矩阵LED实验

1953729 吴浩泽

## 实验目的

掌握S3C2410X芯片的I/O控制寄存器的配置；

了解ARM芯片中复用I/O口的使用方法；

掌握矩阵LED的应用原理。

## 实验设备

硬件：Embest EduKit-IV平台，ULINK2仿真器套件，PC机。

软件：μVision IDE for ARM集成开发环境，Windows 98/2000/NT/XP。

## 实验内容

编写S3C2410X处理器的点阵屏驱动程序；

编写S3C2410X处理器的点阵屏应用程序。

## 实验原理

**1. 硬件原理**

本实验平台设计了一个16×16的点阵屏。点阵屏由发光LED矩阵块组成。16×16点阵屏即屏上有16×16个LED发光二极管，每个发光二极管可理解为一个像素点，它们被按着行与列的形式整齐地排列，通过控制每个LED（像素点）的亮灭，点阵屏可与计算机同步显示汉字、英文文本和图形。本实验平台的点阵屏电路如下图：



点阵屏的行扫描信号



点阵屏的列扫描信号



点阵屏的保护电阻

****

点阵屏的行驱动信号

****

点阵屏的接口电路

本实验平台使用的16×16点阵屏上，每一行16个LED，它们采用共阳极的接法；每一列16个LED，它们采用共阴极的接法。如上图8-1-5中，QL1～QL16是点阵屏的行驱动信号，每一个信号控制一行；LR1～LR16是点阵屏的列驱动信号，每一个信号控制一列。故通过利用相应行线输出高电平，相应列线输出低电平，就可以点亮点阵屏上相应的LED。如果按着一定的控制或扫描方法，就可以实现汉字、图形的显示。

为了能够稳定地控制点阵屏的显示，本实验平台采用了CD4094作为行线和列线扫描信号的控制芯片。CD4094芯片简单来说就是一串入并出的功能，将CPU的串行数据转化为并行数据输出。

如上图8-1-1与图8-1-2，行（列）扫描信号分别采用了两片CD4094级连的方式来构成，第一片的数据溢出信号LQS连接到第二片的串行数据输入口。在图8-1-1中，当CPU通过DATA引脚串行输入16位的行扫描信号时，第一片CD4094（标号U1101）存储先输入的8位数据，后输入的8位数据通过引脚LQS溢出，然后输入并存储到第二片CD4094（标号U1102）。CLK信号是时钟信号，当时钟信号为高电平时，CD4094允许串行输入的数据发生变化。LL1～LL8是并行输出信号。LOE为全局输出使能信号，当LOE输入高电平时，CD4094上锁存的并行数据全部输出。列线扫描信号也是同样的方式，如图8-1-2。

在图8-1-3与图8-1-4中，行线扫描信号LL1～LL16分别外接电阻（限流）后接至NPN型三极管，最后引出行线驱动信号QL1～QL16。三极管提高了行线输出信号的驱动能力。在图8-1-4中，如果RQx（x为1～16的整数）输出低电平，那么相应的三极管导通，行驱动信号QLx（x为1～16的整数）将输出高电平。

**2. 软件设计**

本实验要求编写点阵屏驱动程序以及点阵屏应用程序。驱动程序要求实现字符、图形显示；应用程序可以调用驱动接口，在点阵屏中显示字符与图形。

要在点阵屏上显示字符、图形，只需要按照字符/图形的编码，点亮矩阵屏上相应的LED即可。在本实验平台上，矩阵屏为16×16点阵，每屏显示两个字符，只需要把16×8点阵的字库码送到可容纳两个字符字库码（2×16字节）的显示缓冲区，然后根据显示缓冲区的内容控制点阵屏上LED灯的亮灭。字库码表明了显示某个字符时点阵屏上的LED的亮灭布局，每一个LED灯的状态用一位二进制表示，1表示该LED亮，0表示该LED灭，对于16×8点阵的字库码，每个字符的字库码所占用的存储容量为16\*8/8 = 16 Byte。

在点阵屏显示驱动程序中，开辟一个显示缓冲区用来接受字库码，（字库码是由应用程序传递过来的），然后采用动态扫描的方法来实现字符、图形的显示。每次扫描一行，这样依次循环扫描16行，经过一定的延时处理，就可以实现字符、图片的静态显示。

为了实现字符和图片的动态流动显示，还需要一个大小为可容纳3个字符字库码（3×16字节）的缓冲区，该缓冲区中每次保存3个字符的字库码。在显示过程中，为了达到流动效果，每次将该缓冲区中的前两个字符字库码的各字节左移i位与其后第16字节右移8-i位相或的结果复制到显示缓冲区中显示，直到第一个字符完全移出、第三个字符完全移入时，更新缓冲区。如此循环即可实现字符的流动显示。

在显示时，根据字库码，对点阵屏逐行动态扫描。扫描时，行线的数据与列线的数据都通过CD4094锁存起来，然后经过片选使能再同时输出。

在本实验中，使用到的通用端口有GPIOB4/5，GPIOC0/7，GPIOD10，GPIOG8，因此对它们的工作模式应配置为输出模式。

## 实验步骤

**1. 准备实验环境**

使用ULINK2仿真器连接Embest EduKit-IV实验平台的主板JTAG接口；使用Embest EduKit-IV实验平台附带的交叉串口线，连接实验平台主板上的COM2和PC机的串口（一般PC只有一个串口，如果有多个请自行选择，笔记本没有串口设备的可购买USB转串口适配器扩充）；使用Embest EduKit-IV实验平台附带的电源适配器，连接实验平台主板上的电源接口。

**2. 串口接收设置**

在PC机上运行windows自带的超级终端串口通信程序，或者使用实验平台附带光盘内设置好了的超级终端，设置超级终端：波特率115200、1位停止位、无校验位、无硬件流控制，或者使用其它串口通信程序。（注：超级终端串口的选择根据用户的PC串口硬件不同，请自行选择，如果PC机只有一个串口，一般是COM1）

**3. 打开实验例程**

1）拷贝实验平台附带光盘DISK3\_S3C2410\03-Codes\01-MDK\Mini2410-IV文件夹到MDK的安装路径：Keil\ARM\Boards\Embest\（如果本实验之前已经拷贝，可以跳过这一步）。（注：用户也可拷贝工程到任意目录，本实验为了便于教学，故统一实验路径）；

2）运行μVision IDE for ARM软件，点击菜单栏“Project”，选择“Open Project…”，在弹出的对话框选择实验例程目录8.1\_DotLed\_Test子目录下的DotLed\_Test.Uv2工程。

3）默认打开的工程在源码编辑窗口会显示实验例程的说明文件readme.txt，详细阅读并理解实验内容。

4）工程提供了两种运行方式：一是下载到SDRAM中调试运行，二是固化到Nor Flash中运行。用户可以在工具栏Select Target下拉框中选择在RAM中调试运行还是固化Flash中运行。如下图所示：



选择运行方式

下面实验将介绍下载到SDRAM中调试运行，所以我们在Select Target下拉框中选择DotLed\_Test IN RAM。

5）接下来开始编译链接工程，在菜单栏“Projiet”选择“Build target”或者“Rebuild all target files”编译整个工程，用户也可以在工具栏单击“”或者“”进行编译。

6）编译完成后，在输出窗口可以看到编译提示信息，比如“".\SDRAM\DotLed\_Test.axf" - 0 Error(s), 1 Warning(s).”，如果显示“0 Error(s)”即表示编译成功。

7）拨动实验平台电源开关，给实验平台上电，单击菜单栏Debug->Start/Stop Debug Session项将编译出来的映像文件下载到SDRAM中，或者单击工具栏“”按钮来下载。

8）下载完成后，单击菜单栏Debug->Run项运行程序，或者单击工具栏“”按钮来全速运行程序。用户也可以使用进行单步调试程序。

9）全速运行后，用户可以在超级终端看到程序运行的信息“Please Look At The 16X16 LEDS”。

10）用户可以Stop程序运行，使用μVision IDE for ARM的一些调试窗口跟踪查看程序运行的信息。

注：如果在第4）步用户选择在Flash中运行，则编译链接成功后，单击菜单栏Flash->Download项将程序固化到NorFlash中，或者单击工具栏按钮“”固化程序，从实验平台的主板拔出JTAG线，给实验平台重新上电，程序将自动运行。

**4. 观察实验结果**

在执行到第8）步时，可以看到超级终端上输出如下字符。

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\* 英蓓特EduKit系列嵌入式教学系统平台 \*\*  \*\* Embest EduKit Series Embedded Teaching Platform \*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Please Look At The 16X16 LEDS |

观察矩阵LED，可以看到字符串“WELCOME TO EMBEST EDUKIT IV ^\_^”将显示在矩阵LED上，并随时间流逝而从右向左不断循环流动显示。

## 实验参考程序

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: write\_L4094  \* func: write list data  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void write\_L4094(UINT16T data)  {  UINT8T i;  Lstr\_H;  for(i=0;i<16;i++)  {  CLK\_L;  if(data & 0x01==1)  Ldata\_H;  else  Ldata\_L;  data = data >> 1;  CLK\_H;  }  Lstr\_L;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: write\_R4094  \* func: write row data  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void write\_R4094(UINT8T data)  {  UINT8T i;  Rstr\_H;  for(i = 0; i < 8; i++)  {  CLK\_L;  if(data & 0x01==1)  Rdata\_H;  else  Rdata\_L;  data = data >> 1;  CLK\_H;  }  Rstr\_L;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* name: led\_char\_disp  \* func: led char display  \* para: none  \* ret: none  \* modify:  \* comment:  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void led\_char\_disp(void)  {  UINT8T i=0;  UINT8T j=2,k=0;  UINT16T x;  for(i = 0; i < 32; i++)  dispram[i] = ~l\_display\_array[i];  i=0;  while(i < 16)  {  Rstr\_L; // Enable RSTR  Lstr\_L; // Enable LSTR  DisEn\_Dataout; // lock the data  do { // Write the row data  j--;  write\_R4094(dispram[i \* 2 + j]);  }while(j);  x=~(0x8000 >> i); // Write one list data  write\_L4094(x);  En\_Dataout; // Output the data  for(k = 0;k < 250; k++);  i++;  j = 2; }} |

## 读书笔记

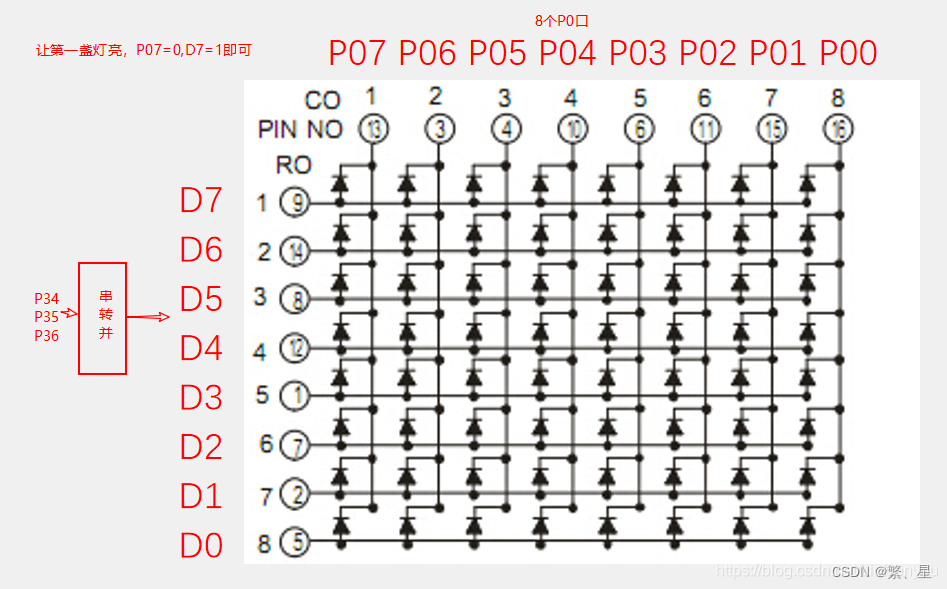
LED点阵屏由若干个独立的LED组成，LED以矩阵的形式排列，以灯珠亮灭显示文字、图片、视频等。LED点阵屏广泛应用于各种公共场合，如汽车报站器、广告屏以及公告牌等。

LED点阵屏分类

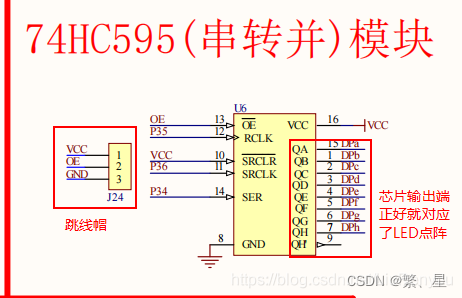
按颜色：单色、双色、全彩。

按像素：8\*8、16\*16

其内部连接与数码管相似，单色的可以共阴极或共阳极



其运用需要借助74HC595移位寄存器



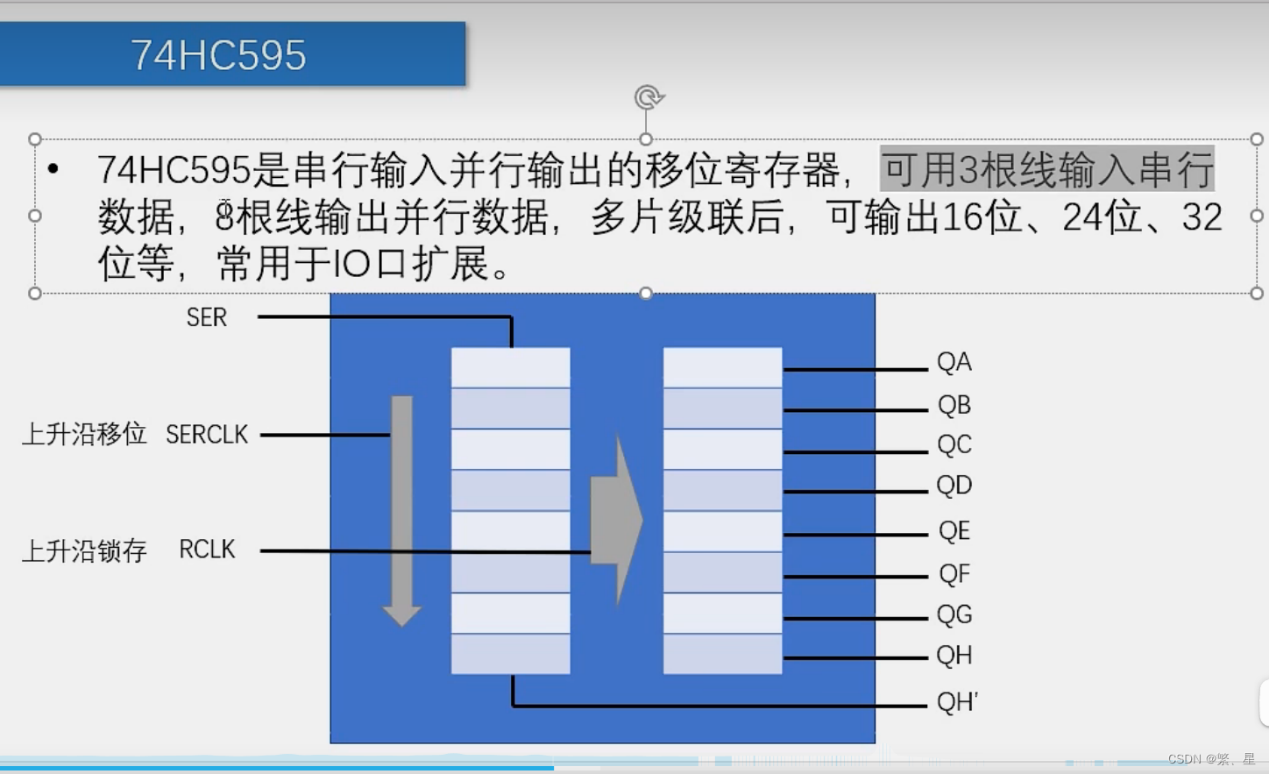
输出端：DPa-DPh正好对应LED点阵，QH’是用来做多个led点阵级联的。

输入端：

P34-SER：代表串行输入，我们串行输入就是把数据放到这。

P35-SCK ：上升沿移位，即RCK从0-1电位变化会让数据往下移一位

P36-RCK：上升沿锁存，搬运工，即SCK从0-1电位变化会把数据搬运到芯片输出端



接下来利用LED点阵屏显示静态笑脸  
#include <REGX52.H>

#include "Delay.h"

sbit RCK=P3^5; //定义RCLK

sbit SCK=P3^6; //定义SRCLK

sbit SER=P3^4; //定义SER

#define LED\_JZjk P0 //代替P0

void \_74HC595\_WriteBety(unsigned char Byte)

{

unsigned char i;

for(i=0;i<8;i++)

{

SER=Byte&(0x80>>i);

SCK=1;

SCK=0;

}

RCK=1;

RCK=0;

}

void LED\_JZ(unsigned char Lie,Data)

{

\_74HC595\_WriteBety(Data);

LED\_JZjk=~(0x80>>Lie);

Delay(1);

LED\_JZjk=0xFF;

}

void main()

{

SCK=0;

RCK=0;

while(1)

{

LED\_JZ(0,0x3C);

LED\_JZ(1,0x42);

LED\_JZ(2,0xA9);

LED\_JZ(3,0x85);

LED\_JZ(4,0x85);

LED\_JZ(5,0xA9);

LED\_JZ(6,0x42);

LED\_JZ(7,0x3C);

}

}

