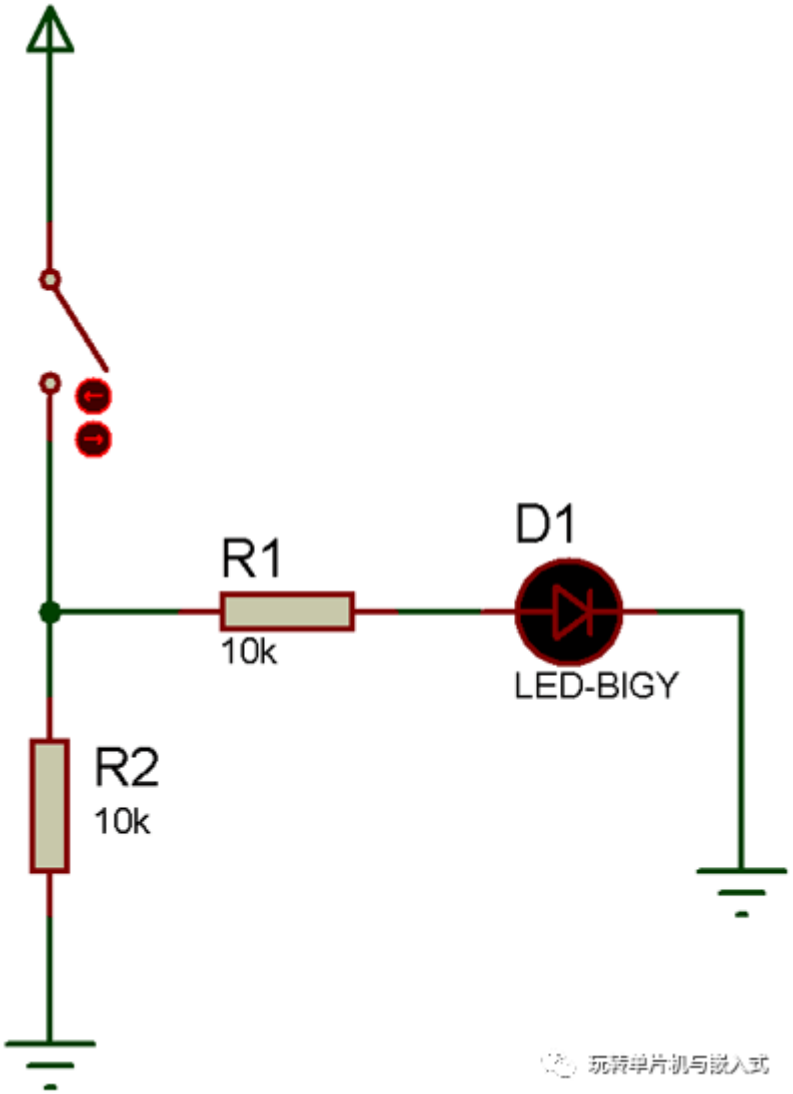
當我們按下按鈕或撥動開關或微動開關時，兩個金屬部件會接觸以短路電源。但它們不會立即連接，而是金屬部件在實際穩定連接之前連接和斷開幾次。釋放按鈕時也會發生同樣的事情。這會導致誤觸發或多次觸發，例如多次按下按鈕。這就像一個彈跳的球從高處落下，它一直在表面彈跳，直到它靜止。



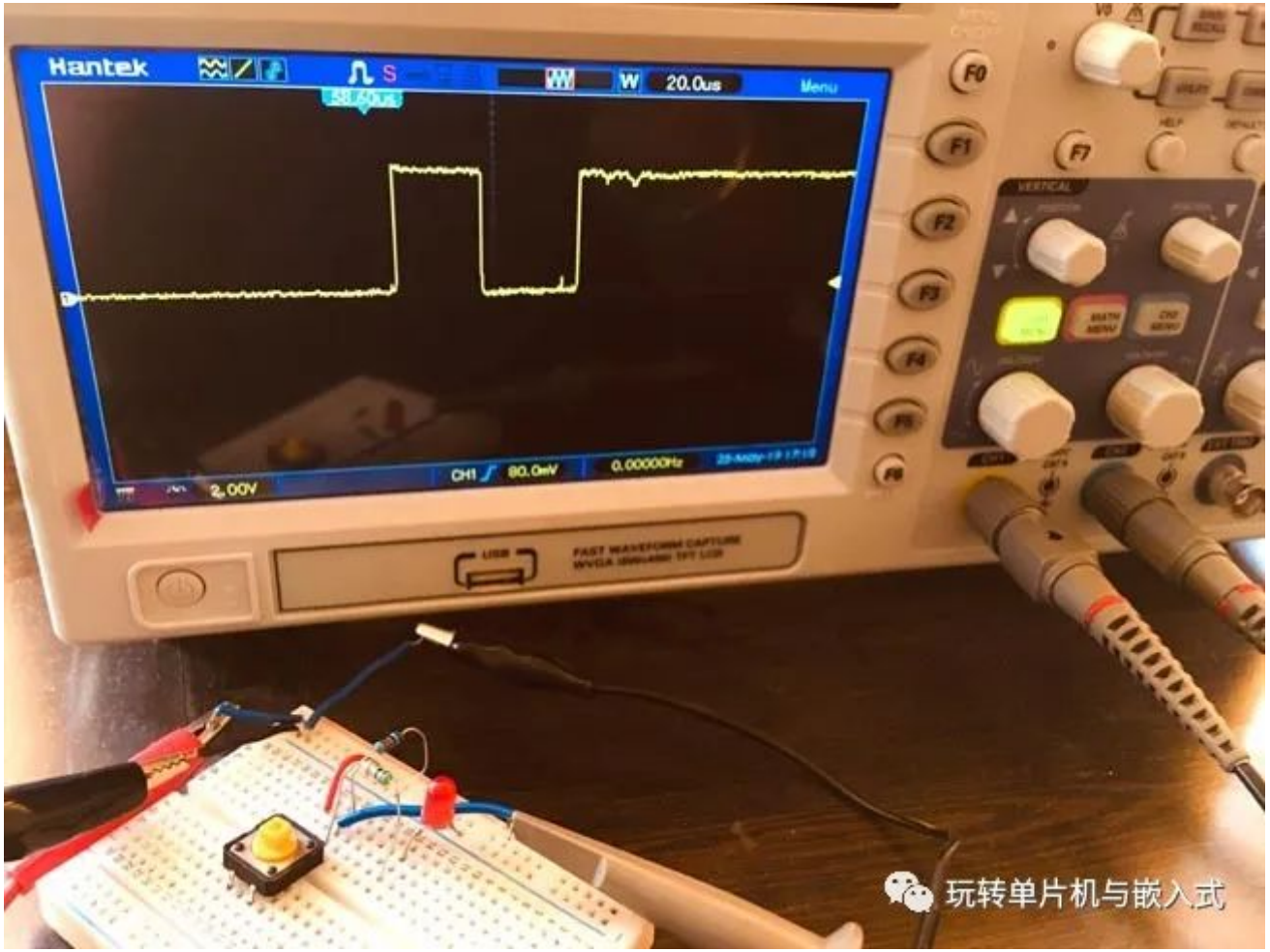
換句話說，我們可以說開關彈跳是任何開關的非理想行為，它會生成單個輸入的多個轉換。當我們處理電源電路時，開關彈跳不是主要問題，但當我們處理邏輯或數字電路時，它會引起問題。因此，為了消除電路中的抖動，使用了開關去抖動電路。

二：電路及波形

首先，我們將演示沒有開關去抖動的電路



通過示波器抓取信號的波形如下：



您還可以在按下按鈕時在示波器中看到波形。它顯示在按鈕切換期間發生了多少彈跳。

三：硬件去抖動

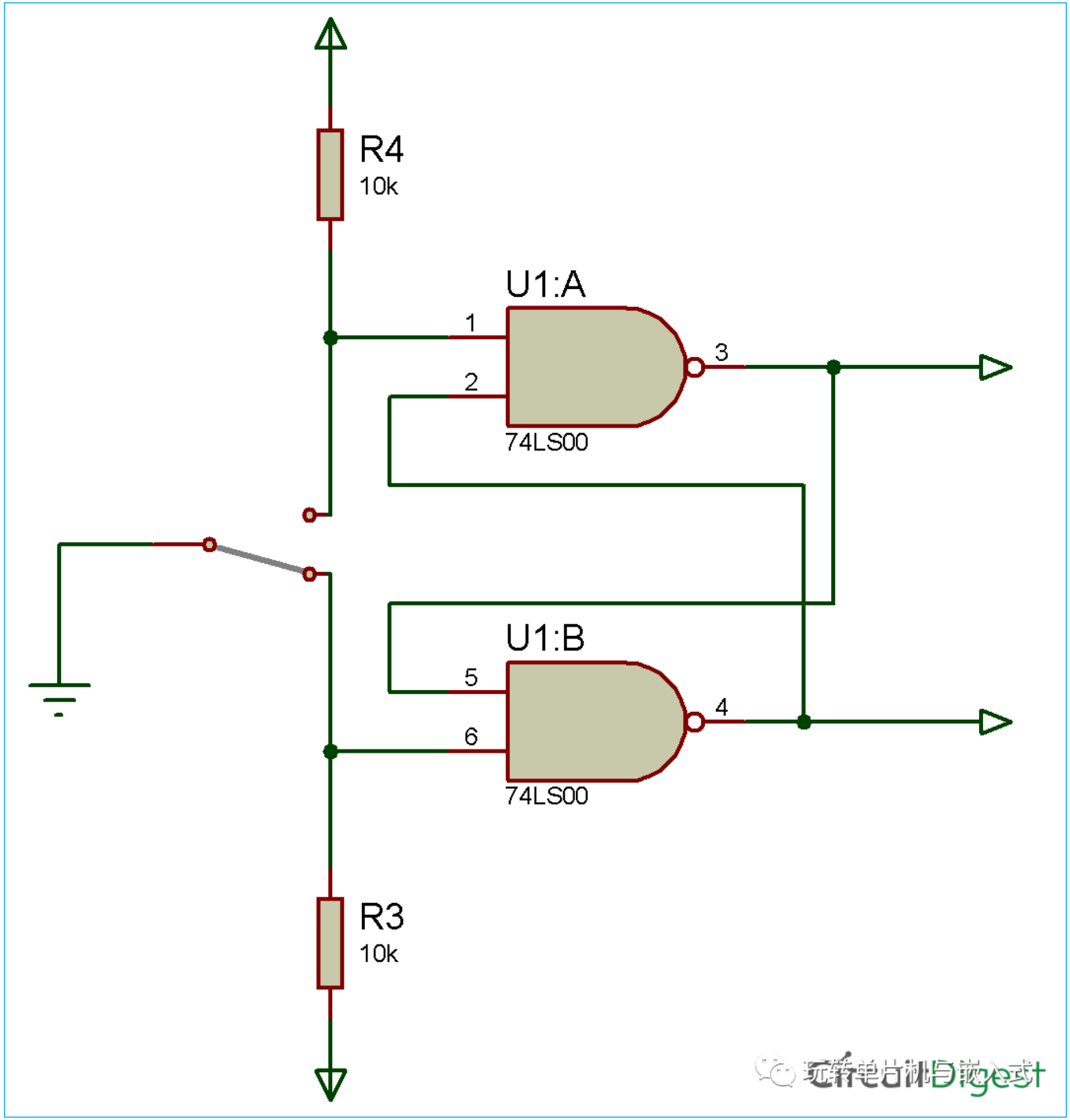
防止電路開關彈跳的常用方法有3種。

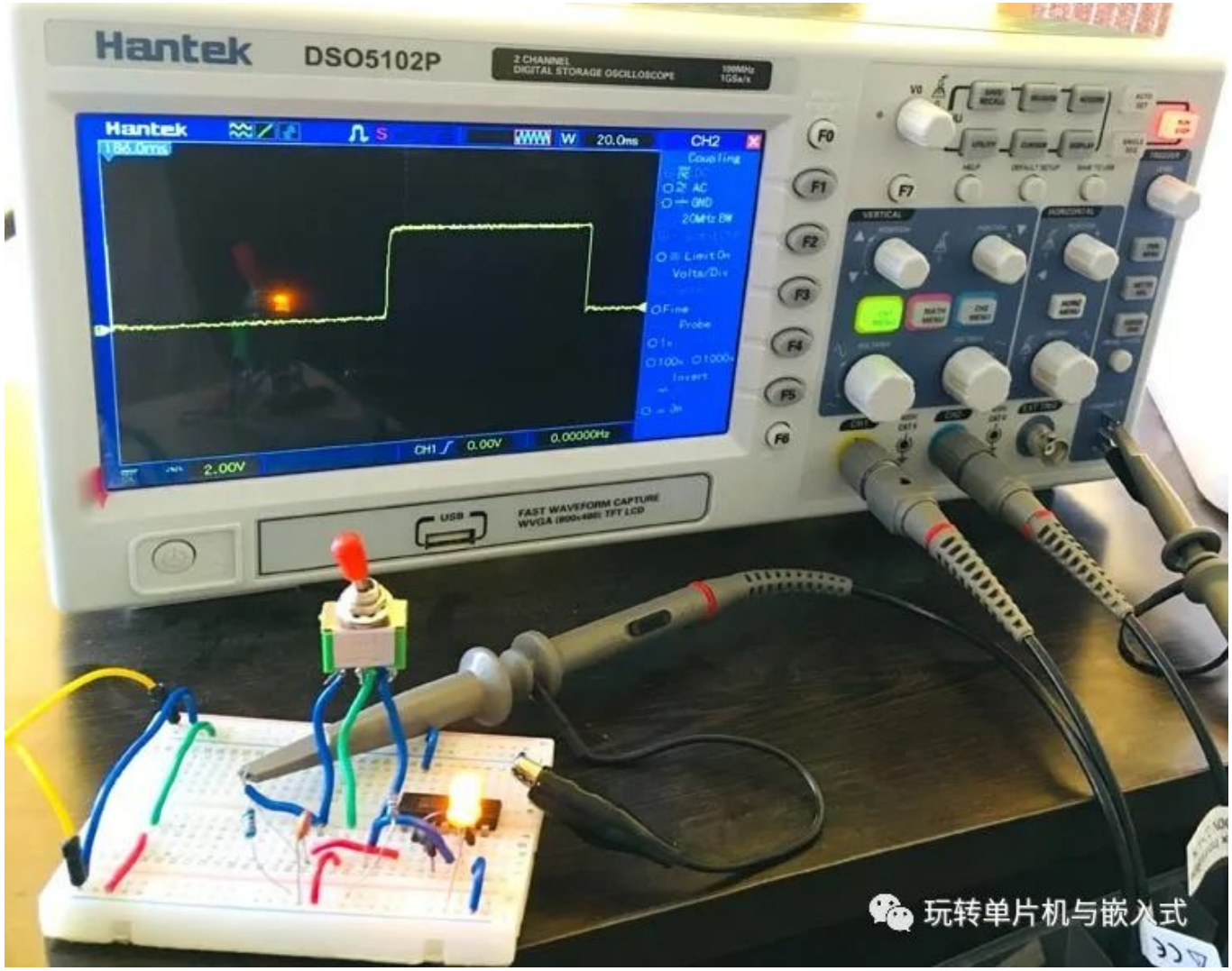
- 硬件去抖

- RC 去抖
- 開關去抖IC

01 硬件電路去抖

在硬件去抖動技術中，我們使用SR 觸發器來防止電路發生開關抖動。這是所有方法中最好的去抖動方法。


R4
10k
U1:A
1
2
3
74LS00
U1:B
5
6
4
74LS00
R3
10k

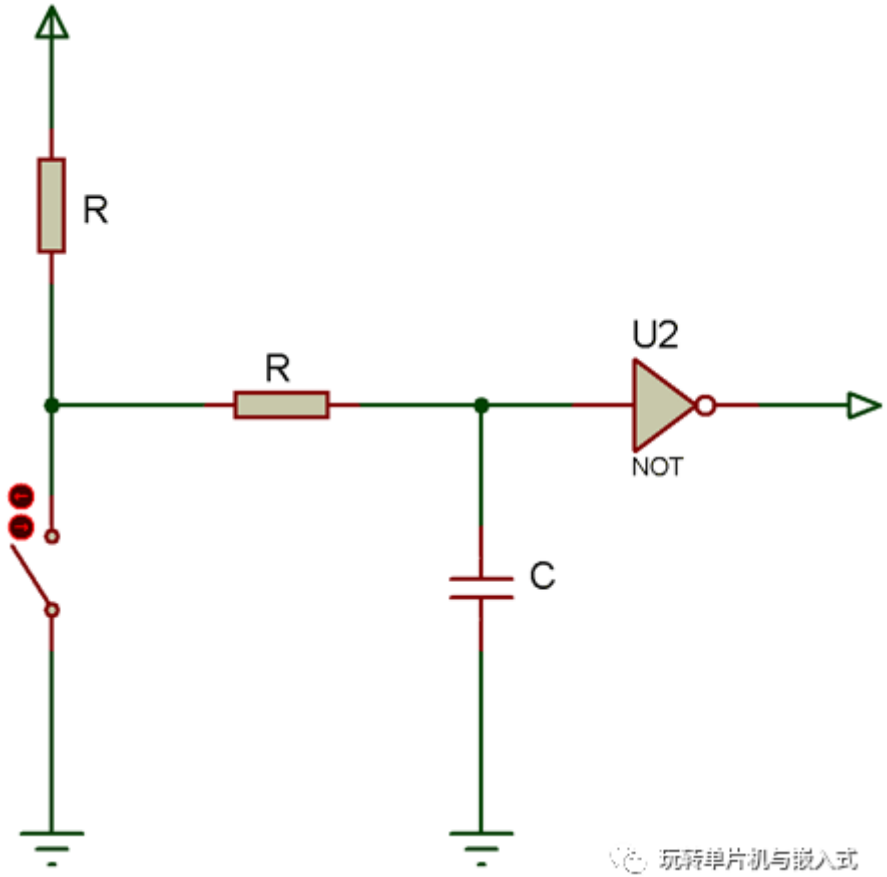


該電路由兩個與非門（74HC00 IC）組成，形成一個SR 觸發器。正如您在電路圖中看到的，只要撥動開關切換到A 側，輸出邏輯就會變為“高”。在這裡，我們使用示波器來檢測彈跳。而且，正如您在下面給出的波形中看到的那樣，邏輯正在以輕微的曲線移動而不是彈跳。電路中使用的電阻是上拉電阻。

每當開關在觸點之間移動以產生反彈時，觸發器都會保持輸出，因為“0”是從與非門的輸出反饋的。

02 RC 去抖

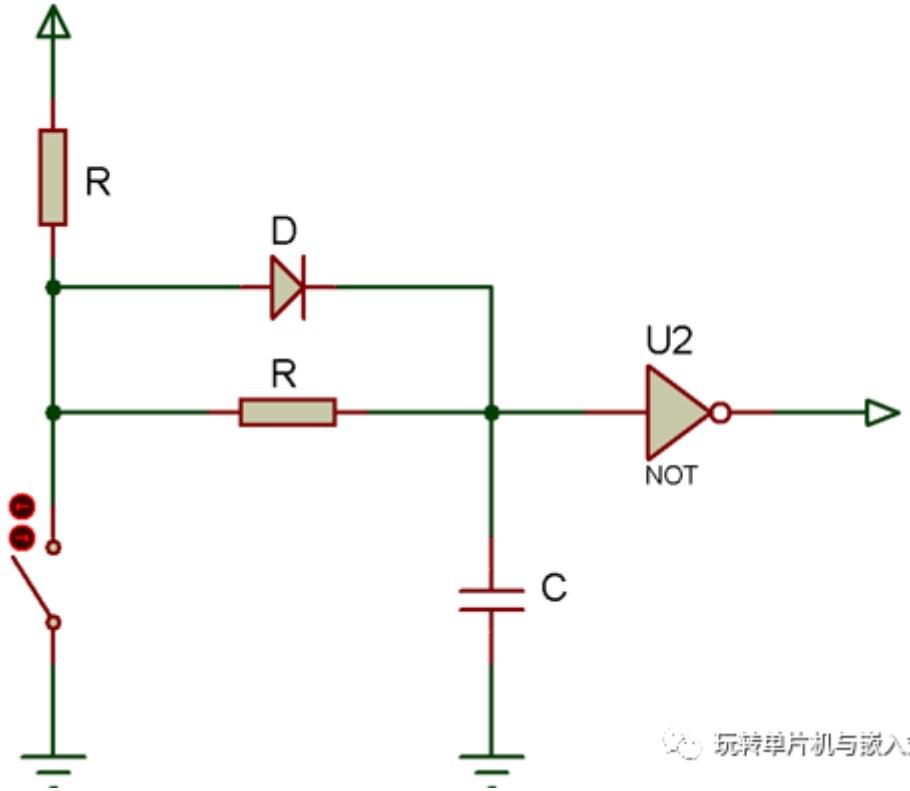
RC 僅由其名稱定義，該電路使用RC 網絡來防止開關彈跳。電路中的電容器濾除開關信號的瞬間變化。當開關處於打開狀態時，電容器兩端的電壓保持為零。最初，當開關打開時，電容器通過R1 和R2 電阻器充電。



當開關閉合時，電容器開始放電至零，因此反相施密特觸發器輸入端的電壓為零，因此輸出變為高電平。

在彈跳情況下，電容器停止Vin 處的電壓，直到它達到Vcc 或接地。

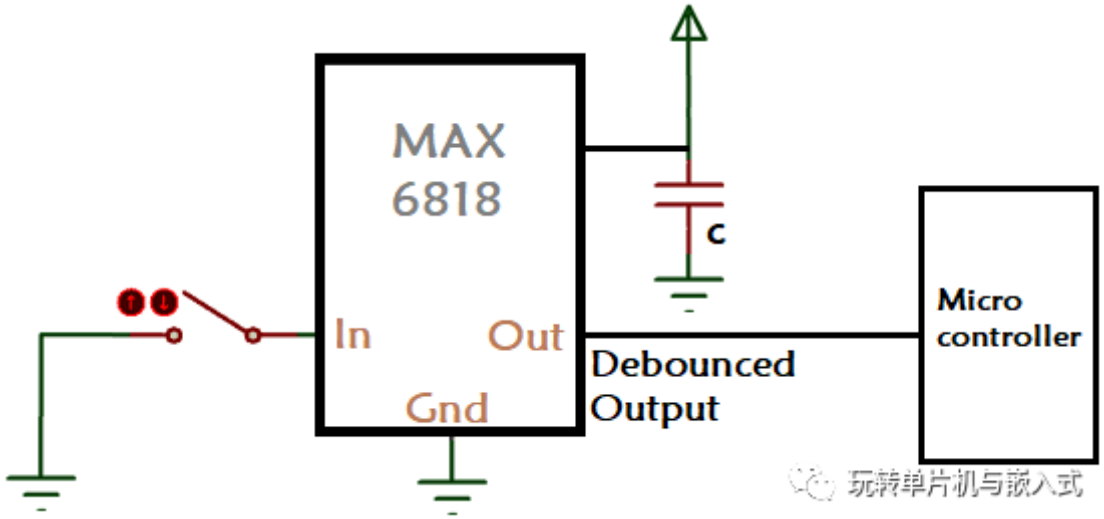
為了提高RC 去抖動的速度，我們可以連接一個二極管，如下圖所示。因此，它減少了電容器的充電時間。



03 開關去抖IC

市場上有用於開關去抖動的IC。一些去抖IC 是MAX6816、MC14490 和LS118。

下面是使用MAX6818進行開關去抖的電路圖。



所以在這裡，我們學習了按鈕如何產生開關反彈效應，以及如何通過使用硬件的方式來防止按鍵抖動。

四：軟件消抖

我們都知道，並且也是我們使用最多的場合是通過軟件實現按鍵消抖。

最簡單的方式是增加延遲以消除軟件去抖。添加延遲會強制控制器在特定時間段內停止，但在程序中添加延遲並不是一個好的選擇，因為它會暫停程序並增加處理時間。最好的方法是在代碼中使用中斷來進行軟件彈跳。

01 軟件延時

```
sbit KEY = P1^3;
///按键读取函数
uint8_t GetKey(void)
{
    if(KEY == 1)
    {
        DelayMs(20);        //延时消抖
        if(KEY == 1)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
    else
    {
        return 0;
    }
}
```

上面是最簡單的軟件延時方法，也可以通過多個按鍵組合增加相關軟件濾波的方式進行按鍵判斷，其實原理相似。

但是這種純延時的實現方式太過暴力，在延時的時候一直佔用cpu的資源，如果在延時的時候，有其他外部中斷或者搶占事件，系統完全沒有響應的。

所以我們CPU需要一個獨立的定時裝置，來完成這個計時工作，而且需要在計時時間到達時再檢測一次按鍵的電平值。

02 中斷消抖

首先初始化管腳，打開管腳的外部中斷:

```
/*Configure GPIO pins : KEY_1_Pin KEY_2_Pin */
  GPIO_InitStruct.Pin = KEY_1_Pin|KEY_2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 5, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
```

初始化TIM1，打開其update中斷:

```

static void MX_TIM1_Init(void)
{
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 7200 - 1;                // 72000000 / 7200 = 10000
```

```
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 200 - 1;                    // 200 * 0.01 = 20ms
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
```

```
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
{
    if(htim_base->Instance==TIM1)
    {
   /* Peripheral clock enable */
        __HAL_RCC_TIM1_CLK_ENABLE();
  /* USER CODE BEGIN TIM1_MspInit 1 */
        HAL_NVIC_SetPriority(TIM1_UP_IRQn,1,3);
        HAL_NVIC_EnableIRQ(TIM1_UP_IRQn);
    }
}
```

在stm32f1xx_hal_it.c中去註冊中斷回調函數(關鍵的步驟，需要在按鍵中斷處理函數中打開定時器，開始計時):

```
void EXTI15_10_IRQHandler(void)            // 按键的中断处理函数
{

  HAL_TIM_Base_Start_IT(&htim1);    //   开启定时器1，开始计时

  printf("key down\r\n");

  __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_11);
  __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_12);
}

```

定時器的中斷處理函數:

```
void TIM1_UP_IRQHandler(void)
{

  HAL_TIM_IRQHandler(&htim1);   //这个是所有定时器处理回调的入口，在这个函数里对应
  printf("TIM IRQ\r\n");

}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)        // 定时器u
{
   /* USER CODE BEGIN Callback 0 */
```

```
11
12    /* USER CODE END Callback 0 */
13    if (htim->Instance == TIM2) {
14      HAL_IncTick();
15    }
16
17    if (htim->Instance == TIM1) {            // 在这里选择tim1
18
19      printf("TIM1 updata\r\n");
20
21     HAL_TIM_Base_Stop_IT(&htim1);       //     关闭tim1 及清除中断
22
23      if (GPIO_PIN_RESET == HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_11) )    //再次判断
24      {
25       printf("KEY1 be pressed!!!\r\n");
26      }
27
28     if (GPIO_PIN_RESET == HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_12) )//再次判断管脚的
29     {
30       printf("KEY2 be pressed!!!\r\n");
31     }
32    }
33    /* USER CODE BEGIN Callback 1 */
34
35    /* USER CODE END Callback 1 */
36  }
```

總結一下,實現用定時器中斷來完成按鍵延時去抖的關鍵步驟:

1. 初始化GPIO腳，初始化TIM ， 算好時間，填入分頻值。

2. 打開GPIO中斷，在中斷處理函數中打開定時器，讓其計數。

3. 定時器溢出中斷函數中，再次判斷按鍵電平值。關閉定時器，清除pending。

歡迎關注我的公眾號，回复【加群】或掃碼加我好友，限時免費進入技術交流群，也可免費加入我的知識星球。

☆推薦閱讀☆

- 【專輯】器件選型
- 【專輯】經驗分享
- 【專輯】硬件設計
- 【專輯】開源項目
- 【專輯】單片機
- 【專輯】STM32
- 【專輯】軟件設計
- 【專輯】職業發展

感謝大家閱讀，如果喜歡
請點贊和“在看”吧，或者分享到朋友圈。
點擊跳轉到原文，限時優惠加入我們的知識星球（加好友獲取免費券）

收錄於合集#經驗分享 13

下一篇 · 【經驗】學習電子及產品開發的10個簡單的步驟

喜歡此內容的人還喜歡

PLC最全編程算法，資深電氣工程師總結，建議收藏備用!

機器人學習教程

自舉電路工作原理和自舉電阻和電容的選取

英飛凌工業半導體

開關電源芯片內部電路搞不懂？那還怎麼玩轉芯片？這份解析安排起來！

電巢射頻