

**接口技术综合实验报告**

|  |
| --- |
| 姓 名： 洪炜豪 |
| 学 号： U202115512 |
| 小组成员： 金亚璞 |
| 学 院： 计算机科学与技术学院 |
| 专 业： 计算机科学与技术 |
| 班 级： CS 2106 |
| 指导教师： 胡迪青 |

2023年 11月 20 日

目录

[**1.实验目的** 1](#_Toc102151226)

[**2.实验环境** 1](#_Toc102151227)

[**3.项目设计与实现** 2](#_Toc102151228)

[**3.1 总体设计与实现** 2](#_Toc102151229)

[**3.2 硬件实现** 2](#_Toc102151230)

[**3.3 软件实现** 6](#_Toc102151231)

[**4.系统运行结果** 7](#_Toc102151232)

[**5.总结** 9](#_Toc102151233)

[**6.源代码** 10](#_Toc102151234)

1. **实验目的**

基于 Imagination 大学计划开源的 RVfpgaSoC，通过讲课和实验，使学生了解和掌握接口技术， 并采用基本的接口模块，运用EDA工具通过IP集成的方式搭建一个简单的嵌入式应用系统。

目标1：熟悉硬件设计工具的功能、特点及使用方法，掌握基于主流设计工具设计硬件模块或系统的流程和方法，使学生具备硬件系统的开发和调试能力； 目标2：基于接口技术理论知识，根据实验任务要求，设计硬件功能部件和简单系统，掌握硬件及系统设计方法，培养学生发现问题、分析问题、解决问题的实践能力； 目标3：通过实验检查和验收过程中的问答，阐述实验设计等活动，培养学生与专业有关的沟通与表达能力。

实验要求：在前面实验的基础上，在RVfpga\_SoC处理器系统中至少再添加一个新的外设模块，完成系统的硬件设计，生成比特流。

注： 添加的新外设必须是前面实验没有使用过的，这个外设模块可以是Vivado自带的、也可以是现成的免费的IP模块、还可以是自定制的IP模块。

在上述实现的RVfpga\_SoC处理器系统上，编写一个综合应用软件。该综合应用系统要能够充分展示同学们的想象力和创造性。

1. **实验环境**

Vivado:

v2019.2 (64-bit)  
SW Build: 2708876 on Wed Nov 6 21:40:23 MST 2019  
IP Build: 2700528 on Thu Nov 7 00:09:20 MST 2019

VSCode:

版本: 1.84.2 (user setup)

提交: 1a5daa3a0231a0fbba4f14db7ec463cf99d7768e

日期: 2023-11-09T10:51:52.184Z

OS: Windows\_NT x64 10.0.19044

处理器：

11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz

操作系统：

版本 Windows 10 专业版

版本号 21H2

安装日期 ‎2021/‎8/‎5

操作系统内部版本 19044.3086

**3.项目设计与实现**

**3.1 总体设计与实现**

先设计pwm音频输出外设模块，集成内置音频文件，然后将PWM模块集成到RVfpga\_SoC处理器系统中。完成硬件设计后，使用Vivado生成比特流文件。接着在platform上编写c语言代码运行一个系统，这个系统是时钟系统，支持设定日期时间，闹铃，倒计时，时间到就激活pwm模块响铃。最后进行测试。

**3.2 硬件实现**

1. 设计PWM音频输出功能：

- 设计PWM模块以实现音频输出。

- 集成内置音频文件。

PWM（Pulse Width Modulation）最简单的理解就是MCU内部有个定时器，定时器特定的时间内将GPIO的电平翻转一下。对于翻转的时间间隔的比例就是占空比，而持续这两个过程则可以算为一个周期。

PWM模块被设计用来生成对应于音乐音符的音频音调，我的设计中简单地演奏了《See You Again》的前奏。模块设置为能够产生不同频率的方波，这些频率对应于音乐音符。下面是代码的详细解释：

### 模块和宏定义

```verilog

`timescale 1ns / 1ps

```

这行代码定义了在仿真中使用的时间单位和精度。1ns表示时间单位是纳秒，1ps表示时间精度是皮秒。

```verilog

// 高低中音宏定义

`define L1 262 //低音1

...

`define H7 1976 //高音7

```

这些行定义了音符的频率值，低音（L）、中音（C）、高音（H），用于后续PWM信号的产生。

### 模块接口

```verilog

module My\_Controller(

input clk,

input rst,

input [31:0] enable,

output reg AUD\_PWM,

output wire AUD\_SD

);

```

- `clk` 是模块的时钟输入。

- `rst` 是复位信号，用于重置模块状态。

- `enable` 是一个32位的输入信号，用于启用或禁用音频输出。

- `AUD\_PWM` 是音频输出的PWM信号。

- `AUD\_SD` 是一个输出信号，当 `enable` 为高时，它允许音频输出。

### 状态计数器

```verilog

localparam state\_top=24'd12500000-1;

reg [23:0] state\_cnt;

```

这里定义了一个局部参数 `state\_top` 和一个状态计数器 `state\_cnt`。每当 `state\_cnt` 达到 `state\_top`（假定时钟频率下的250ms计数值）时，状态计数器复位。

### 状态计数器逻辑

```verilog

always@(posedge clk or negedge rst)

if(!rst)

state\_cnt<=0;

else if(state\_cnt<state\_top)

state\_cnt<=state\_cnt+1;

else

state\_cnt<=0;

```

这个始终块控制状态计数器的行为。在每个时钟上升沿，如果复位信号为低，则计数器清零；如果计数器值小于 `state\_top`，则计数器增加；否则，计数器复位。

### 状态机逻辑

```verilog

reg [20:0] cnt\_top;

wire state\_cnt\_done=(state\_cnt==state\_top)?1:0;

always@(posedge clk or negedge rst)

if(!rst)

state<=0;

else if(state\_cnt\_done)begin

if(state<73)

state<=state+1;

else

state<=0;

end

else

state<=state;

```

这部分定义了一个7位的状态寄存器 `state`，且在每个时钟上升沿更新状态。如果复位信号为低，状态清零；如果状态计数器完成（`state\_cnt\_done`），则状态递增或复位。

### 音符选择逻辑

```verilog

always@(\*)

begin

case(state)

0 :cnt\_top<=`C5;

...

73 :cnt\_top<=`C5;

default:;

endcase

end

```

这个组合逻辑块根据当前状态选择对应的音符频率值。每种状态对应一个 `case` 分支，用于设置 `cnt\_top`，这决定了PWM信号的频率。

### PWM频率计数器

```verilog

reg [26:0] cnt;

always@(posedge clk or negedge rst)

if(!rst)

cnt<=0;

else if(cnt<50\_000\_000/cnt\_top-1)

cnt<=cnt+1;

else

cnt<=0;

```

定义了一个计数器 `cnt`，用于产生PWM信号。

### PWM产生逻辑

```verilog

always@(posedge clk or negedge rst)

if(!rst)

AUD\_PWM<=0;

else

AUD\_PWM<=(cnt<cnt\_top/2)?1:0;

```

这个始终块生成PWM信号。它根据 `cnt\_top` 控制PWM信号的占空比，`cnt` 在到达 `cnt\_top/2` 时翻转 `AUD\_PWM` 信号的状态。

### 主要功能

主要实现了一个有限状态机，每个状态对应旋律中的一个音符。`state\_cnt`计数器为状态转换之间的延迟，以保持每个音符的持续时间。当`state\_cnt\_done`信号被断言时，状态切换，改变正在播放的音符。

`cnt\_top`值是当前音符频率的周期，用于生成PWM信号。`AUD\_PWM`信号以音符的频率切换，当连接到扬声器或音频输出设备时，创建可听见的音调。

`AUD\_SD`信号用于根据`enable`输入启用或禁用音频输出，允许控制音频活动的时间。

总体来说，这个模块通过控制 `AUD\_PWM` 信号的占空比，产生一系列对应不同音符频率的方波信号，用于在连接的扬声器上播放音乐。

文本, 信件

描述已自动生成

图3-1 封装好后的pwm模块

2. 选择和集成PWM音频输出模块：

- 在Vivado中选择自定义的IP模块PWM模块。

- 将PWM模块集成到RVfpga\_SoC处理器系统中。在处理器与PWM模块之间建立适当的接口，并确保它们能够有效地通信。

图形用户界面

描述已自动生成

图3-2 地址分配

文本

描述已自动生成

图3-3 引脚约束

图示, 示意图

描述已自动生成

图3-4 块设计

3. 生成比特流：

- 完成硬件设计后，使用Vivado生成比特流文件。

**3.3 软件实现**

1. 编写时钟系统软件：

- 使用C语言在RVfpga\_SoC处理器系统上开发时钟系统的软件部分。

- 实现用户界面，允许用户设置日期、时间、响铃时间和倒计时。

- 设计软件以侦听时钟信号，并在指定的时间激活PWM模块播放预设的音频。

2. 系统功能：

- 日期和时间设置：允许用户设置当前日期和时间。

- 闹铃功能：用户可以设置闹铃时间。

- 倒计时功能：用户可以设置倒计时，到时间后触发闹铃。

- 音频播放：当时钟响起或倒计时结束时，通过PWM模块播放音频。

3. 测试和调试：

- 在RVfpga\_SoC系统上部署软件，并进行测试以确保所有功能按预期工作。

- 调试可能出现的问题，如音频播放质量、定时准确性等。

**4.****系统运行结果**

我的时钟系统通过读取Switch的输入进行设置，共分配了5个操控的开关，分别是16,4,3,2,1号Switch，完成以下几个部分的功能：

16号开关：进行日期设定，开启后第15-10号配置日期，9-6 号配置月份，5-1号配置日期，支持2000-1-1到2063-12-31日期设定。

图片包含 游戏机, 电路

描述已自动生成

图4-1 日期配置测试

4号开关：进行日期的查看，打开后查看当前日期。

3号开关：打开后进行定时器设定，可设定倒计时xx分xx秒，时间到了出发闹铃。

图片包含 图形用户界面

描述已自动生成

图4-2 配置定时器

3号＋4号开关：启动定时器。

2号开关：设定闹铃，当系统时间等于闹铃时间时响铃。

2号＋4号开关：启动闹铃。

1号开关：设定当前时间xx时xx分。

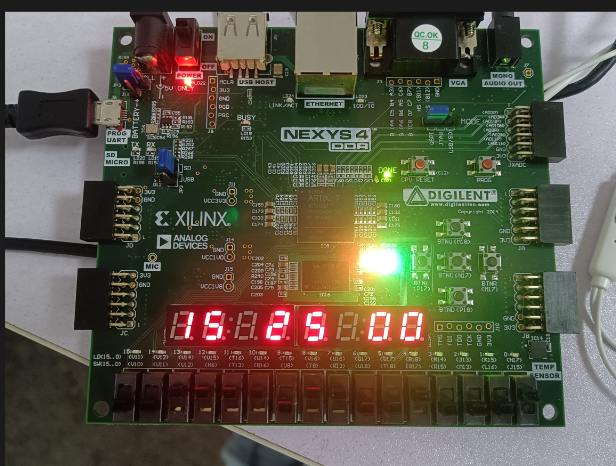


图4-3 设定当前时间

所有开关闭合时显示当前时间xx时xx分xx秒。

支持日期的自动转换，考虑了闰年，各个月份天数不同的情况。经测试均正常。

**问题发现与解决：**

1. 发现在设定日期时间的时候系统时钟也应该继续运行而不是就此暂停，于是在进入其他case时将h，m，s保存到hh，mm，ss中，在case语句结束后，如果发现hh，mm，ss改变后，就进行自增，返回case0时再把其读到h，m，s中进行自增，与此同时把hh，mm，ss置为默认值0xff。
2. 发现pwm模块不发出声音：因为人耳只能听到特定频率的声音，而我没有正确设定声音的频率。正确配置频率后解决了问题。

**5.总结**

**实验体验**

* **探索与学习**：通过这个实验，我得以深入探索RVfpga\_SoC处理器系统和硬件设计工具（如Vivado）。这个过程不仅丰富了我的理论知识，还提高了我实践操作的能力。
* **创意实现**：设计时钟系统允许我将创造性思维应用于实际项目中。从设计到实现的过程，让我体验到了从无到有创造一个完整系统的成就感。

**技术技能的提升**

* **硬件设计**：通过在RVfpga\_SoC处理器系统中添加PWM音频输出模块，我学习了如何设计和集成新的硬件模块。
* **软件开发**：编写时钟系统的软件部分提升了我的编程技能，特别是在嵌入式系统环境中编写和调试代码的能力。

**实际问题解决**

* **调试与优化**：在实验过程中，我遇到了各种技术挑战，如硬件与软件的兼容性问题，以及信号没有正常输出，模块设计不理想等问题。这些问题的解决过程锻炼了我的问题分析和解决能力。
* **创新思维**：设计一个时钟系统需要创意，特别是在功能和用户交互方面。这激发了我的创新思维，让我学会了如何在技术项目中融入创意元素。

**沟通与表达**

* **技术沟通**：通过与同伴和教师的交流，我学会了如何清晰地表达技术观点和设计思路。这种沟通对于理解复杂概念和解决实际问题非常重要。并且我学会了在遇到问题时如何自主寻找解决方案，这增强了我的独立思考和问题解决能力。
* **表达与报告**：撰写实验报告和展示设计过程是这个实验的重要部分。这不仅让我学会了如何书面表达技术思想，还锻炼了我的口头表达能力。

总体来说，这个实验不仅提升了我的技术能力，还锻炼了我的创新思维和解决问题的能力。通过实际设计和实现一个综合应用系统，我得到了宝贵的实践经验，这将对我未来的学习和职业生涯产生深远影响。

最后，再次感谢胡迪青老师和李丹阳助教在实验过程中对我的点拨和帮助！

**6.源代码**

**My\_Controller.v**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2023/11/10 18:05:11

// Design Name:

// Module Name: My\_Controller

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

//高低中音宏定义

`define L1 262 //低音1

`define L2 294 //低音2

`define L3 330 //低音3

`define L4 349 //低音4

`define L5 392 //低音5

`define L6 440 //低音6

`define L7 494 //低音7

`define C1 523 //中音1

`define C2 587 //中音2

`define C3 659 //中音3

`define C4 699 //中音4

`define C5 784 //中音5

`define C6 880 //中音6

`define C7 988 //中音7

`define H1 1046 //高音1

`define H2 1175 //高音2

`define H3 1319 //高音3

`define H4 1397 //高音4

`define H5 1568 //高音5

`define H6 1760 //高音6

`define H7 1976 //高音7

module My\_Controller(

input clk,

input rst,

input [31:0] enable,

output reg AUD\_PWM,

output wire AUD\_SD

);

//250ms跳转一次,需要计数12500000

localparam state\_top=24'd12500000-1;

reg [23:0] state\_cnt;

assign AUD\_SD = enable[0];

always@(posedge clk or negedge rst)

if(!rst)

state\_cnt<=0;

else if(state\_cnt<state\_top)

state\_cnt<=state\_cnt+1;

else

state\_cnt<=0;

//状态跳转标志信号

wire state\_cnt\_done=(state\_cnt==state\_top)?1:0;

reg [4:0] state;

reg [20:0] cnt\_top;

always@(posedge clk or negedge rst)

if(!rst)

state<=0;

else if(state\_cnt\_done)begin

if(state<31)

state<=state+1;

else

state<=0;

end

else

state<=state;

always@(\*)

begin

case(state)

0 :cnt\_top<=`C5;

1 :cnt\_top<=`H2;

2 :cnt\_top<=`H1;

3 :cnt\_top<=`C5;

4 :cnt\_top<=`C5;

5 :cnt\_top<=`H1;

6 :cnt\_top<=`H2;

7 :cnt\_top<=`H3;

8 :cnt\_top<=`H2;

9 :cnt\_top<=`H1;

10 :cnt\_top<=`H2;

11 :cnt\_top<=`C5;

12 :cnt\_top<=`H2;

13 :cnt\_top<=`H1;

14 :cnt\_top<=`C5;

15 :cnt\_top<=`C5;

16 :cnt\_top<=`H1;

17 :cnt\_top<=`H2;

18 :cnt\_top<=`H3;

19 :cnt\_top<=`H2;

20 :cnt\_top<=`H1;

21 :cnt\_top<=`H2;

22 :cnt\_top<=`C5;

23 :cnt\_top<=`H2;

24 :cnt\_top<=`H1;

25 :cnt\_top<=`C5;

26 :cnt\_top<=`C5;

27 :cnt\_top<=`H1;

28 :cnt\_top<=`H3;

29 :cnt\_top<=`C5;

30 :cnt\_top<=`C6;

31 :cnt\_top<=`C5;

32 :cnt\_top<=`C5;

33 :cnt\_top<= 0;

34 :cnt\_top<=`C1;

35 :cnt\_top<=`C2;

36 :cnt\_top<=`C2;

37 :cnt\_top<=`C1;

38 :cnt\_top<=`C2;

39 :cnt\_top<=`C3;

40 :cnt\_top<= 0;

41 :cnt\_top<=`C3;

42 :cnt\_top<=`C5;

43 :cnt\_top<=`C6;

44 :cnt\_top<=`C7;

45 :cnt\_top<=`C6;

46 :cnt\_top<=`C5;

47 :cnt\_top<=`C3;

48 :cnt\_top<=`C2;

49 :cnt\_top<=`C2;

50 :cnt\_top<=`C1;

51 :cnt\_top<=`C2;

52 :cnt\_top<=`C2;

53 :cnt\_top<=`C3;

54 :cnt\_top<=`C1;

55 :cnt\_top<=`C1;

56 :cnt\_top<= 0;

57 :cnt\_top<= 0;

58 :cnt\_top<=`C1;

59 :cnt\_top<=`C3;

60 :cnt\_top<=`C5;

61 :cnt\_top<=`C6;

62 :cnt\_top<=`C5;

63 :cnt\_top<=`C5;

64 :cnt\_top<= 0;

65 :cnt\_top<=`C1;

66 :cnt\_top<=`C2;

67 :cnt\_top<=`C2;

68 :cnt\_top<=`C1;

69 :cnt\_top<=`C2;

70 :cnt\_top<=`C3;

71 :cnt\_top<= 0;

72 :cnt\_top<=`C3;

73 :cnt\_top<=`C5;

default:;

endcase

end

reg [26:0] cnt;

always@(posedge clk or negedge rst)

if(!rst)

cnt<=0;

else if(cnt<50\_000\_000/cnt\_top-1)

cnt<=cnt+1;

else

cnt<=0;

always@(posedge clk or negedge rst)

if(!rst)

AUD\_PWM<=0;

else

AUD\_PWM<=(cnt<cnt\_top/2)?1:0;

endmodule

**AUDIO.c**

#define PWM\_OUT       0x80140000

#define SegEn\_ADDR    0x80001038

#define SegDig\_ADDR   0x8000103C

#define RPTC\_CNTR     0x80001200

#define RPTC\_HRC      0x80001204

#define RPTC\_LRC      0x80001208

#define RPTC\_CTRL     0x8000120c

#define IO\_LEDR     0x80100000

#define IO\_SW       0x80100008

#define READ\_IO(dir) (\*(volatile unsigned \*)dir)

#define WRITE\_IO(dir, value) { (\*(volatile unsigned \*)dir) = (value); }

unsigned int getshow(unsigned int h, unsigned int m, unsigned int s) {

    unsigned int show = 0;

    unsigned int hshow = 0, mshow = 0, sshow = 0;

    unsigned int high = 0, low = 0;

    high = h / 10;

    low = h % 10;

    hshow |= high;

    hshow <<= 4;

    hshow |= low;

    high = m / 10;

    low = m % 10;

    mshow |= high;

    mshow <<= 4;

    mshow |= low;

    high = s / 10;

    low = s % 10;

    sshow |= high;

    sshow <<= 4;

    sshow |= low;

    show |= hshow << 24;

    show |= mshow << 12;

    show |= sshow;

    return show;

}

int main ( void ) {

    static int dayt[][13]={

        {0,31,28,31,30,31,30,31,31,30,31,30,31},

        {0,31,29,31,30,31,30,31,31,30,31,30,31}

    };

    struct date{

        int year;

        int month;

        int day;

    };

    int counter\_value;

    unsigned int h = 0, m = 0, s = 0, show = 0 ,show1 = 0, control=0 ,control1 = 0;

    unsigned int seth = 25, setm = 61;

    unsigned int noisy = 0;

    unsigned int hh = 0xFF, mm = 0xff, ss = 0xff;

    unsigned int sw;

    unsigned int alarm\_hour = 99; // 闹钟小时，初始设置为无效值

    unsigned int alarm\_minute = 99; // 闹钟分钟，初始设置为无效值

    unsigned int alarm\_second = 99;

    unsigned int alarm\_read = 0;

    unsigned int date\_read = 0;

    unsigned int date\_display = 0;

    int leap;

    WRITE\_IO(PWM\_OUT, 0);

    WRITE\_IO(SegEn\_ADDR, 0x24);

    WRITE\_IO(RPTC\_LRC, 0x2FFFFFF);

    struct date date\_;

    while (1) {

        sw = READ\_IO(IO\_SW);

        control = sw & 0xf;

        control1 = sw & 0x8000;

        switch(control) {

            case 0:

                leap=(((date\_.year%4==0)&&(date\_.year%100!=0))||(date\_.year%400==0));

                alarm\_read = 0;

                date\_read = 0;

                date\_display = 0;

                if(hh!=0xFF || mm!=0xFF || ss!=0xFF) {

                    h = hh;

                    m = mm;

                    s = ss;

                    hh = 0xff;

                    mm = 0xff;

                    ss = 0xff;

                }

                s++;

                if(s==60) {

                    s = 0;

                    m++;

                }

                if(m==60) {

                    m = 0;

                    h++;

                }

                if(h==24) {

                    h = 0;

                    date\_.day += 1;

                    if ( date\_.day == dayt[leap][date\_.month] + 1){

                        date\_.day = 0;

                        date\_.month += 1;

                        if ( date\_.month == 13){

                            date\_.month = 0;

                            date\_.year += 1;

                        }

                    }

                }

                break;

            case 1:

                h = (sw >> 10) & 0x3f;

                m = (sw >> 4) & 0x3f;

                s = 0;

                break;

            case 2:

                if(hh==0xff && mm==0xff && ss==0xff) {

                    hh = h;

                    mm = m;

                    ss = s;

                }

                alarm\_read = 1;

                alarm\_hour = (sw >> 10) & 0x3f;

                alarm\_minute = (sw >> 4) & 0x3f;

                alarm\_second = 0;

                break;

            case 4:

                if(hh==0xff && mm==0xff && ss==0xff) {

                    hh = h;

                    mm = m;

                    ss = s;

                }

                h = 0;

                m = (sw >> 10) & 0x3f;

                s = (sw >> 4) & 0x3f;

                break;

            case 6:

                if(hh==0xff && mm==0xff && ss==0xff) {

                    hh = h;

                    mm = m;

                    ss = s;

                }

                // 启动闹钟

                show1 = 1;

                break;

            case 8:

                if(hh==0xff && mm==0xff && ss==0xff) {

                    hh = h;

                    mm = m;

                    ss = s;

                }

                date\_display = 1;

                break;

            case 12:

                if(h==0 && m==0 && s==0) {

                    noisy = 30;

                    h = 99;

                    m = 99;

                    s = 99;

                }

                else if(h==99 && m==99 && s==99) {

                    ;

                }

                else {

                    if(s>0) {

                        s--;

                    }

                    else if(m>0) {

                        m--;

                        s = 59;

                    }

                }

                break;

        }

        switch(control1) {

            case 0x8000:

                if(hh==0xff && mm==0xff && ss==0xff) {

                    hh = h;

                    mm = m;

                    ss = s;

                }

                date\_.year = (sw >> 9) & 0x3f;

                date\_.month = (sw >> 5) & 0xf;

                date\_.day = sw  & 0x1f;

                date\_read = 1;

                break;

        }

        if(h==seth && m==setm) {

            noisy = 10;

            seth = 25;

            setm = 61;

        }

        if(hh!=0xFF || mm!=0xFF || ss!=0xFF) {

            ss++;

            if(ss==60) {

                ss = 0;

                mm++;

            }

            if(mm==60) {

                mm = 0;

                hh++;

            }

            if(hh==24) {

                hh = 0;

            }

        }

        if (alarm\_read == 1){

            show = getshow(alarm\_hour, alarm\_minute, alarm\_second);

        }

        else if (date\_read == 1){

            show = getshow(date\_.year, date\_.month, date\_.day);

        }

        else if (date\_display == 1){

            show = getshow(date\_.year, date\_.month, date\_.day);

        }

        else {

            show = getshow(h, m, s);

        }

        WRITE\_IO(SegDig\_ADDR, show);

        if (show1 && h == alarm\_hour && m == alarm\_minute) {

            noisy = 10;

            show1 = 0; // 重置闹钟设置，避免重复激活

        }

        if(noisy>0) {

            WRITE\_IO(PWM\_OUT, 1);

            noisy--;

        }

        else {

            WRITE\_IO(PWM\_OUT, 0);

        }

        WRITE\_IO(RPTC\_CTRL, 0xC0);

        WRITE\_IO(RPTC\_CTRL, 0x21);

        while(1){

            counter\_value = READ\_IO(RPTC\_CTRL);

            counter\_value = counter\_value & 0x40;

            if(counter\_value!=0){

                break;

            }

        }

    }

    return(0);

}

1. 原创性声明

本人郑重声明本报告内容，是由本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明的内容外，本报告不包含任何其他个人或集体已经公开的发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：洪炜豪**图片包含 物体, 手表, 钟表, 鸟

描述已自动生成**

1. 对课程实验的学术评语（教师填写）
2. 对课程实验的评分（教师填写）

|  |  |  |  |
| --- | --- | --- | --- |
| 评分项目  （分值） | 报告撰写  （40分） | 实验过程  （60分） | 最终评定  （100分） |
| 得分 |  |  |  |

指导教师签字：