南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

计算机科学与工程学院

“硬件课程设计 II”报告书

题目：可沿导轨运动的机械臂控制器设计

小组长：　蒋旭钊

小组成员：王迪

小组成绩：

日期：　　2021年 12 月29 日

小组成员学号	姓名	承担的任务	成绩
918106840727	蒋旭钊	写控制函数，机械臂调试，报告编写	
918106840746	王迪	写功能函数，机械臂调试，报告编写	

题目的要求

该题的被控对象“沿导轨运动的机械臂”具有串行通信接口，可以通过串口来接收命令，以便控制沿导轨运动的机械臂动作。

1、串口通信协议遵循 MODBUS 协议。
2、以 Zynq 芯片为核心来构建控制器的硬件平台，硬件平台上至少应该包括 UART、键盘、LED 指示灯等部件。
3、系统基于无操作系统环境下开发，所有硬件接口部件的操作不得调用开发环境中提供的函数来实现。
4、控制器的基本功能要求：可以与沿轨道运动的机械臂进行串口通信，通信协议见；可以手动控制机械臂在导轨上运动；可以手动控制 6 个轴的顺时针、逆时针转；可以手动控制 3 号传送带启动、停止；可以手动控制出箱子。
5、完成复位功能设计，即按一个按键后，机械臂恢复最初位置。
6、设计自动搬运物体十次的功能，每个颜色的箱子放在对应颜色的方框内。

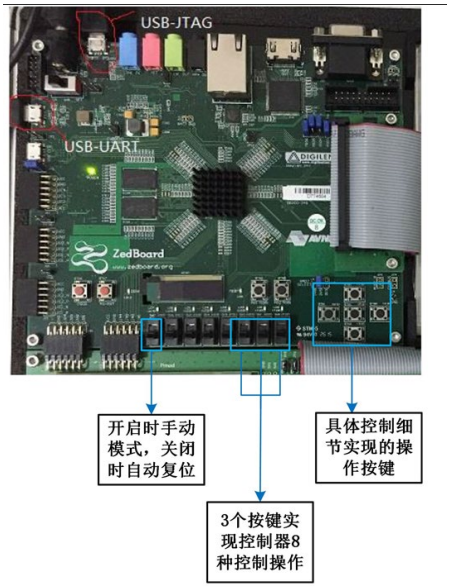
总体设计

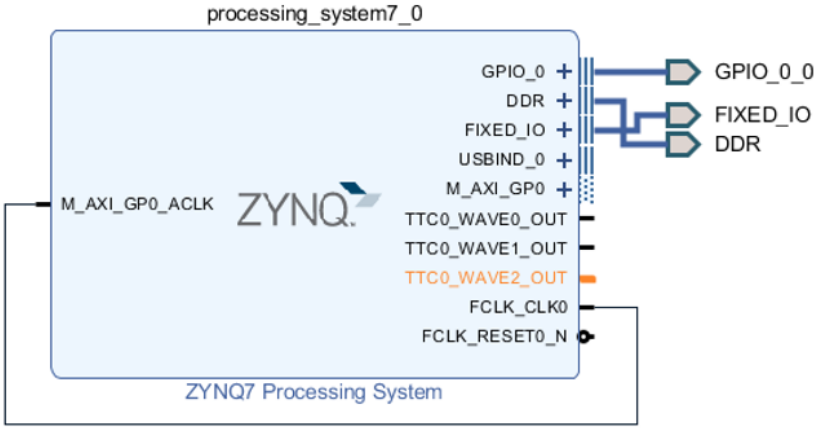
一、系统的组成及硬件环境

1、系统总体结构图，指出控制器与被控对象之间的连接结构

嵌入式系统的开发，通常需要构造一个交叉编译环境（即建立宿主机—目标机的开发架构）。

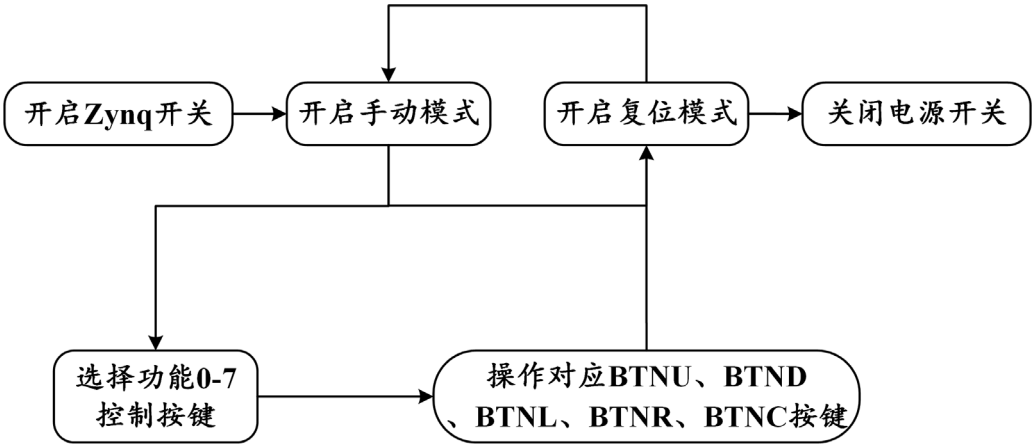
2、控制器的硬件组成框图，即基于 ZYNQ 芯片的控制器硬件组成（只与该项目有关的硬件），并在框图中表明主要的信号。



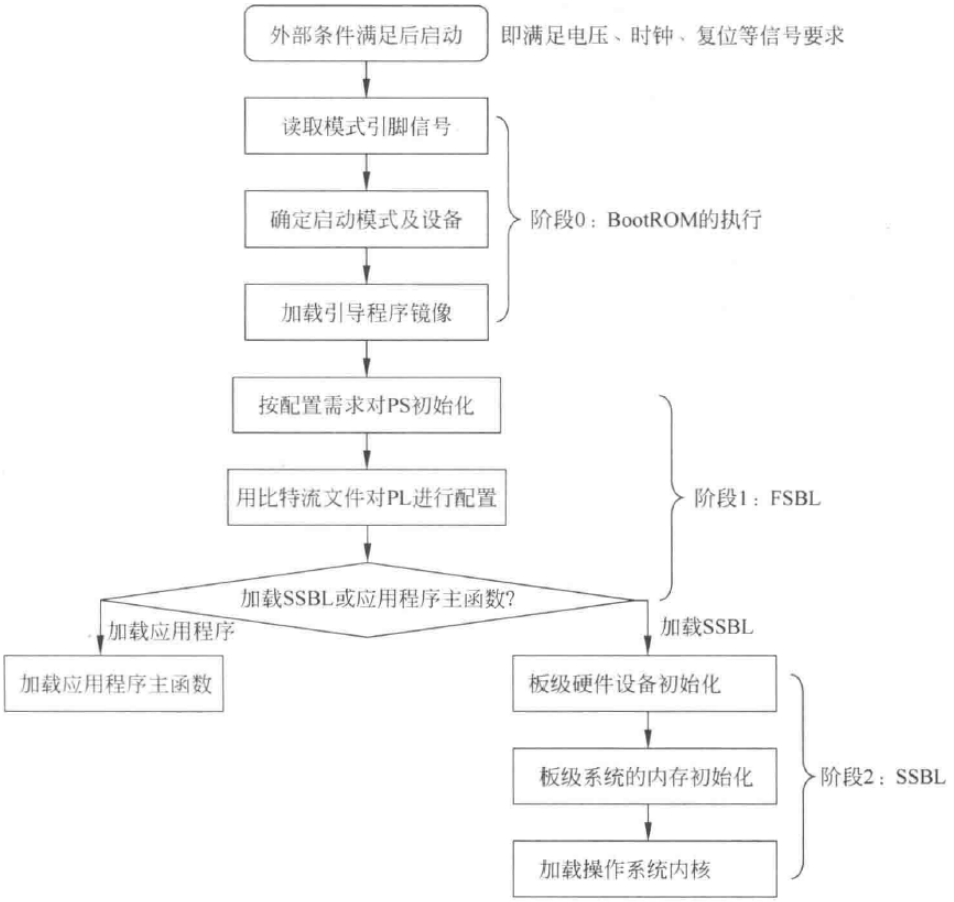


二、描述系统的软件总体流程

1、画出控制器的控制程序总体流程图



2、描述控制器中的软件启动流程，并指出启动程序如何引导控制器的应用程序



- 阶段 0:BootROM 的执行

BootROM 是指固化在 Zynq 芯片内部 ROM 中的一段代码,该段代码完成模式引脚(MIO)上的信号读取及判断;对外部设备控制器进行初始化,并读写这些外部设备;根据启动模式,加载第一阶段引导程序到片上存储器中, 或者直接在线性的 NOR Flash 存储器中执行引导程序。

- 阶段 1:第一阶段引导程序

FSBL 被称为第一阶段引导程序。其主要功能是初始化 PS 部分和 PL 部分，并加载第二阶段引导程序代码或应用程序的主函数。对 PS 部分初始化,实际上就是根据需要来配置 PS 部分的通用外部设备及接口。

FSBI 阶段的最后,将根据 Flash 分区镜像,来确定是加载 SSBL 还是直接加载应用程序的主函数。若系统的应用程序不需要基于操作系统上开发， 那么就不需要加载 SSBL，而是直接加载应用程序主函数。

- 阶段 2:第二阶段引导程序

第二阶段引导程序的主要工作是引导操作系统，为操作系统的运行进行存储空间初始化，并初始化必要的外设。这一阶段的程序功能实际上就是 Bootloader 的功能，根据需要引导的操作系统。

3、描述相关硬件部件的驱动程序流程，如串口部件、键盘部件等

```
//设置 MIO 引脚地址
#define MIO_PIN_07      (*(volatile unsigned int *)0xF800071C)
#define MIO_PIN_50      (*(volatile unsigned int *)0xF80007C8)
#define MIO_PIN_51      (*(volatile unsigned int *)0xF80007CC)
```

```
//设置 GPIO 端口方向寄存器地址
#define DIRM_0          (*(volatile unsigned int *)0xE000A204)
#define DIRM_1          (*(volatile unsigned int *)0xE000A244)
#define DIRM_2          (*(volatile unsigned int *)0xE000A284)
#define DIRM_3          (*(volatile unsigned int *)0xE000A2C4)
//设置 GPIO 端口输出使能寄存器地址
#define OEN_0           (*(volatile unsigned int *)0xE000A208)
#define OEN_1           (*(volatile unsigned int *)0xE000A248)
#define OEN_2           (*(volatile unsigned int *)0xE000A288)
#define OEN_3           (*(volatile unsigned int *)0xE000A2C8)
//设置 GPIO 端口输出寄存器地址
#define DATA_0          (*(volatile unsigned int *)0xE000A040)
#define DATA_1          (*(volatile unsigned int *)0xE000A044)
#define DATA_2          (*(volatile unsigned int *)0xE000A048)
#define DATA_3          (*(volatile unsigned int *)0xE000A04C)
//设置 GPIO 端口输入寄存器地址
#define DATA_0_RO       (*(volatile unsigned int *)0xE000A060)
#define DATA_1_RO       (*(volatile unsigned int *)0xE000A064)
#define DATA_2_RO       (*(volatile unsigned int *)0xE000A068)
#define DATA_3_RO       (*(volatile unsigned int *)0xE000A06C)
//设置 UART1 引脚地址的宏定义
#define rMIO_PIN_48          (*(volatile unsigned long*)0xF80007C0)
#define rMIO_PIN_49     (*(volatile unsigned long*)0xF80007C4)
#define rUART_CLK_CTRL   (*(volatile unsigned long*)0xF8000154)
#define rControl_reg0       (*(volatile unsigned long*)0xE0001000)
#define rMode_reg0          (*(volatile unsigned long*)0xE0001004)
//设置 UART1 端口波特率等参数地址寄存器的宏定义
#define rBaud_rate_gen_reg0 (*(volatile unsigned long*)0xE0001018)
#define rBaud_rate_divider_reg0 (*(volatile unsigned long*)0xE0001034)
#define rTx_Rx_FIFO0 (*(volatile unsigned long*)0xE0001030)
#define rChannel_sts_reg0 (*(volatile unsigned long*)0xE000102C)
```

4、描述控制器与被控对象之间的通信协议，请详细描述协议格式。

协议采用 MODBUS ASCII 模式，协议的格式如下（注：共 9 个字节，CRC 校验暂时不用）：

前导码 (1 字节)	地址 (1 字节)	命令 1 (1 字节)	命令 2 (1 字节)	命令 3 (1 字节)	命令 4 (1 字节)	命令 5 (1 字节)	命令 6 (1 字节)	命令 7 (1 字节)	CRC 校验 (4 字节)

(1)前导码：一个字节的 ASCII 码，其值固定为 0x23，即字符“#”。

(2)地址：一个字节的 ASCII 码，其值代表地址，如：若地址是 1 时，该字节值为：0x31，或者字符‘1’。

(3)命令字节：共有 7 个命令字节，每一个字节均为 ASCII 码，具体命令的功能要视情况而定。

(4)协议格式中的地址为 0x32，命令 1~命令 6 分别控制机械臂的第 1 轴~第 6 轴的转动，命令 7 控制机械臂在导轨上运动。命令 1~命令 6 的值与

其对应命令功能如下：

0x30 轴不动

0x31 轴按顺时针方向动作，转速角度为 1 度

0x32 轴按顺时针方向动作，转速角度为 2 度

0x33 轴按顺时针方向动作，转速角度为 3 度

0x34 轴按顺时针方向动作，转速角度为 5 度

0x35 轴按逆时针方向动作，转速角度为 1 度

0x36 轴按逆时针方向动作，转速角度为 2 度

0x37 轴按逆时针方向动作，转速角度为 3 度

0x38 轴按逆时针方向动作，转速角度为 5 度

命令 7 的值与其对应命令功能如下：

0x30 机械臂在轨道上不动

0x31 机械臂在轨道左移，移动速度为 1 档

0x32 机械臂在轨道左移，移动速度为 2 档

0x33 机械臂在轨道左移，移动速度为 3 档

0x34 机械臂在轨道右移，移动速度为 1 档

0x35 机械臂在轨道右移，移动速度为 2 档

0x36 机械臂在轨道右移，移动速度为 3 档

(5)箱子随机出现的控制命令协议格式中的地址为 0x34。命令 2 为控制 2 号站的箱子是否出现。

0x30 不出箱子

0x31 出箱子

(6)传送带的控制命令协议格式中的地址为 0x36。控制 2 号传送带的开和关

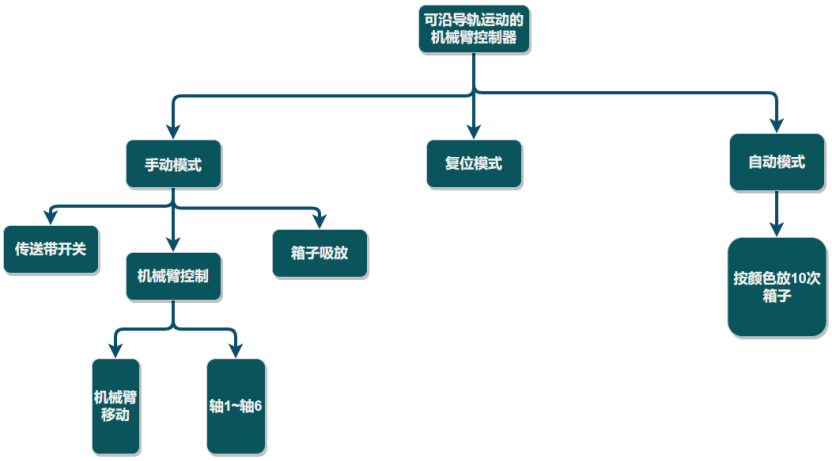
0x30 不动作

0x31 开传送带

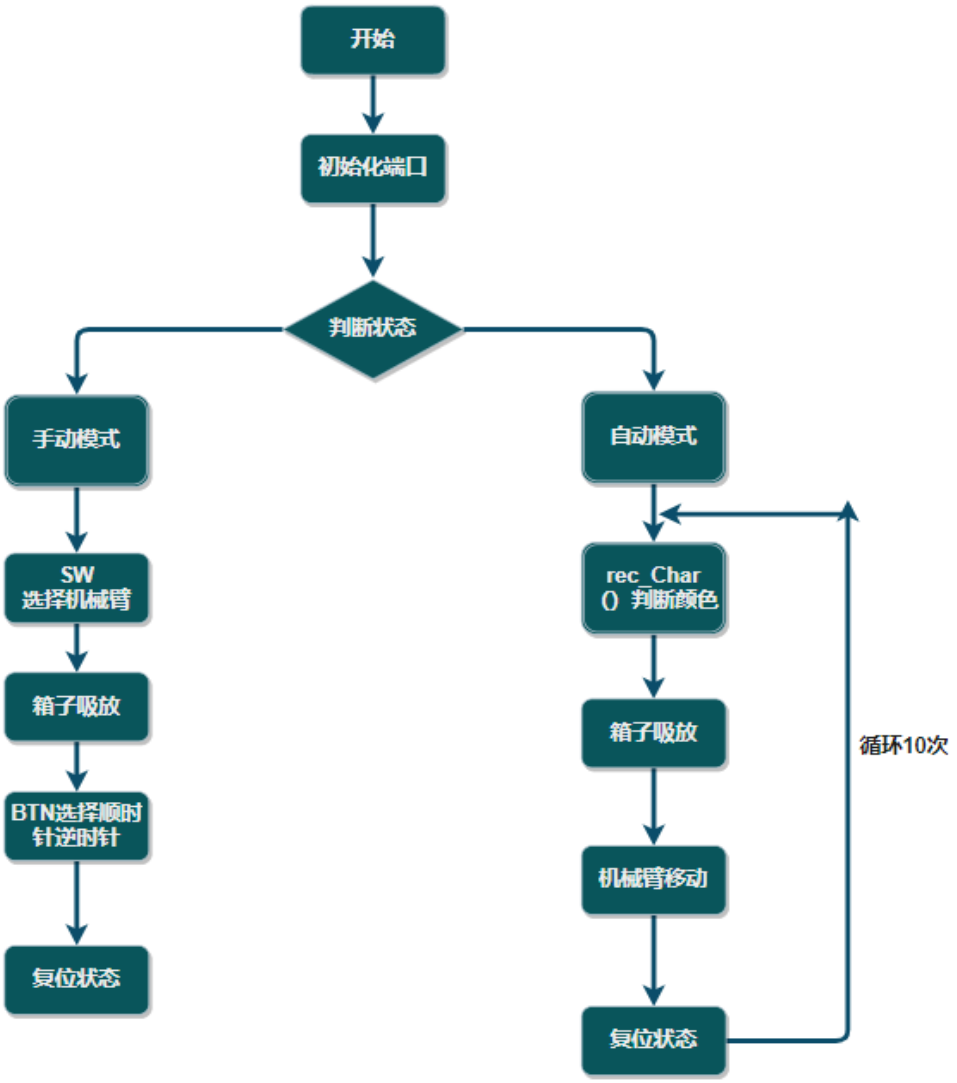
0x32 关传送带

5、画出客户端程序的功能框图及其程序流程图。

- 功能框图：



- 程序流程图


开始
初始化端口
判断状态
手动模式
SW
选择机械臂
箱子吸放
BTN选择顺时
针逆时针
复位状态
自动模式
rec_Char
() 判断颜色
箱子吸放
机械臂移动
复位状态
循环10次

详细设计

一、控制器软件的详细设计

1、通信驱动程序代码

```
#define MIO_PIN_07      (*(volatile unsigned int *)0xF800071C)
#define MIO_PIN_50      (*(volatile unsigned int *)0xF80007C8)
#define MIO_PIN_51      (*(volatile unsigned int *)0xF80007CC)
```

此处设置 MIO 的引脚地址，本次实验中所用到的 MIO 端口为 07、50 以及 51 端口，这是所使用的 GPIO 信号（端口 0 对应的 7 号以及端口 1 对应的 50、51 号）对应的控制引脚，通过初始地址加上偏移量使得其地址得以呈现。

我们还设置了其 GPIO 端口控制方向寄存器 DIRM，用来控制是输入还是输出：

```
#define DIRM_0          (*(volatile unsigned int *)0xE000A204)
#define DIRM_1          (*(volatile unsigned int *)0xE000A244)
#define DIRM_2          (*(volatile unsigned int *)0xE000A284)
#define DIRM_3          (*(volatile unsigned int *)0xE000A2C4)
```

以及 GPIO 端口的使能控制，控制各个端口的使能，此处表示对 GPIO 的0、1、2、3号端口进行使能控制：

```
#define OEN_0           (*(volatile unsigned int *)0xE000A208)
#define OEN_1           (*(volatile unsigned int *)0xE000A248)
#define OEN_2           (*(volatile unsigned int *)0xE000A288)
#define OEN_3           (*(volatile unsigned int *)0xE000A2C8)
```

下图是对 GPIO端口的输入控制，是控制机械臂的主要端口。

```
#define DATA_0_RO       (*(volatile unsigned int *)0xE000A060)
#define DATA_1_RO       (*(volatile unsigned int *)0xE000A064)
#define DATA_2_RO       (*(volatile unsigned int *)0xE000A068)
#define DATA_3_RO       (*(volatile unsigned int *)0xE000A06C)
```

下图是对 GPIO端口的输出控制，是控制 LED灯的主要端口。

```
#define DATA_0          (*(volatile unsigned int *)0xE000A040)
#define DATA_1          (*(volatile unsigned int *)0xE000A044)
#define DATA_2          (*(volatile unsigned int *)0xE000A048)
#define DATA_3          (*(volatile unsigned int *)0xE000A04C)
```

下图是 UART1引脚地址的宏定义以及波特率等参数地址的宏定义：

```
#define rMIO_PIN_48     (*(volatile unsigned long*)0xF80007C0)
#define rMIO_PIN_49     (*(volatile unsigned long*)0xF80007C4)
#define rUART_CLK_CTRL  (*(volatile unsigned long*)0xF8000154)
#define rControl_reg0   (*(volatile unsigned long*)0xE0001000)
#define rMode_reg0      (*(volatile unsigned long*)0xE0001004)
#define rBaud_rate_gen_reg0 (*(volatile unsigned long*)0xE0001018)
#define rBaud_rate_divider_reg0 (*(volatile unsigned long*)0xE0001034)
#define rTx_Rx_FIFO0 (*(volatile unsigned long*)0xE0001030)
#define rChannel_sts_reg0 (*(volatile unsigned long*)0xE000102C)
```

```
1.  //UART1 的初始化函数
2.  void RS232_Init()
3.  {
4.      rMIO_PIN_48=0x000026E0;
5.      rMIO_PIN_49=0x000026E0;
```

```
        rUART_CLK_CTRL=0x00001402;
        rControl_reg0=0x00000017;
        rMode_reg0=0x00000020;
        rBaud_rate_gen_reg0=62;
        rBaud_rate_divider_reg0=6;
    }


    //单个字节数据的发送函数
    void send_Char(unsigned char data)
    {
         while((rChannel_sts_reg0&0x10)==0x10);
         rTx_Rx_FIFO0=data;
    }


    //9 个字节数据的发送函数，调用 send_Char(unsigned char data);
    void send_Char_9(unsigned char modbus[])
    {
        int i;
        char data;
        for(i=0;i<9;i++){
            data=modbus[i];
            send_Char(data);
            delay(100,10,10);       //延时
        }
    }
```

2、其他功能模块的流程及主要程序代码

- 机械臂控制：

```
    //arm_id 机械臂朝着对应的方向按档位转动一次
    void turnArm(int arm_id,int dir,int spd){
        unsigned char modbus_com[9];
        modbus_com[0]='#';  //起始符，固定为#
        modbus_com[1]='2';
        modbus_com[2]='0';
        modbus_com[3]='0';
        modbus_com[4]='0';
        modbus_com[5]='0';
        modbus_com[6]='0';
        modbus_com[7]='0';
        modbus_com[8]='0';

        if(arm_id==-1 || dir==0)return;
        else if(dir==1)modbus_com[arm_id+2]=0x30+spd;    //顺时针
```

```
        else if(dir==-1)modbus_com[arm_id+2]=0x34+spd;  //逆时针

        send_Char_9(modbus_com);
    }
```

- Main 函数：

```
    int main()
    {

        init_platform();
        u32 flag;       //变量 flag 用于记录 SW0~SW7 按键按下信息;

        //注：下面 MIO 引脚和 EMIO 引脚的序号是统一编号的，MIO 序号为 0~31 及 32~53，EMIO 序号为 54~85 及 86~117
        //配置及初始化 MIO07 引脚的相关寄存器，MIO07 作为 LED 灯控制的输出引脚
        MIO_PIN_07 = 0x00003600;
        DIRM_0 = DIRM_0|0x00000080;
        OEN_0 = OEN_0|0x00000080;
        //配置及初始化 MIO50、MIO51 引脚的相关寄存器，MIO50、MIO51 作为按键输入引脚
        MIO_PIN_50 = 0x00003600;
        MIO_PIN_51 = 0x00003600;
        DIRM_1 = DIRM_1 & 0xFFF3FFFF;
        //初始化 EMIO54~EMIO58 的引脚，它们对应 BTNU、BTND、BTNL、BTNR、BTNC 按键，输入
        DIRM_2 = DIRM_2 & 0xFFFFFFE0;
        //初始化 EMIO59~EMIO66 的引脚，它们对应 SW7~SW0 拨动开关，输入
        DIRM_2 = DIRM_2 & 0xFFFFE01F;
        //初始化 EMIO67~EMIO74 的引脚，它们对应 LED7~LED0，输出
        DIRM_2 = DIRM_2|0x001FE000;
        OEN_2 = OEN_2|0x001FE000;

        //初始化 UART1
        RS232_Init();

        int fgauto=1;
        while(1){
            //读模式信息，即读 SW7、SW6 的输入信息
            flag = DATA_2_RO&0x00000060;//【EMIO 59-60】
            switch(flag){
            case 0x00:                  //复位模式
                DATA_2 = DATA_2&0xFFE01FFF;     //模式指示灯 LED7~LED0 灭
                reset();
                fgauto=1;
```

```
                delay(100,10,10);
                break;
            case 0x20:                          //手动控制模式
                DATA_2 = (DATA_2|0x00002000)&0xFFFFBFFF;     //指示灯 LED7 亮、LED6 灭【EMIO 67 68】
                singleStep();
                break;
            case 0x40:                          //自动控制模式
                DATA_2 = (DATA_2|0x00004000)&0xFFFF7FFF;     //LED7 灭、LED6 亮
                if(fgauto){
                    autoExColor();
                    delay(1000,100,500);
                }
                fgauto=0;
                break;
            case 0x60:                          //机械臂示教模式（该模式暂不实现）
                DATA_2 = DATA_2|0x00006000;                  //LED7 亮、LED6 亮
                break;
            }
        }
        return 0;
    }
```

- 复位函数：

```
    //复位函数
    void reset(){
        for(int i=0;i<6;i++){
            if(CurPos[i]!=0){
                if(CurPos[i]>0){
                    while(CurPos[i]!=0){
                        CurPos[i]=(CurPos[i]-5)%360;
                        turnArm(i,-1,4);
                    }
                 }
                 if(CurPos[i]<0){
                     while(CurPos[i]!=0){
                        CurPos[i]=(CurPos[i]+5)%360;
                        turnArm(i,1,4);
                    }
                 }
            }
        }
```

```
19.
20.         //复位导轨
21.         if(track!=0){
22.             if(track>0){
23.                 moveArm(2,2);
24.                 track-=2;
25.             }
26.             else{
27.                 moveArm(1,2);
28.                 track+=2;
29.             }
30.         }
31.     }
```

- 自动模式下根据箱子颜色执行

```
1.     //根据颜色执行动作
2.     void autoExColor(){
3.         initArm();
4.         for(int count=0;count<10;count++){
5.             swTrans(1);
6.             if(count)for(int j=0;j<25;j++){
7.                 delay(1000,500,10);
8.             }
9.             proBox(1);
10.             RS232_Init();
11.             delay(1000,100,50);
12.             unsigned char color = rec_Char();
13.             if(color=='B'){
14.                 GoBlue();
15.             }else if(color=='G'){
16.                 GoGreen();
17.             }else if(color=='R'){
18.                 GoRed();
19.             }
20.         }
21.         endArm();
22.     }
```

- 蓝色箱子摆放，红色同理，绿色不需要移动导轨

```
1.     void GoBlue(){
2.         catch(1);
3.         for(int i=1;i<=36;i++){
4.             turnArm(0,-1,4);
5.
```

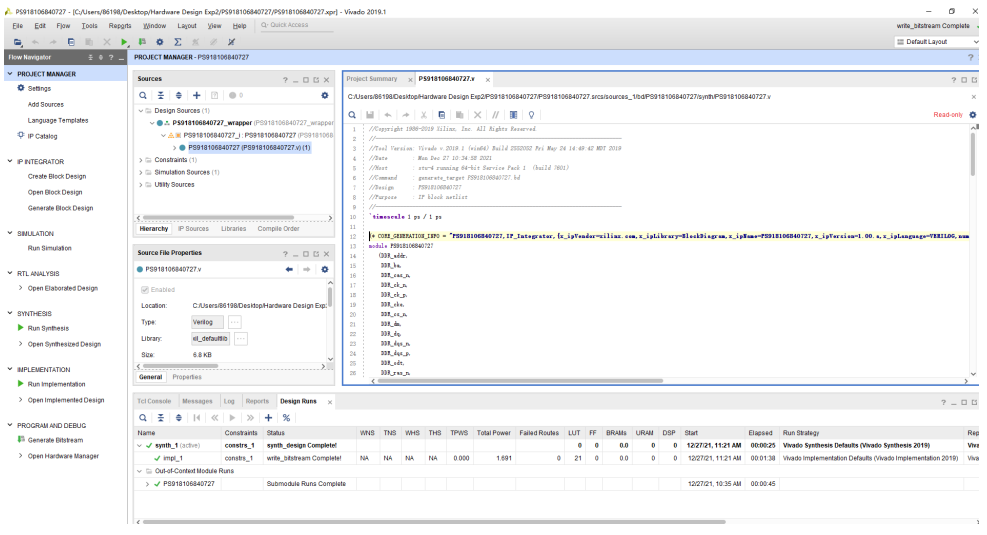
```
        }delay(100,100,50);
        for(int i=1;i<=20;i++){
            moveArm(1,3);

        }delay(100,100,50);
        catch(2);
        for(int i=1;i<=36;i++){
            turnArm(0,1,4);

        }delay(100,100,50);
        for(int i=1;i<=20;i++){
            moveArm(2,3);

        }delay(100,100,50);
    }
```

3、被控对象的动作截图 3，附上被控对象的动作截图（即机械臂等进行搬运物体的截屏图）。

- 抓取各类颜色并摆放

- 复位

- 工程文件截图



体 会	**蒋旭钊：** 这次硬件课设，我主要负责的是功能模块的编写、机械臂的调试以及报告的撰写。 通过这次硬件课程设计，我再次温习了嵌入式的相关知识，了解了软硬件协同的相关知识，掌握了Vivado中项目建立的大体流程。其中GPIO端口操作，UART 串口控制，关于 EMIO 端口的读写等等都是我们上学期嵌入式考试的内容，需要合理地设置与或关系，才能让端口寄存器的值保持正确，从而能够实现我们想要的操作。在这次实验中，我发现自己原先学习的知识都在实践中得以验证，真正做到了“纸上得来终觉浅，绝知此事要躬行”。 这次硬件课设也在不断锤炼我的纠错能力，硬件编程和我们软件不同，可能没有很好的debug展示，只能通过上板验证的效果来逆向推理出可能出现的问题，然后根据自己的猜想去修改相关的数据结构以及逻辑代码，然后正向继续验证，如此反复。 我认为，在这学期的课设当中，不仅培养了独立思考的能力，动手实践的能力，还培养了我们的合作能力：我同小组成员一起理解并掌握老师给的框架，并熟练运用起来，这本身就能让我受益匪浅。 改进意见：我认为学校可以增加 Verilog 培训课程或者 Vivado 工程教学课程，帮助学生更好地掌握硬件编程相关知识。 **王迪：** 本次硬件课程设计让我受益匪浅，对于《嵌入式操作系统》中学习到的知识有了更深入的理解。在本次设计过程中，我们遇到了很多困难，有通信部分的也有操作逻辑部分的。其中，让我印象最为深刻的是，我们最先使用的通信部分的接受函数通过 RFUL 位即确定接收 FIFO 缓存为未满来进行颜色状态的信号接收，但是这会导致我们在自动执行过程的最后一次无法正常执行完成，出现错误。最终，我们选择按照《嵌入式操作系统》书中的接收函数将我们的判断条件更改为判断 REMPTY 位即确定接收 FIFO 缓存为未空来进行数据接收，最终奇怪的 bug 得以解决。这让我更加意识到软硬件协同过程中各部分协调的重要性。除此之外，经过本次硬件实验的操作，我还掌握了对于查阅手册进行管脚约束，控制信号端口的能力。这让我再未来实际工作当中遇到相应问题有了更强的解决能力，大大提高了我的工程应用能力！真得很感谢学院开设这样实用的硬件实验课程，也非常感谢老师对于我们实验所给予的指导！