目录

更新

2024-04-12

封装模块

- buzzer
- delay
  - 11.0592MHz
    - 10us * t
    - 1ms * t
  - 12MHz
    - 10us * t
    - 1ms * t
- hc_sr04
- key_4x4
- lcd1602
- motor
  - dc motor
  - stepping motor

使用方法

导入项目

正常创建 Keil 项目, 并将需要的封装库加入到项目.

- 如果整个文件夹一起装入项目, 可以选择:
  - 在 Keil C51/C 里面添加文件路径
  - include 加入路径: '#include "hc_sr04/hc_sr04.h"'
- 如果直接将文件复制到了项目路径, 那正常导入便好

在项目目录下创建'config.h'文件, 并按照每个库头文件的要求进行宏定义

- 每个头文件都将需要定义和可以选择定义的宏解释了的, 项目里面也有示例
- 示例

'__config__.h'

```
/* ------- define for lcd1602 ------- */

#if 0
#define UNUSED_LCD1602              // 将不编译 LCD1602 相关函数
#endif
#if 0
#define LCD1602_TIMEOUT_BREAK       // 设置可手动跳出查忙
#endif
#if 0
#define LCD1602_NEED_READ_DATA      // 需要编译 LCD1602_Read
#endif
#define LCD1602_DATA P0             // 数据 to LCD1602
#define LCD1602_DEFINE_RS P1 ^ 0    // 寄存器选择
#define LCD1602_DEFINE_RW P1 ^ 1    // 读/写
#define LCD1602_DEFINE_EN P1 ^ 2    // 使能
```

'lcd1602.h'

```
#ifndef LCD1602_H
#define LCD1602_H
/**
 * 在使用之前需要在 '__config__.h' 中定义以下宏:
 *   - LCD1602_DEFINE_RS: 寄存器选择引脚
 *   - LCD1602_DEFINE_RW: 读/写引脚
 *   - LCD1602_DEFINE_EN: 使能引脚
 *   - LCD1602_DATA: LCD1602 与 单片机连接的 数据引脚s
 *
 * 如果需要使用从 LCD1602 上读取数据, 需要定义宏:
 *   - LCD1602_NEED_READ_DATA: 这将会编译 LCD1602_Read 函数
 *
 * 当包含多文件测试时, 有些文件暂时用不了 Keil 会警告, 作为妥协可以定义宏:
 *   - UNUSED_LCD1602: 这将不会编译 LCD1602 相关函数
 */

/**
 * 为了解决单片机: 无连接或者连接的lcd1602故障, 导致查忙一直无响应的死循环的问题
 * 可以在 '__config__.h' 中定义宏: LCD1602_TIMEOUT_BREAK
 * 便可以自行通过定时器中断或者外部中断将 'lcd1602_timeout' 置零 来跳出查忙
 * 注意: 如果手动将 lcd1602_timeout 置于1 请手动归0, 否则查忙将不可用
 * 更详细的内容请查看 'lcd1602.c' 文件
 */
```

在'main.c'文件中导入需要的头文件, 并根据头文件给出的接口进行调用

- 接口都给的很简单, 可以直接使用
- 示例

'main.c'

```c
#include "__config__.h"
#include "delay/delay_c52.h"
#include "key_4x4/key_4x4.h"
#include "lcd1602/lcd1602.h"
#include "hc_sr04/hc_sr04.h"

// c语言标准库函数
extern int sprintf (char *, const char *, ...);

void LCD1602_ShowString(unsigned char* s)
{
    while (*s)
    {
        LCD1602_Write(*s++);
    }
}

void main(void)
{
    unsigned char num[10];

    LCD1602_Cmd(Set_8bit_2line_5x7);    // 命令6
    LCD1602_Cmd(Show_CursorOn);         // 命令4
    LCD1602_Cmd(Mode_CursorRightMove); // 命令3
    LCD1602_Cmd(Clear_Screen);          // 命令1
    LCD1602_Write('A');
    Delay1ms(1);

    while (1)
    {
        LCD1602_Cmd(Move_Cursor_Row1_Col(3));
        // 调用超声波传感器获取距离(单位: mm)
        sprintf(num, "%dmm", HC_SR04_Millimeter());
        // 显示在 lcd 屏幕上
        LCD1602_ShowString(num);
        Delay1ms(1000);
    }
}
```

'lcd1602.h'

```c
// 命令1: 清屏
#define Clear_Screen              0x01    // ... ...

// 命令2: 光标返回0位
#define Return_Cursor             0x02    // 0x02-0x03

// 命令3: 输入字符后 屏幕/光标 移动
#define Mode_ScreenLeftMove       0x05    // 屏幕左移
#define Mode_ScreenRightMove      0x07    // 屏幕右移
#define Mode_CursorLeftMove       0x04    // 光标自减
#define Mode_CursorRightMove      0x06    // 光标自增

// 命令4: 控制屏幕和光标状态
#define Show_ScreenOff            0x08    // 屏幕关闭 0x08-0x0b
#define Show_CursorOff            0x0c    // 亮屏无光标 0x0c 0x0d
#define Show_CursorOn             0x0e    // 亮屏有光标
#define Show_CursorFlicker        0x0f    // 亮屏光标闪烁

// 命令5: 光标/屏幕 左/右移
/**
 * 值得注意的是 屏幕不会真的移动，屏幕移动是靠 DDRAM 整体的循环移动呈现的
 */
#define Shift_CursorLeft          0x10    // 光标左移 0x10-0x13
#define Shift_CursorRight         0x14    // 光标右移 0x14-0x17
#define Shift_ScreenLeft          0x1c    // 屏幕左移 0x1c-0x1f
#define Shift_ScreenRight         0x18    // 屏幕右移 0x18-0x1b

// 命令6: 设置数据位 显示行数 点阵
#define Set_4bit_1line_5x7        0x20    // 0x20-0x23
#define Set_4bit_1line_5x10       0x24    // 0x24-0x27
#define Set_4bit_2line_5x7        0x28    // 0x28-0x2b
```

'hc_sr04.h'

```c
/**
 * 注意事项:
 *    - 探测并返回，整个过程不会用到中断，请勿开启所使用的定时器中断('__config__.h'中定义的)
 *      虽然一般情况不会出错，但是难免会有意外
 *    - 虽然不会用到中断，但是如果有中断打断过程可能会造成以下问题:
 *        - 发送触发信号被打断，导致探测失败，将会返回不确定的数字
 *        - 计时被干扰，导致计时错误，会返回不确定的数字
 *    - 此封装，虽然对 TMOD 进行了操作，但是不会干扰到另一个定时器工作
 *    - 为了防止 TMOD 在外部赋值出现意外，再每次开始前都会对定时器初始化，略微的开销是值得的
 * 返回值: 计算出的距离，单位: mm
 */
unsigned int HC_SR04_Millimeter(void);
```

'hc_sr04.c'

```c
/**
 * 计算方法: 340m/s(来回路程距离) / 2 * TIME_VALUE
 * 换算后: 0.17 mm/us(单程距离) * Tus(0<=T<=65535)
 * 最大可计算 约: 650cm, HC_SR04
 * 探测距离: 2cm-450cm, 足够了
 * 返回值: us * mm/us = mm
*/
unsigned int HC_SR04_Millimeter(void)
{
    TIMER_INIT;
    TRIG = 0;
    TRIG = 1; // 触发信号
    HC_SR04_DELAY_10US;
    TRIG = 0;
    while (!ECHO)
        ; // 等待开始探测
    TIMER_START;
    while (ECHO)
        ; // 等待结束探测
    TIMER_STOP;
    return (unsigned int)(TIME_VALUE * 0.17);
}
```

c51_lib

关于c51常用的一些模块的封装库