**Microcontroller Experiment**

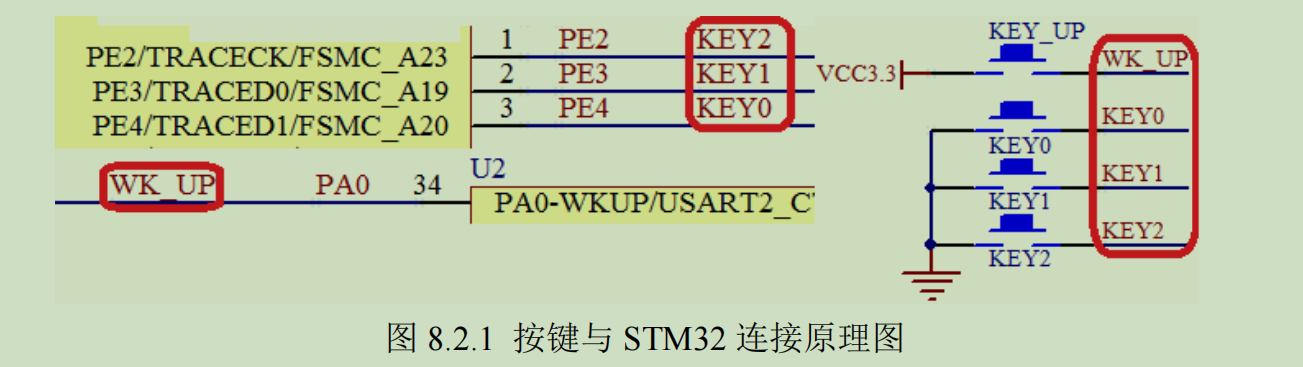
Key Input Experiment

Name: Han Yichen 韩一尘

Student ID: 22722051

Date: 2023/11/1

1. **Experimental Principles**

The key input is realized through the IO port of STM32F1, and the user uses the four keys on board to control the on-off of the two LEDs and the on-off of the buzzer. When the IO port of STM32F1 is used as input, the state of the IO port is read by calling the function GPIO\_ReadInputDataBit (). On the other hand, there are four buttons (WK\_UP, KEY0, KEY1 and KEY2) on the development board of warship STM32 to control the two LEDs (DS0 and DS1) and buzzer on the board, in which WK\_UP controls the buzzer, and presses it once to stop; KEY2 controls DS0, pressing it once to light up and then pressing it once to turn it off; KEY1 controls DS1 with the same effect as Key 2; KEY0 controls DS0 and DS1 at the same time. Press it once, and their states will flip once. The principle of connecting the key with STM32 is shown in the following figure:

It should be noted here that KEY0, KEY1 and KEY2 are active at low level, while WK\_UP is active at high level, and there is no pull-up resistor outside, so it is necessary to set up pull-up and pull-down inside STM32.

1. **Main program analysis**

#include "led.h"

#include "delay.h"

#include "key.h"

#include "sys.h"

#include "beep.h"

//ALIENTEK 战舰 STM32 开发板实验 3

//按键输入实验

int main(void)

{

u8 key;

delay\_init();

//延时函数初始化

LED\_Init();

//LED 端口初始化

KEY\_Init(); //初始化与按键连接的硬件接口

BEEP\_Init(); //初始化蜂鸣器端口

LED0=0;

//先点亮红灯

while(1)

{

key =KEY\_Scan(0);

//得到键值

if(key)

{

switch(t)

{

case WKUP\_PRES:

//控制蜂鸣器

BEEP=!BEEP;break;

case KEY2\_PRES: //控制 LED0 翻转

LED0=!LED0;break;

case KEY1\_PRES: //控制 LED1 翻转

LED1=!LED1;break;

case KEY0\_PRES: //同时控制 LED0,LED1 翻转

LED0=!LED0;

LED1=!LED1;break;

}

}else delay\_ms(10);

}

}

There are two functions in the key.c file, and KEY\_Init () is used to initialize the IO port of key input. First, the GPIOA and GPIOE clocks are enabled, and then the input setting. KEY\_Scan () function of PA0 and PE2~4 is realized, which is used to scan whether the four IO ports are pressed. The KEY\_Scan () function supports two functions.

This scanning mode is set by the mode parameter.

When mode is 0, the KEY\_Scan () function will not support continuous pressing to scan a key, and after the key is pressed.

You must release it before you can trigger it for the second time, otherwise you won't respond to this button again. This has the advantage of preventing you from pressing it more than once.

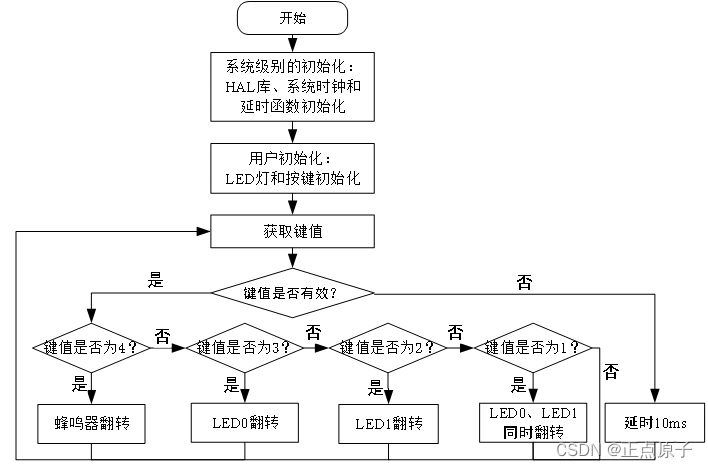
The second trigger, and the disadvantage is that it is not suitable when long press is needed.

When mode is 1, the KEY\_Scan () function will support continuous pressing, and if a key is pressed all the time, it will continue.

Returns the key value of this key, so that long press detection can be realized conveniently.

The first is to call system-level initialization: initialize HAL library, system clock and delay function. Next, call led\_init to initialize the LED lamp, and call the key\_init function to initialize the key. Finally, the key value is scanned in the infinite loop, and then the key value is used to determine which key is pressed. If there is a key pressed, the corresponding lamp is turned over, and if there is no key pressed, the delay is 10 ms.

**3.Flow chart**



**4.Experimental result :**

WK\_UP press call once, and then press stop once; KEY2 controls DS0, pressing it once to light up and then pressing it once to turn it off; KEY1 controls DS1 with the same effect as Key 2; KEY0 controls DS0 and DS1 at the same time. Press it once, and their states will flip once.

1. **Improvement and perfection**

At first, only one drop-down button was selected for debugging, but later, a pull-up button was added, and then the if judgment loop of the first three buttons would not stop, resulting in failure.

1. The basic configuration of 1.IO port must be correct, the clock function should not be used wrongly, and the parameters should be corrected.

2. The pull-down mode on 2.IO port should be correct.

3.IO port level initialization configuration

4. directly read the IO position 0/1 without pressing the key.