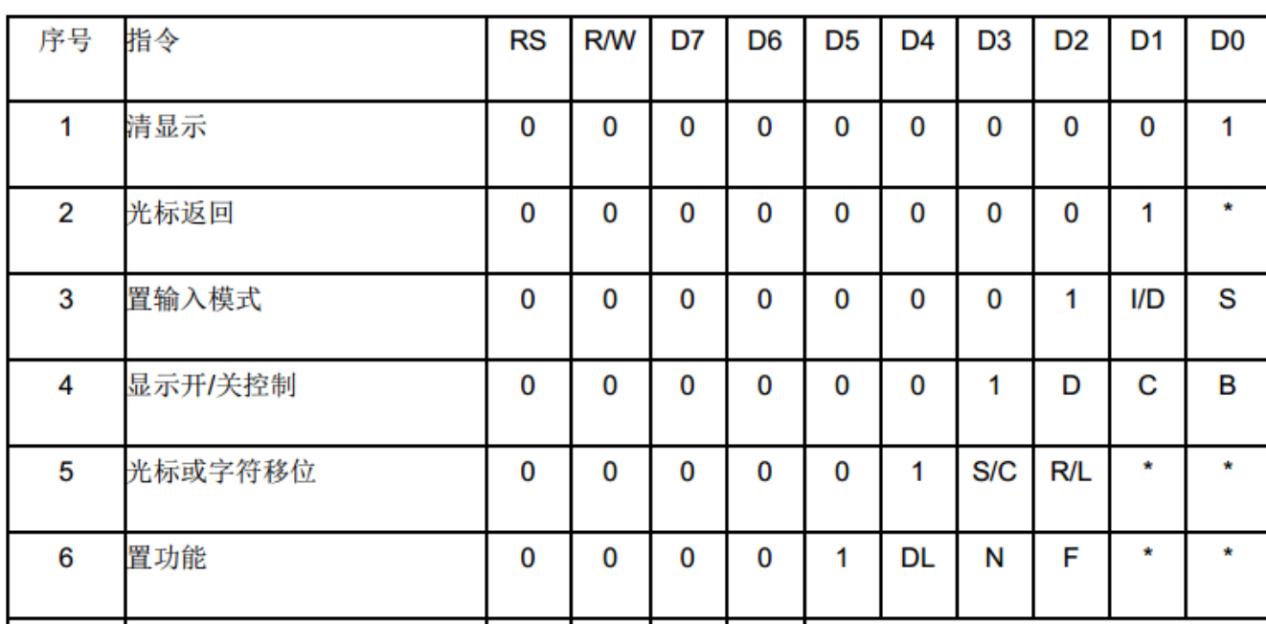
**STM32G071RB连接外设LCD1602**

说明：将驱动从单片机STC8A8K64S4A12移植到ARM处理器STM32G071RB

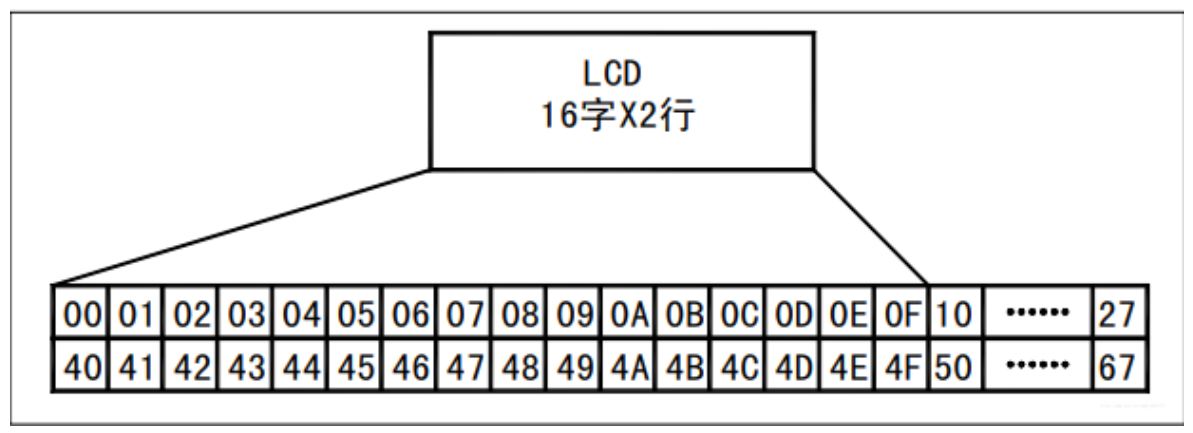
**第一步**：了解LCD1602相关知识

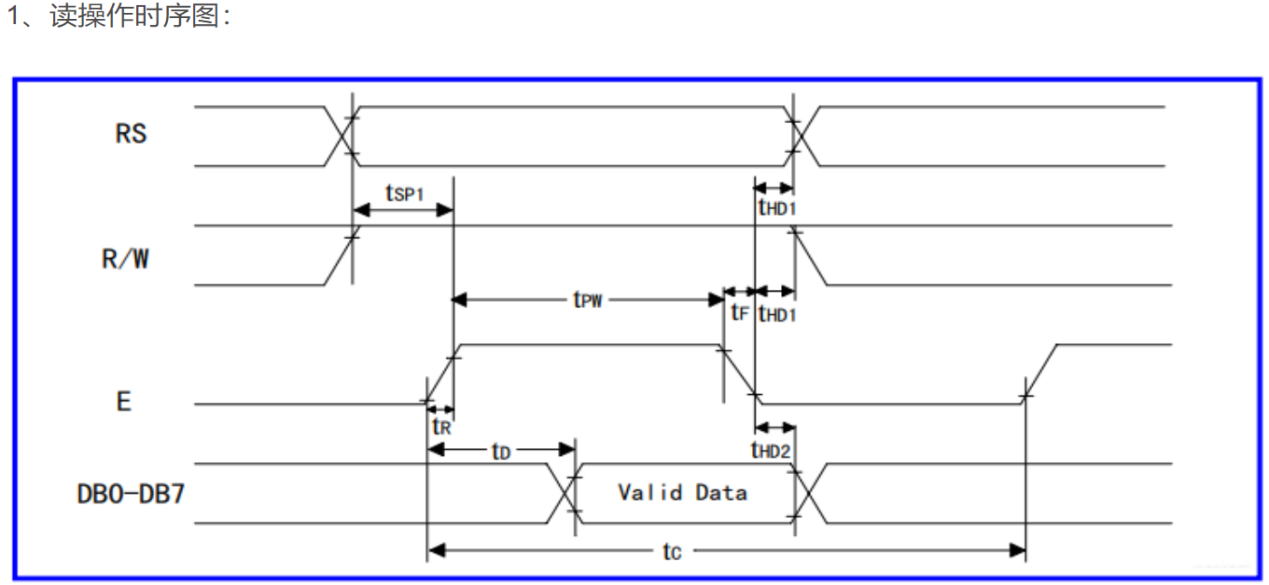
****

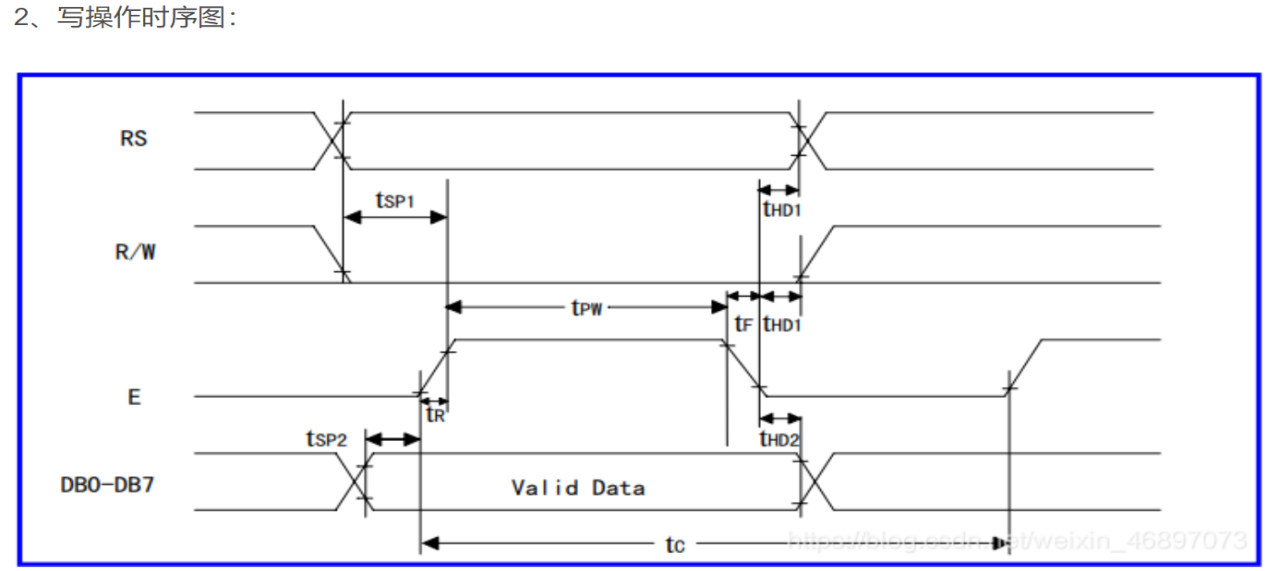
****

****

****

****

****

****

**第二步**：查找之前写过的STC8A8K64S4A12与LCD1602的使用

**①led1602.h文件：**

#ifndef \_1602\_

#define \_1602\_

#include "reg51.h"

#include "intrins.h"

#include "stdio.h"

#define u8 unsigned char //自定义数据类型u8 与u16

#define u16 unsigned int

#define EN\_GOL\_INT EA=1

sbit LCD1602\_RS=P2^5; //LCD1602的RS信号线连接单片机P2.5管脚

sbit LCD1602\_RW=P2^6; //LCD1602的RW信号线连接单片机P2.6管脚

sbit LCD1602\_E =P2^7; //LCD1602的E信号线连接单片机P2.7管脚

sfr LCD1602\_DB=0x80; //LCD1602的数据总线连接到P0端口

sfr P0M1=0x93; //定义P0端口P0M1寄存器地址 P0连接1602数据总线

sfr P0M0=0x94; //定义P0端口P0M0寄存器地址

sfr P2M1=0x95; //定义P2端口P2M1寄存器地址 P2连接1602控制总线

sfr P2M0=0x96; //定义P2端口P2M0寄存器地址

void delay(void); //定义函数--延时

void set\_P0\_mode(u8 a,u8 b); //定义函数--设置端口P0工作模式

void set\_P2\_mode(u8 a,u8 b); //定义函数--设置端口P2工作模式

void lcdwait(); //定义函数--等待空闲

void lcdwritecmd(unsigned char cmd); //定义函数--向1602发送指令

void lcdwritedata(unsigned char dat); //定义函数--向1602发送数据

void lcdinit(); //定义函数--1602初始化

void lcdsetcursor(unsigned char x, unsigned char y); //定义函数--1602输出定位

void lcdshowstr(unsigned char x, unsigned char y, //定义函数--LCD1602输出

unsigned char \*str);

#endif

**②led1602.c文件**

#include "led1602.h"

void delay(void) //利用指令来延时

{

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

\_nop\_();

}

void lcdwait() //读取BF标志，以此等待空闲

{

LCD1602\_DB=0xFF; //数据总线全部置一

delay();

LCD1602\_RS=0;

LCD1602\_RW=1;

LCD1602\_E=1;

while(LCD1602\_DB & 0x80); //等待标志BF为低，LCD1602空闲

LCD1602\_E=0;

}

void lcdwritecmd(unsigned char cmd)

{

lcdwait(); //等待空闲

delay();

LCD1602\_RS=0;

LCD1602\_RW=0;

LCD1602\_DB=cmd; //发送命令

LCD1602\_E=1;

delay();

LCD1602\_E=0; //执行命令（下降沿执行）

}

void lcdwritedata(unsigned char dat)

{

lcdwait();

delay();

LCD1602\_RS=1;

LCD1602\_RW=0;

LCD1602\_DB=dat; //发送数据

LCD1602\_E=1;

delay();

LCD1602\_E=0; //执行

}

void lcdinit()

{

lcdwritecmd(0x38); //发送命令0x38，2行模式，5x8点阵，8位宽度

lcdwritecmd(0x0c); //发送命令0x0c，打开显示，关闭光标

lcdwritecmd(0x06); //发送命令0x06，文字不移动，地址自动加一

lcdwritecmd(0x01); //发送命令0x01，清屏

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x; //y=0（第一行），起始地址0x00加偏移地址x（列）

else

address=0x40+x; //y=0（第二行），起始地址0x40加偏移地址x（列）

lcdwritecmd(address|0x80); //写存储器地址命令

}

void lcdshowstr(unsigned char x, unsigned char y,

unsigned char \*str)

{

lcdsetcursor(x,y); //地址定位

while((\*str)!='\0') //不是字符串的结尾则继续

{

lcdwritedata(\*str); //将str指向的数据发送给1602（成功输出）

str++; //指针指向下一个需要输出的数据

}

}

void set\_P0\_mode(u8 a,u8 b) //设置端口P0（连接1602数据总线）工作模式

{

P0M0=a;

P0M1=b;

}

void set\_P2\_mode(u8 a,u8 b) //设置端口P2（连接1602地址总线）工作模式

{

P2M0=a;

P2M1=b;

}

**③主文件1602.c**

#include "led1602.h"

void main()

{

set\_P0\_mode(0x00,0x00); //设置端口模式

set\_P2\_mode(0x00,0x00);

EN\_GOL\_INT; //使能单片机全局中断

lcdwait(); //确认LCD为空闲状态

lcdinit(); //初始化LCD

delay();

lcdshowstr(0,0,"Xiao Xin Xin"); //输出

lcdshowstr(6,1,"V");

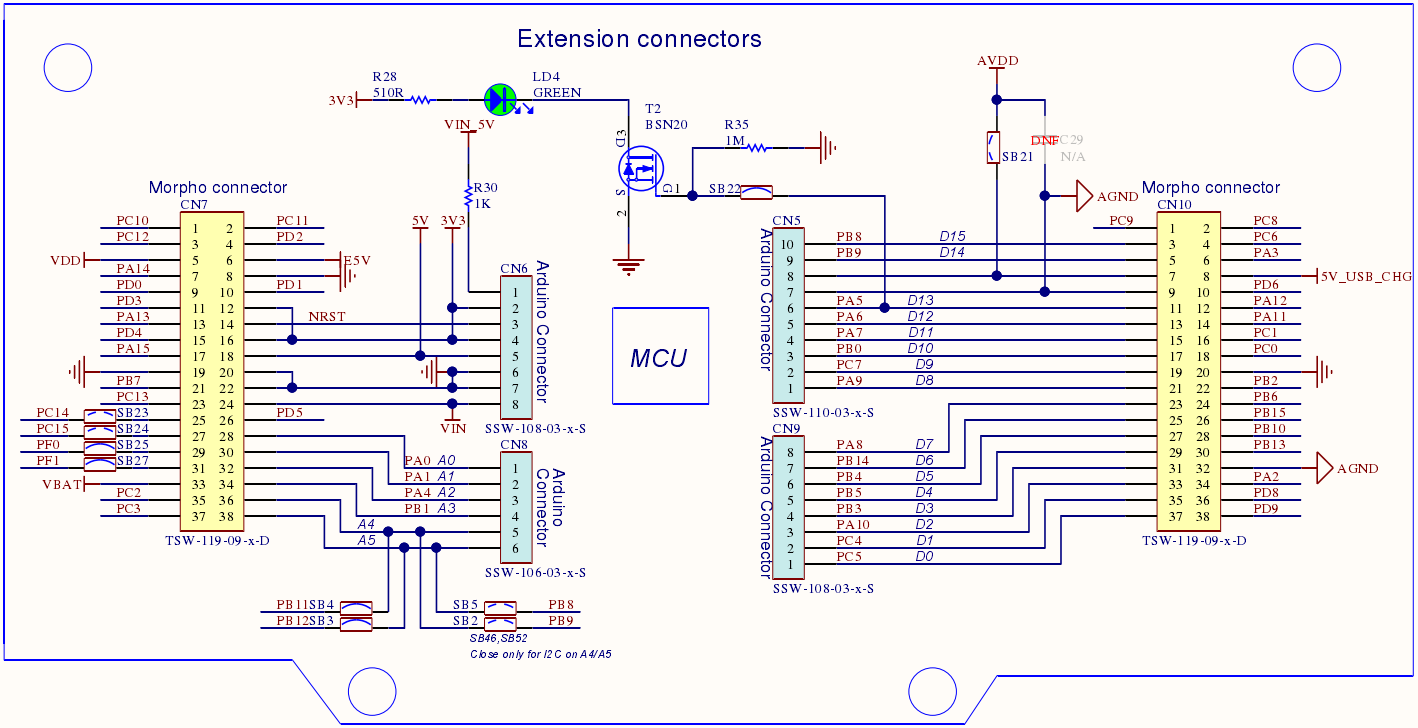
while(1); //保持

}

**第三步**：硬件连接

思路：数据总线连接到GPIOA[7:0]，E、RW、RS分别连接到GPIOB[2:0]。

具体引脚见下方原理图：

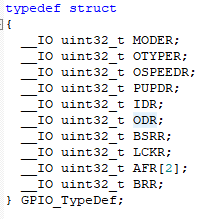


**第四步**：建立新的ARM工程

将GPIOA[7:0]与GPIOB[2:0]管脚设置为GPIO\_OUT，并生成代码。

**第五步**：确定驱动移植的思路

对LCD1602的数据总线GPIOA[7:0]改写用结构体形式，改写其输出数据寄存器。



该结构体存储在如下文件：

ba00ec8bf58c74c34176b4994852e07

改写输出寄存器方式如下：

35c0372dc399af25a77c4a976296f02

对RS、RW、E信号GPIOB[2:0]的按位改写用HAL\_GPIO库函数HAL\_GPIO\_WritePin()函数。

**第六步**：解决问题实现移植

**问题①**：已经有一个生成的库文件了，不能成功添加单片机的led1602.h和led1602.c

解决：将LCD1602驱动用到的函数粘贴在main.c文件下。

**问题②**：单片机延迟函数delay()中嵌套的\_nop\_()函数无法成功调用。

解决：用库函数HAL\_Delay()代替此延迟函数。

**问题③**：去除多余部分。

解决：去除设置P0和P2端口的工作模式，因为创建工程生成代码之前我们就已经设置对应管脚工作模式为通用输出。也去除单片机中断使能EN\_GOL\_INT()。

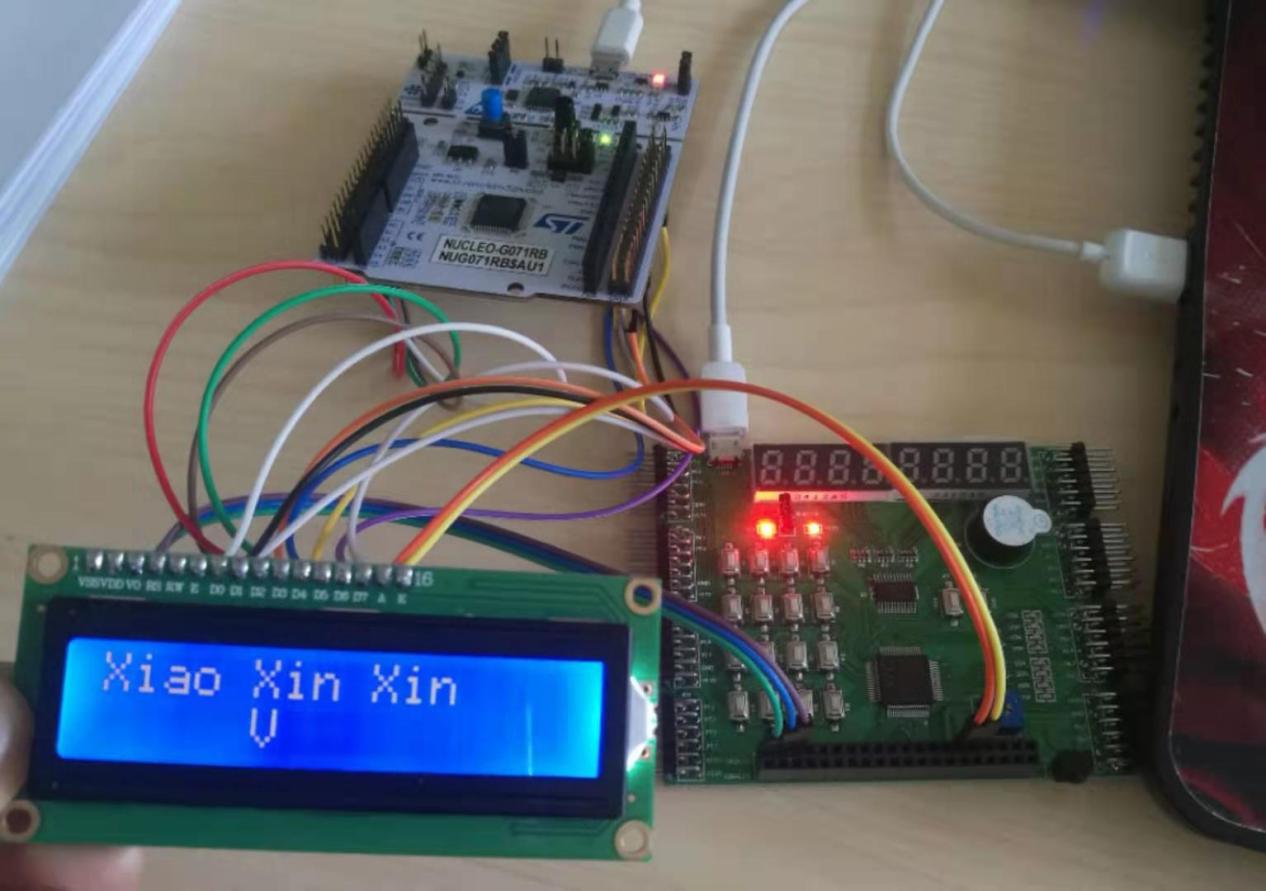
**问题④**：显示在LCD1602的字符串出现定位错误（中间以空格隔开）。

问题分析：回顾设置：LCD1602的工作模式设置是文字不动，地址自动加一；GPIOA和GPIOB的工作模式为通用输出。在代码中其中一个子函数为读取LCD1602的BF标志位，在其忙时等待，在空闲时才执行以下命令。但是GPIOA和GPIOB没有输入数据的功能，所以这个子函数执行完毕之后，没能读取BF标志位并且让LCD1602的地址自动加一，造成了地址过度右移，故字符与字符之间产生了空格。

解决办法设想：该读取BF标志位并等待的子函数作用，可以理解为一个时间不定的延时函数。只要用延时库函数HAL\_Delay(time)代替它即可，只要这里time够大即可避开LCD的忙状态。

解决：实际上发现直接注释掉这个等待函数即可，没有因为LCD1602的忙状态而输出错误。【之后的设计如果有影响即可在对应位置加HAL\_Delay(time)，这样造成的后果是会造成时间上的浪费。】

第七步：效果演示



**第八步**：代码设计

①在main.c文件起始位置添加：

#include "stdio.h"

#define u8 unsigned char

#define u16 unsigned int

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y,

unsigned char \*str);

②在主函数GPIO初始化与while(1)之间添加

lcdinit();

HAL\_Delay(10);

lcdshowstr(0,0,"Xiao Xin Xin");

lcdshowstr(5,1,"V");

③在主函数之后添加如下子函数

void lcdwritecmd(unsigned char cmd)

{

// lcdwait();

HAL\_Delay(10);

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_0,GPIO\_PIN\_RESET);// LCD1602\_RS=0;

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_1,GPIO\_PIN\_RESET);

GPIOA->ODR=cmd;

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_2,GPIO\_PIN\_SET);

HAL\_Delay(10);

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_2,GPIO\_PIN\_RESET);

}

void lcdwritedata(unsigned char dat)

{

// lcdwait();

HAL\_Delay(10);

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_0,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_1,GPIO\_PIN\_RESET);

GPIOA->ODR=dat;

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_2,GPIO\_PIN\_SET);

HAL\_Delay(10);

HAL\_GPIO\_WritePin(GPIOB,GPIO\_PIN\_2,GPIO\_PIN\_RESET);

}

void lcdinit()

{

lcdwritecmd(0x38);

lcdwritecmd(0x0c);

lcdwritecmd(0x06);

lcdwritecmd(0x01);

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x;

else

address=0x40+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str)

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

**第九步**：总结与展望

总结：为什么LCD1602第一行结束地址为0x27而第二行起始地址为0x40？中间为什么要空出来？这样设计第一行输出字符串如果溢出一部分不会影响到第二行的输出。

展望①：思考E、RW、RE在每个时间段的值，即可用GPIOB的结构体给输出数据寄存器整体赋值GPIOB->ODD=0x\*\*\*\*\*\*\*\*，不再用HAL\_GPIO\_WritePin()函数。

展望②：学习结构体相关知识，自己定义结构体。

展望③：实现读取BF标志来等待LCD的功能lcdwait()，要求管脚双向传输。

拓展思考：我是利用STC8A8K64S4A12给LCD1602供电，用STM32G071RB控制LCD1602输出。那么能不能把单片机一个端口和ARM的一个端口串联，单片机为输出端口，ARM为中断输入端口。单片机作为协控制器逻辑：按下单片机按键---->单片机输出端口为高电平---->ARM接收端口收到控制信号---->ARM中断触发---->响应中断

**第十步**：改进

去除HAL库的调用：GPIOB也用结构体赋值，自定义延时函数

①mian.c文件开头添加

#define u8 unsigned char

#define u16 unsigned int

void delay(void);

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y,

unsigned char \*str);

②main.c文件GPIO初始化和while(1)之间添加

lcdinit();

delay();

lcdshowstr(0,0,"Good Boy");

lcdshowstr(0,1,"Success");

③添加子函数

void delay ()

{

for(int i=0;i<99;i++)

for(int j=0;j<99;j++)

{}

}

void lcdwritecmd(unsigned char cmd)

{

// lcdwait();

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

GPIOA->ODR=cmd;

GPIOB->ODR=0x04; //E=1\RW=0\RS=0

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

}

void lcdwritedata(unsigned char dat)

{

// lcdwait();

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

GPIOA->ODR=dat;

GPIOB->ODR=0x05; //E=1\RW=0\RS=1

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

}

void lcdinit()

{

lcdwritecmd(0x38);

lcdwritecmd(0x0c);

lcdwritecmd(0x06);

lcdwritecmd(0x01);

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x;

else

address=0x40+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str)

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

