矩阵扫描键盘实验

1953729 吴浩泽

## 实验目的

掌握四型实验箱扩展输入输出端口的原理和应用。

掌握矩阵式键盘原理及编程方法。

## 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

学习四型实验箱扩展输入输出IO口硬件原理。

学习键盘扫描原理并编程实现。

## 实验原理

四型实验箱采用了扩展输入输出IO口以节省资源。该平台不仅支持键盘模块还支持矩阵键盘。首先，让我们一起来了解一下什么是矩阵键盘。



矩阵键盘中行线与列线交叉但是不相连通，而是由一个按键相连接。在硬件设计上，相应的列线通过电阻接到正电源上，此时列线就全为高电平；行线输出低电平；如果没有键按下，则所有的列线都为高电平；一旦有键按下，那么相应的行线与列线就会导通，有键按下的列线就会由高电平转变为低电平，从而可以检测到有键按下并最终通过一定的方法获得键值。

**硬件原理图**



扩展IO口

在上图中，标号为U32的芯片Y4VHC541DT控制了6根列线KR1～KR6；标号为U34的芯片74VHC573DT控制了6根行线KL1～KL6。列线KR1～KR6可以通过访问8位只读寄存器0x21080000来获得其电平状态；行线KL1～KL6可以通过访问8位只写寄存器0x21140000来控制其电平状态。6根列线KR1～KR6的状态是由硬件决定的，初始态全为高电平，行线状态可以由软件控制。在软件上将所有行线置低电平，那么只要有键按下，相应的列线就会变为低电平。

在上图中，所有的列线都通过二极管接到了引脚IRQKEY上，只要有键按下，那么相应的列线就会变为低电平，二极管导通，从而引脚IRQKEY就会变为低电平。IRQKEY引脚的信号是用于键盘中断事件的。在硬件上，IRQKEY引脚通过主板的中断扩展最终连通到了CPU的中断脚。这样，只要有键按下，如果软件允许，就会触发键盘中断事件。当然也可用查询的方式实现。

前面已经说明，对行线与列线的控制和访问是通过读或写相应的寄存器来进行的。在上图中，芯片Y4VHC541DT由寄存器0x21080000控制；芯片74VHC573DT由寄存器0x21140000控制。从上图6-1-2标号为U32的图可以看出，寄存器0x21080000的D1～D6位（图中为标为A2～A7）连接着列线KR1～KR6；从上图标号为U34的图可以看出，寄存器0x21140000的D2～D7位（图中标为Q2～Q7）控制着行线KL1～KL6。

**矩阵键盘接口电路**

本教学实验平台的电路中，EduKit-IV设计了1个支持6×6矩阵键盘的接口

****

矩阵键盘接口电路

## 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。最后请将随实验平台一起附带的键盘插在键盘接口上，注意方向和位置，具体请查看四型实验箱硬件手册的键盘接口部分。

**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6.1\_Keypad\_Test子目录下的Keypad\_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Keypad\_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\Keypad\_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，此时，用户可单击键盘上的按键，会在超级终端上显示所按下的键值。

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 press some keys on the keypad

此时用户可在键盘上按下任意键，会在超级终端上显示所按下的键，如：

You have pressed key < 0 >

You have pressed key < 1 >

You have pressed key < 2 >

You have pressed key < 3 >

You have pressed key < 4 >

You have pressed key < 5 >

You have pressed key < 6 >

You have pressed key < 7 >

You have pressed key < 8 >

You have pressed key < 9 >

You have pressed key < A >

You have pressed key < B >

You have pressed key < C >

You have pressed key < D >

You have pressed key < E >

You have pressed key < F >

You have pressed key < + >

You have pressed key < - >

You have pressed key < \* >

You have pressed key < FUN >

## 实验参考程序

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File： keypad\_test.c

\* Author: embest

\* Desc： Keypad\_Test

\* History:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*------------------------------------------------------------------------------------------\*/

/\* include files \*/

/\*------------------------------------------------------------------------------------------\*/

#include "2410lib.h"

/\*------------------------------------------------------------------------------------------\*/

/\* constant define \*/

/\*------------------------------------------------------------------------------------------\*/

#define KPLAddr (\*(volatile unsigned char\*)0x21140000) // Keypad list io address

#define KPRAddr (\*(volatile unsigned char\*)0x21080000) // Keypad row io address

/\*------------------------------------------------------------------------------------------\*/

/\* global variables \*/

/\*------------------------------------------------------------------------------------------\*/

UINT16T KeyNo ; // Key No.

UINT8T KPRData; // Read Data from the row of keypad

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* name: keyscan

\* func: scan the keypad

\* para: none

\* ret: 1--one key have been pressed 0--no key was pressed

\* modify:

\* comment:

UINT8T keyscan(void)

{

UINT8T i, j;

// Scan the keypad

for(i = 0; i < 4; i++)

{

KPLAddr = ~(0x4 << i); // Give one list a low lever

for(j = 0; j < 5; j++) // Check the row ,if have low lever ,that is one key was pressed.

{

KPRData = KPRAddr;

if(!(KPRData & (0x1 << j)))

{

delay(10);

KPRData = KPRAddr;

if(!(KPRData & (0x1 << j))) // Avoid dithering

{

do{

KPRData = KPRAddr;

}while(!(KPRData & (0x1 << j))); // Wait the key to be released

KeyNo = i << 8 | j;

return 1; // One key was pressed

}

}

}

}

return 0; // No key was pressed

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* name: keypad\_test

\* func: scan the keypad and display the pressed key

\* para: none

\* ret: none

\* modify:

\* comment:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void keypad\_test(void)

{

UINT8T keychar;

if(keyscan())

{

switch(KeyNo)

{

case 0x0000: keychar = 'U'; break; // FUN key

case 0x0001: keychar = 'D'; break;

case 0x0002: keychar = '-'; break;

case 0x0003: keychar = '0'; break;

case 0x0004: keychar = '+'; break;

case 0x0100: keychar = '\*'; break;

case 0x0101: keychar = 'C'; break;

case 0x0102: keychar = '3'; break;

case 0x0103: keychar = '2'; break;

case 0x0104: keychar = '1'; break;

case 0x0200: keychar = 'F'; break;

case 0x0201: keychar = 'B'; break;

case 0x0202: keychar = '6'; break;

case 0x0203: keychar = '5'; break;

case 0x0204: keychar = '4'; break;

case 0x0300: keychar = 'E'; break;

case 0x0301: keychar = 'A'; break;

case 0x0302: keychar = '9'; break;

case 0x0303: keychar = '8'; break;

case 0x0304: keychar = '7'; break;

default: break;

}

if( keychar == 'U')

uart\_printf(" You have pressed key < FUN >\n");

else

uart\_printf(" You have pressed key < %c >\n", keychar);

}

}

## 读书笔记

微机原理与接口实验箱系统组成：

·硬件：

(1)微机系统扩展：8088CPU，64K监控程序存储器，64K用户程序和数据存储器，64K实验程序存储器，开放式4\*6键盘电路，开放式6位动态数码管显示电路，配有串口或USB通信方式，出厂配置为串口。

(2)输入/输出：128\*64LCD图文电路(或字符式1602LCD)，16\*16LED点阵电路，开关逻辑电平输入/输出，8MHZ时钟与分频，单脉冲发生器。

(3)常用接口芯片：8250、8251、8253、8255、8279、0809、0832、74LS273、74LS244、8237、8259、RS232等；

(4)控制对象：小型直流电机、步进电机、温度、压力、继电器、电子音响喇叭等；

·软件：

系统配置Win98/2000/xp等操作平台的微机仿真调试软件，支持汇编、C语言编译、调试。

·电源：

系统内置高性能直流稳压电源。

·机箱：

系统配置轻便铝合金箱子作为实验仪机箱。

三、微机原理与接口实验箱主要特点

(1)二种工作方式：一是联PC机运行，在与上位软件联机的状态下，实现各种调试和行运的操作；二是脱机运行，实验系统自带键盘、显示电路和监控管理程序，在无PC机的情况下，同样可以通过内存读写、单步、断点、全速等方法调试实验程序。

(3)实验开放性：实验电路单元尽可能独立开放,如开放式键盘,开放式显示器,开放式串口等,为适应多种方式实验提供可能。

(4)二次开发：主机板留有系统总线接口，可开发用户系统。

第一行的行控制线接到p17，第二行的行控制线接到p16，第三行的行控制线接到p15，第4行的行控制线接到p14

第一列的列控制线接到p13，第二列的列控制线接到p12，第三列的列控制线接到p11，第四列的列控制线接到p10

矩阵键盘的原理和独立按键类似，另外我们可以把矩阵键盘的任意一行或一列作为一个独立键盘使用，假如我们把第一行作为独立键盘，那么我们只需要让P17输出高电平，其余7个io口输出低电平即可，假如我们按下了s1，那么p13的电平就会被拉低，变为低电平，所以我们可以通过查找低4位里哪一位为低电平就可以知道哪个按键按下了。

下面来说说矩阵按键扫描原理（即当我们按下一个矩阵键盘的按键时，如何获取按键的位置）

方法有2种，一种是逐行扫描，一种是行列扫描.接下来就主要讲讲行列扫描.

行列扫描的话，就是一开始让p1口高4位输出高电平，低4位输出低电平，若这4行按键里，有按键按下了，那么那一行按键对应的io的电平就会被拉低，我们就可以知道按键的行坐标.获取按键列坐标的方法也是类似的，就是一开始让p1口高4位输出低电平，低4位输出高电平，若这4列按键里，有按键按下了，那么那一列按键对应的io的电平就会被拉低，我们就可以知道按键的列坐标，获得了行坐标x，列坐标y后，4\*（x-1）+y就是按键的编号.

两端都接在IO端口，逐行扫秒，逐列读取状态（扫描哪一行，哪一行就是低电平）

注意，定义C1的P4^4时，加上str P4 =0xc0(此时对应的头文件为inclued "reg52.h"),避免定义失效，因为再reg52.h对应的头文件中没有这一块寄存器的定义，所以要注意。因此，我们可以得到矩阵键盘的基本扫描步骤：

<1> R1输出点电平，R2、R3、R4输出高电平，逐个读取判断列信号，如果都为高电平则R1行上没有按键按下。

<2> R2输出点电平，R1、R3、R4输出高电平，逐个读取判断列信号。

<3> R3输出点电平，R1、R2、R4输出高电平，发现C2列信号为低电平，那么可以判断得R3行的C2列的按键有按下动作。

<4> R4输出点电平，R1、R3、R4输出高电平，逐个读取判断列信号。

如此循环往复，扫描的按键的状态。

我们知道有按键按下动作，那么又怎么知道是哪一个按键按下呢？这时，我们最好定义一个键值全局变量，给矩阵行列上的每一个的按键编一个唯一的号码。当扫描的某一行某一列的按键动作后，把对应的编号复制给键值变量，这样我们判断这个键值，就知道是那个按键有触发动作了。

代码如下：

#include "reg52.h"

sfr P4 = 0xc0;

sbit R1 = P3^0;

sbit R2 = P3^1;

sbit R3 = P3^2;

sbit R4 = P3^3;

sbit C4 = P3^4;

sbit C3 = P3^5;

sbit C2 = P4^2;

sbit C1 = P4^4;

unsigned char code SMG\_duanma[18]=

{0xc0,0xf9,0xa4,0xb0,0x99,0x82,0xf8,

0x80,0x90,0x88,0x80,0xc6,0xc0,0x86,0x8e,

0xbf,0x7f};

void SelectHC573(unsigned char channel1)

{

switch(channel1)

{

case 4:

P2 = (P2 & 0x1f| 0x80);

break;

case 5:

P2 = (P2 & 0x1f| 0xa0);

break;

case 6:

P2 = (P2 & 0x1f| 0xc0);

break;

case 7:

P2 = (P2 & 0x1f| 0xe0);

break;

}

}

void DisplayKeyNum(unsigned char value)

{

SelectHC573(6);

P0 = 0x01;

SelectHC573(7);

P0 = value;

}

unsigned char key\_num;

void ScanKeyMulti()

{

R1 = 0;

R2 = R3 = R4 = 1;

C1 = C2 = C3 =C4 =1;

if(C1 == 0)

{

while(C1 == 0);

key\_num = 0;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C2 == 0)

{

while(C2 == 0);

key\_num = 1;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C3 == 0)

{

while(C3 == 0);

key\_num = 1;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C4 == 0)

{

while(C4 == 0);

key\_num = 1;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

R2 = 0;

R1 = R3 = R4 = 1;

C1 = C2 = C3 =C4 =1;

if(C1 == 0)

{

while(C1 == 0);

key\_num = 4;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C2 == 0)

{

while(C2 == 0);

key\_num = 5;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C3 == 0)

{

while(C3 == 0);

key\_num = 6;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C4 == 0)

{

while(C4 == 0);

key\_num = 7;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

R3 = 0;

R1 = R3 = R4 = 1;

C1 = C2 = C3 =C4 =1;

if(C1 == 0)

{

while(C1 == 0);

key\_num = 8;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C2 == 0)

{

while(C2 == 0);

key\_num = 9;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C3 == 0)

{

while(C3 == 0);

key\_num = 10;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C4 == 0)

{

while(C4 == 0);

key\_num = 11;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

R4 = 0;

R2 = R3 = R4 = 1;

C1 = C2 = C3 =C4 =1;

if(C1 == 0)

{

while(C1 == 0);

key\_num = 12;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C2 == 0)

{

while(C2 == 0);

key\_num = 13;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C3 == 0)

{

while(C3 == 0);

key\_num = 14;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

else if(C4 == 0)

{

while(C4 == 0);

key\_num = 15;

DisplayKeyNum(SMG\_duanma[key\_num]);

}

}

void main()

{

while(1)

{

ScanKeyMulti();

}

}