实验三:GPIO中断/定时器实验

1．实验目的

* 掌握GPIO中断工作模式设置。
* 学习中断服务函数的编写。
* 掌握定时器外设的操作原理和编程。

2．实验设备

* 硬件：PC机一台

H0 ARM实验板一套

* 软件：WindowsXP系统，Keil uVision 4.x集成开发环境

3．实验内容

（1）编程将GPIO设置为低电平触发外部中断；然后等待中断事件。中断服务程序控制LED指示灯闪烁，给出反馈。使用Keil uVision调试运行程序。

（2）编程使用LPC1100的定时器匹配功能和外部匹配输出功能，用外部匹配输出引脚控制LED指示灯闪烁。

4．实验预习要求

(1)复习LPC1100中断工作原理和编程方法；

(2)复习LPC1100定时器工作结构、原理和编程方法。

5．实验步骤

（一）GPIO中断实验：

(1)开发板按键连接到LPC1114的P0\_1引脚，见图3-1，本实验中该引脚被配置成GPIO中断输入功能。

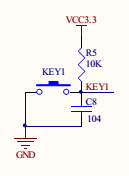
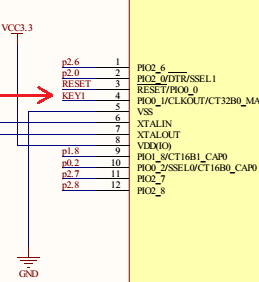
 

图3-1 按键输入

(2)启动Keil uVision，新建一个工程ex03-1。建立C文件ex03-1.c，编写实验程序，然后添加到工程中。设置工程调试选项。具体步骤参考实验二。

参考程序见程序清单3.1。

(4)实验程序分为三个部分：

a）设置中断向量表，这部分使用系统提供的startup\_LPC11xx.s即可；

b）初始化ARM处理器、中断控制器、IO输入引脚的设置，允许中断发生和处理，然后主程序进入空循环，等待中断事件；

c）准备中断处理程序，对中断事件进行相应的处理；

(5) 编译链接工程。连接实验板，进行仿真调试。

（二）定时器实验：

(1) 本实验使用LPC1114微控制器16位定时器1。16位定时器与32位定时器结构和功能都相同，只是计数范围受限为16位整数大小（65535）。如图3-2所示，H0实验板的LED指示灯连接的引脚PIO1\_9，又是16位定时器1的匹配输出0引脚，只要程序将该引脚做相应的配置，就能用作定时器匹配输出。

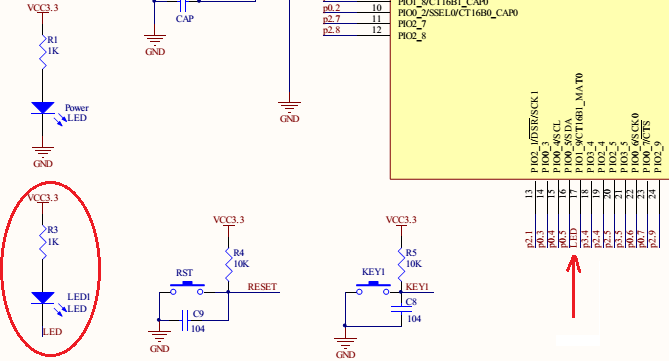


图3-2 LED指示灯连接到16位定时器1的匹配输出0引脚

(2)启动Keil uVision，新建一个工程ex03-2。建立C文件ex03-2.c，编写实验程序，然后添加到工程中。设置工程调试选项。具体步骤参考实验二。

(3)实验程序timer1Init函数初始化16定时器1，使用预分频将48MHz的系统时钟分频为1KHz，然后设置配置寄存器MR0为500-1，实现每500ms发生一次匹配事件。注意16位定时器计数最大值为65535。

(4) 编译链接工程，连接实验板，进行仿真调试。

6．实验参考程序

GPIO中断的参考程序见程序清单3.1。定时器实验的参考程序见程序清单3.2。

程序清单3.1 GPIO中断实验参考程序

#include "LPC11XX.h"

/\* 宏定义 \*/

#define BEEP (1ul << 9) /\* BEEP定义PIOX\_7 \*/

#define KEY (1ul << 1) /\* 按键定义PIO0\_1 \*/

#define BEEPOFF() LPC\_GPIO1->DATA |= BEEP /\* BEEP开 \*/

#define BEEPON() LPC\_GPIO1->DATA &= (~BEEP) /\* BEEP关 \*/

void myDelay (uint32\_t ulTime)

{

uint32\_t i;

i = 0;

while (ulTime--) {

for (i = 0; i < 5000; i++);

}

}

// Switch LED signal to output port with no pull up or pulldown

void LedOutputCfg(void)

{

LPC\_SYSCON->SYSAHBCLKCTRL = LPC\_SYSCON->SYSAHBCLKCTRL | (1<<16) | (1<<6);

LPC\_IOCON->PIO1\_9 = (0x0) + (0<<3) + (0<<5);

// Set pin 9 as output

LPC\_GPIO1->DIR = LPC\_GPIO1->DIR | (1<<9);

return;

}

void KEYInit( void )

{

LPC\_IOCON->PIO0\_1 &= (~0x07); /\* 将P0.1初始化为GPIO功能 \*/

LPC\_GPIO0->DIR &= (~KEY); /\* 设置P0.1为输入 \*/

LPC\_GPIO0->IS = 0x00; /\* P0.1为边沿中断 \*/

LPC\_GPIO0->IEV |= KEY; /\* 上升沿中断 \*/

LPC\_GPIO0->IE |= KEY; /\* P0.1中断不屏蔽 \*/

NVIC\_EnableIRQ(EINT0\_IRQn);

}

// Switch the CPU clock frequency to 48MHz

void SystemInit(void)

{

LPC\_SYSCON->PDRUNCFG = LPC\_SYSCON->PDRUNCFG & 0xFFFFFF5F;

// Select PLL source as crystal oscillator

LPC\_SYSCON->SYSPLLCLKSEL = 1;

// Update SYSPLL setting (0->1 sequence)

LPC\_SYSCON->SYSPLLCLKUEN = 0;

LPC\_SYSCON->SYSPLLCLKUEN = 1;

LPC\_SYSCON->SYSPLLCTRL = (3 + (1<<5)); // M = 4, P = 2

// wait until PLL is locked

while(LPC\_SYSCON->SYSPLLSTAT == 0);

LPC\_SYSCON->MAINCLKSEL = 3;

// Update Main Clock Select setting (0->1 sequence)

LPC\_SYSCON->MAINCLKUEN = 0;

LPC\_SYSCON->MAINCLKUEN = 1;

return;

}

void PIOINT0\_IRQHandler(void)

{

BEEPON();

myDelay(100);

BEEPOFF();

myDelay(100);

BEEPON();

myDelay(100);

BEEPOFF();

myDelay(100);

LPC\_GPIO0->IC |= KEY; /\* 此句要放到中断处理退出前 \*/

}

int main(void)

{

// Initialize LED output

LedOutputCfg();

KEYInit();

while(1)

；

}

程序清单3.2 定时器实验参考程序

#include "LPC11XX.h"

void SystemInit(void)

{

LPC\_SYSCON->PDRUNCFG = LPC\_SYSCON->PDRUNCFG & 0xFFFFFF5F;

// Select PLL source as crystal oscillator

LPC\_SYSCON->SYSPLLCLKSEL = 1;

// Update SYSPLL setting (0->1 sequence)

LPC\_SYSCON->SYSPLLCLKUEN = 0;

LPC\_SYSCON->SYSPLLCLKUEN = 1;

LPC\_SYSCON->SYSPLLCTRL = (3 + (1<<5)); // M = 4, P = 2

// wait until PLL is locked

while(LPC\_SYSCON->SYSPLLSTAT == 0);

LPC\_SYSCON->MAINCLKSEL = 3;

// Update Main Clock Select setting (0->1 sequence)

LPC\_SYSCON->MAINCLKUEN = 0;

LPC\_SYSCON->MAINCLKUEN = 1;

return;

}

void timer1Init (void)

{

LPC\_SYSCON->SYSAHBCLKCTRL |= (1 << 16); /\* 打开IOCON模块时钟 \*/

LPC\_IOCON->PIO1\_9 |= 0x01; /\* 将P1.9配置为MAT输出引脚 \*/

LPC\_SYSCON->SYSAHBCLKCTRL |= (1 << 8); /\* 打开16位定时器1模块的时钟 \*/

LPC\_TMR16B1->PR = 47999; /\* 设置分频系数 \*/

LPC\_TMR16B1->MCR = 2; /\* 设置MR0匹配后复位TC \*/

LPC\_TMR16B1->EMR = (0x03 << 4); /\* MR0匹配后MAT1.0输出翻转 \*/

LPC\_TMR16B1->MR0 = 500-1; /\* 频率控制,500ms后翻转输出 \*/

LPC\_TMR16B1->TCR = 0x01; /\* 启动定时器 \*/

}

int main(void)

{

timer1Init();

while(1)

;

}

7．实践、观察、思考

（1）能否使用多个GPIO中断，如何区分中断来源？

（2）用按键中断控制指示灯的状态切换。

（3）用定时器中断控制指示灯的闪烁。