# 嵌入式手札七 LCD，键盘的应用

## 实验目的：

1、利用MCU中的sysTick模块用来计时；

2、通过UART模块实现输入发送系统时间至MCU中并且计时发送到LCD上显示

3、实验一：通过串口发送系统时间到MCU中，MCU计时并发送LCD动态显示；

4、实验二：如何通过试验箱的键盘输入6位数的时间数值，发送给g\_time[]数组中，开始计时送LCD显示

5、实验三：通过在C#窗口程序上设置虚拟键盘实现MCU实体键盘和虚拟键盘的同步

## 实验环境：

1）PC机一台，KL25开发板一片，导线若干

2) KDS开发环境，VS2017或更高平台

## 实验内容：

## 实验一：

### 1）设计串口接收模块UART

#### #1目的：通过设计串口接收模块来接受系统发送的时间，通过组帧函数进行判断从而将系统发送过来的时间存放到g\_time[ ]数组中，从而让sysTick模块开始计时发送到LCD中；

#### #2问题：

##### 问题1）如何正确将发送过来的时间存放到g\_time[ ]数组里

@方法：（这里用到的其实和上次定时器实验很接近，与上次相比，多作的事情就是将LCD组件利用起来每秒显示时间）

#1考虑到C#输入的数字都是字符型，每个数字字符都对应一个（码）值；buffer缓存中的数字部分每位字节运算都是用对应的值计算；但是发送给c#的时候，必须发送数字相应的字符码；通过减去字符码和实际数值的差值进行存放；

问题2）如何在组帧成功时同步让sysTick模块开始同步计时

@方法：设置全局变量flagTime标志，同志sysTick模块开始计时

@源码：

**void** **UART1\_IRQHandler**(**void**) {

uint\_8 ch;

uint\_8 flag;

**static** uint\_8 buffer[64];

DISABLE\_INTERRUPTS;

**if** (uart\_get\_re\_int(UART\_1)) {

ch = uart\_re1(UART\_1, &flag);

//flagInput = 1;

//用于接收输入到MCU的字符

//g\_LCDBuffer[31] = ch;

**if** (CreateFrame(ch, buffer) != 0) {

//组帧成功，更改标志

flagTime = 1;

**if** (buffer[1] == 8) { //C#输入的数字都是字符型，每个数字字符都对应一个ASCII（码）值

//LCD\_buffer[]更新时间，更新显示内容

uint\_8 g\_temp[32] = " ";

uint\_8 i = 0;

**for** (i = 0; i < 32; i++) {

g\_LCDBuffer[i] = g\_temp[i];

} //LCD缓冲区赋值

//buffer缓存中的数字部分每位字节运算都是用对应的ASCII码值计算

//但是发送给c#的时候，必须发送数字相应的字符ASCII码值；

//计算hh:mm:ss

uint\_8 g\_time0 = (buffer[2]);

g\_time0 -= 48;

g\_time0 \*= 10;

buffer[3] -= 48;

g\_time0 += (buffer[3]);

uint\_8 g\_time1 = (buffer[5]);

g\_time1 -= 48;

g\_time1 \*= 10;

buffer[6] -= 48;

g\_time1 += (buffer[6]);

uint\_8 g\_time2 = (buffer[8]);

g\_time2 -= 48;

g\_time2 \*= 10;

buffer[9] -= 48;

g\_time2 += (buffer[9]);

g\_time[0] = g\_time0;

g\_time[1] = g\_time1;

g\_time[2] = g\_time2;

}

}

}

**if** (flag) {

uart\_send1(UART\_1, ch);

}

ENABLE\_INTERRUPTS;

}

问题三：

### 2）设计sysTick模块

#### #目的：控制sysTick模块进行计时，改变g\_time[ ]的数值,并且动态更新时间到LCD

#### #问题：

##### 1）如何将MCU内部的g\_time[ ]数组内部的时间分别赋给LCD发送缓存g\_LCDBuffer[]中；

@方法：

通过将g\_time[ ]内部的三位时分秒的数值进行位拆分，拆分出时分秒每个数值的十位，个位；加入字符‘ ：’，组合成时间格式发送LCD发送缓存g\_LCDBuffer[]中

2）

@源码：

**if** (flagTime == 1) {

//静态变量,负责控制每次亮灭间隔的时间

**static** uint\_8 SysTickcount = 0;

SysTickcount++;

**if** (SysTickcount >= 100) {

SysTickcount = 0;

SecAdd1(g\_time); //时间加一

g\_LCDBuffer[24] = ((g\_time[0] / 10) + 48);

g\_LCDBuffer[25] = ((g\_time[0] % 10) + 48);

g\_LCDBuffer[26] = ':';

g\_LCDBuffer[27] = ((g\_time[1] / 10) + 48);

g\_LCDBuffer[28] = ((g\_time[1] % 10) + 48);

g\_LCDBuffer[29] = ':';

g\_LCDBuffer[30] = ((g\_time[2] / 10) + 48);

g\_LCDBuffer[31] = ((g\_time[2] % 10) + 48);

LCDShow(g\_LCDBuffer); //LCD显示

}

}

## 实验二：

### 1）设计sysTick模块

#### #目的：通过键盘按下通过试验箱的键盘输入6位数的时间数值，发送给g\_time[]数组中，开始计时送LCD显示

#### #问题：

##### 问题1）如何准确识别按下的六个时间数值后，开始计时

@方法：设计一个全局变量计时器flagInput=0，初始按键时钟计时标志flagTime=0,停止计时，通过按六次，每次将时间数值送往g\_time[]更改计时标志flagTime开启计时

@源码：

**if** (kb != 0xff) //有实际按键

{

//设计实验实验

flagTime = 0; //输出时间，暂停标志

**if** (countIn == 0 || countIn == 1 || countIn == 2 || countIn == 3

|| countIn == 4 || countIn == 5) {

countIn++; //数值加一

**switch** (countIn) {

**case** 0: {

g\_time0 = (KBDef(kb) - 48);

}

**case** 1: {

g\_time0 \*= 10;

g\_time0 += (KBDef(kb) - 48);

}

**case** 2: {

g\_time1 = (KBDef(kb) - 48);

}

**case** 3: {

g\_time1 \*= 10;

g\_time1 += (KBDef(kb) - 48);

}

**case** 4: {

g\_time2 = (KBDef(kb) - 48);

}

**case** 5: {

g\_time2 \*= 10;

g\_time2 += (KBDef(kb) - 48);

//赋值给时间数组

g\_time[0] = g\_time0;

g\_time[1] = g\_time1;

g\_time[2] = g\_time2;

//复位计数值，表示输入结束;

countIn = 0;

flagTime = 1; //显示时间标志赋值

}

}

}

}

## 实验三：

### 1）设计设计串口接收模块UART

#### #目的：接收C#虚拟键盘发送的数据，同步显示在LCD上

#### #问题

##### 问题1）：如何识别虚拟键盘发送的按键

@方法：通过组帧的方式，C#按下一个键，发送一个帧；通过条件判断语句筛选出发送的组帧对应的按键，将按键值赋给LCD输出缓存g\_LCDBuffer[]，调用显示函数进行显示；

PS:帧尾用到了C,所以在发送C按键对应的值时，用小写的c进行组帧发送

@源码：

**if** (CreateFrame(ch, buffer) != 0) {

//组帧成功，更改标志

**if** (buffer[1] == 1) {

//更改buffer显示内容

uint\_8 g\_temp[32] = " ";

uint\_8 i = 0;

**for** (i = 0; i < 32; i++) {

g\_LCDBuffer[i] = g\_temp[i];

}

**switch** (buffer[2]) {

**case** '1': {

g\_LCDBuffer[31] = '1';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '2': {

g\_LCDBuffer[31] = '2';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '3': {

g\_LCDBuffer[31] = '3';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '4': {

g\_LCDBuffer[31] = '4';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '5': {

g\_LCDBuffer[31] = '5';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '6': {

g\_LCDBuffer[31] = '6';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '7': {

g\_LCDBuffer[31] = '7';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '8': {

g\_LCDBuffer[31] = '8';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** '9': {

g\_LCDBuffer[31] = '9';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'A': {

g\_LCDBuffer[31] = 'A';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'B': {

g\_LCDBuffer[31] = 'B';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'c': {

//这里有个需要注意的地方，因为，组帧最后用的是C，所以按键发送的内容需要用c来判断

g\_LCDBuffer[31] = 'C';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'D': {

g\_LCDBuffer[31] = 'D';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'E': {

g\_LCDBuffer[31] = 'E';

LCDShow(g\_LCDBuffer);

**break**;

}

**case** 'F': {

g\_LCDBuffer[31] = 'F';

LCDShow(g\_LCDBuffer);

**break**;

}

**default**: {

}

}

}

}

}

### 2）设计sysTick模块

#### #目的：通过追踪试验箱键盘按下的键，发送到LCD上显示并发送给C#串口助手进行虚拟键盘同步显示

#### #问题：

##### 问题1）：如何通过sysTick模块发送给C#;且让C#识别出发送的按键值

@方法:利用组帧的思想，将每一个按键的值组成 【“\*”+”键值”】的形式，从而在c#端让程序自动识别

@源码：

**if** (kb != 0xff) //有实际按键

{

//进阶性实验

uint\_8 g\_temp[32] = " ";

uint\_8 i = 0;

**for** (i = 0; i < 32; i++) {

g\_LCDBuffer[i] = g\_temp[i];

} //LCD缓冲区赋值

uint\_8 pressKey = KBDef(kb);

**switch** (pressKey) {

**case** '1': {

g\_LCDBuffer[31] = '1';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '1');

**break**;

}

**case** '2': {

g\_LCDBuffer[31] = '2';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '2');

**break**;

}

**case** '3': {

g\_LCDBuffer[31] = '3';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '3');

**break**;

}

**case** '4': {

g\_LCDBuffer[31] = '4';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '4');

**break**;

}

**case** '5': {

g\_LCDBuffer[31] = '5';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '5');

**break**;

}

**case** '6': {

g\_LCDBuffer[31] = '6';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '6');

**break**;

}

**case** '7': {

g\_LCDBuffer[31] = '7';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '7');

**break**;

}

**case** '8': {

g\_LCDBuffer[31] = '8';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '8');

**break**;

}

**case** '9': {

g\_LCDBuffer[31] = '9';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, '9');

**break**;

}

**case** 'A': {

g\_LCDBuffer[31] = 'A';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'A');

**break**;

}

**case** 'B': {

g\_LCDBuffer[31] = 'B';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'B');

**break**;

}

**case** 'C': {

//这里有个需要注意的地方，因为，组帧最后用的是C，所以按键发送的内容需要用c来判断

g\_LCDBuffer[31] = 'C';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'C');

**break**;

}

**case** 'D': {

g\_LCDBuffer[31] = 'D';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'D');

**break**;

}

**case** 'E': {

g\_LCDBuffer[31] = 'E';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'E');

**break**;

}

**case** 'F': {

g\_LCDBuffer[31] = 'F';

LCDShow(g\_LCDBuffer);

uart\_send1(UART\_1, '\*');

uart\_send1(UART\_1, 'F');

**break**;

}

**default**: {

**break**;

}

}

}

### 3）进行C#窗口编程

#### #目的：实现虚拟键盘和实体键盘同步显示

#### #方法：

##### 1）通过在C#的串口接收程序中定义switch语句，于每个MCU发送的str 进行判断；根据需要完成相关操作；

@操作：

#1：在设计见面设计如下的用按钮组成的键盘（只有串口成功打开时，才能显示）



#2、对于实体按键发送的帧只要识别就动态控制对那个按键进行状态切换以达到按压的效果

#3：对于从虚拟键盘按键的每个按键编写对应的click（）方法，按压后发送帧，按钮使能关闭；对从MCU返回帧进行识别后，识别成功则按钮是能开启；达到按键按压的效果；

；

#### #源码：

##### 1）按键按压（篇幅所限，只列出样例）按键0的对应key0\_Click（）方法

private void key0\_Click(object sender, EventArgs e)

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P10C");//发送消息；

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.key0.Enabled = false;

}

##### 2）对按下虚拟键盘或实体键盘的返回帧的条件筛选

private void SCIPort\_DataReceived(object sender,

System.IO.Ports.SerialDataReceivedEventArgs e)

{

String str = String.Empty;

bool Flag;//标记串口接收数据是否成功

int len;//标记接收的数据的长度

byte[] ch2 = new byte[2];

//ComDevice.Encoding = System.Text.Encoding.GetEncoding("GB2312");

//调用串口接收函数,并返回结果

Flag = sci.SCIReceiveData(ref PublicVar.g\_ReceiveByteArray);

if (Flag == true)//串口接收数据成功

{

len = PublicVar.g\_ReceiveByteArray.Length;

//对于字符串形式,考虑到可能有汉字,

//直接调用系统定义的函数,处理整个字符串

str = Encoding.GetEncoding("GB2312").GetString(PublicVar.g\_ReceiveByteArray);

//

SCIUpdateRevtxtbox(TbShowString, str);

switch (str)

{

//虚拟键盘返回帧判断

case "P10C":

{

this.key1.Enabled = true;

break;

}

.

.

.

//实体键盘返回帧判断

case "\*1":

{

this.keyF.Enabled = false;

this.key1.Enabled = true;

break;

}

.

.

.

default: { break; }

}

.

.

.

## 实验总结：

1、这次试验真的花了很多的时间，我发现，越往后越是对编程能力的考验。对整个程序框架的大致原理理解不难，难在灵活地通过编程控制这些组件，来达到自己想要的目的，编程技术不到家是很致命的。

2、老师课上说过要能够扛得住挫折，让我体会很深，越深入的学习越有难度，不可能像起初入门那样一帆风顺，肯定会有很多很多的困难，不能灰心丧气，要怀着兴奋的态度去挑战问题，解决问题，自己才能有所成长。

3、学习的节奏一定要掌控好，加快学习效率。抓住重点学习；