STM32 infrared decoding (NEC)

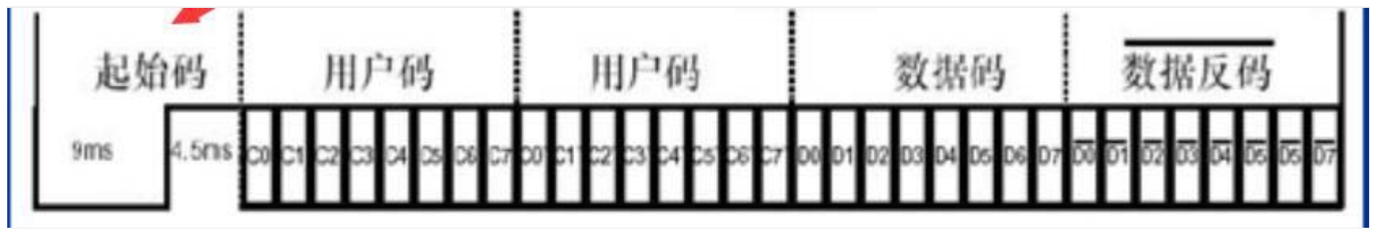
tags: MCU stm32

Article Directory

This experiment is implemented with STM32F4

Infrared remote control cannot partition walls and has strong anti-interference.

If you are not interested in the process, you can directly look at the code

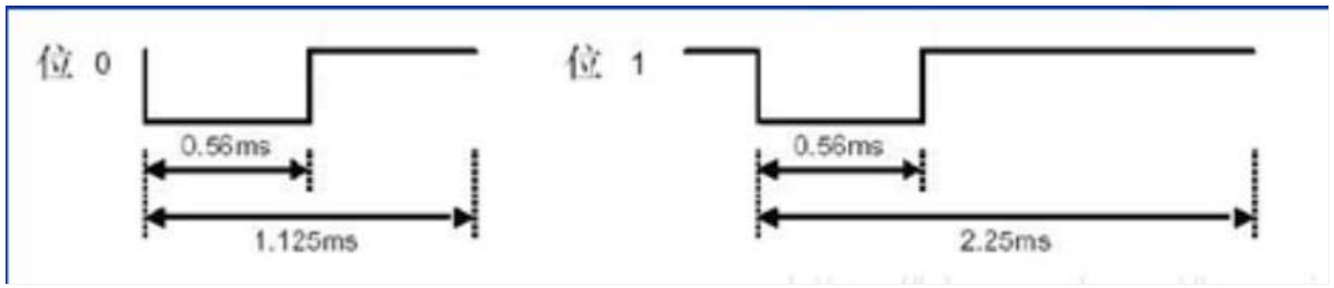


NEC code timing

is roughly received the pilot code (9ms low level + 4.5ms high level) + address code + inverted code (for verification) + data + data inverted code, at this time it has been received When the data is complete, if you don't let go, there will be a 9ms low level + 2.5ms high level + 0.56ms low level + 97ms high level. If it is still pressed, the same will happen. The timing of one cycle occurs. Use this to judge how many times to press

Be careful! This is the timing of the sender, the timing of the receiver is reversed, that is, he sends a high level and we receive a low level! And we only need to measure the high level.

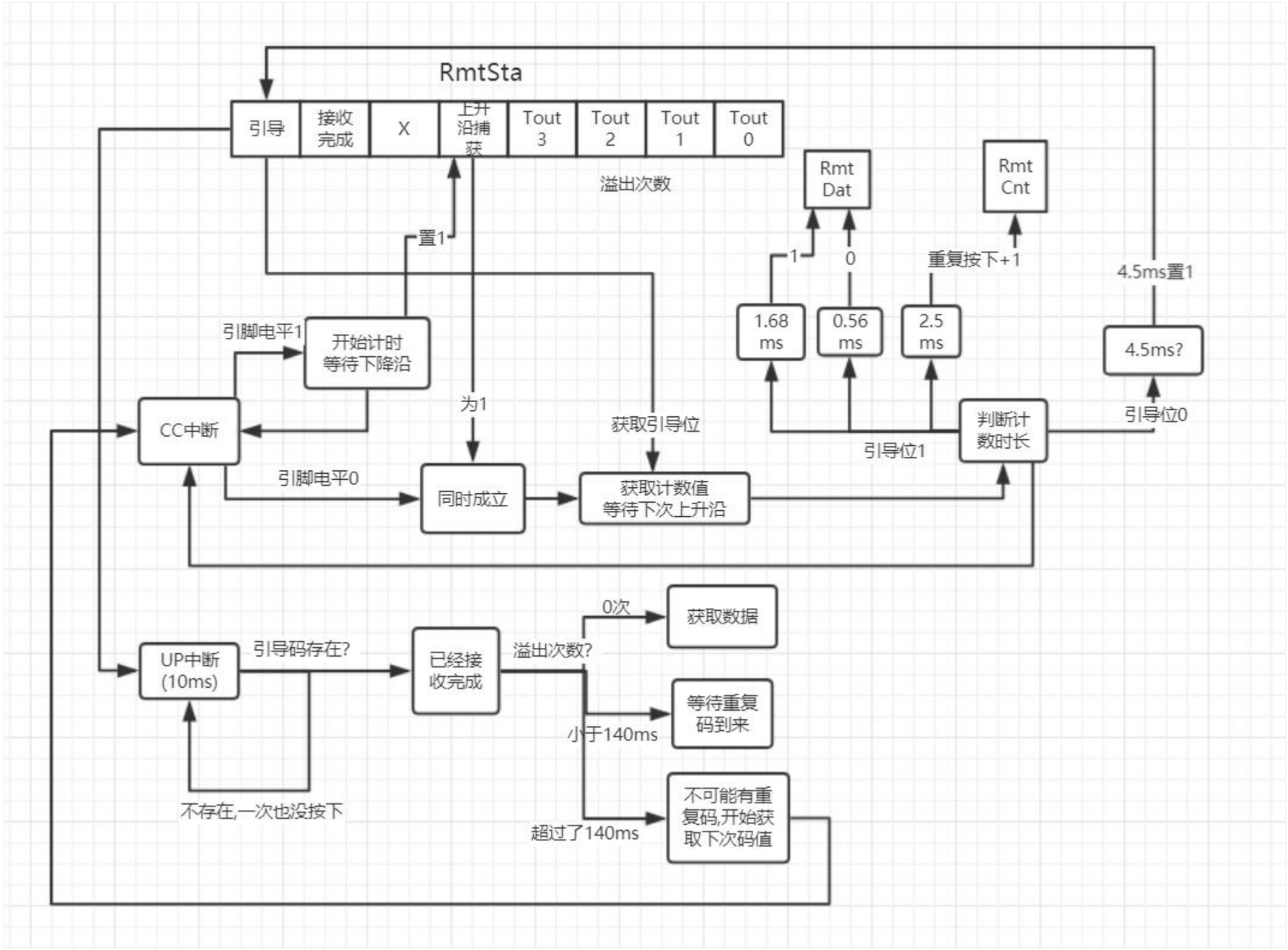
Whether a code is 0 or 1, it is expressed by a carrier level + different signal levels, as shown in the figure above



For example, bit 0 is composed of 0.56msLow+0.56msHigh, for 1.125ms, bit 1 is 0.56ms L+1.68ms H This expresses a signal of one bit, remember! The received signal is reverse, and then the signal can be obtained by measuring the high-level pulse width.

Here STM32 uses input capture to measure pulse width (single timer input mode), and there is also a PWM dual input mode.

First, let's express the program realization more vividly through the flowchart (refer to the punctual atom)!



Let's talk about it roughly, define a status byte to access the current decoding status. By capturing the CC interrupt and the UP overflow interrupt to operate the status bit, in fact, the CC interrupt is used to receive data, and the UP interrupt is used to determine whether the repeated pressing time has expired and whether the data has been received is completed. After entering the CC interrupt, judge the pulse width time, is it 0? it's 1? Is it a pilot code or a repeat code

Catch interrupt

```
1                                   //<Read high level time to get data>
```

```
                                                //[7]: received the boot code flag
                                              //[6]: got all the information of a button
                                                        //[5]: reserved
          //[4]: Mark whether the rising edge has been captured
                                                     //[3:0]: overflow timer
                                             u8 RmtSta = 0;//Remote control status bit
                                              u16 CntVal;     //Counter value 0-10000
                    u32 RmtDat = 0; //Received data address code-address inversion-data code-digital inversion
                                            u8 RmtCnt = 0;  //Number of repeated presses
```

External variables are defined as above

```
                                                #define REMOTE_IC_TIMx TIM1
                                                  #define RMT_IN PAin(8)
                                               //Capture interrupt-detect key value
                                                  void TIMx_CC_IRQHandler(void)
                                                                {
                                               if(TIM_GetITStatus(REMOTE_IC_TIMx,TIM_IT_CC1))
                                                                  {
                                                       if(RMT_IN)      //At this time at high level
                                                                      {
                                             REMOTE_IC_TIMx->CCER |= (1<<1);         //Wait for the falling edge
                                 REMOTE_IC_TIMx->CNT = 0;                        //The counter is cleared to 0 to start counting-1us once
                          RmtSta |= (1<<4);                                    //The rising edge has been captured, start to capture the falling edge
                                                                      }
                                                            else    //At this time at low level
                                                                      {
                                              CntVal = REMOTE_IC_TIMx->CCR1;          //Read capture register-get high level time
                                                        REMOTE_IC_TIMx->CCER &= (~(1<<1));      //Turn to rising edge
                                             if(RmtSta & (1<<4))                                    //There has been a rising edge
                                                                        {
                                                        if(RmtSta & (1<<7))                             //Guide code received
                                                                       {
                                                                    //Judgment of three codes
                                                    if((CntVal > 300)&&(CntVal < 800))              //560us threshold --0
                                                                        {
                                                                      RmtDat <<= 1;
                                                                        }
                                                  else if((CntVal > 1400)&&(CntVal < 1800))//1680us threshold --1
                                                                        {
                                                                      RmtDat <<= 1;
                                                                        RmtDat++;
                                                                        }
                                                else if((CntVal > 2200)&&(CntVal < 2600))//2500us threshold - press repeatedly
                                                                        {
                                                                        RmtCnt++;
```

```
                RmtSta &= 0xF0;//Empty the overflow times and wait for the next press
                }
            }
else if((CntVal > 4200) && (CntVal < 4700))     //4.5ms boot code, start to receive data
            {
            RmtSta |= (1<<7);
        RmtCnt = 0;//The number of repetitions is cleared
            }
        }
    RmtSta &= (~(1<<4));//Clear the rising edge flag
    }
}
TIM_ClearITPendingBit(REMOTE_IC_TIMx,TIM_IT_CC1);
}
```

Only when the pilot code is received can it be judged whether 0 is 1 or a repeated code. Since the capture interrupt is in single input mode, it is impossible to determine whether it is a rising edge or a falling edge. But if the current pin is high, then it must be a rising edge that was received just now. After receiving the pilot code, determine the counter threshold (counter+1 = 1us), typically 1.68 is 1, 0.56 is 0, and 2.5 is repeated pressing. The data is stored in RmtDat (32bit).

Overflow interrupt

```
//Update interrupt--detect repeated press
    void TIMx_UP_IRQHandler(void)
    {
    if(TIM_GetITStatus(REMOTE_IC_TIMx,TIM_IT_Update))
        {
  if(RmtSta & (1<<7))     //Under the premise of receiving the guide code, check whether to press repeatedly
        {
      //After more than 10ms, it may enter the repeated press event, stop checking the key time,
       //Wait for the next rising edge to calculate whether it is a pilot code or a repeat code
            RmtSta &= (~(1<<4));
if((RmtSta & 0x0F )== 0x00)     RmtSta |= (1<<6);//If it never overflowed before, it means that the data code is obtained
          if((RmtSta & 0x0F) < 14) //Less than 14*10ms, indicating that there may be repeated codes coming
            {
            RmtSta++;
            }
        else //Timed out, there is no possibility of duplicate codes coming
            {
        RmtSta &= 0b01110000;//Clear the boot bit and overflow value, and re-receive
            }
        }
    }
    TIM_ClearITPendingBit(REMOTE_IC_TIMx,TIM_IT_Update);
    }
```

The overflow is 10ms once, so if the infrared remote control is pressed, it is impossible to overflow! Because there is a pulse when it is pressed, it will be cleared when there is a pulse, and the maximum count is 4.5ms<10ms, so if it reaches the overflow interrupt, it means that the reception is completed or it is not pressed at all or enters the repeat mode.
To save space, each sentence above has a very detailed explanation.

Verify data for output

```
//Scan the remote control press
//return key value
u8 Remote_Scan(void)
{
u8 sta = 0;
u8 tmp1 = 0,tmp2 = 0;

if(RmtSta & (1<<6))     //Receive the complete key value, start to verify the data
{
tmp1 = RmtDat >> 24;    //address code
tmp2 = RmtDat >> 16;    //Address inversion
if((tmp1 == (u8)~tmp2) && (tmp1 == REMOTE_ID))
{
tmp1 = RmtDat >> 8;
tmp2 = RmtDat;
if(tmp1 == (u8)~tmp2) sta = tmp1; //Return key value
}
if((sta == 0) || ((RmtSta & 0x80) == 0)) RmtSta &= ~(1<<6),RmtCnt=0;//At this point it has been released
}
return sta;
}
```

The verification is very simple, that is, address code = inverted address code, data code = inverted data code. In particular, remember to strengthen the type conversion before negating the sign.
For the address code, almost every remote control is fixed, but the key code (data code) is not fixed, from 0 to 255.

Some other codes

```
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
TIM_TimeBaseInitStructure.TIM_Period = (10000-1);
TIM_TimeBaseInitStructure.TIM_Prescaler = (168-1);      //1us
TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(REMOTE_IC_TIMx,&TIM_TimeBaseInitStructure);
TIM_ICInitTypeDef TIM_ICInitStructure;
//Input capture
TIM_ICInitStructure.TIM_Channel = TIM_Channel_1;
TIM_ICInitStructure.TIM_ICFilter = 3;
```

```
    TIM_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising;
    TIM_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV1;
TIM_ICInitStructure.TIM_ICSelection = TIM_ICSelection_DirectTI;
```

Share this part of the code. GPIO is multiplexed into timer x. Then enable CC1 (pay attention to the pin) and UP interrupt. The interrupt vector is the priority of CC>UP
My test code:

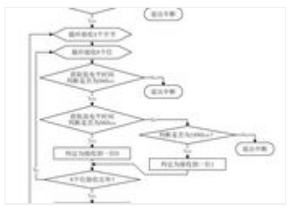
```
                p1:
  RemoteData = Remote_Scan();
       if(RemoteData)
                {
     if(last == RemoteData) goto p1;
printf("\r\nRemoteData:%d",RemoteData);
      printf("\r\nCNT:%d",RmtCnt);
           last = RemoteData;
                }
```

Infrared decoding felt nothing when I looked at the principle at first, and it was quite frustrating to do--!! But it was not difficult.

Intelligent Recommendation

NEC infrared remote control protocol

There are two common encoding formats for remote controllers, one is NEC format and the other is RC5 format. The signal from the remote control passes through an infrared receiver and the signal is se...



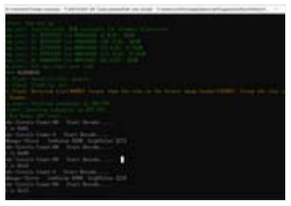
NEC infrared remote control protocol

Remote control home appliances are often less demanding communication distance, and the infrared cost lower than other wireless devices, therefore the infrared remote control home appliances applicati...

STM32HAL----Infrared Remote Control (NEC)

An infrared program of NEC agreement, NEC agreement, physical environment is F103 atomic battleship V3, and STM32CubeMX is used to generate the initial program. Timer

configuration72 frequency divisio...



ESP32 Development Notes (3) Source code example 12_IR_Rev_RMT Use RMT to realize infrared remote control receiving and decoding (NEC encoding)

Development board purchase link https://item.taobao.com/item.htm?spm=a2oq0.12575281.0.0.50111deb2Ij1As&ft=t&id=626366733674 Introduction to Development Board Development environment build wind...

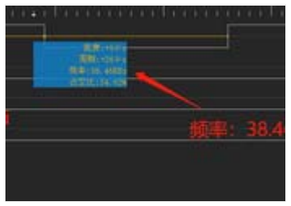
About infrared remote control protocol (NEC format)

We are in contact with the ordinary operation of the remote control NEC format, here NEC talk format. When a key is pressed the remote control, will send the data of one frame, the frame data is compo...

More Recommendation

N76E003 infrared coding program and NEC protocol analysis

N76E003 infrared coding program and circuit (38kHZ, NEC agreement) N76E003 is an enhanced 8-bit 8051 MCU with flash, the instruction set is fully compatible with the standard 80C51, and built-in 16M c...

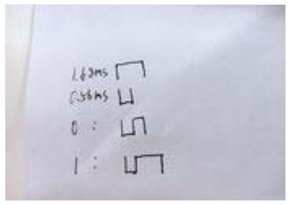


Detailed IR NEC infrared driver debugging data

Detailed IR NEC infrared driver debugging data: TX debugging attention: 1). During data processing, LSB processing is required, that is, the low bit is first, and it can be shifted to the right, but l...

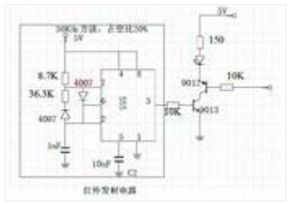
Speed up the transmission rate of infrared nec protocol

Traditional NEC protocol The traditional nec protocol generally takes about 100ms to complete the transmission of a frame, with a 13.5ms frame header, plus a 32-bit 0 or 1. A 0 is a low level of 560us...



Nec code infrared remote control principle finishing

Remote control: NEC code 960nm wavelength crystal oscillator is 455KHZ, the corresponding transmission frequency (carrier frequency) is 38KHZ, The remote control ID is 0 (that is, the system identific...



NEC protocol-the use of infrared remote control

The NEC protocol is one of many infrared remote control protocols, and the infrared decoding is realized by the Lanqiao Cup MCU development board. Introduction of related chips and components This pic...