1 电容触摸按键实验

1.1 实验目的

通过 STM32F429 的 TIM2 的通道 1（PA5）实现输入捕获功能，并实现一个简单的电容触摸按键，通过该按键控制灯 DS1 的亮灭。了解触摸按键的原理，并能针对此动作进行简单的反馈。

1.2 实验原理

使用检测电容充放电时间的方法来判断是否有触摸，图 1 中 R 是外接的电容充电电阻，Cs 是没有触摸按下时 TPAD 与 PCB 之间的杂散电容。而 Cx 则是有手指按下的时候，手指与 TPAD 之间形成的电容。图中的开关是电容放电开关（由实际使用时，由 STM32F429 的 IO 代替）。先用开关将 Cs（或 C+Cx）上的电放尽，然后断开开关，让 R 给 Cs（或 Cs+Cx）充电，当没有手指触摸的时候，Cs 的充电曲线如图中的 A 曲线。而当有手指触摸的时候，手指和 TPAD 之间引入了新的电容 Cx，此时 Cs+Cx 的充电曲线如图中的 B 曲线。从上图可以看出，A、B 两种情况下，Vc 达到 Vth 的时间分别为 Tcs 和 Tcs+Tcx。在本次实验中，只要能够区分 Tcs 和 Tcs+Tcx，就已经可以实现触摸检测了，当充电时间在 Tcs 附近，就可以认为没有触摸，而当充电时间大于 Tcs+Tx 时，就认为有触摸按下（Tx 为检测阀值）。

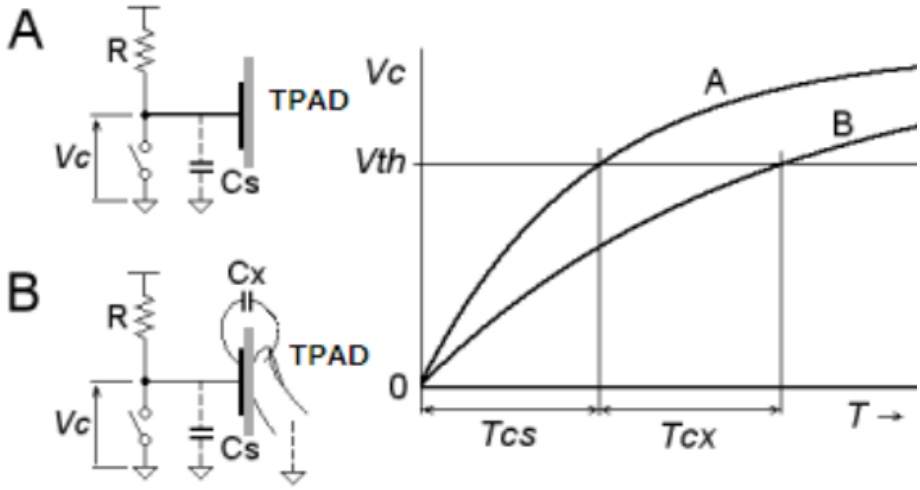

图 **1**　电容触摸按键原理

1.3 代码描述

main.c 片段

```c
void space_ratio(u8 delayms, float rate, _Bool rev) {
  // 根据占空比调整 LED 亮度
  LED1 = rev;
  delay_ms((u8)(delayms * rate));
  LED1 = !rev;
  delay_ms((u8)(delayms * (1 - rate)));
}
int main(void) {
  HAL_Init();                         // 初始化 HAL 库
  Stm32_Clock_Init(360, 25, 2, 8); // 设置时钟,180Mhz
  delay_init(180);                    // 初始化延时函数
  uart_init(115200);                  // 初始化 USART
  LED_Init();                         // 初始化 LED
  TPAD_Init(2); // 初始化触摸按键,以 90/8=11.25Mhz 频率计数
  u8 stage = 0; // 共有 5 个状态，0 - 3 有不同闪烁频率，4 是亮度高低闪烁。
  LED1 = 1, LED0 = 0;                                    // 初始状态
  uint16_t delay = 10, temp[] = {1000, 500, 100, 50}; // 设置闪烁间隔
  for (uint16_t t = 0;; ++t) {                           // t 用于计时
    if (TPAD_Scan(0)) { // 检测到按键，改变当前状态
      ++stage, stage %= 5;
    }
    if (stage < 4 && t > temp[stage] / delay / 4) {
      // 若处于闪烁状态且该次闪烁已打到时间，闪烁
      LED1 = !LED1, LED0 = !LED1;
      t = 0;
    } else if (stage == 4) { // 在亮度调节状态
      LED1 = 1;                // 初始亮度最低
      if (t > 20)
        t = 0;                       // 反复闪烁
      for (int _ = 0; _ < 3; ++_) // 延时
        space_ratio(15, (float)t / 15, 0);
    }
    if (stage < 4) // 4 状态的 space_ratio 自带延时，因此不需要再 delay
      delay_ms(delay);
  }
}
```

1.4 实验结果

在本次实验中，修改了部分代码，实现了开始时红黄两灯以较慢的速度交替发光，通过不断触摸覆铜区域，增加交替变化的频率，最终达到以不同亮度闪烁的状态，总体上实现了触摸按键增加频闪频率的功能。

1.5 心得体会

开始时我封装了一个以不同频率闪烁的函数，然而实际测试中发现很难接受到按键信号，因为将 `delay_ms` 放在该函数内，函数运行会阻塞主线程（外层的 `while` 循环），使传感器无法检测到按键信号。

在前期测试中有出现两灯常亮的情况，但其实是频率过快人眼不能辨别，因此需要调整参数。`temp[] = {1000, 500, 100, 50}` 和 `t > temp[stage] / delay / 4` 的具体数字是调整的结果。

2 光照 & 接近传感器实验外部中断实验

2.1 实验目的

使用 STM32F429 的普通 IO 口模拟 IIC 时序，来驱动 AP3216C 传感器模块，从而检测环境光强度（ALS）和接近距离（PS）这类环境参数，并作出反馈。

在实际使用中，还学习了 LCD 显示屏的使用方法。

2.2 实验原理

本次实验使用的环境传感器为AP3216C，这是敦南科技推出的一款三合一环境传感器，它包含了：数字环境光传感器（ALS）、接近传感器（PS）和一个红外 LED（IR）。该芯片通过 IIC 接口和 MCU 连接，并支持中断（INT） 输出。AP3216C 的框图如图 2 所示。

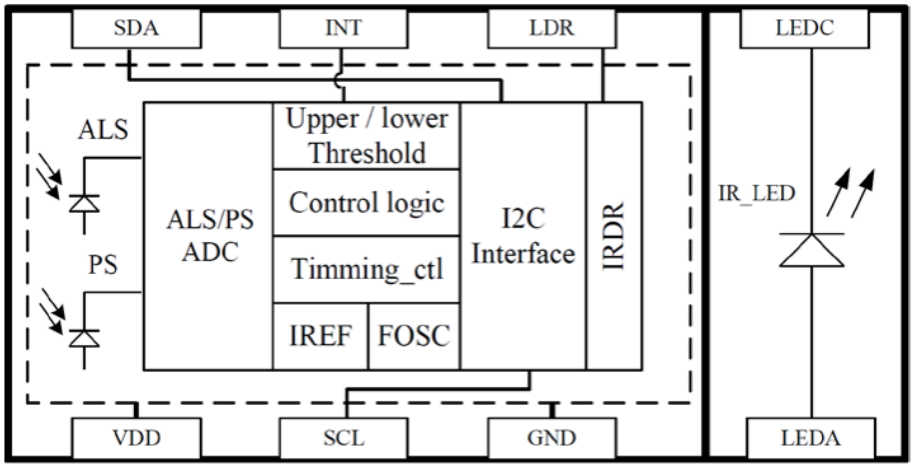

图 **2** **AP3216C** 框图

2.3 代码描述

main.c 片段

```c
#include "math.h"
int main(void) {
  u16 ir, als, ps;
  HAL_Init();                         // 初始化 HAL 库
  Stm32_Clock_Init(360, 25, 2, 8); // 设置时钟,180Mhz
  delay_init(180);                    // 初始化延时函数
  uart_init(115200);                  // 初始化 USART
  LED_Init();                         // 初始化 LED
  KEY_Init();                         // 初始化按键
  SDRAM_Init();                       // 初始化 SDRAM
  LCD_Init();                         // 初始化 LCD
  POINT_COLOR = BLUE;                 // 设置字体为蓝色
  const int center[] = {230, 400}; // 圆圈的中心点
  u8 key;
  _Bool mode = 0; // 模式，0 为距离模式，1 为光照模式
  while (1) {
    key = KEY_Scan(0); // 监测按键，改变工作模式
    if (key == WKUP_PRES)
      mode = !mode;
    AP3216C_ReadData(&ir, &ps, &als); // 读取数据
    LCD_Clear(99999); // 每个循环清屏，让圆形能动态显示
    LCD_ShowString(30, 130, 200, 16, 16,
                   !mode ? "mode: distance"
                         : "mode: brightness"); // 在显示屏上显示工作模式
    const float temp =
        !mode ? (float)ps / 10 : sqrt(als) * 1.5; // 预处理距离/亮度，坐标变换
    LCD_Draw_Circle(center[0], center[1], temp);
    LCD_Draw_Circle(center[0], center[1], temp + 1);
    LCD_Draw_Circle(center[0], center[1], temp + 2);
    // 画圆。LCD_Draw_Circle 函数没有调节粗细的功能，因此画三个圆来加粗。
    delay_ms(90);
  }
}
```

2.4 实验结果

在本次的实验中，修改了部分代码，没有显示参数值，取而代之的是一个会随着光照强度和距离远近而变化面积的圆。具体表现是,当距离越近或光照强度越大时,圆半径越大.可以通过按键 *KEY_UP* 切换检测光照和检测距离的检测模式。

2.5 心得体会

在 `lcd.h` 头文件中内置了许多图形 & 字符绘制的函数，可以多加利用；光照传感器的值是非线性的，因此用 sqrt 将其映射为相对线性值，便于观察；当绘制的圆形超出了 LCD 屏幕的左右边界时，根据其绘制策略，可以看到“反射”回来的“波纹”，比较有意思。