实3 uC/os II 实时操作系统

“hello\_ucosii.c”文件内容：

#include <stdio.h>

#include "includes.h"

#include "system.h"

#include <io.h>

/\* Definition of Task Stacks \*/

#define TASK\_STACKSIZE 2048

OS\_STK task1\_stk[TASK\_STACKSIZE];

OS\_STK task2\_stk[TASK\_STACKSIZE];

/\* Definition of Task Priorities \*/

#define TASK1\_PRIORITY 1

#define TASK2\_PRIORITY 2

/\* Prints "Hello World" and sleeps for three seconds \*/

void task1(void\* pdata)

{

while (1)

{

printf("Hello uCOS-II\n");

OSTimeDlyHMSM(0, 0, 3, 0);

}

}

void task2(void\* pdata)

{

while (1)

{

unsigned int i;

// read switch

i = IORD(PIO\_SW\_BASE, 0);

// write ledr

IOWR(PIO\_LEDR\_BASE, 0, i);

}

}

int main(void)

{

OSTaskCreateExt(task1, NULL,

(void \*)&task1\_stk[TASK\_STACKSIZE-1],

TASK1\_PRIORITY, TASK1\_PRIORITY,

task1\_stk,

TASK\_STACKSIZE, NULL, 0);

OSTaskCreateExt(task2,NULL,

(void \*)&task2\_stk[TASK\_STACKSIZE-1],

TASK2\_PRIORITY, TASK2\_PRIORITY,

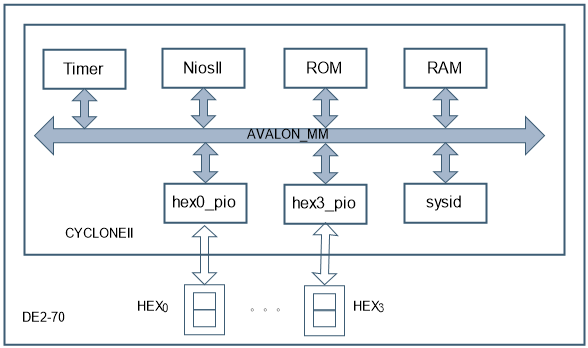
task2\_stk, TASK\_STACKSIZE, NULL, 0);

OSStart();

return 0;

}

实验二 数码管定时实验



* 编写C程序，参见图5.

#include "system.h"

#include "altera\_avalon\_timer\_regs.h"

#include "altera\_avalon\_pio\_regs.h"

#include "alt\_types.h"

#include "sys/alt\_irq.h"

#include <unistd.h>

alt\_u16 flag; //毫秒定时计数器

alt\_u16 second; //秒计数器

//alt\_u8 seg[16] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

alt\_u8 seg[16] = {

63, 6, 91, 79, 102, 109, 125, 7,

127, 111, 119, 124, 57, 94, 121, 113

}; // 0,1,2,....9, a, b, c, d, e,f

void delay(alt\_u32 cnt);

void seg\_dis(alt\_u16 dis\_num);

//毫秒定时中断函数

static void handle\_Timer0\_interrupts(void)

{

IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER\_BASE,0);//清TO标志

if(flag < 1000) flag = flag+1;

else

{ flag = 0; second++;

if(second==60) second=0; }

}

//主函数

int main (void)

{ flag = 0; second = 0;

//IOWR\_ALTERA\_AVALON\_PIO\_DATA(LED\_PIO\_BASE,0x00); //数码管位选管脚关闭//IOWR\_ALTERA\_AVALON\_PIO\_DATA(SEGCS4\_PIO\_BASE,0xf); //数码管片选管脚关闭alt\_irq\_register(TIMER\_IRQ,TIMER\_BASE,handle\_Timer0\_interrupts); //注册定时器中断函数 IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER\_BASE,7); //启动timer允许中断，连续计数

while(1) {//IOWR\_ALTERA\_AVALON\_PIO\_DATA(SEGCS4\_PIO\_BASE,0xf); //数码管片选管脚关闭IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_HEX0\_BASE,seg[(flag%100)/10]); //数码管位选显示变量second的个位//IOWR\_ALTERA\_AVALON\_PIO\_DATA(SEGCS4\_PIO\_BASE,0x7); //数码管片选个位

delay(2500); //delay5ms IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_HEX1\_BASE,seg[(flag%1000)/100]);

delay(2500); //delay5ms IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_HEX2\_BASE,seg[(second%10)]);

delay(2500); //delay5ms IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_HEX3\_BASE,seg[(second%100)/10]);

delay(2500); //delay5ms

}

}

//延时函数（延时时间为(2+2\*i)us）

void delay(alt\_u32 cnt)

{ alt\_u32 i =0;

while(i < cnt)

{ i++;}

}

实验七：C2H编译器

#include "system.h"

#include "altera\_avalon\_pio\_regs.h"

#include "alt\_types.h"

int main(void)\_\_attribute\_\_((

weak,alias("alt\_main")));

void delay (void)

{ alt\_u32 i=0, j=0;

while (i<100000) i++;

while (j<100000) j++;

return; }

int alt\_main (void)

{ alt\_u32 led = 0x2;

alt\_u8 dir = 0;

while (1)

{ if (led & 0x00020001)

{ dir = (dir ^ 0x1); }

if (dir)

{ led = led >> 1; }

else

{ led = led << 1; } IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_LEDR\_BASE,led);

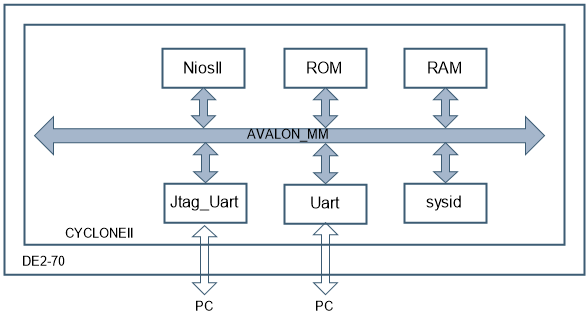
delay( );

}

return 0;

}

实验三 串口收发实验



uart\_printf.c

#include <stdio.h>

int main()

{ while(1)

{

printf("Hello from Nios II!\n");

int i=0;

while(i<100000) { i++;}

}

return 0;

}

uart.c

#include "alt\_types.h"

#include "io.h"

#include "sys/alt\_irq.h"

#include "system.h"

#include <stdio.h>

#include <unistd.h>

void Uart\_init(void);

void Uart\_tx(alt\_u8 txdb);

alt\_u8 uart\_rx\_temp;

alt\_u8 rx\_flag;

//函数名：Uart\_rx\_ISR

//功 能：串口接收数据中断服务函数

//参 数：无 返 回：无

void Uart\_rx\_ISR(void)

{ alt\_u8 uart\_status;

do{ uart\_status =

IORD\_16DIRECT(UART232\_BASE,8); }

while((uart\_status & 0x80)!=0x80); uart\_status = IORD\_16DIRECT(UART232\_BASE,8); uart\_rx\_temp = IORD\_8DIRECT(UART232\_BASE,0);

rx\_flag = 1;//串口数据接收标志位置位

}

//函数名：main 功能：主函数

//输入：无 返回：alt\_u8

int main()

{ Uart\_init(); //串口初始化函数 while(1)

{if(rx\_flag == 1)//接收到UART数据

{ rx\_flag = 0;//串口数据接收标志位清零

Uart\_tx('R');

Uart\_tx('X');

Uart\_tx('D');

Uart\_tx('=');

Uart\_tx(uart\_rx\_temp);

Uart\_tx(';');

}

}

return 0;

}

//函数名Uart\_init功能：串口初始化函数

//参 数：无 返 回：无

void Uart\_init(void)

{rx\_flag = 0;//串口数据接收标志位清零

IOWR\_16DIRECT(UART232\_BASE,12,0xc0);

IOWR\_16DIRECT(UART232\_BASE,8,0x0); //清状态标志

alt\_irq\_register

(UART232\_IRQ,UART232\_BASE,Uart\_rx\_ISR);

}

/\*

IOWR\_16DIRECT(基地址，偏移量，数据)16位写数据函数

IORD\_16DIRECT(基地址，偏移量，数据)16位读数据函数

串口的基地址是：UART\_BASE

偏移量0：rxdata,接收数据寄存器

偏移量4：txdata,发送数据寄存器

偏移量8：status,状态寄存器

偏移量12：control,控制寄存器

控制寄存器：第6位，ITRDY 准备好传输中断

第7位，IRRDY 准备好读取中断\*/

//函数名：Uart\_tx

//功 能：串口发送数据函数

//参 数：alt\_u8 txdb 返回：无

void Uart\_tx(alt\_u8 txdb)

{ alt\_u8 uart\_status;

do{uart\_status = IORD\_16DIRECT(UART232\_BASE,8); }//读状态寄存器

while((uart\_status & 0x40) != 0x40); //判断数据(TRDY == 1)是否发送完毕

uart\_status = IORD\_16DIRECT(UART232\_BASE,8); //再次读状态寄存器，清状态寄存器

IOWR\_8DIRECT(UART232\_BASE,4,txdb); //发送数据

}

//函数名：Uart\_bps\_change

//功 能：串口波特率设置函数

//参 数：alt\_u8 newbps

void Uart\_bps\_change(alt\_u8 newbps)

{

alt\_u16 div;

switch(newbps)

{

case(0x00): { div = 12; } break; //1200

case(0x01): { div = 24; } break; //2400

case(0x02): { div = 48; } break; //4800

case(0x03): { div = 96; } break; //9600

case(0x04): { div = 192; } break; //19200

case(0x05): { div = 384; } break; //38400

case(0x06): { div = 576; } break; //57600

case(0x07): { div = 1152; } break; //115200

case(0x08): { div = 288; } break; //28800

case(0x09): { div = 768; } break; //76800

case(0x0a): { div = 625; } break; //62500

case(0x0b): { div = 1250; } break; //125000

case(0x0c): { div = 2500; } break; //250000

case(0x0d): { div = 2304; } break; //230400

case(0x0e): { div = 3456; } break; //345600

case(0x0f): { div = 6912; } break; //691200

default: {} break;

}

div = (500000/div)-1;

IOWR\_16DIRECT(UART\_BASE,16,div);//设置波特率分频计数器

}

/\*Bps\_set指令值 0x00 0x01 0x02

0x03 0x04 0x05 0x06 0x07

波特率1200 2400 4800 9600 19200 38400 57600 115200

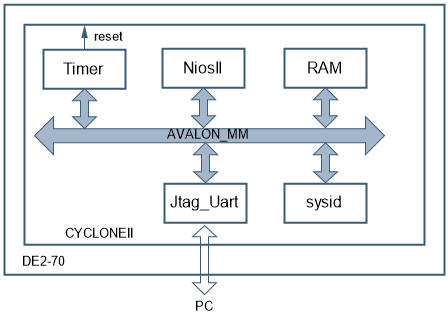
Bps\_set指令值 0x08 0x09 0x0a 0x0b

0x0c 0x0d 0x0e 0x0f

波特率 28800 76800 62500 125000

250000 230400 345600 691200\*/

实验四 看门狗定时器实验



#include "altera\_avalon\_timer\_regs.h"

#include "system.h"

#include <stdio.h>

//#include <unistd.h>

#include "alt\_types.h"

void delay(alt\_u32 cnt);

int main(void)

{ printf("\n\reseting..\n");

IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER\_0\_BASE,4); //启动看门狗

while(1)

{ delay(25000);

printf("running...,");

IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER\_0\_BASE,0); //喂狗

}

}

//延时函数（延时时间为(2+2\*i)us）

void delay(alt\_u32 cnt)

{ alt\_u32 i =0;

while(i < cnt) { i++; }

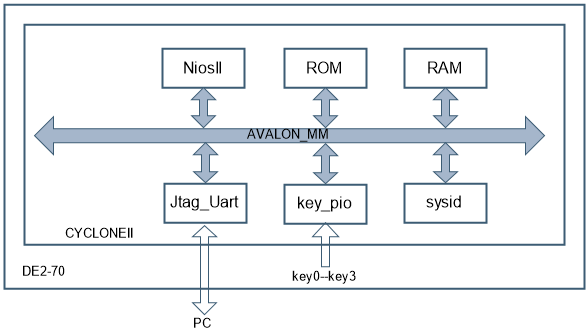
}

思考题

1、看门狗的作用是什么？如何实现？

2、应如何写喂狗程序？

实验五 按键中断实验



思考题

1、在边沿触发中断的设置下，按键电路是否要消除抖动？DE2-70的按键是否有抖动消除电路？

2、IORD\_ALTERA\_AVALON\_PIO\_DATA(base)与

IORD\_ALTERA\_AVALON\_PIO\_EDGE\_CAP(base)有何区别？

#include "sys/alt\_irq.h"

#include "altera\_avalon\_pio\_regs.h"

#include "system.h"

#include <stdio.h>

alt\_u8 int\_flag; //中断标志位

alt\_u8 edge\_capture\_value; //IO当前状态存储变量

/\* 按键中断服务程序\*/

static void handle\_key\_interrupts(void)

{ edge\_capture\_value=

IORD\_ALTERA\_AVALON\_PIO\_DATA(KEY\_PIO\_BASE);//EDGE\_CAP(KEY\_PIO\_BASE); //读取当前产生中断的IO值IOWR\_ALTERA\_AVALON\_PIO\_EDGE\_CAP(KEY\_PIO\_BASE,0x00);

int\_flag = 1; }

/\* 按键IO初始化 \*/

static void init\_button\_pio()

{ edge\_capture\_value = 0x00;

/\*\*//\* Enable all 4 button interrupts.\*/

IOWR\_ALTERA\_AVALON\_PIO\_IRQ\_MASK(KEY\_PIO\_BASE, 0x0f);

/\*\*//\* Reset the edge capture register. \*/

IOWR\_ALTERA\_AVALON\_PIO\_EDGE\_CAP(KEY\_PIO\_BASE, 0x00);

/\*\*//\* Register the interrupt handler. \*/

alt\_irq\_register(KEY\_PIO\_IRQ, KEY\_PIO\_BASE, handle\_key\_interrupts); }

int main(void) //主函数

{ alt\_u8 key\_value =0x00;

int\_flag = 0; init\_button\_pio();//初化按键寄存器 while(1) {

if(int\_flag == 1)

{ switch(edge\_capture\_value)

{ case 0x1:

{ key\_value = 0x00; break; }

case 0x2:

{key\_value = 0x01; break; }

case 0x4:

{key\_value = 0x02; break; }

case 0x8:

{key\_value = 0x03; break; }

default:

{ key\_value = edge\_capture\_value; break; }

}

int\_flag = 0;

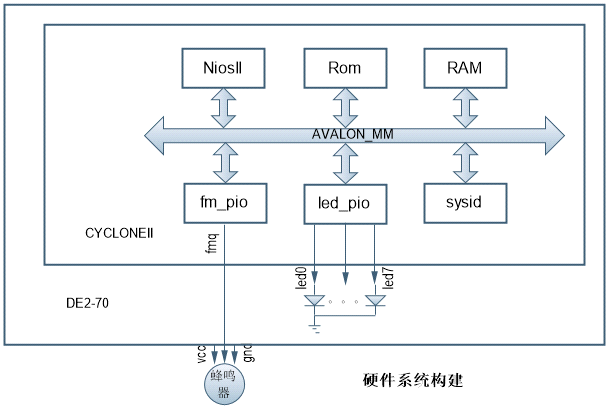
printf("key value

= %d\n",key\_value);//JTAG UART打印当前键值}

}

}

实验一 蜂鸣器和流水灯实验



思考题

1、如何改变蜂鸣器鸣叫的强度？

2、如何改变流水灯的流水方向？

3.、如何测量延时函数Delay(1)和Delay(2)的实际延时时间？

蜂鸣器软件工程

#include "alt\_types.h"

#include "altera\_avalon\_pio\_regs.h"

#include "sys/alt\_irq.h"

#include "system.h"

#include <stdio.h>

#include <unistd.h>

void delay(alt\_u32 cnt); //延时函数申明

int main()

{ //alt\_u8 cnt;

IOWR\_ALTERA\_AVALON\_PIO\_DATA(FM\_PIO\_BASE,0x0); //初始化蜂鸣器控制端口

while(1)

{

IOWR\_ALTERA\_AVALON\_PIO\_DATA(FM\_PIO\_BASE,0x1);

delay(500000);

IOWR\_ALTERA\_AVALON\_PIO\_DATA(FM\_PIO\_BASE,0x0);

delay(500000);

}

return 0;

}

void delay(alt\_u32 cnt) //延时函数（延时时间为(2+2\*i)us）

{

alt\_u32 i =0;

while(i < cnt) { i++;}

}

流水灯的程序

#include "alt\_types.h"

#include "altera\_avalon\_pio\_regs.h"

#include "sys/alt\_irq.h"

#include "system.h"

#include <stdio.h>

#include <unistd.h>

void delay(alt\_u32 cnt);

int main()

{

alt\_u8 cnt;

IOWR\_ALTERA\_AVALON\_PIO\_DATA(LED\_PIO\_BASE,0xff);//初始化8个LED都处于熄灭状态

while(1)

{ for(cnt=0;cnt<8;cnt++)//循环LED显示

{

delay(500000); //延时大约1s

IOWR\_ALTERA\_AVALON\_PIO\_DATA(LED\_PIO\_BASE,~(1 << cnt)); //给8个LED送数据}

}

return 0;

}

void delay(alt\_u32 cnt) //延时函数（延时时间为(2+2\*i)us）

{

alt\_u32 i =0;

while(i < cnt) { i++;}

}

大作业：

#include "system.h"

#include <sys/alt\_irq.h>

#include "alt\_types.h"

#include <io.h>

#define SEG7\_SET(index, seg\_mask) IOWR(SEG7\_DISPLAY\_BASE,index,seg\_mask)

#define SEG7\_NUM 4

static alt\_u16 shu[] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBF, 0xE0,0xFF,0xF6};// 0,1,2,....9

void SEG7\_Clear(void){

int i;

for(i=0;i<SEG7\_NUM;i++){

SEG7\_SET(i, 0x00);

}

}

void SEG7\_tt(void){

SEG7\_SET(0, 0x8C);

SEG7\_SET(1, 0xE0);

SEG7\_SET(2, 0x8C);

SEG7\_SET(3, 0xE0);

}

int alt\_main(void)

{

SEG7\_tt();

usleep(5000000);

alt\_u16 s1[][4] = {{shu[0],shu[2],shu[0],shu[2]}, {shu[0],shu[1],shu[0],shu[1]}};

SEG7\_Clear();

int i = 0;

int time;

while(1){

for(time = 2 ; time > 0 ; time--){

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

{

SEG7\_SET(i, s1[2-time][i]);

}

IOWR(LEDG\_BASE,0,0x01);

IOWR(LEDR\_BASE,0,0x08);

usleep(1000000);

}

for(time = 2 ; time > 0 ; time--){

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

{

SEG7\_SET(i, s1[2-time][i]);

}

IOWR(LEDR\_BASE,0,0x01);

IOWR(LEDG\_BASE,0,0x02);

usleep(1000000);

}

for(time = 2 ; time > 0 ; time--){

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

{

SEG7\_SET(i, s1[2-time][i]);

}

IOWR(LEDR\_BASE,0,0x06);

IOWR(LEDG\_BASE,0,0x00);

usleep(1000000);

}

}

}