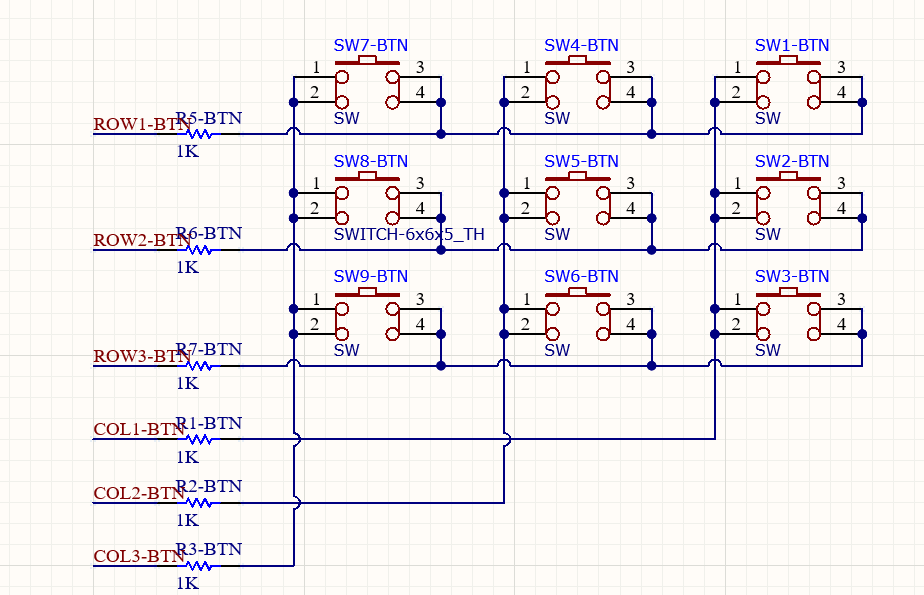
# 按钮矩阵模块

## 1.模块介绍

本键盘由9个按键开关构成，原理图如下：



这9个按键开关采用行列式布局，可以发现9个按键开关只用了6个I/O口就实现了控制，有效地提高I/O口利用率。驱动原理也很简单，例如我们将3个行引脚ROW1、ROW2、ROW3都配置为上拉输入，将3个列引脚COL1、COL2、COL3都配置为低电平推挽输出，然后扫描监听行引脚状态。当任意按键摁下时，行引脚就变为低电平，这样就知道哪一行的按键触发，然后翻转行列引脚状态，3个行引脚配置为低电平推挽输出，3个列引脚配置为上拉输入，接着扫描监听列状态，哪一列引脚变成低电平，就知道那一列的按键触发，两次结果相结合，就知道哪一行哪一列的按键触发，从而确定具体按键。

**2.程序实现**

本例程使用的MCU为沁恒的CH32V307VCT6,频率72MHz.

首先需要先初始化GPIO,本例程默认先监听行引脚，所以3个行引脚配置为上拉输入，3个列引脚配置为低电平推挽输出。

    //GPIO初始化

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    EXTI\_InitTypeDef EXTI\_InitStructure = {0};    //开启时钟

    //所有行配置为上拉输入

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|KEY\_ROW\_GPIO\_CLK, ENABLE);

    //GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_Disable,ENABLE);//关闭swd引脚

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    //所有列配置为低电平推挽输出

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|KEY\_COL\_GPIO\_CLK, ENABLE);

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_COL\_PORT, KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN);

当然也可以先监听列引脚，原理是相同的。接着为3个行引脚配置外部中断，中断配置为下降沿中断。

//行因引脚中断初始化,按键按下进入对应中断

    GPIO\_EXTILineConfig(KEY\_ROW\_PORTSOURCE , KEY\_ROW1\_PINSOURCE|KEY\_ROW2\_PINSOURCE|KEY\_ROW3\_PINSOURCE);

    EXTI\_InitStructure.EXTI\_Line = KEY\_ROW1\_EXTILINE|KEY\_ROW2\_EXTILINE|KEY\_ROW3\_EXTILINE; // 外部中断线2

    EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; // 配置为外部中断

    EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;

    EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; // 使能外部中断

    EXTI\_Init(&EXTI\_InitStructure);

    //按键矩阵

    NVIC\_InitTypeDef NVIC\_InitStructure = {0};

    NVIC\_InitStructure.NVIC\_IRQChannel = KEYBOARD\_EXTI\_IRQn ;

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

    NVIC\_Init(&NVIC\_InitStructure);

引脚初始化完后，还需要行列引脚状态的翻转函数，因为一开始行引脚上拉输入，列引脚下拉输出，摁键摁下后，进入行引脚中断，中断中需要把行引脚改为下拉输出，列引脚改为上拉输入，然后依次查看列引脚状态，判断触发的按键是哪个。退出中断前，还需要将引脚状态改回行引脚上拉输入，列引脚下拉输出。

//检测到哪一行后，进入列扫描模式，行引脚设置为低电平推挽输出模式，列引脚设置为上拉输入模式

void Keyboart\_ScanCol\_Mode(void)

{

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    //所有行配置为低电平推挽输出模式

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_ROW\_PORT, KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN);

    //所有列配置为上拉输入模式

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

}

void Keyboart\_ScanRow\_Mode(void)

{

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    //所有列配置为低电平推挽输出

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_COL\_PORT, KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN);

}

接着写行引脚的外部中断函数

void KEYBOARD\_EXTI\_IRQHandler(void)

{

    //行1中断触发

    if(EXTI\_GetITStatus(KEY\_ROW1\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW1\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW1\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("11\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("12\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("13\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW1\_EXTILINE);

        }

    }

    //行2中断触发

    else if(EXTI\_GetITStatus(KEY\_ROW2\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW2\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW2\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("21\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("22\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("23\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW2\_EXTILINE);

        }

    }

    //行3中断触发

    else if(EXTI\_GetITStatus(KEY\_ROW3\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW3\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW3\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("31\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("32\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("33\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW3\_EXTILINE);

        }

    }

}

需要注意的是Delay\_Ms(50)的作用是为按键提供软件滤波，但一般情况下并不推荐在中断中使用Delay\_Ms()延时，因为主函数或其余地方正在调用延时函数，可能导致systick定时器冲突，有需要可改为使用其他定时器来实现软件滤波。

完整代码如下所示：

**Keyboard.h**

#ifndef \_\_KEYBOARD\_H

#define \_\_KEYBOARD\_H

#include "debug.h"

#define KEY\_ROW\_PORT        GPIOA

#define KEY\_ROW1\_PIN        GPIO\_Pin\_13

#define KEY\_ROW2\_PIN        GPIO\_Pin\_14

#define KEY\_ROW3\_PIN        GPIO\_Pin\_15

#define KEY\_ROW\_GPIO\_CLK    RCC\_APB2Periph\_GPIOA

#define KEY\_ROW\_PORTSOURCE       GPIO\_PortSourceGPIOA

#define KEY\_ROW1\_PINSOURCE       GPIO\_PinSource13

#define KEY\_ROW1\_EXTILINE         EXTI\_Line13

#define KEY\_ROW2\_PINSOURCE       GPIO\_PinSource14

#define KEY\_ROW2\_EXTILINE         EXTI\_Line14

#define KEY\_ROW3\_PINSOURCE       GPIO\_PinSource15

#define KEY\_ROW3\_EXTILINE         EXTI\_Line15

#define KEY\_COL\_PORT        GPIOE

#define KEY\_COL1\_PIN        GPIO\_Pin\_3

#define KEY\_COL2\_PIN        GPIO\_Pin\_4

#define KEY\_COL3\_PIN        GPIO\_Pin\_5

#define KEY\_COL\_GPIO\_CLK    RCC\_APB2Periph\_GPIOE

#define KEYBOARD\_EXTI\_IRQn            EXTI15\_10\_IRQn

#define KEYBOARD\_EXTI\_IRQHandler      EXTI15\_10\_IRQHandler

void Keyboard\_Init(void);

void Keyboart\_ScanCol\_Mode(void);

void Keyboart\_ScanRow\_Mode(void);

#endif

**Keyboard.c**

#include "keyboard.h"

void KEYBOARD\_EXTI\_IRQHandler(void) \_\_attribute\_\_((interrupt("WCH-Interrupt-fast")));

void Keyboard\_Init(void)

{

    //GPIO初始化

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    EXTI\_InitTypeDef EXTI\_InitStructure = {0};    //开启时钟

    //所有行配置为上拉输入

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|KEY\_ROW\_GPIO\_CLK, ENABLE);

    //GPIO\_PinRemapConfig(GPIO\_Remap\_SWJ\_Disable,ENABLE);//关闭swd引脚

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    //所有列配置为低电平推挽输出

    RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO|KEY\_COL\_GPIO\_CLK, ENABLE);

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_COL\_PORT, KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN);

    //行因引脚中断初始化,按键按下进入对应中断

    GPIO\_EXTILineConfig(KEY\_ROW\_PORTSOURCE , KEY\_ROW1\_PINSOURCE|KEY\_ROW2\_PINSOURCE|KEY\_ROW3\_PINSOURCE);

    EXTI\_InitStructure.EXTI\_Line = KEY\_ROW1\_EXTILINE|KEY\_ROW2\_EXTILINE|KEY\_ROW3\_EXTILINE; // 外部中断线2

    EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; // 配置为外部中断

    EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;

    EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; // 使能外部中断

    EXTI\_Init(&EXTI\_InitStructure);

    //按键矩阵

    NVIC\_InitTypeDef NVIC\_InitStructure = {0};

    NVIC\_InitStructure.NVIC\_IRQChannel = KEYBOARD\_EXTI\_IRQn ;

    NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

    NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2;

    NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

    NVIC\_Init(&NVIC\_InitStructure);

}

//检测到哪一行后，进入列扫描模式，行引脚设置为低电平推挽输出模式，列引脚设置为上拉输入模式

void Keyboart\_ScanCol\_Mode(void)

{

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    //所有行配置为低电平推挽输出模式

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_ROW\_PORT, KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN);

    //所有列配置为上拉输入模式

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

}

void Keyboart\_ScanRow\_Mode(void)

{

    GPIO\_InitTypeDef GPIO\_InitStructure = {0};

    GPIO\_InitStructure.GPIO\_Pin = KEY\_ROW1\_PIN|KEY\_ROW2\_PIN|KEY\_ROW3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;

    GPIO\_Init(KEY\_ROW\_PORT, &GPIO\_InitStructure);

    //所有列配置为低电平推挽输出

    GPIO\_InitStructure.GPIO\_Pin = KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN;

    GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

    GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

    GPIO\_Init(KEY\_COL\_PORT, &GPIO\_InitStructure);

    GPIO\_ResetBits(KEY\_COL\_PORT, KEY\_COL1\_PIN|KEY\_COL2\_PIN|KEY\_COL3\_PIN);

}

void KEYBOARD\_EXTI\_IRQHandler(void)

{

    //行1中断触发

    if(EXTI\_GetITStatus(KEY\_ROW1\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW1\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW1\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("11\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("12\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("13\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW1\_EXTILINE);

        }

    }

    //行2中断触发

    else if(EXTI\_GetITStatus(KEY\_ROW2\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW2\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW2\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("21\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("22\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("23\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW2\_EXTILINE);

        }

    }

    //行3中断触发

    else if(EXTI\_GetITStatus(KEY\_ROW3\_EXTILINE)!=RESET)

    {

        EXTI\_ClearITPendingBit(KEY\_ROW3\_EXTILINE);

        Delay\_Ms(50);

        if(GPIO\_ReadInputDataBit(KEY\_ROW\_PORT, KEY\_ROW3\_PIN)==0)

        {

            Keyboart\_ScanCol\_Mode();

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL1\_PIN)==0)

            {

                printf("31\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL2\_PIN)==0)

            {

                printf("32\r\n");

            }

            if(GPIO\_ReadInputDataBit(KEY\_COL\_PORT, KEY\_COL3\_PIN)==0)

            {

                printf("33\r\n");

            }

            Keyboart\_ScanRow\_Mode();

            EXTI\_ClearITPendingBit(KEY\_ROW3\_EXTILINE);

        }

    }

}