简答题 20分

分析设计题 20分

分析程序，把程序补充完整 20

分析程序，给出运行结果 20

编程序 20

**1.一般定义**

2、嵌入式系统的几个重要特征

3.常见的嵌入式操作系统

嵌入式系统  单片机系统

嵌入式系统  PC机

ARM处理器工作模式

ARM寄存器组成

异常和中断的区别？

中断和调子程序的区别？

调子程序和跳转的区别？

ARM的异常中断

指令

1.已知在内存单元0x40000000中存放自变量x，求如下函数的函数值y，并将其存入内存单元0x40000001.使用条件域指令编写程序。

Y=1,0,-1(当x分别x>0,x=0,x<0时)

area example,code,readonly

entry

code32

Start ldr r2,=0x40000000

ldrb r1,[r2]

cmp r1,#0

moveq r1,#0

movlt r1,#0xffffffff

movgt r1,#1

add r2,r2，#1

str r1,[r2]

end

2. 堆栈练习:下面是ARM指令中的堆栈模式，其中初始值R13=0x2000, r2=0x20, r3=0x30,r4=0x40,r5=0x50, r6=0x60,r7=0x70, r8=0x80,r9=0x90请说出指令执行后寄存器与内存的变化。

STMFD R13！，{R2-R9}；

LDMFD R13！，{R2-R9}；

3. 块拷贝练习:下面是ARM指令中的块传送模式，其中初始值R0=0x2000, r2=0x20, r3=0x30,r4=0x40,r5=0x50, r6=0x60,r7=0x70, r8=0x80,r9=0x90请说出指令执行后寄存器与内存的变化。

STMIA R0！{R2-R9}；

STMIB R0！{R2-R9}；

STMDA R0！{R2-R9}；

STMDB R0！{R2-R9}；

LDMIA R0！{R2-R9}；

LDMIB R0！{R2-R9}；

LDMDA R0！{R2-R9}；

LDMDB R0！{R2-R9}

4. 形成两个矢量的标量积：每个矢量有20个分量，每个分量为一个字，但它的值小于两个字节。

MOV R11,#20 ;初始化循环次数

MOV R10,#0；初始化总和

LOOP LDR R0,[R8],#4；读取第一分量

LDR R1,[R9],#4

MLA R10,R0,R1, R10；乘积累加

SUBS R11,R11,#1；减循环记数

BNE LOOP

5. 执行下面的程序后，r3,r4,r5,r6分别是多少？

mov r0,#0x0c000000

ldr r9,=0x12345678

str r9,[r0]

ldr r3,[r0]

add r0,r0,#1

ldr r4,[r0]

add r0,r0,#1

ldr r5,[r0]

add r0,r0,#1

ldr r6,[r0]

6. 下面为内存分配伪指令，假设内存从0x30001000处存放，请用图的办法

表示出内存的占用情况。

**DCD 0X12345678**

**Space 4**

**dcb 1,2,3,4**

**dch 0xff,0X11**

7. 软中断使用

8. 利用内嵌汇编程序来实现中断的使能和禁止。使能IRQ中断

void enable\_IRQ(void)

{

int tmp;

\_ \_asm

{

MRS tmp, CPSR

BIC tmp,tmp,#0x80

MSR CPSR\_c,tmp

}

}

9. 利用内嵌汇编程序来实现中断的使能和禁止。禁止IRQ中断

void EnableIRQ(void)

{

\_\_asm{

MRS R1，CPSR

BIC R1，R1，#0x80

MSR CPSR\_c，R1

}

return;

}

10. 计算一个N的阶层，如：10的阶层

area jc,code,readonly  
FACTOREAL MOV R6,#10  
 MOV R4,R6   
LOOP SUBS R6,R6,#1  
 MULNE R5,R4,R6   
 mov r4,r5   
 BNE LOOP

11.编写一个子程序，从存储器某处拷贝一个字符串到存储器另一处。源字符串的开始地址放入R1中，长度放入R2,目的字符串的开始地址在R3.（对他们的位置进行判断）

12.求1－－100的平方和？

13. 伪操作的执行时间和对齐情况，请用图的办法表示出内存的占用情况。

area test, code ,readonly

entry

mov r0,#1;

b stop

dcd 1,2,3,4

dcd 1,2,3,4

dcb 0xff

dcdu 0x12345678

dcd 0x12345678

stop

b stop

end

14. 看ALIGN指令在内存存放中的作用，请用图的办法表示出内存的占用情况。

area aa, code, readonly

ENTRY

mov R1,#1

dcd 0x12345678

label DCB 0x48;

label2 dcb 0x48

dcb 0x48

dcb 0x48

dcb 0x48

ALIGN 2

label3 dcb 0x48

align 4

label4 dcb 0x48

stop

b stop

END

15. 把16进制数转换成ASC2编码

AREA Hex\_Ou,CODE,READONLY

ENTRY ；代码的入口

LDR R1,=VALUE；要转换的数据

LDR R1,[R1]；取出0x12345678

LDR R3,=STORE;存放转换后的数字

BL HexOut ；调用十六进制输出

STOP B STOP；结束

VALUE DCD 0X12345678；测试数据

STORE space 32

HexOut MOV R2,#8 ；

LOOP MOV R0,R1,LSR #28；读取高位的半字节

CMP R0,#9 ；0~9还是A~F

ADDGT R0,R0,#’A’-10；ASC字母

ADDLE R0,R0,#’0’； ASC字母

STR R0,[R3],#4 ；存放数据

MOV R1,R1,LSL #4；左移4位

SUBS R2,R2,#1 ；显示个数减1

BNE LOOP ；若还有，则继续进行

MOV PC,R14 ；返回

END

17. 数据块复制举例

本程序将数据从源数据区src复制到目的数据区dst,复制时以8个字为单位进行。对于最后所剩不足8个字的数据，以字为单位复制，这时程序跳转到copywords处执行。在进行以8个字为单位的数据复制时，保存了所用的8个工作寄存器

AREA Block,CODE,READONLY

NUM EQU 20 ；设置要复制的字数

ENTRY

LDR R0,=Src

LDR R1,=Dst

MOV R2,#NUM

MOV SP,#&400

Bcopy MOVS R3,R2,LSR #3；8字为单位的次数

BEQ Cword ；R3为0，转到按字传送

STMFD SP!,{R4-R11}；保存工作寄存器

Ocopy LDMIA R0!,{R4-R11}；按‘8字传

STMIA R1!,{R4-R11}

SUBS R3,R3,#1

BNE Ocopy

LDMFD SP!, {R4-R11}；出栈

Cword ANDS R2,R2,#7；获剩余字个数

BEQ stop ；如为0，退出

Wcopy LDR R3,[R0],#4；按字传

STR R3,[R1],#4

SUBS R2,R2,#1

BNE Wcopy

Stop

B stop

AREA Bdata, DATA, READWRITE

Src DCD 1,2,3,4,5,6,7,8,1,2,…….

Dst DCD 0,0,0,0,0,0,0,0,0,…….

END

18. 本程序将字符从源字符数据区srcstr复制到目标字符数据区dststr。调用子函数strcopy以字符为单位进行字符串复制。

AREA StrCopy，CODE，READCONLY ；定义程序名及属性

ENTRY ；标识程序人口点

start

LDR R1，=srcstr ; R1指向源字符数据区

LDR R0，=dststr ; R0指向目标字符数据区

BL strcopy ; 调用字符串复制子程序strcopy

 Stop

B stop

strcopy

LDRB R2，[r1]，#1 ; 读取字符，并更新源字符地址R1

STRB R2，[r0]，#1 ；写入字符，并更新目标字符地址R0

CMP R2，#0 ；判断源字符串是否复制完成

BNE strcopy ；循环执行复制，直到字符串复制完成

MOV pc，lr ；返回主程序

；定义数据段及属性

AREA Strings，DATA，READWRITE

；定义源字符数据区srcstr和目标字符数据区dststr

srcstr DCB “First string-source”,0

dststr DCB “Second string—deslinatiOn”,0

END ; 程序结束

19. 利用跳转表实现程序跳转 举例

在程序中常常要根据一定的参数选择执行不同的子程序。本例演示通过跳转表实现程序跳转。跳转表中存放的是各子程序的地址。在本例中，**R3寄存器中存放的跳转表的基地址。R0寄存器的值用于选择不同的子程序：**当R0为0时，选择的是子程序DoAdd;当R0为1时，选择的是子程序DoSub。

AREA Jump,CODE,READONLY

NUM EQU 2；跳转表中的子程序个数

ENTRY

Start MOV R0,#0；选择调用哪个子程序

MOV R1,#3；R1，R2子程序要用的参数

MOV R2,#2

BL Func

Stop

B stop

Func CMP R0,#NUM

MOVCS PC,LR；>=2超出范围程序返回

ADR R3,Jtable；跳转表的基地址

LDR PC,[R3,R0,LSL#2];跳到对应子程序

Jtable DCD DoAdd；跳转表首地址

DCD DoSub

DoAdd ADD R0,R1,R2

MOV PC,LR ;子程序返回到那儿？

DoSub SUB R0,R1,R2

MOV PC,LR

END

20. 求最大公约数

**AREA gcd, CODE**

**ENTRY**

**MOV r0, #9**

**MOV r1, #15**

**start**

**CMP r0, r1 SUBLT r1, r1, r0 SUBGT r0, r0, r1**

**BNE start**

**stop**

**B stop**

**END**

21. 写一段 ARM汇编程序：循环累加队列中的所有元素，直到碰上零值元素，结果放在r4。

**AREA total, CODE**

**ENTRY**

**MOV r4, #0**

**ADR r0, array**

**loop**

**LDR r1, [r0], #4 ADD r4, r4, r1**

**CMP r1, #0**

**BNE loop**

**stop**

**B stop**

**array DCD 0x11**

**DCD 0x22**

**DCD 0**

**END**

22. **写一个汇编主程序，把一个含64个带符号的16-bit数据组成的队列求平方和**

**AREA dspcode, CODE**

**ENTRY**

**MOV r2, #64**

**MOV r4, #0**

**LDR r0, =x**

**loop**

**LDRSH r1, [r0], #2  
 MLA r4, r1, r1, r4**

**SUBS r2, r2, #1**

**BGT loop**

**stop**

**B stop**

**AREA dspdata, DATA**

**x**

**DCW 0x7777**

**DCW 0x1111**

**DCW 0xeeef**

**...**

**END**

23. 字符串复制(例子C2)

该例主要介绍如何使用指令BL调用子程序。在内嵌的BL指令中，除了正常的操作数域外，还必须增加可选的寄存器列表。例如,主函数main()中的“BL strcpy,{R0，R1}”指令的输入寄存器列表为{R0，R1}，没有输出寄存器列表。子程序使用的工作寄存器为ATPCS默认工作寄存器R0～R3、R12、LR及CPSR。

#include <stdio.h>

void strcpy(char \* src，const char \*dst)

{

int ch;

\_ \_asm

{

loop：

LDRB ch，[src]，#1

STRB ch，[dst]，#1

CMP ch，#0

BNE loop

}

}

int main(void)

{

const char \*a=“Hello world!”;

Char b[20]；

\_ \_asm

{

MOV R0,a

MOV R1,b

BL strcpy,{R0，R1}

}

printf("Originalstring：％s＼n”，a);

printf("Copied string：％s＼n”，b)

return 0;

}

23. 下面是一个在汇编语言程序中访问C语言程序全局变量的实例。

#include <stdio.h>

extern void asmsubroutine(void);

int globvar;

int main(){

globvar=1;

asmsubroutine();

printf("globvar=：%d\n",globvar);

}

AREA globals，CODE，READONLY

EXPORT asmsubroutine

; 用EXPORT伪操作声明该变量可被其它文件引用，相当于声明了一个全局变量

IMPORT globvar ; 用IMPORT伪操作声明该变量是在其它文件中定义的，在本文件中可能要用到该变量。

asmsubroutine

LDR R1，=globvar ; 从文字池读globvar的地址，并将其保存到R1

LDR R0，[R1]; 再将其值读人到寄存器R0中

ADD R0，R0，#2

STR R0，[R1] ; 修改后再将寄存器R0的值赋予变量global

MOV PC，LR

END

24. 下面是一个C语言程序调用汇编语言程序完成字符串复制的实例。

#include <stdio.h>

／／用extern声明一个函数为外部函数可被其他文件中的函数调用

extern void strcopy(char \*d，const char \*s)；

int main()

{

const char \*srcstr=“First string - source”;

char \*dststr="Second string - destination"；

printf(“Before copying: \n”)；

printf(“％s＼n％s＼n”，srcstr，dststr)；

strcopy(dststr，srcstr)；//调用汇编函数strcopy()

printf(“After copying：＼n，’)；

printf(“％s＼n％s＼n”，srcstr，dststr)；

return(0)；

}

AREA Scopy，CODE,READONLY

EXPORT strcopy ;用EXPORT伪指令声明该变量为全局变量

Strcopy

LDRB R2,[R1],#1 ；R0指向目标字符串dststr

STRB R2,[R0],#1 ；R1指向源字符串srcstr

CMP R2,#0

BNE strcopy

MOV PC,LR

END

25. **下面是一个汇编语言调用C语言程序实例**

int sum(int a, int b, int c, int d, int e)

{

return a+b+c+d+e;

}

EXPORT mul

AREA mul，CODE，READONLY

IMPORT sum ; 伪操作IMPORT声明C程序sum()

STR LR,[SP,#-4]! ; 保存返回地址LR

ADD R1,R0,R0

ADD R2,R1,R0

ADD R3,R1,R2

STR R3,[SP,#-4]!

ADD R3,R1,R1

BL sum ; 调用C语言程序sum()

ADD SP,SP,#4 ; 调整数据栈指针，准备返回

LDR PC,[SP],#4 ; 从子程序返回

END

26. 如图所示，通过控制GPF3，GPF4，GPF5，GPF6的控制实现对LED1，LED2，LED3，LED4 亮灭的控制。LED灯通过电源，限流电阻与ARM的I/O口相连，当I/O口为低电平时，点亮led。反之I/O为高电平时，则LED不会亮。

程序如下：

#define GPFCON (\*(volatile unsigned \*)0x56000050)

#define GPFDAT (\*(volatile unsigned \*)0x56000054)

#define GPFUP (\*(volatile unsigned \*)0x56000058)

void Delay(unsigned int);

int Ledlamp()//流水灯程序

{

unsigned char ledtab[]={0xf7,0xef,0xdf,0xbf};

//流水灯接口输出赋值

int i;

GPFUP&=0XFFFFFF87; //使能上拉F3-6

GPFCON&=0XC03f;

GPFCON|=0X1540; //将F3-6 设为输出口

while(1)

{

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

{

GPFDAT=ledtab[i];

Delay(70);

}

}

return(0);

}

void Delay(unsigned int x ) //延时程序

{

unsigned int i,j,k;

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

for(j=0;j<=0xff;j++)

for(k=0;k<=0xff;k++);

}

// init.s 在汇编语言中，请记住只有标识符才顶格写，其他的不要顶格写

//主函数入口

AREA ledtest,CODE,READONLY

ENTRY

ldr r13,=0x1000

IMPORT Ledlamp；调用LED子函数

b Ledlamp

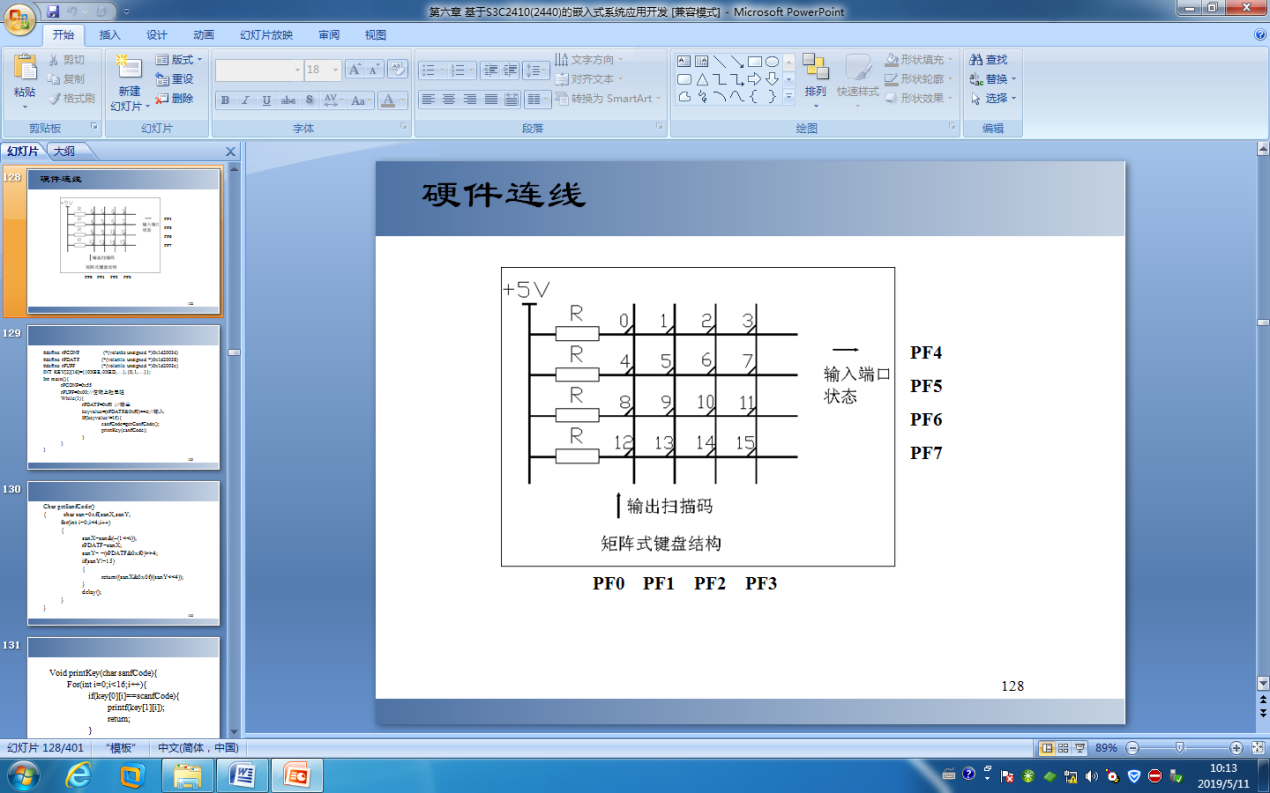
END

27. 设计S3C2410与一个4×4的键盘接口。（参考实验三）

1。确定键盘的检测方法－键盘循环扫描法(请描述键盘的识别过程) 。

2。S3C2410与一个4×4的键盘硬件连接。

硬件连接：PF0~3为输出口，PF4~7为输入口



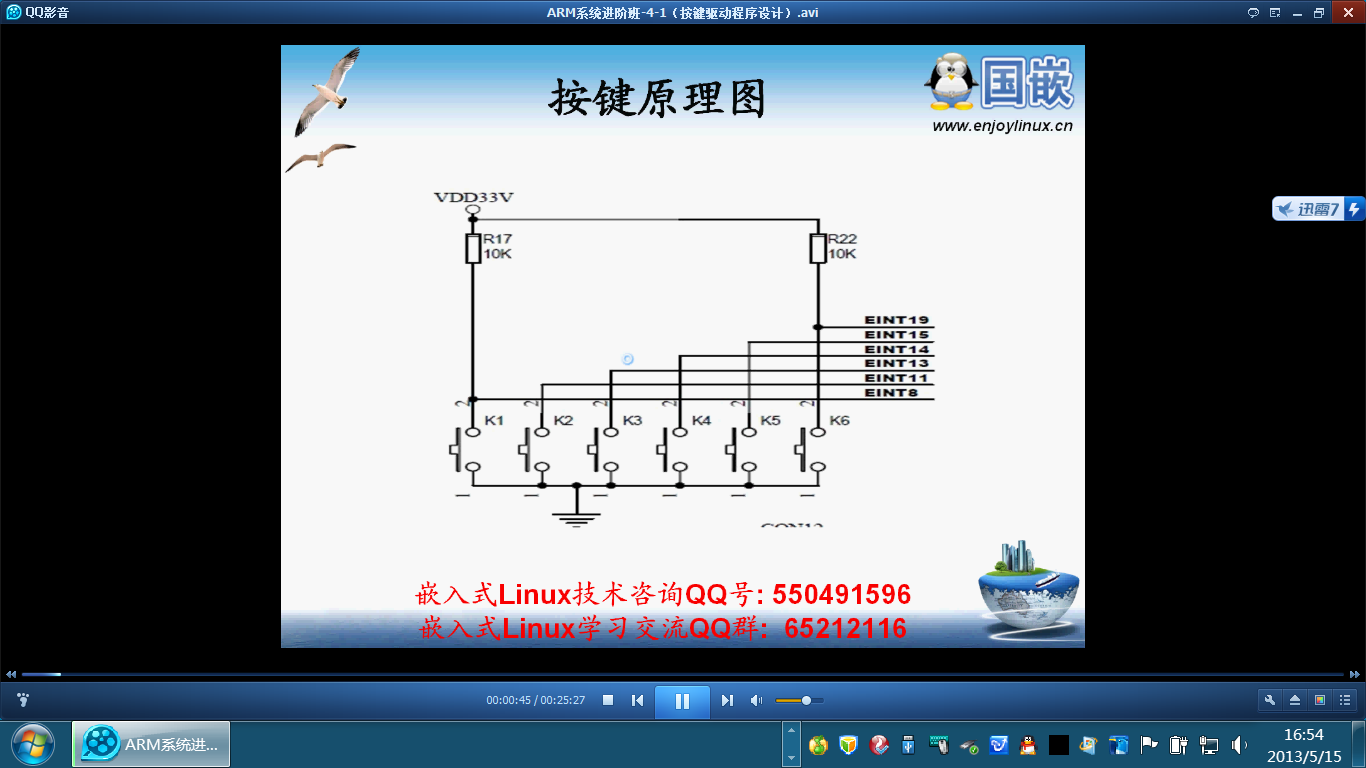
3。定义端口地址

4。初始化I/O

5。编程序实现

28. 根据串行口的工作原理，编程实现ARM的UART 通讯。（参考实验二）

29. 设计S3C2410与6个按键的接口，如下图所示，利用中断原理设计硬件连接，并编写程序实现按键1按下led1点亮其它灯灭，按键2按下led2点亮其它灯灭，按键3按下led3点亮其它灯灭，按键4按下led4点亮其它灯灭，按键5按下led5点亮其它灯灭，按键6按下led6点亮其它灯灭。



30.设计系统，系统中有两个led灯和两个控制按钮，按第一个按钮灯交替闪烁，按第二个按钮一下两个灯同时闪烁。

31. 存储控制寄存器的初始化程序设计

**memsetup: @ 设置存储控制器以便使用SDRAM等外设**

**mov r1, #MEM\_CTL\_BASE @ 存储控制器的13个寄**

**存器的开始地址**

**adrl r2, mem\_cfg\_val @ 这13个值的起始存储地址**

**add r3, r1, #52 @ 13\*4 = 52**

**1:**

**ldr r4, [r2], #4 @ 读取设置值，并让r2加4**

**str r4, [r1], #4 @ 将此值写入寄存器，并让r1加4**

**cmp r1, r3 @ 判断是否设置完所有13个寄存器**

**bne 1b @ 若没有写成，继续**

**mov pc, lr** .align 4

mem\_cfg\_val: @ 存储控制器13个寄存器的设置值

.long 0x22011110 @ BWSCON

.long 0x00000700 @ BANKCON0

.long 0x00000700 @ BANKCON1

.long 0x00000700 @ BANKCON2

.long 0x00000700 @ BANKCON3

.long 0x00000700 @ BANKCON4

.long 0x00000700 @ BANKCON5

.long 0x00018005 @ BANKCON6

.long 0x00018005 @ BANKCON7

.long 0x008C07A3 @ REFRESH

.long 0x000000B1 @ BANKSIZE

.long 0x00000030 @ MRSRB6

.long 0x00000030 @ MRSRB7

例2，如图5-7所示，蜂鸣器是通过I/O口GPB0通过NPN 型三级管控制的。当三极管的发射极（C）为低电平，即GPB0 为高电平时，蜂鸣器会响，反之则不响。通过设置两者之间的时间则可以让蜂鸣器发出不同的响声，甚至唱歌。

程序如下：

int BellMain()

{

#define GPBCON (\*(volatile unsigned \*)0x56000010)

#define GPBDAT (\*(volatile unsigned \*)0x56000014)

#define GPBUP (\*(volatile unsigned \*)0x56000018)

void Delay(unsigned int);

GPBUP&=0XFFFFFFFE; //上拉使能GPB0

GPBCON&=0XFFFFFFC; //GPB0 设为输出

GPBCON|=0X0000001;

while(1)

{

GPBDAT &= 0xfe; //不能用!(非)

9013

20Ω

2k

VDD5V

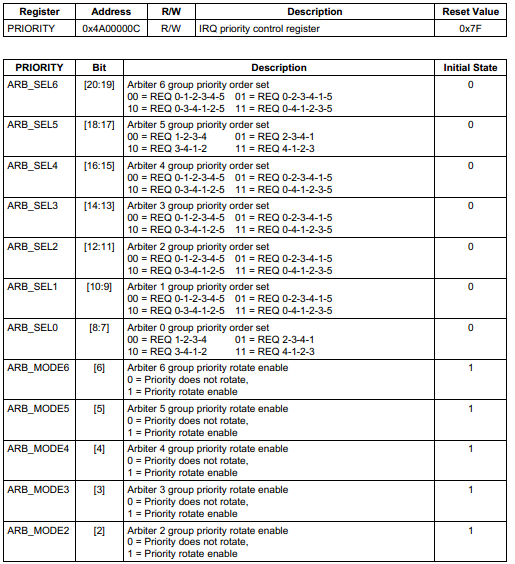
GPB0/TOUT0

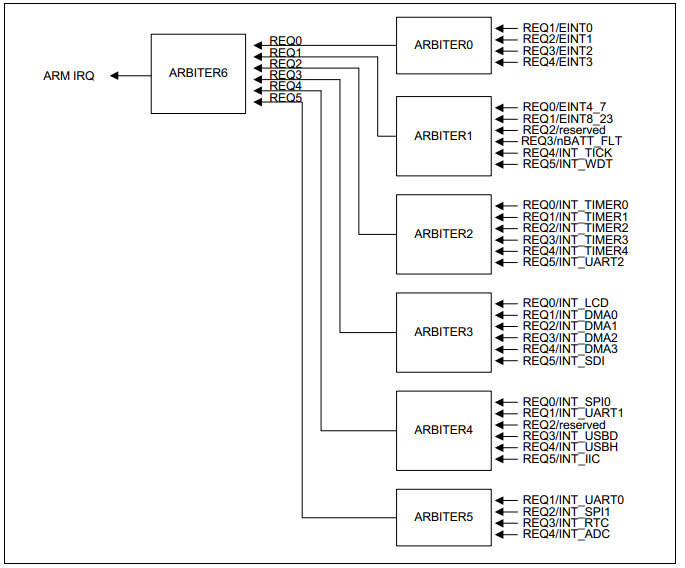
图5-7控制蜂鸣器原理图

Delay(40);

（4）优先级寄存器PRIORITY

根据要求，配置中断源的优先级





叙述工作原理

