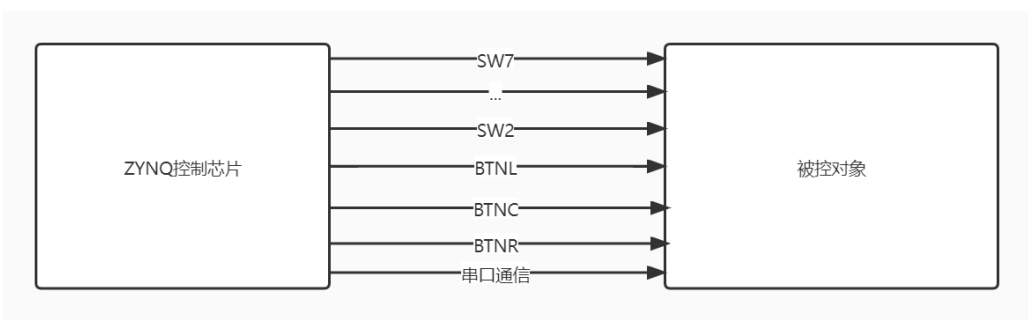
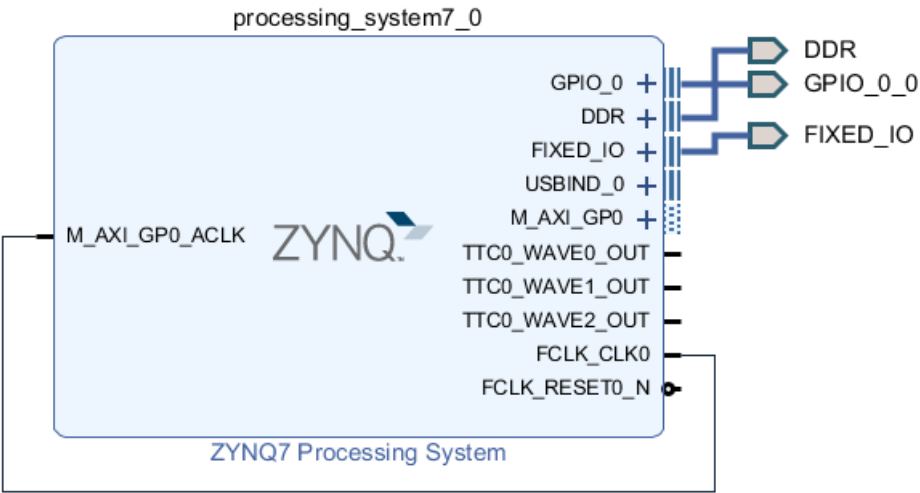


题目的要求	<div>1、可以与沿轨道运动的机械臂进行串口通信，串口通信协议遵循MODBUS 协议。</div> <div>2、可以手动控制机械臂在导轨上运动。</div> <div>3、可以手动控制 6 个轴的顺时针、逆时针转。</div> <div>4、可以手动控制 3 号传送带启动、停止。</div> <div>5、可以手动控制出箱子。</div> <div>6、完成复位功能设计，即按一个按键后，机械臂恢复最初位置。</div>																																																				
总体设计	<div>一，描述系统的组成及硬件环境，要求：</div> <div>1，画出系统总体结构图，指出控制器与被控对象之间的连接结构；</div> <div></div> <table><tr><th>SW7 – SW2</th><th>模式选择</th><th>BTNL\BTNR</th><th>控制指令</th></tr><tr><td>00 xxxx</td><td>复位模式</td><td></td><td></td></tr><tr><td>01 xxxx</td><td>手动模式</td><td></td><td></td></tr><tr><td rowspan="2">01 0000</td><td rowspan="2">第 1 轴</td><td>BTNL</td><td>第 1 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 1 轴逆时针转动</td></tr><tr><td rowspan="2">01 1000</td><td rowspan="2">第 2 轴</td><td>BTNL</td><td>第 2 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 2 轴逆时针转动</td></tr><tr><td rowspan="2">01 0100</td><td rowspan="2">第 3 轴</td><td>BTNL</td><td>第 3 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 3 轴逆时针转动</td></tr><tr><td rowspan="2">01 1100</td><td rowspan="2">第 4 轴</td><td>BTNL</td><td>第 4 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 4 轴逆时针转动</td></tr><tr><td rowspan="2">01 0010</td><td rowspan="2">第 5 轴</td><td>BTNL</td><td>第 5 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 5 轴逆时针转动</td></tr><tr><td rowspan="2">01 1010</td><td rowspan="2">第 6 轴</td><td>BTNL</td><td>第 6 轴顺时针转动</td></tr><tr><td>BTNR</td><td>第 6 轴逆时针转动</td></tr><tr><td>01 0110</td><td>导轨运动</td><td>BTNL</td><td>机械臂在轨道左移</td></tr></table>	SW7 – SW2	模式选择	BTNL\BTNR	控制指令	00 xxxx	复位模式			01 xxxx	手动模式			01 0000	第 1 轴	BTNL	第 1 轴顺时针转动	BTNR	第 1 轴逆时针转动	01 1000	第 2 轴	BTNL	第 2 轴顺时针转动	BTNR	第 2 轴逆时针转动	01 0100	第 3 轴	BTNL	第 3 轴顺时针转动	BTNR	第 3 轴逆时针转动	01 1100	第 4 轴	BTNL	第 4 轴顺时针转动	BTNR	第 4 轴逆时针转动	01 0010	第 5 轴	BTNL	第 5 轴顺时针转动	BTNR	第 5 轴逆时针转动	01 1010	第 6 轴	BTNL	第 6 轴顺时针转动	BTNR	第 6 轴逆时针转动	01 0110	导轨运动	BTNL	机械臂在轨道左移
SW7 – SW2	模式选择	BTNL\BTNR	控制指令																																																		
00 xxxx	复位模式																																																				
01 xxxx	手动模式																																																				
01 0000	第 1 轴	BTNL	第 1 轴顺时针转动																																																		
		BTNR	第 1 轴逆时针转动																																																		
01 1000	第 2 轴	BTNL	第 2 轴顺时针转动																																																		
		BTNR	第 2 轴逆时针转动																																																		
01 0100	第 3 轴	BTNL	第 3 轴顺时针转动																																																		
		BTNR	第 3 轴逆时针转动																																																		
01 1100	第 4 轴	BTNL	第 4 轴顺时针转动																																																		
		BTNR	第 4 轴逆时针转动																																																		
01 0010	第 5 轴	BTNL	第 5 轴顺时针转动																																																		
		BTNR	第 5 轴逆时针转动																																																		
01 1010	第 6 轴	BTNL	第 6 轴顺时针转动																																																		
		BTNR	第 6 轴逆时针转动																																																		
01 0110	导轨运动	BTNL	机械臂在轨道左移																																																		

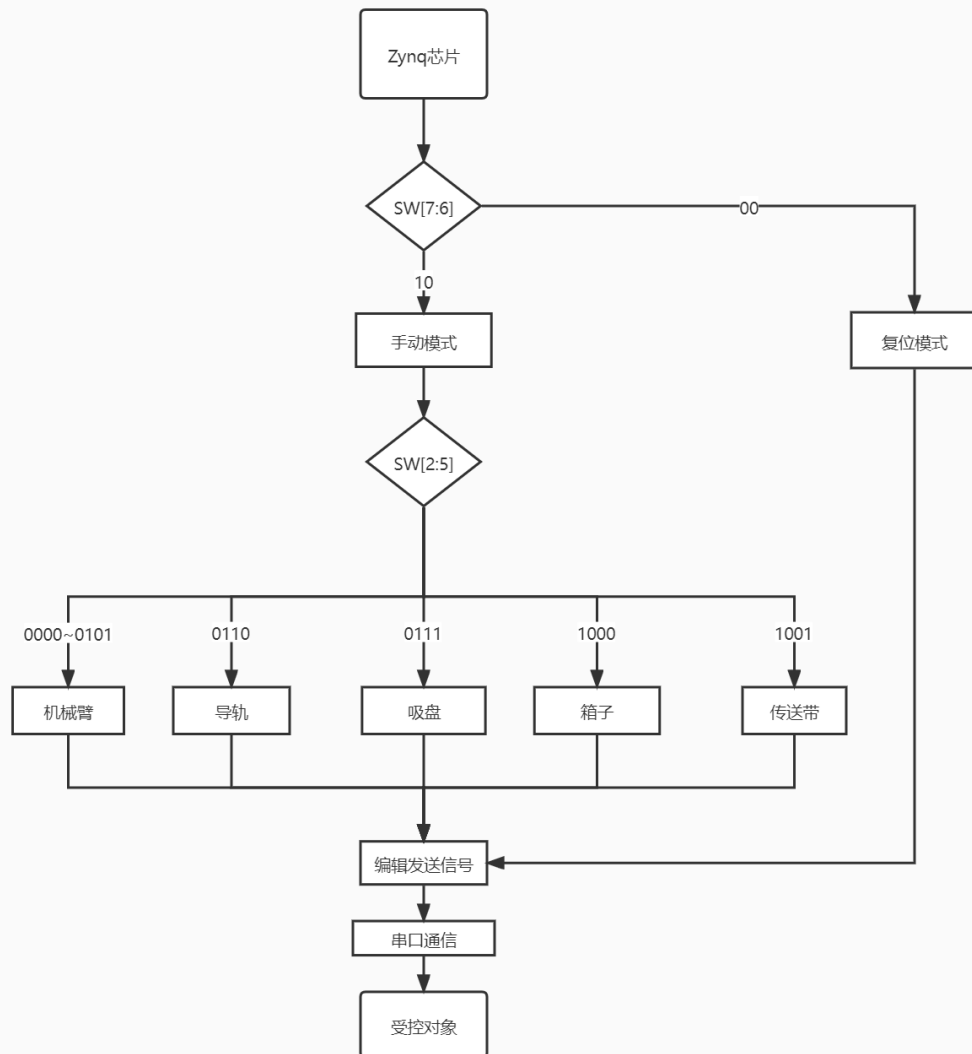
		BTNR	机械臂在轨道右移	
01 1110	吸盘	BTNL	2 号站的吸盘为吸	
		BTNR	2 号站的吸盘为放	
01 0001	箱子	BTNC	2 号站的箱子出现	
01 1001	传送带	BTNL	3 号传送带开	
		BTNR	3 号传送带关	

2，画出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，要求：

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；

Step1——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上的信号读取及判断；对四线 _SPI、NOR、NAND、SD 等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，或直接在现行的 NOR Flash 存储器中执行引导程序。

Step2——第一阶段引导程序：

FSBL 被称为第一阶段引导程序，它的主要功能是初始化 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。FSBL 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。（此处没有相应的操作系统，故不需要加载 SSBL）。

BootROM 在加载 FSBL 或应用程序时，不是直接转移到其代码上执行，而是加载其一个合法的程序镜像。镜像时一种可由 BootROM 加载时

进行解析的文件，该文件中的信息是在可执行代码前加上一些说明信息，以便 BootROM 加载时进行解析。

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；
GPIO 驱动：

- (1) 设置 MIO/EMIO 引脚为 GPIO 功能，通过向系统级寄存器 SLCR 中的相应引脚功能配置寄存器 MIO_PIN_N 中写入相应的参数，来设置 MIO 引脚为 GPIO。
- (2) 设置 MIO/EMIO 引脚为输入还是输出，通过向寄存器 DIRM_N 中的相应位写入参数来设置 MIO/EMIO 引脚的方向。
- (3) 若方向配置为输出，还需要设置 OEN_N 寄存器来使能输出。否则不需要配置该寄存器。
- (4) 根据方向配置为输入还是输出，完成对寄存器 DATA_N_RO 或 DATA_N 的读取还是写入。

URAT 驱动：

- (1) 初始化函数，向 MIO_PIN_N 寄存器中写入相应的参数来设置 MIO 引脚为 URAT 的引脚功能，并设置数据格式，波特率参数等。
- (2) 发送函数，通过向 Tx_Rx_FIFO 寄存器中写入参数来发送一个字符的信息，在写入该寄存器前需判断发送 FIFO 是否不满。
- (3) 接收函数，通过读 Tx_Rx_FIFO 寄存器中的值来接收一个字符的信息，在读该寄存器前需判断接收 FIFO 是否不空。

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，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)
---------------	--------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	------------------

前导码：一个字节的 ASCII 码，其值固定为 0x23，即字符 ‘#’ 。

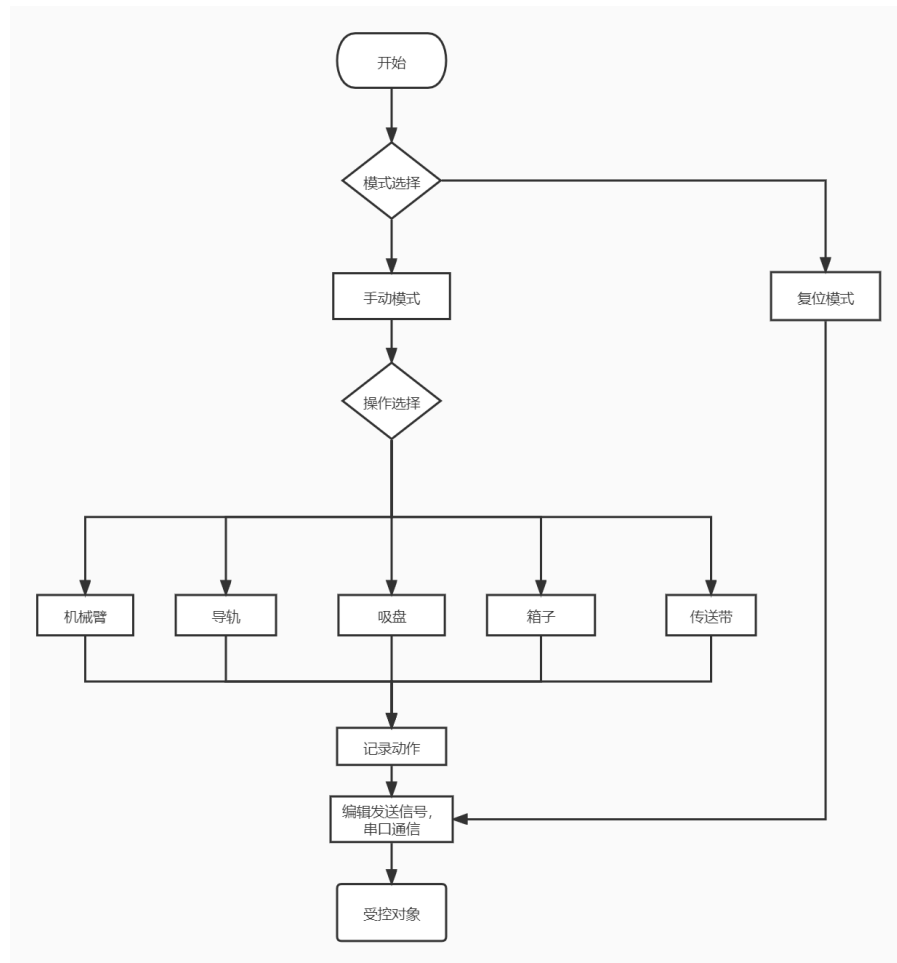
地址：一个字节的 ASCII 码，其值代表地址，如：若地址是 1 时，该字节值为：0x31，或者字符 ‘1’ 。

命令字节：共有 7 个命令字节，每一个字节均为 ASCII 码，具体命令的功能如下，

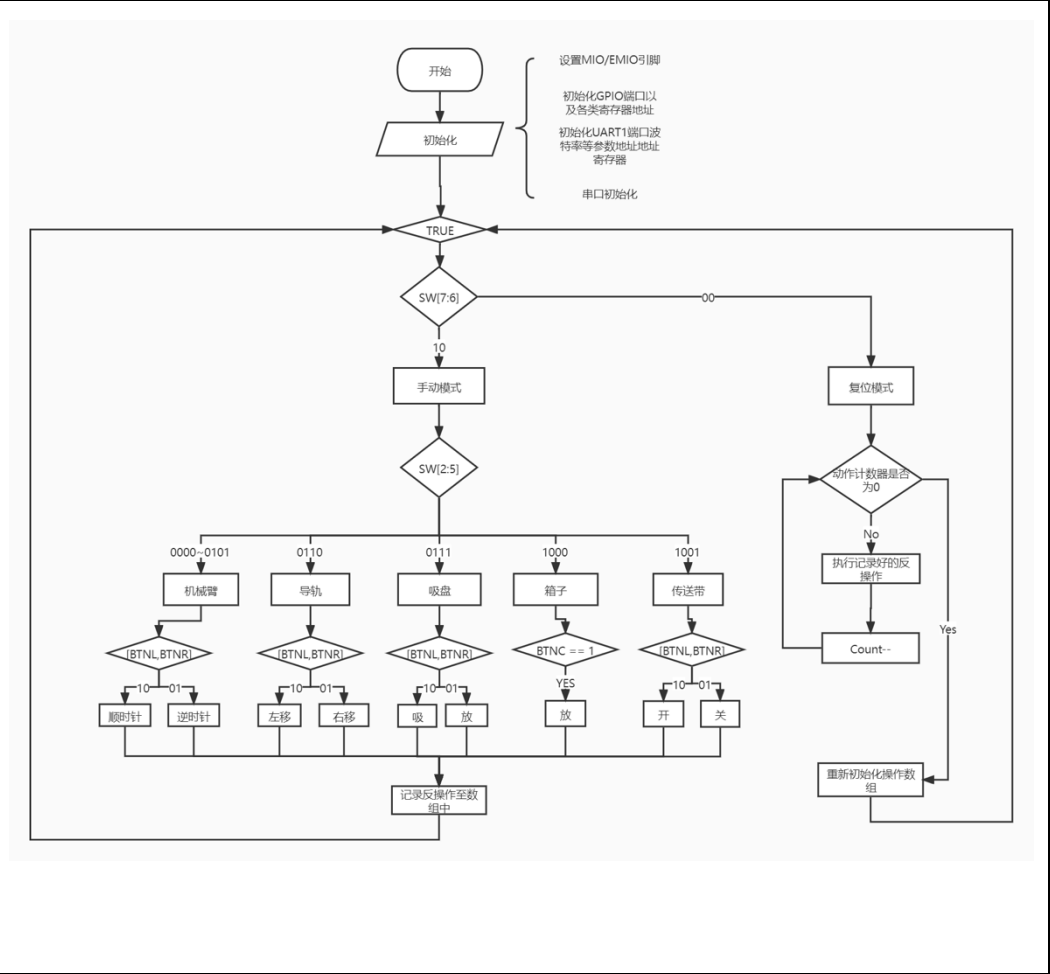
协议格式中的地址为 0x32，命令 1~命令 6 分别控制机械臂的第 1 轴~第 6 轴的转动，命令 7 控制机械臂在导轨上运动。命令 1~命令 6 的值与其对应命令功能如下：

- 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

	<p>0x38 轴按逆时针方向动作，转速角度为 5 度</p> <p>箱子随机出现的控制命令： 协议格式中的地址为 0x34。命令字节为： 命令 1：控制 1 号站的箱子是否出现，具体命令码及对应的功能如下。 0x30 不出箱子 0x31 出箱子 命令 2：控制 2 号站的箱子是否出现，具体命令码与命令 1 相同。 命令 3：控制 3 号站的箱子是否出现，具体命令码与命令 1 相同。</p> <p>吸盘的控制命令： 协议格式中的地址为 0x35。命令字节为： 命令 1：控制 1 号站的吸盘是吸还是放，具体命令码及对应的功能如下。 0x30 不动作 0x31 吸 0x32 放 命令 2：控制 2 号站的吸盘是吸还是放，具体命令码与命令 1 相同。 命令 3：控制 3 号站的吸盘是吸还是放，具体命令码与命令 1 相同。</p> <p>传送带的控制命令： 协议格式中的地址为 0x36。命令字节为： 命令 1：控制 2 号传送带的开和关，具体命令码及对应的功能如下。 0x30 不动作 0x31 开传送带 0x32 关传送带 命令 2：控制 3 号传送带的开和关，具体命令码与命令 1 相同。</p> <p>5，画出客户端程序的功能框图及其程序流程图。</p> <p>功能框图：</p>
--	--



程序流程图：



详细设计	<p>一，控制器软件的详细设计，要求：</p> <p>1，给出通信驱动程序代码，并解释：</p> <p>UART1 的初始化函数：</p> <pre>void RS232_Init() { rMIO_PIN_48=0x000026E0; rMIO_PIN_49=0x000026E0; rUART_CLK_CTRL=0x00001402; rControl_reg0=0x00000017; rMode_reg0=0x00000020; rBaud_rate_gen_reg0=62; rBaud_rate_divider_reg0=6; }</pre> <p>单个字节数据的发送函数：</p> <pre>void send_Char(unsigned char data) { while((rChannel_sts_reg0&0x10)==0x10); rTx_Rx_FIFO0=data; }</pre> <p>9 个字节数据的发送函数：</p> <pre>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 } }</pre> <p>2，给出其他功能模块的流程及主要程序代码，并解释。如机械臂转动控制功能的流程等。</p> <p>控制功能模块：</p> <p>（1）机械臂和导轨：</p> <p>//机械臂相关各部件动作函数</p> <pre>void arm(int Arm_ID,int Arm_dir) { unsigned char modbus_com[9]; modbus_com[0]='#'; //起始符，固定为# modbus_com[1]='2'; //机械臂相关 modbus_com[2]='0';</pre>
------	--


```

modbus_com[3]='0';
modbus_com[4]='0';
modbus_com[5]='0';
modbus_com[6]='0';
modbus_com[7]='0';
modbus_com[8]='0';

switch(Arm_ID){
case 1: //第一个轴
    if (Arm_dir==0){
        modbus_com[2]='3';
    }
    else if(Arm_dir==1){
        modbus_com[2]='7';
    }
    break;
case 2: //第二个轴
    if (Arm_dir==0){
        modbus_com[3]='3';
    }
    else if(Arm_dir==1){
        modbus_com[3]='7';
    }
    break;
case 3: //第三个轴
    if (Arm_dir==0){
        modbus_com[4]='3';
    }
    else if(Arm_dir==1){
        modbus_com[4]='7';
    }
    break;
case 4: //第四个轴
    if (Arm_dir==0){
        modbus_com[5]='3';
    }
    else if(Arm_dir==1){
        modbus_com[5]='7';
    }
    break;
case 5: //第五个轴
    if (Arm_dir==0){
        modbus_com[6]='3';
    }

```

```

        else if(Arm_dir==1){
            modbus_com[6]='7';
        }
        break;
case 6:                                //第六个轴
    if (Arm_dir==0){
        modbus_com[7]='3';
    }
    else if(Arm_dir==1){
        modbus_com[7]='7';
    }
    break;
case 7:                                //轨道上的移动
    if (Arm_dir==0){
        modbus_com[8]='2';
    }
    else if(Arm_dir==1){
        modbus_com[8]='5';
    }
    break;
}
send_Char_9(modbus_com);
}
(2) 箱子:
void box(void)
{
    unsigned char modbus_com[9];
    modbus_com[0]='#';                //起始符，固定为#
    modbus_com[1]='4';                //箱子
    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';
    //2 号站
    modbus_com[3] = '1';
    send_Char_9(modbus_com);
}
(3) 吸盘:
void plate(int Plate_ID,int Plate_dir)
{
    unsigned char modbus_com[9];

```

```

modbus_com[0]='#';           //起始符，固定为#
modbus_com[1]='5';           //吸盘
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';

```

```

switch(Plate_ID){
case 1:                       //1 号站
    if(Plate_dir == 0){
        modbus_com[2]='1';
    }else if(Plate_dir == 1){
        modbus_com[2]='2';
    }
    break;
case 2:                       //2 号站
    if(Plate_dir == 0){
        modbus_com[3]='1';
    }else if(Plate_dir == 1){
        modbus_com[3]='2';
    }
    break;
case 3:                       //3 号站
    if(Plate_dir == 0){
        modbus_com[4]='1';
    }else if(Plate_dir == 1){
        modbus_com[4]='2';
    }
}

```

```

    send_Char_9(modbus_com);
}

```

(4) 传送带:

```

void trans(int Trans_dir)
{
    unsigned char modbus_com[9];
    modbus_com[0]='#';           //起始符，固定为#
    modbus_com[1]='6';           //传送带
    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';
```

```
//控制 3 号传送带
    if(Trans_dir == 0){
        modbus_com[3]='1';
    }else if(Trans_dir == 1){
        modbus_com[3]='2';
    }
    send_Char_9(modbus_com);
}
```

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）。
部分程序以及动作截图:

