STC 单片机第二次作业

实验二：

（1）将 1602 显示模块，与 STC 单片机实验箱正确连接，并在 1602 上显示学号（在教材上 P365 页上有 1602 显示模块原理介绍和参考程序。

（2） 通过外部按键触发中断，实现学号在 1602 上的左移/右移

1、设计思路

该实验可以分解为两个部分：移位和方向。基本想法为：移位由主函数的无限循环实现，方向由外部按键中断确定。之所以不把移位放在中断中实现是因为，移位涉及到无限循环，在中断放入无限循环显然是不可取的。

2、所有代码

代码供有 3 个版本，以下为全部代码，后面将逐一分析。

```
#include "reg51.h"
#include "led1602.h"


// 用于 V2.0 版本
const char * StuNum = "2016014503";
// 用于 V2.1 版本
const char StuNum2[17] = {'2', '0', '1', '6', '0', '1', '4', '5', '0', '3', '\0', '\0', '\0', '\0', '\0', '\0', '\0'};

volatile unsigned char dir = 0;
void delay(unsigned int xms);
void EXINT0() interrupt 0
{
    dir = 1;
}
void EXINT1() interrupt 2
{
    dir = -1;
}

void main(void)
{
    char i = 0;
    /**********LCD Initial**********/
    P0M1=0;
  P0M0=0;
    P2M1=0;
```

```
P2M0=0;
lcdwait();
lcdinit();

/**********EXIT Initial**********/
IT0 = 1;
IT1 = 1;
EX0 = 1;
EX1 = 1;
EA = 1;

while (1)
{
	lcdwritecmd(0x01); // clear the screen

	/****************V1.0****************
	// 最初的代码，无法循环显示
	lcdshowstr(i,0,"2016014503");
	i += dir;
	if (i < 0)
		i = 6;
	else if (i > 6)
		i = 0;
	****************V1.0****************/

	/****************V2.0****************
	// 加入移位循环显示版本，  超出边界的部分将在另一侧显示
	if (i >= 0 && i <= 6)
		lcdshowstr(i, 0, StuNum);
	else if (i < 0)
	{
		lcdshowstr(0, 0, StuNum - i);
		lcdshowstr(16 + i, 0, StuNum);
		if (i < -9)
			i = 6;
	}
	else if (i > 6)
	{
		lcdshowstr(i, 0, StuNum);
		lcdshowstr(0, 0, StuNum + 16 - i);
		if (i > 15)
			i = 0;
	}
	i += dir;
```

```
            ****************V2.0****************/

            /****************V2.1****************/
            //  循环移位显示简化代码结构版本
            i %= 16;
            lcdshowstr(i, 0, StuNum2);
            lcdshowstr(0, 0, &StuNum2[16 - i]);
            i += dir;
            if (i < 0)
                  i = 15;
            /****************V2.1****************/

            delay(1000);
      }

}

void delay(unsigned int xms)    // xms 代表需要延时的毫秒数
{
      unsigned int x,y;
      for(x=xms;x>0;x--)
            for(y=110;y>0;y--);
}
```

3、代码分析

头文件：

包含 lcd1602 库函数和 51 单片机头文件

```
#include "reg51.h"
#include "lcd1602.h"
```

外部按键中断函数：

全局量 dir 为表示方向的有符号量，由外部按键中断改变，因此设为 volatile 型，防止编译器优化。初始化为 0，则表示没有移位，这样在程序刚初始化，没有按键中断时，屏幕上的学号是静止的。

```
volatile unsigned char dir = 0;
void EXINT0() interrupt 0
{
  dir = 1;
}
void EXINT1() interrupt 2
{
  dir = -1;
}
```

主函数初始化：

初始化部分，主要为寄存器配置，及 lcd 库函数调用。PxM0,PxM1 为两个 8 位寄存器，组合起来用于控制 Px 端口的方向和驱动模式。中断寄存器参考书 p83，图 5-17。

```
void main(void)
{
  char i = 0;
  /**********LCD Initial**********/
  P0M1=0; // P0, P2端口设为准双向端口模式
  P0M0=0;
  P2M1=0;
  P2M0=0;
  lcdwait();
  lcdinit();

  /**********EXIT Initial**********/
  IT0 = 1; // 按键均使用下降沿触发
  IT1 = 1;
  EX0 = 1; // 开按键中断
  EX1 = 1;
  EA = 1;  // 开总中断
```

移位实现：

移位在 while（1）无限循环中实现，在每次循环的开始，对 lcd 写入 0x01 命令，用于清屏

```
while (1)
{
  lcdwritecmd(0x01); // clear the screen
```

（1）V1.0 版本，无移位循环

```
/*****************V1.0*****************
// 最初的代码，无法循环显示
lcdshowstr(i,0,"2016014503");
i += dir; // 通过dir更新i的值，dir为正则向左移，反之右移
if (i < 0)
  i = 6;
else if (i > 6)
  i = 0;
*****************V1.0*****************/
```

其中 i 为有符号数，表示在屏幕的一行上学号开始输出的位置，由于 1602 每行有 16 个像素，而学号是 10 位，因此在 0 <= i <=6 时，可以完整的显示学号，当超出这个范围，学号显示就不完整了。当 i 超出这个范围，重新定义 i 值即可。比如当 i 加到 7 时，即学号右移了 7 位，学号超出屏幕范围，i 置 0，学号又回到最左边。当 i<0 时同理。

（2）V2.0 版本，包含移位循环，超出边界的部分将在另一侧显示

```
/*****************V2.0*****************
// 加入移位循环显示版本，  超出边界的部分将在另一侧显示
if (i >= 0 && i <= 6)
  lcdshowstr(i, 0, StuNum);
else if (i < 0)
{
  lcdshowstr(0, 0, StuNum - i);
  lcdshowstr(16 + i, 0, StuNum);
  if (i < -9)
    i = 6;
}
else if (i > 6)
{
  lcdshowstr(i, 0, StuNum);
  lcdshowstr(0, 0, StuNum + 16 - i);
  if (i > 15)
    i = 0;
}
i += dir;
*****************V2.0*****************/
```

在 V1.0 的基础上加入循环显示，效果如下图示意，例如当学号向右移动越界时，越界部分将在左侧显示，右移同理。

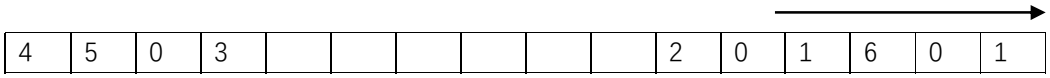

图 1　LCD 循环移位显示示意图

因为涉及到将学号分开打印，所以要将学号定义为字符串指针，方便学号定位。

```
// 用于v2.0版本
const char * StuNum = "2016014503";
```

为实现这个想法，首先对 i 的范围分类。0<=i<=6 为无越界正常情况，此时正常输出即可。当越界时，学号分两部分打印，分别在左边界，和右边界。两者特点不同：

（a）左边界打印，屏幕位置固定为（0，0），需要确定字符串起始位置，所以基本形式为

lcdshowstr(0, 0, StuNum + n);

n 的表达示由具体情况确定。

（b）右边界打印，字符串起始位置固定为第一位，即 StuNum（或&StuNum[0]），需要确定屏幕打印位置，基本形式为

lcdshowstr(n, 0, StuNum);

n 的表达示由具体情况确定。

当 i<0，表明向左越界，因为 i 为负，StuNum – i 指向的位置即左边界剩余学号开始的位置。右边界显示，屏幕起始位置为 16 + i。当 i < -9 表明整个学号都越出左边界，如图 2 所示，为便于理解，需将显示屏左侧补全一个学号的长度

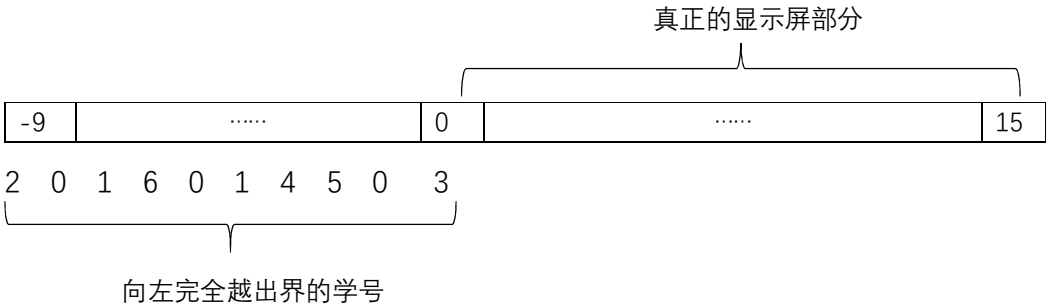

图 2 左越界示意图

此时整个字符串都位于右侧，所以将 i 置 6，i 落入正常区间

```
else if (i < 0)
{
  lcdshowstr(0, 0, StuNum - i);
  lcdshowstr(16 + i, 0, StuNum);
  if (i < -9)
    i = 6;
}
```

当 i>6，表明向右越界，左边界字符串起始位置为 StuNum + 16 – i，右边界屏幕位置为 i。当 i > 15，表明整个学号都越出右边界，此时整个字符串都位于左侧，所以将 i 置 0，i 落入正常区间。原理可参照图 2.

```
else if (i > 6)
{
  lcdshowstr(i, 0, StuNum);
  lcdshowstr(0, 0, StuNum + 16 - i);
  if (i > 15)
    i = 0;
}
```

（3）V2.1 版本，在 V2.0 基础上简化代码结构

```
/*****************V2.1****************/
// 循环移位显示简化代码结构版本
i %= 16;
lcdshowstr(i, 0, StuNum2);
lcdshowstr(0, 0, &StuNum2[16 - i]);
i += dir;
if (i < 0)
  i = 15;
/*****************V2.1****************/
```

这个版本基本想法是左右边界的打印同时进行，不再对 i 分类，只确定在 0~15 的范围，所以结构大大简化。实现的关键是对字符串 StuNum2 的定义

```
const char StuNum2[17] = {'2', '0', '1', '6', '0', '1', '4', '5', '0', '3',
                          '\0', '\0', '\0', '\0', '\0', '\0', '\0'};
```

相比 V2.0，这里字符串结尾多补了'\0'字符。在字符串中'\0'表示字符串的结束。而 LCD 库函数 lcdshowstr()也利用了这一点

```
void lcdshowstr(unsigned char x, unsigned char y,
                unsigned char *str)
{
  lcdsetcursor(x,y);
  while((*str)!='\0')
  {
     lcdwritedata(*str);
     str++;
  }
}
```

当检测到'\0'时，打印结束。如果字符串起始的字符就是'\0'，那么打印根本不会进行。

下面是原理，

这个方法实际是对显示屏 16 个像素做周期延拓，如图 3

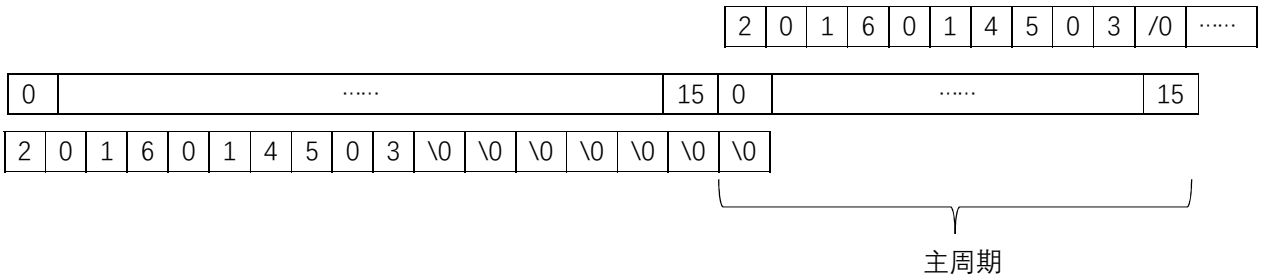

图 3 周期延拓示意图

每次移位时相当于一个无限长周期连续字符串在左右移位，当取其中 16 位观察时，它是循环移位的。这与圆周卷积的原理是一致的。对字符串结尾补'\0'确保当主周期字符串未越界时，上个周期（或下个周期）的字符串不会被打印。正常来说应将字符串补至 16 位，和屏幕同宽，但由于一个循环总是运行两次打印函数，为确保合理性，需将字符串补至 17 位，这样初始时刻，在主周期仍然是运行两次打印函数。