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
  HO 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。

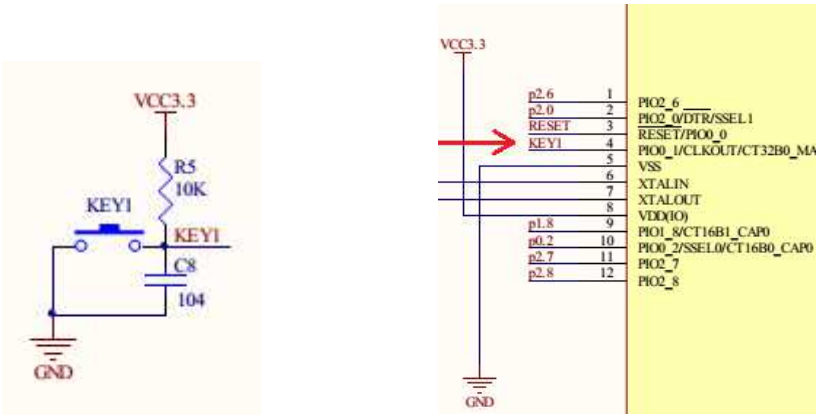

图 3-1 按键输入

(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。

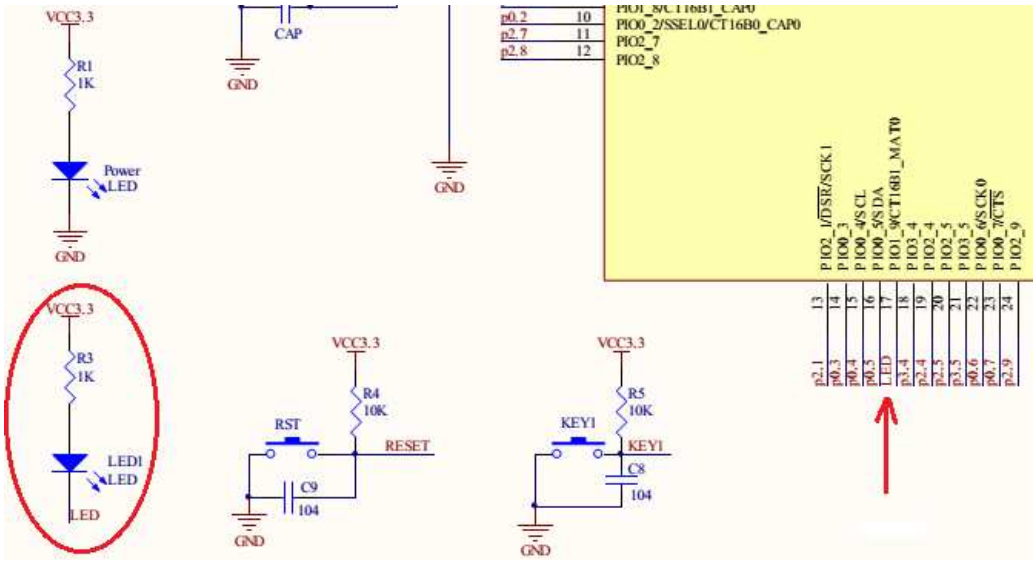

图 3-2 LED 指示灯连接到 16 位定时器 1 的匹配输出 0 引脚

(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。