实验1

1. 实验内容

(1) 接口电路

按键KEY1接口电路如图1.3.3所示，蜂鸣器接口电路如图1.3.1所示。

(2) 实验内容

编写程序，实现以下功能：如果按键以前蜂鸣器响，则按键后，蜂鸣器不响；如果按键以前蜂鸣器不响，则按键后，蜂鸣器响。

(3) 编程要求

采用外部中断技术识别KEY1按键。

2. 实验指导

(1) 外部中断技术

外部中断模块是ARM处理器内的一个独立功能模块，只要通过寄存器对其进行必要的配置，就可以按照设定的方式工作了。

外部中断模块主要的功能是依据寄存器EXTMODE、寄存器EXTPOLAR规定的信号识别规则判别4个外部中断EINT0、EINT1、EINT2、EINT3输入信号是否是中断请求信号。如果是中断请求信号，向外部中断标志寄存器EXTINT的对应位写入1，同时通过内部中断请求线向VIC申请中断。得到响应后，调用对应的中断服务程序。在中断服务程序内需要清除外部中断标志寄存器EXTINT的标志位，清除的方法是只要向寄存器EXTINT对应位写入1即可。外部中断模块相关的寄存器描述如下。

① 外部模式寄存器EXTMODE与外部极性寄存器EXTPOLAR

a. 外部模式寄存器EXTMODE

确定外部中断信号触发模式为边沿触发或电平触发，如表1.4.1所示。

b. 外部极性EXTPOLAR

与EXTMODE配合确认外部中断信号为上升沿或下降沿(边沿模式)触发；高电平或低电平(电平)触发，如表1.4.2所示。

② 外部中断标志寄存器EXTINT

记录EINT0、EINT1、EINT2、EINT3中断请求信息。如表1.4.3所示。

表2.4.1 外部模式寄存器EXTMODE

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 记 号 | 功 能 描 述 | 复 位 值 |
| 0 | EXTMODE0 | 位值=0 EINT0 电平触发 | 0 |
| 位值=1 EINT0 边沿触发 |
| 1 | EXTMODE1 | 位值=0 EINT1 电平触发 | 0 |
| 位值=1 EINT1 边沿触发 |
| 2 | EXTMODE2 | 位值=0 EINT2 电平触发 | 0 |
| 位值=1 EINT2 边沿触发 |
| 3 | EXTMODE3 | 位值=0 EINT3 电平触发 | 0 |
| 位值=1 EINT3 边沿触发 |

表1.4.2 外部极性寄存器EXTPOLAR

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 记 号 | 功 能 描 述 | 复 位 值 |
| 0 | EXTPOLAR0 | 位值=0  EXTMODE D0=0 EINT0低电平触发  EXTMODE D0=1 EINT0下降沿触发 | 0 |
| 0 | EXTPOLAR0 | 位值=1  EXTMODE D0=0 EINT0高电平触发  EXTMODE D0=1 EINT0上升沿触发 |  |
| 1 | EXTPOLAR1 | 位值=0  EXTMODE D0=0 EINT1低电平触发  EXTMODE D0=1 EINT1下降沿触发 | 0 |
| 位值=1  EXTMODE D0=0 EINT1高电平触发  EXTMODE D0=1 EINT1上升沿触发 |

(续表)

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 记 号 | 功 能 描 述 | 复 位 值 |
| 2 | EXTPOLAR2 | 位值=0  EXTMODE D0=0 EINT2低电平触发  EXTMODE D0=1 EINT2下降沿触发 | 0 |
| 位值=1  EXTMODE D0=0 EINT2高电平触发  EXTMODE D0=1 EINT2上升沿触发 |
| 3 | EXTPOLAR3 | 位值=0  EXTMODE D0=0 EINT3低电平触发  EXTMODE D0=1 EINT3下降沿触发 | 0 |
| 位值=1  EXTMODE D0=0 EINT3高电平触发  EXTMODE D0=1 EINT3上升沿触发 |

表1.4.3 EXTINT外部中断标志寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 初始化值  (系统复位) |
| 0 | EINT0 | P0.1或P0.16为EINT0功能时，如果引脚输入信号满足EXTMODE和EXTPOLAR配置的中断触发条件，则此位置1，并且向VIC发送中断。  向此位写入1时，清除中断标志。(电平模式除外) | 0 |
| 1 | EINT1 | P0.3或P0.14为EINT1功能时，如果引脚输入信号满足EXTMODE和EXTPOLAR配置的中断触发条件，则此位置1，并且向VIC发送中断。  向此位写入1时，清除中断标志。(电平模式除外) | 0 |
| 2 | EINT2 | P0.7或P0.15为EINT2功能时，如果引脚输入信号满足EXTMODE和EXTPOLAR配置的中断触发条件，则此位置1，并且向VIC发送中断。  向此位写入1时，清除中断标志。(电平模式除外) | 0 |
| 3 | EINT3 | P0.9或P0.20、P0.30为EINT3功能时，如果引脚输入信号满足EXTMODE和EXTPOLAR配置的中断触发条件，则此位置1，并且向VIC发送中断。  向此位写入1时，清除中断标志。(电平模式除外) | 0 |

(2) 编程策略

本程序由程序框架和2个函数组成，编程策略如图1.4.1所示，包括两个执行线索：第一个是主程序执行线索；第二个是中断服务执行线索。

① 主程序执行线索

主程序执行线索按以下步骤展开。

a. 将管脚P0.7配置为GPIO输出；将管脚P0.20配置为EINT3；初始化使蜂鸣器不响。

参考表1.3.1所示，当PINSEL1的位[9:8]=11时，P0.20为EINT3功能，配置语句如下：

PINSEL1=PINSEL1｜(3<<8)；

b. 通过配置CPSR的 I标志为0来控制开放所有IRQ中断，实现此功能的函数为IRQEnable()。

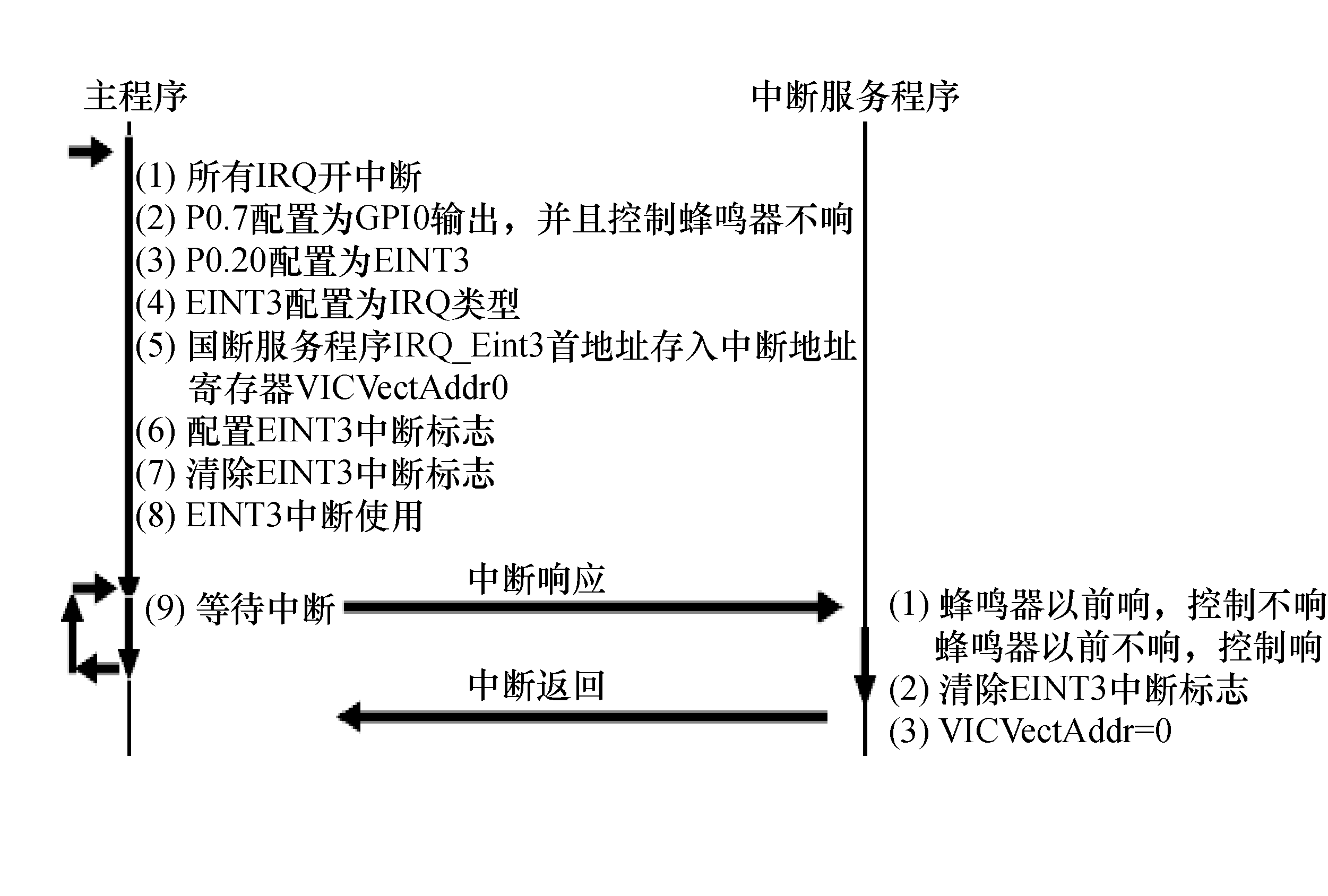


图1.4.1 编程策略

c. 通过配置中断选择寄存器VICIntSelect的位17=0，使EINT3为IRQ类型。EINT3的中断源编号为17，配置语句为：

VICIntSelect=VICIntSelect&(~(1<<17))；

d. 通过配置向量控制寄存器VICVectCntl0、向量地址寄存器VICVectAddr0使中断响应时，系统能够自动找到中断服务程序，进而执行中断服务程序。下面为EINT3中断分配中断向量0，配置语句：

VICVectCntl0=0x20｜17；

VICVectAddr0=(int)IRQ\_Eint3；

其中，数字17为EINT3中断源编号；IRQ\_Eint3为EINT3的中断服务程序。

e. 通过清除EXTINT的位3，使系统可以继续受理EINT3中断请求，配置语句为：

EXTINT = 1<<3；

f. 通过配置寄存器EXTMODE的位3=1，寄存器EXTPOLAR的位3=0使EINT3为下降沿触发，配置语句为：

EXTMODE = 1<<3；

EXTPOLAR = 0x00；

g. 通过配置寄存器VICIntEnable的位17=1，使EINT3开中断。配置语句为：

VICIntEnable = 1<<17；

h. 执行死循环语句“while(1)；”，等待中断。

② 中断服务执行线索

a. 单击KEY1按键，在管脚P0.20输入一个信号，信号波形如图1.4.2所示。



图1.4.2 引脚P0.20输入信号

b. P0.20的功能是EINT3，所以EINT3输入信号也如图2.4.2所示。

c. VIC自动按照寄存器EXTMODE、寄存器EXTPOLAR配置的信号识别规则(下降沿触发)识别EINT3输入信号，一旦确认为中断信号，则将1写入寄存器EXTINT的位3；然后，依次查询控制寄存器VICVectCntl0、VICVectCntl1、…、VICVectCntl15，确认EINT3中断服务程序首地址IRQ\_Eint3存在向量地址寄存器VICVectAddr0中；接着将寄存器VICVectAddr0的值传送至寄存器VICVectAddr；最后，向系统发送中断请求。

d. 由于EINT3为IRQ中断类型，系统响应中断请求后，转到IRQ入口0x00000018处执行，执行的结果是将VICVectAddr的值传送至PC，并自动转到中断服务程序IRQ\_Eint3执行。

e. 中断服务程序IRQ\_Eint3()

首先，中断服务程序根据标志IsBeep控制蜂鸣器在响与不响之间切换。关键语句如下：

if( IsBeep==1)

{ IsBeep=0；

IO0SET =NOBEEP；

} else { IsBeep=1；

IO0CLR = BEEP；

}

其中，语句“IO0SET=NOBEEP；”为控制蜂鸣器不响；语句“IO0CLR=BEEP；”为控制蜂鸣器响。

然后，使用执行语句“EXTINT=1<<3；”清除EXTINT中EINT3中断标志，使系统可以继续响应来自EINT3的中断请求。

最后，清除向量地址寄存器VICVectAddr的值，使其他中断源可以得到中断服务。VICVectAddr用于记录正在服务的中断服务程序地址。

f. 中断服务结束。

(3) 调试策略

① 在语句“IRQEnable()；”的随后语句处，设置一断点，记为断点1。断点1设置的目的是观察调试环境是否满足要求。单击按钮，执行程序，如果程序执行不可控，则调试环境有问题，退出调试，转而解决调试环境问题；如果程序停在断点1处，调试环境无问题，继续向下调试。

② 中断函数IRQ\_Eint3()内，在语句“if(IsBeep==1)”处设置一断点，记为断点2。断点2设置的目的是观察两个问题：一是管脚初始化、EINT3中断初始化是否正确；二是能否识别KEY1按键。单击按钮，执行程序，之后单击按键KEY1，如果程序停在断点2，以上涉及的两个问题全部正确，继续向下调试；如果程序没有停在断点2，则出现问题。问题出现后按如下顺序排除故障：第一检查管脚初始化程序是否正确；第二检查EINT3中断初始化程序是否正确；第三检查KEY1接口电路中的短路块是否短接；第四使用硬件检测方法检测KEY1按键是否损坏。

③ 继续断点2，单击按钮，单步执行程序，观察蜂鸣器是否在“响”与“不响”之间切换。如果不能实现切换，首先检查程序，确认程序正确；其次，采用硬件检测方法，检测蜂鸣器是否损坏；如果能够切换，继续向下调试。

④ 如果前边调试无问题，单击按钮，执行程序，并且按KEY1键，观察程序是否停在断点2。此次执行的目的是确认每按一次KEY1键，都执行一次中断服务程序。

⑤ 去掉所有断点，单击按钮，执行程序，单击按键KEY1，观察程序是否能完成设计功能。

实验2

1. 实验内容

(1) LED接口电路

参考图1.3.4。

(2) 实验内容

编程实现利用定时器技术，控制LED1每隔5秒闪亮一次。

(3) 编程要求

选用定时器0采用中断方式实现定时控制，定时时间为5秒。

2. 实验指导

(1) 定时器技术

定时器模块是ARM处理器内的一个独立功能模块，只要通过寄存器对其进行必要的配置，就可以按照设定的方式开始工作了。现以定时器0为例简要介绍定时器的组成及其工作原理，如图1.4.3所示。

1. Fpclk外围设备时钟

直接作为PR(预分频器)的输入，Fpclk=11.0592Mhz，T为一个时钟时长，如图1.4.4所示。其中，T=1/Fpclk(秒)。

1. PR预分频器

对Fpclk进行N+1分频，输出时长为Tpr的时钟序列，作为定时器TC(定时计数器)的输入。N值为PR寄存器的配置值，如图2.4.5所示，Tpr的计算如公式1.4.1所示。

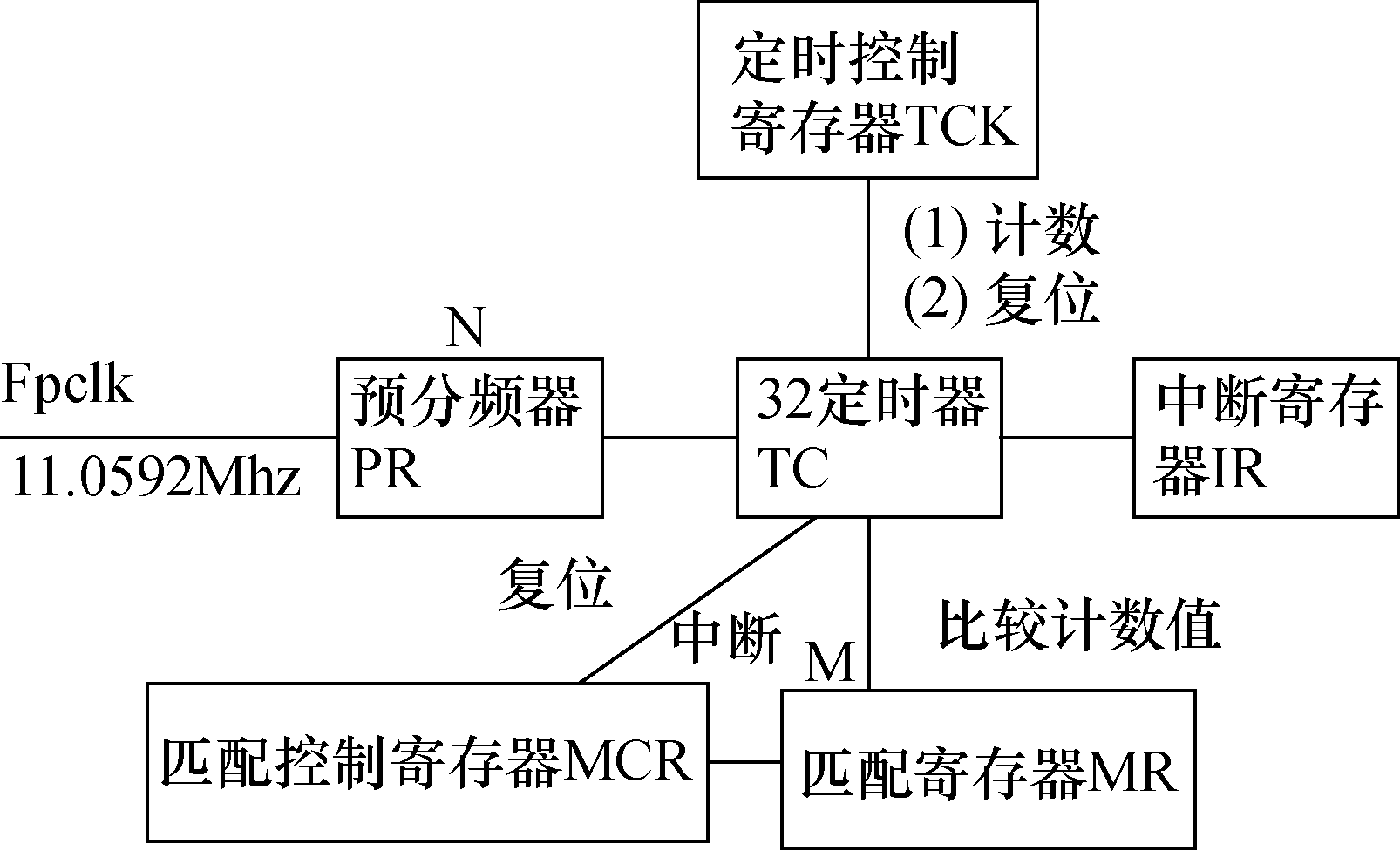


图2.4.3 定时器0逻辑功能框图



图1.4.4 Fpclk外围设备时钟



图1.4.5 T0PR预分频器的输出波形

Tpr=T\*(N+1)=(1/Fpclk)\*(N+1)(秒) (1.4.1**)**

定时器0的预分频器为T0PR；定时器1的预分频器为T1PR。

1. TC定时器

其为32位寄存器，对输入的时钟序列Tpr进行计数，每个时钟周期使TC加1，计数到匹配寄存器MR的值M 时，定时时间到，使用公式1.4.2计算定时时间。

定时时间=Tpr\*M=(1/Fpclk)\*(N+1)\*(M) (秒) (1.4.2)

定时器0的定时计数器为T0TC；定时器1的定时计数器为T1TC。

以定时1秒为例，说明如何配置T0PR寄存器和T0MR寄存器，依据公式1.4.1，将1秒带入公式，1秒=(1/Fpclk)\*(N+1)\*(M) (秒)

(N+1)\*M=11059200

通常情况，N+1取值为10的倍数，这里选取100，也就是N=99，则M=110592。这样，只要在初始化时，配置预分频寄存器T0PR的值为99，匹配寄存器T0MR的值为110592，定时器0就会以1秒定时。

④ IR中断寄存器

中断寄存器IR用于记录MRn(n取值0到3)、CRn(n取值0到3)中断信息。系统响应中断后，执行中断服务程序，在中断服务程序出口处通过写中断寄存器，清除中断标志，如表2.4.4所示。

Timer0(定时器0)的中断标志寄存器为T0IR；Timer1(定时器1)的中断标志寄存器为T1IR。定时器0一旦发生定时中断，则T0IR的位0置1。定时器0得到中断服务后，必须在中断服务程序结束前通过向T0IR位0写入1，清除定时器0的中断标志。

表1.4.4 中断寄存器IR

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 复 位 值 |
| 0 | MR0中断 | 匹配通道0中断标志 | 0 |
| 1 | MR1中断 | 匹配通道1中断标志 | 0 |
| 2 | MR2中断 | 匹配通道2中断标志 | 0 |
| 3 | MR3中断 | 匹配通道3中断标志 | 0 |
| 4 | CR0中断 | 捕获通道0事件中断标志 | 0 |
| 5 | CR1中断 | 捕获通道1事件中断标志 | 0 |
| 6 | CR2中断 | 捕获通道2事件中断标志 | 0 |
| 7 | CR3中断 | 捕获通道3事件中断标志 | 0 |

1. MCR匹配控制寄存器

MCR匹配控制寄存器用于将TC定时计数器的值与MR匹配寄存器的值进行比较，当TC=MR时，称为匹配。一旦匹配成功，匹配控制寄存器MCR控制TC复位，如果需要申请中断，则向IR寄存器写入中断标志，并申请中断。寄存器MCR位信息定义如表1.4.5所示。Timer0(定时器0)的匹配控制寄存器为T0MCR。Timer1(定时器1)的匹配控制寄存器为T1MCR。

⑥ TCR定时控制寄存器

TCR用于清除TC和启动TC计数，如表1.4.6所示。

表1.4.5 MCR匹配控制寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 复 位 值 |
| 0 | MR0I | 1：TC计数值与MR0寄存器匹配时，产生中断  0：不产生匹配中断 | 0 |
| 1 | MR0R | 1：TC计数值与MR0寄存器匹配时，控制TC复位  0：此功能不起作用 | 0 |
| 2 | MR0S | 1：TC计数值与MR0寄存器匹配时，使TC 和  PC停止，TCR[0]=0  0：此功能不起作用 | 0 |
| 3 | MR1I | 1：TC计数值与MR1寄存器匹配时，产生中断  0：不产生匹配中断 | 0 |
| 4 | MR1R | 1：TC计数值与MR1寄存器匹配时，控制TC复位  0：此功能不起作用 | 0 |
| 5 | MR1S | 1：TC计数值与MR1寄存器匹配时，使TC 和  PC停止，TCR[0]=0  0：此功能不起作用 | 0 |
| 6 | MR2I | 1：TC计数值与MR2寄存器匹配时，产生中断  0：不产生匹配中断 | 0 |
| 7 | MR2R | 1：TC计数值与MR2寄存器匹配时，控制TC复位  0：此功能不起作用 | 0 |
| 8 | MR2S | 1：TC计数值与MR2寄存器匹配时，使TC 和  PC停止，TCR[0]=0  0：此功能不起作用 | 0 |
| 9 | MR3I | 1：TC计数值与MR3寄存器匹配时，产生中断  0：不产生匹配中断 | 0 |
| 10 | MR3R | 1：TC计数值与MR3寄存器匹配时，控制TC复位  0：此功能不起作用 | 0 |
| 11 | MR3S | 1：TC计数值与MR3寄存器匹配时，使TC 和  PC停止，TCR[0]=0  0：此功能不起作用 | 0 |

表1.4.6 TCR定时控制寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 复 位 值 |
| 0 | 计数使能 | 1:定时计数器开始计数。  预分频计数器开始计数。  0:不计数。 | 0 |
| 1 | 复位使能 | 1:定时计数器、预分频计数器在PCLK的下一个时钟的上升沿同步复位，复位动作持续到此位回复0值。 | 0 |

Timer0(定时器0)的计数控制寄存器为T0TCR。

Timer1(定时器1)的计数控制寄存器为T1TCR。

(2) 编程策略

本程序由程序框架和两个函数组成，函数Timer0\_Init()用于定时器0初始化；函数IRQ\_Timer0()为定时器0的中断服务程序，编程策略如图1.4.6所示。程序包括两个执行线索：第一个是主程序执行线索；第二个是中断服务执行线索。

① 主程序执行线索

主程序执行线索按以下步骤展开。

a. 通过配置CPSR的I标志为0，控制开放所有IRQ中断。

b. 将管脚P2.16配置为GPIO输出，配置语句为：

IO2DIR=0xff<<16；

c. 通过配置预分频器T0PR=99、匹配寄存器T0MR=110592\*5，使定时器0定时5秒。

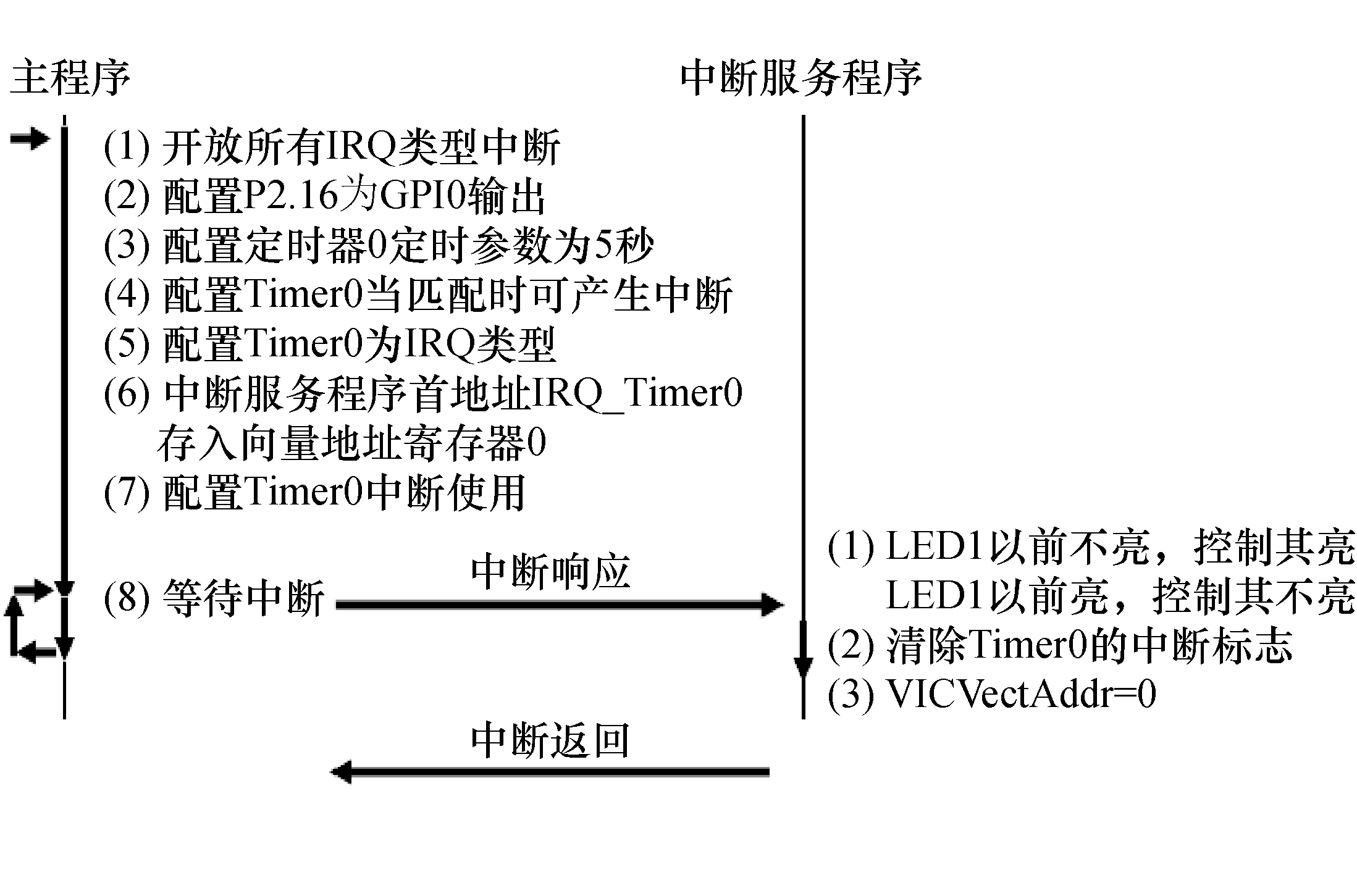


图1.4.6 程序执行策略

d. 通过配置匹配控制寄存器T0MCR 位0=1、位1=1使定时器0匹配时，复位T0TC并产生中断。

e. 通过配置定时控制寄存器T0TCR 位0=1、位1=1使寄存器T0TC立即复位并开始计数。

f. 通过配置中断选择寄存器VICIntSelect的位4=0，使Timer0为IRQ类型。Timer0的中断源编号为4，配置语句为：

VICIntSelect=VICIntSelect&(~(1<<4))；

g. 通过配置向量控制寄存器VICVectCntl0、向量地址寄存器VICVectAddr0使中断响应时，系统能够自动找到定时器0的中断服务程序，然后执行中断服务程序。配置语句为：

VICVectCntl0=0x20｜4；

VICVectAddr0=(int)IRQ\_Timer0；

其中，整数4为定时器0的中断源编号。IRQ\_Timer0为定时器0的中断服务程序地址。

h. 通过配置寄存器VICIntEnable的位4=1，使Timer0开中断，配置语句为：

VICIntEnable = 1<<4

i. 执行死循环语句“while(1)；”，等待中断。

② 中断服务执行线索

a. T0TC自动计数，当T0TC=T0MR时，定时到，由T0MCR控制T0TC复位并产生中断。

b. 系统响应中断后，执行中断服务程序IRQ\_Timer0()。

c. 中断服务程序IRQ\_Timer0()。

首先，中断服务程序根据标志nFlag控制LED1在亮与不亮之间切换。关键语句如下：

if(nFlag==0)

{ nFlag=1；

IO2CLR=0xff01ffff；

} else { nFlag=0；

IO2SET=0xff01ffff；

}

其中，语句“IO2CLR=0xff01ffff；”控制LED1亮；语句“IO2SET=0xff01ffff；”控制LED1不亮；语句“T0IR = 0x01；”用于清除T0IR中Timer0中断标志，使系统可以继续响应来自Timer0的中断请求。

最后，清除向量地址寄存器VICVectAddr的值，使其他中断源可以得到中断服务。VICVectAddr寄存器用于记录当前正在服务的中断服务程序地址。

d. 中断服务结束。

(3) 调试策略

① 在语句“IRQEnable()；”的随后语句处，设置一断点，记为断点1。断点1设置的目的是观察调试环境是否满足要求。单击按钮，执行程序，如果程序执行不可控，则调试环境有问题。退出调试，转而解决调试环境问题；如果程序停在断点1处，调试环境无问题，继续向下调试。

② 中断函数内，在语句“if(nFlag==0)”处设置一断点，记为断点2。断点2设置的目的是观察定时器0中断初始化是否正确。单击按钮，执行程序，如果程序停在断点2，定时器0初始化正确，继续向下调试；如果程序没有停在断点2，则出现问题。出现问题后，检查定时器0初始化程序，确认程序编写正确。

③ 继续断点2，单击按钮单步执行程序，观察LED1是否在“亮”与“不亮”之间切换。如果不能实现切换，首先检查程序，确认程序正确；其次，采用硬件测试方法，检测LED1是否损坏。

④ 如果前面调试无问题，单击按钮，执行程序；观察程序是否停在断点2。此次执行的目的是确认每一次定时到，都执行一次中断服务程序。

1. 最后，去掉所有断点，单击按钮，全速执行程序，观察LED1的变化。

### 实验3

1、实验内容

(1) A/D接口电路

A/D接口电路如图1.5.1所示。

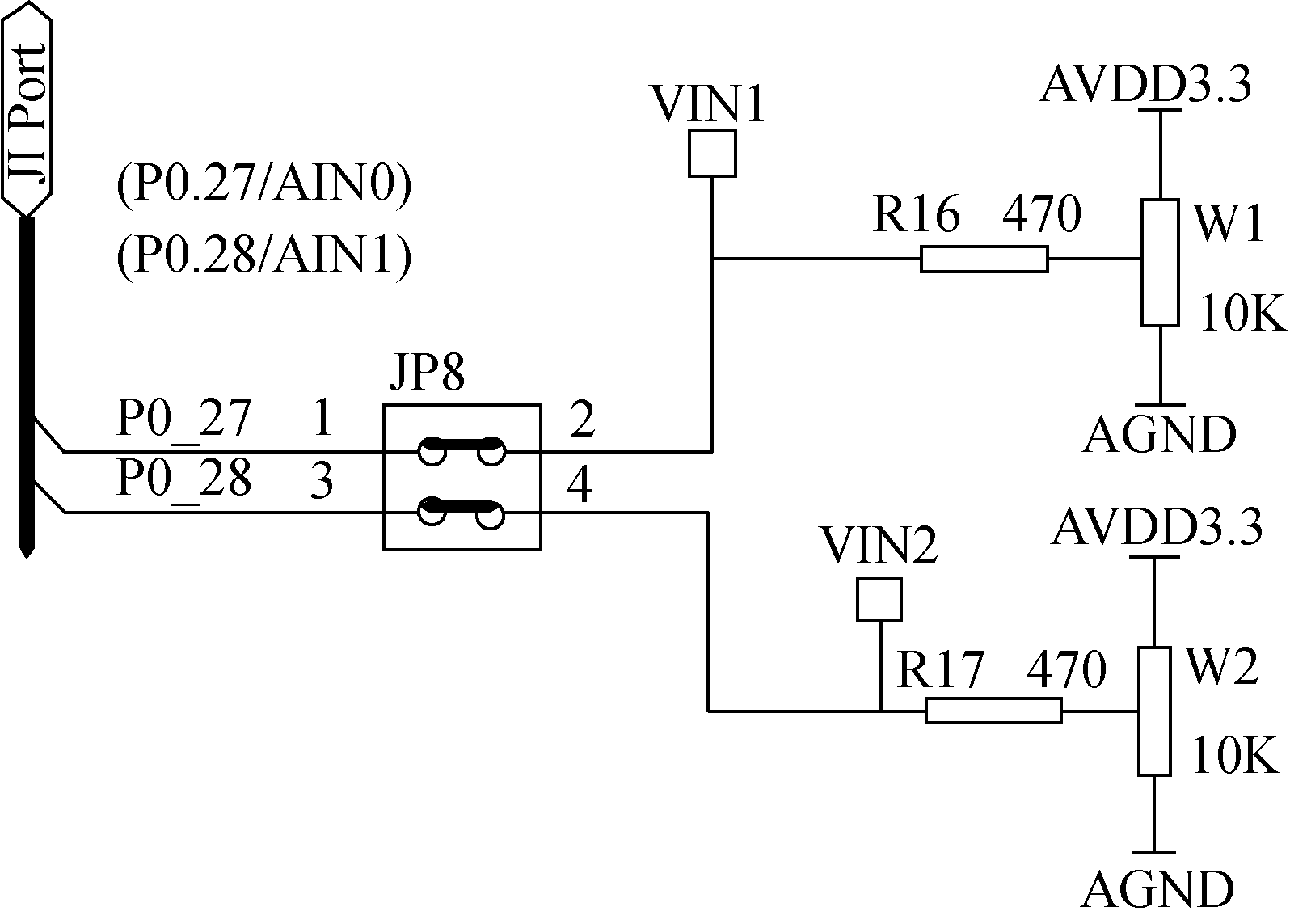


图1.5.1 A/D接口电路

(2) 实验内容

使用A/D模块测量P0.27(AIN0)引脚的直流电压，然后将测量结果通过UART0发送给计算机，计算机在屏幕上显示AIN0的电压。

2、实验指导

(1) A/D技术

A/D模块是ARM处理器内的一个独立功能模块，只要通过寄存器对其进行必要的配置，就可以按照设定的方式开始工作了。

A/D转换模块的功能就是对AIN0…AIN7模拟输入进行模数转换，并将转换结果保存到数据寄存器中，系统通过读数据寄存器获取转换结果，其中转换结果是一个10位二进制数。与A/D转换相关的寄存器描述如下。

① ADC控制寄存器ADCR

此寄存器用于选择模拟输入、配置转换时钟、配置转换通道、启动转换等，详细信息如表2.5.1所示。

② 32位的ADC数据寄存器ADDR

此寄存器用于获取A/D转换结束信息以及转换结果。详细信息如表2.5.2所示。

表2.5.1 ADC控制寄存器ADCR

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 复 位 值 |
| 7:0 | SEL | 每一位对应一路AIN，即位0对应AIN0；位1对应AIN1；…；位7对应AIN7。  当某位=1时，对应此位的AIN被选择用于模数转换。 | 0x01 |
| 15:8 | CLKDIV | 对PCLK进行分频，得到A/D转换时钟。  此时钟必须满足<=4.5MHZ | 0 |
| 16 | BURST | 0:由软件启动A/D转换，转换时长为11个时钟周期。  1：按CLKS配置自动重复转换。 | 0 |
| 19:17 | CLKS | 在Burst模式下，一次转换的时长，转换的结果存入寄存器ADDR中。  000 11个时钟/10位 001 10个时钟/9位  010 9个时钟/8位 011 8个时钟/7位  100 7个时钟/6位 101 6个时钟/5位  110 5个时钟/4位 111 4个时钟/3位 | 000 |
| 21 | PDN | 0:ADC可用  1:ADC掉电模式 | 0 |
| 26:24 | START | 当BURST为0时(软件启动AD转换)  000 不启动  001 立即启动  010 由EDGE决定采用P0.16/EINT0/MAT0.2/CAP0.2的上升沿  (或下降沿)启动A/D转换  011由EDGE决定采用P0.22/CAP0.0/MAT0.0的上升沿(或下降沿)  启动A/D转换  100由EDGE决定采用MAT0.1的上升沿(或下降沿)启动A/D转换  101由EDGE决定采用MAT0.3的上升沿(或下降沿)启动A/D转换  110由EDGE决定采用MAT1.0的上升沿(或下降沿)启动A/D转换  111由EDGE决定采用MAT1.1的上升沿(或下降沿)启动A/D转换 | 000 |
| 27 | EDGEI | 该位在START为010至111时有效，  0:在信号的下降沿启动A/D转换  1:在信号的上升沿启动A/D转换 | 0 |

表2.5.2 ADC数据寄存器ADDR

|  |  |  |  |
| --- | --- | --- | --- |
| 位 | 功 能 | 描 述 | 复 位 值 |
| 15:6 | RESULT | A/D转换结果 | NA |
| 26:24 | CHN | 对那个通道进行的A/D转换 | 0 |
| 31 | DONE | 当A/D转换结束时，此位置“1”。当配置ADCR或从ADDR读数据时，此位清“0” | 0 |

③ 采样频率

采样频率指的是每秒A/D转换次数，由寄存器ADCR的CLKDIV项决定，如公式2.5.1所示，其中Fpclk为外围设备时钟。

ADC时钟=Fpclk/(CLKDIV+1) (2.5.1)

④ 模拟信号输入范围

使用P0.27引脚检测直流电压，可检测的直流电压范围决定模拟参考电压，本实验可检测的电压范围为0…AVDD。

⑤ A/D转换关系

A/D模块对电压在0至AVDD之间的模拟量进行A/D转换，转换的结果是一个10位二进制数，模拟量与转换结果之间的对应关系如下。

a. 模拟量电压0V经过A/D转换后，转换的结果为10位二进制数0000000000。

b. 模拟量电压AVDD经过A/D转换后，转换的结果为10位二进制数1111111111。

c. 假设模拟电压为Vin，X为A/D转换结果，则Vin与X之间的关系是Vin=(X\*AVDD)/(210-1)V。

(2) 接口电路

① A/D接口电路中的元器件

参考图2.5.1，A/D接口电路中使用的元器件包括：P0.27为LPC2290 P0口的27引脚；JP8为短路块；R16为电阻；W1为电位计；AVDD3.3为模拟电源正，电压值为3.3V；AGND为模拟电源地。

② 电路连接

参考图2.5.1，P0.27经过短路块JP8连接到电阻R16的一端；R16的另一端连接电位计W1的滑动触点引脚；电位计W1另两个引脚分别连接模拟电源正和模拟电源地。

③ 工作原理

参考图2.5.1，通过转动W1的旋钮，改变滑动触点位置，从而改变滑动触点的电压，使P0.27引脚输入的模拟电压跟随改变。P0.27具有AIN0功能，使用A/D对AIN0进行A/D转换，从而达到测量P0.27引脚直流电压的目的。

(3) 编程策略

① MagicARM2200实验平台程序Exp501.axf

本程序由程序框架和5个函数组成，编程策略如图2.5.2所示。

a. 函数PIN\_Init()

本函数用于将管脚P0.0配置为TxD0；管脚P0.1配置为RxD0。参考表2.2.4，配置语句为PINSEL0 = 0x05。其中位[1:0]=01，将P0.0配置为TxD0；位[3:2]=01，将P0.1配置为RxD0。

参考表2.3.1，管脚功能选择寄存器PINSEL1位[23:22]=01，将管脚P0.27配置为AIN0，配置语句为PINSEL1=0x01<<22。

b. 函数DelayNs

本函数用于延时，程序编写与前面实例相同。

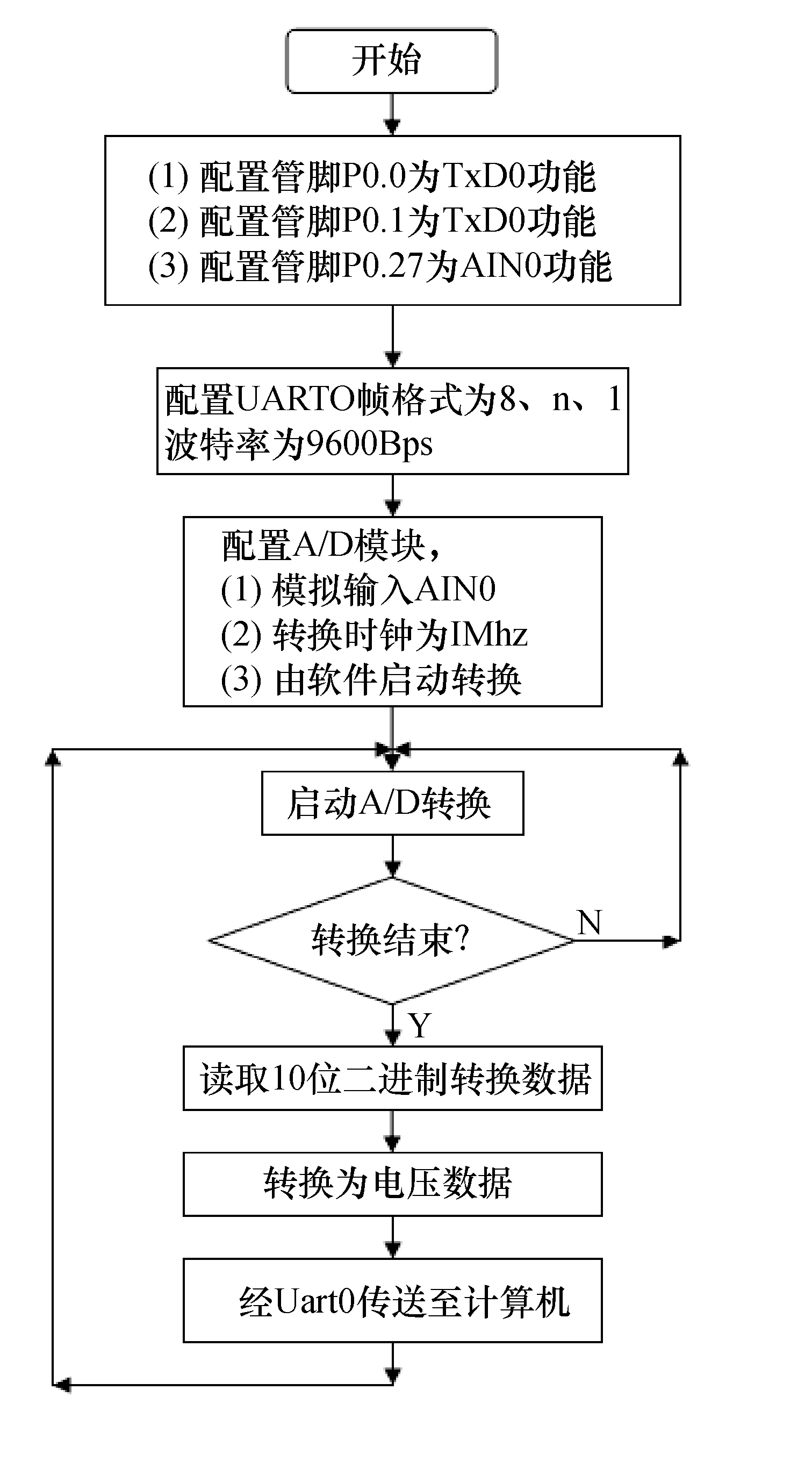


图2.5.2 流程图

c. 函数AD\_Init()

本函数用于ADC初始化，配置语句如下：

ADCR = (1 << 0)｜

((Fpclk / 1000000 - 1) << 8)｜

(0 << 16)｜

(0 << 17)｜

(1 << 21)｜

(0 << 22)｜

(1 << 24)｜

(0 << 27)；

其中，位0=0，用于选择模拟输入AIN0；“CLKDIV = Fpclk/1000000 – 1”是A/D转换时钟，为1MHZ；BURST= 0，用于选择软件控制启动A/D转换；CLKS = 0，用于选择A/D转换为11clock时钟；“PDN =1”与“TEST[1:0] = 00”用于配置模式为正常工作模式(非掉电转换模式)；“START[26:24] = 001”用于直接启动ADC转换。

d. 函数UART0\_Init()、函数UART0\_SendStr()

与UART0实验相同。

e. 主程序main()

首先，调用函数PIN\_Init()、UART0\_Init()、AD\_Init()进行管脚、UART0、A/D模块初始化；然后，采用while语句实现死循环，作为程序主框架。

在程序框架中，首先，设置ADCR寄存器位[26:24]=001，软件启动A/D转换。其次，通过查询ADDR寄存器的位31获知A/D转换结束。然后，从寄存器ADDR位[15:6]提取转换结果数据。最后，将10位A/D转换结果换算成对应的电压值，通过UART0发送给计算机。

(4) SerialPort501.exe程序

SerialPort501.exe程序编写策略与SerialPort201.exe编写相似，这里不再赘述。

(5) 调试策略

① 运行计算机串口程序SerialPort501.exe，打开串口。选择菜单“接收、显示A/D转换数据”命令。

② 对于程序Exp501.axf，在“while( (ADDR&0x80000000)==0)；”语句处，设置一断点，记为断点1，断点1设置的目的是观察初始化是否正确。单击按钮，执行程序，程序停在断点1处，然后，单击按钮，单步执行程序，观察程序是否会停在语句“ADCR = ADCR｜(1<<24)”。如果是，则管脚、ADC初始化正确；如果不是，管脚、ADC初始化程序有问题，分析初始化程序，改正错误。

③ 对于程序Exp501.axf，在语句“ADC\_Data = ADDR；”处设置一断点，记为断点2。断点2设置的目的是观察A/D转换后的数据。单击按钮，执行程序，程序会停在断点2，查看寄存器ADDR位[15:6]的值。

④ 对于程序Exp501.axf，在语句“UART0\_SendStr((unsigned char \*)str)；”处设置一断点，记为断点3。单击按钮，执行程序，程序停在断点3处，检查转换结果与ADDR寄存器内容是否相符。

⑤ 单击按钮，单步执行UART0发送语句，观察计算机端是否收到A/D转换数据。如果没收到，按实验2指示的顺序排除UART0故障。

1. 最后，去掉所有断点，单击按钮，全速执行程序。旋动W1的旋钮，观察计算机接收、显示A/D模块的测量数据。