

前言

现代科技的发展，使得改革传统教学方式迫在眉睫！通过增加实验和培训课程，重点培养学生的创造力和实际操作能力是教学改革的重要内容之一。

作为教学改革的新浪潮的一名成员，德普施科技凭借自己多年来在远程教育、网络化测控、自动控制和机电一体化等领域进行科研和教学实践所取得的丰硕成果，结合国内外多家名校的成果技术及经验，始终致力于开放式教学设备的研究与开发工作，并取得骄人的成绩。

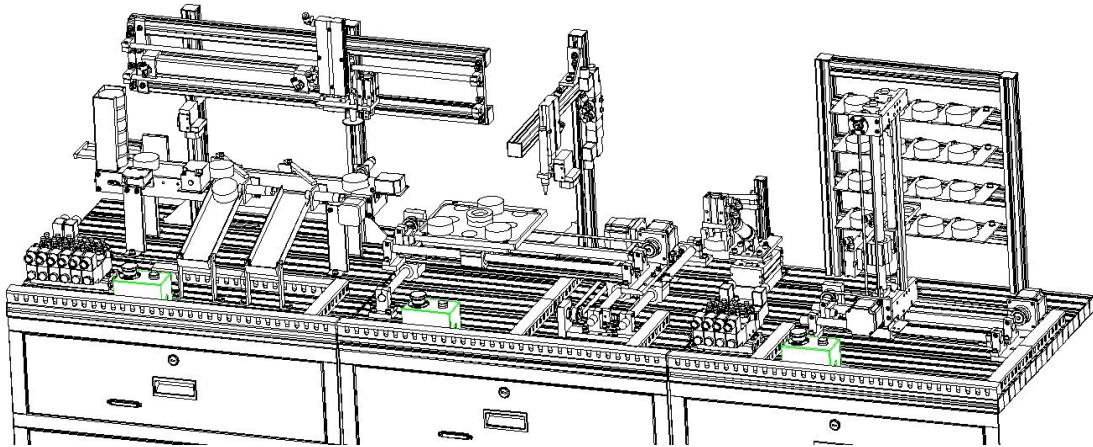
本套可重构机电实训平台系统包括机械、电子与控制为一体，提供从实验对象、信号获取、信号分析以及运动控制等相关专业课程的优秀开放式实训平台。可广泛应用于机械、自动控制、控制工程、机电一体化、电机等专业。系统所支持的测控软件有 C++、LabVIEW 以及 Matlab 等各种环境以适用于学校不同的课程需求。使得学校能根据教学需求组建一个高水平的实验室，提高学科的建设水平。

目录

前言.....	- 1 -
第一章 机械结构.....	- 3 -
1.1 一号台结构.....	- 3 -
1.2 二号台结构.....	- 9 -
1.3 三号台结构.....	- 11 -
第二章 电气连接部分.....	- 14 -
2.1 接口板定义.....	- 14 -
2.2 限位开关连线.....	- 15 -
2.3 驱动器与电机连线.....	- 15 -
第三章 软件部分.....	- 16 -
3.1 版权信息与命名规定.....	- 16 -
3.2 使用纲要.....	- 16 -
3.3 设备操作接口函数介绍.....	- 17 -
3.4 底层设备通讯协议.....	- 31 -
3.5 样例程序使用介绍.....	- 32 -
第四章 实训部分.....	- 39 -
4.1 与控制器 “say hello ”	- 39 -
4.2 电机驱动.....	- 43 -
4.3 IO 口控制.....	- 55 -
4.4 获取平台输入口状态.....	- 62 -
4.5 获取传感器电平.....	- 67 -
4.6 颜色传感器应用.....	- 71 -
4.7 一号平台控制编程.....	- 75 -
4.8 二号平台控制编程.....	- 82 -
4.9 三号平台控制编程.....	- 85 -
4.10 综合控制编程.....	- 89 -
注意和警告提示.....	- 92 -
储运常识.....	- 93 -
安全事项.....	- 94 -
维护 和 保养	- 95 -

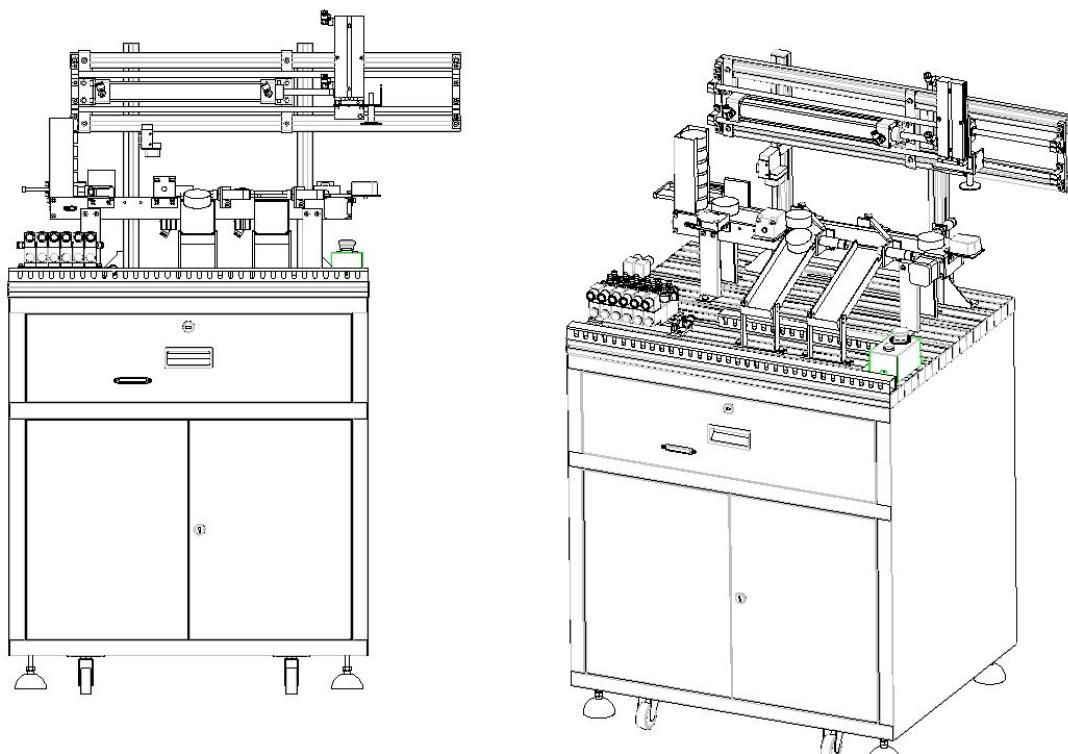
第一章 机械结构

本套系统共由三个工作台组成，下图所展示的为机械连接整体效果图。

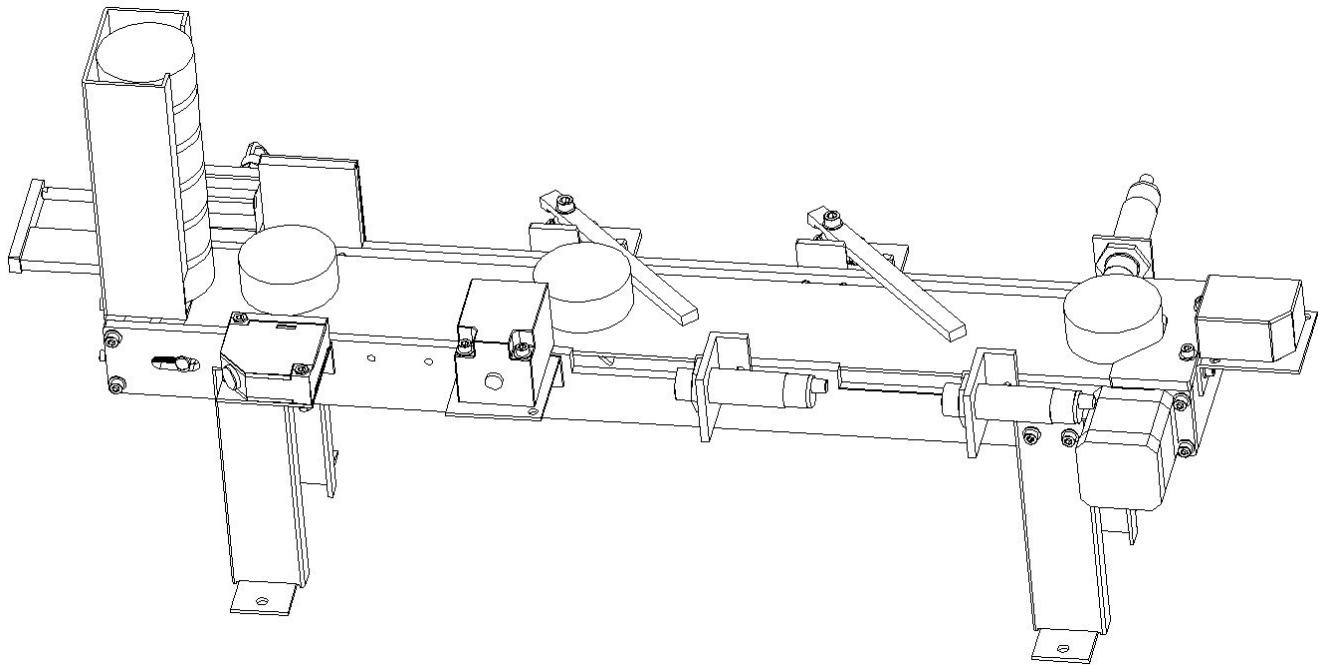


1.1 一号台结构

一号台 3D 形体图如下图所示：



一号台送料机构 3D 形体如下图所示。



如图 1.1 为一号台的传输运料机构，上面装配有检查物料性质的传感器以及推料、选料的气缸。

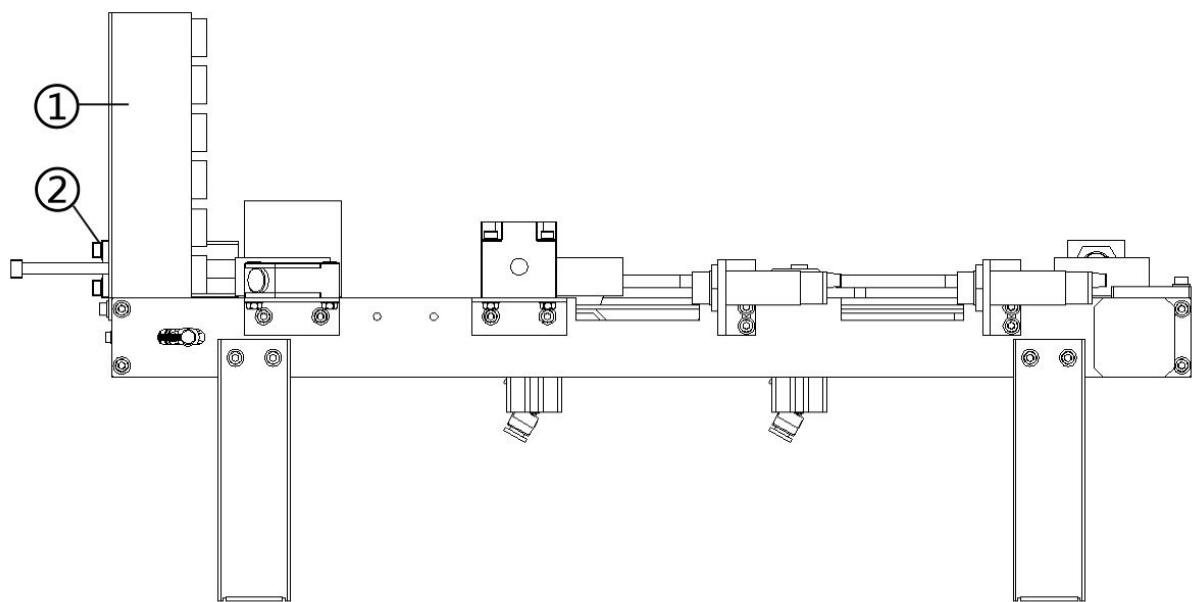


图 1.1 一号台—运料机构 (视图 1)

在图 1.1 中，①是放料槽②是推料气缸。

在实验过程中将物料放入机构①中通过控制机构②推料气缸将机构①中的物料推放到输送带上面。

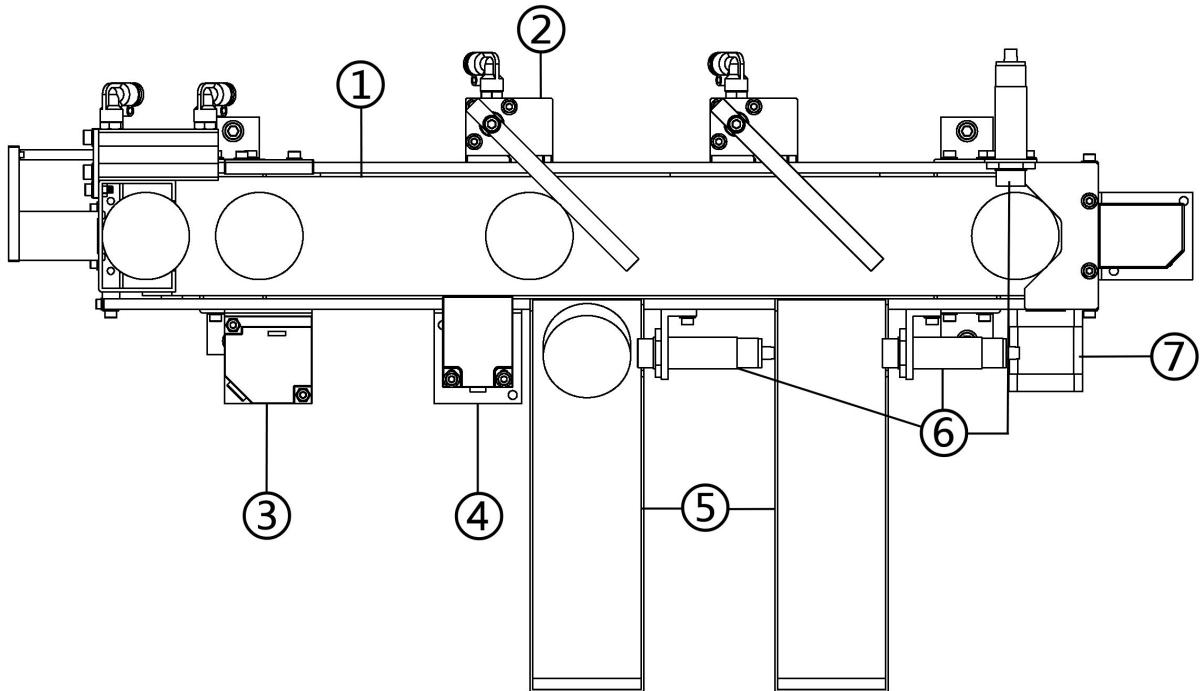


图 1.2 一号台—运料机构 (视图 2)

在图 1.2 中，①是输送带②是旋转气缸③是光电漫射传感器④是电涡流传感器⑤是选料槽⑥是电容式接近开关⑦是步进电机。

输送带由⑦步进电机带动完成运送物料的工作，③④传感器完成物料属性检测工作，②旋转气缸机构可以将不符合要求的物料推送到⑤选料槽中，⑥则配合检测物料到位情况。

从上面的一号台运料机构的视图 1 与视图 2 中看到运料机构可以完成推料、检测物料、挑选物料的工作。

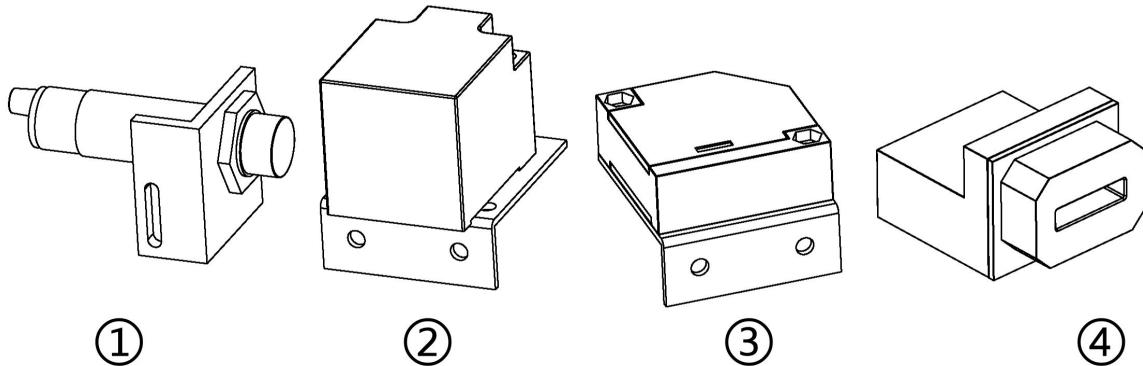
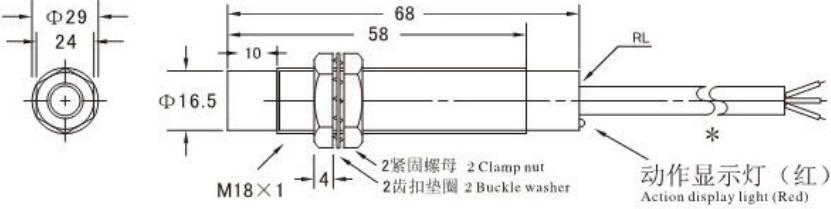
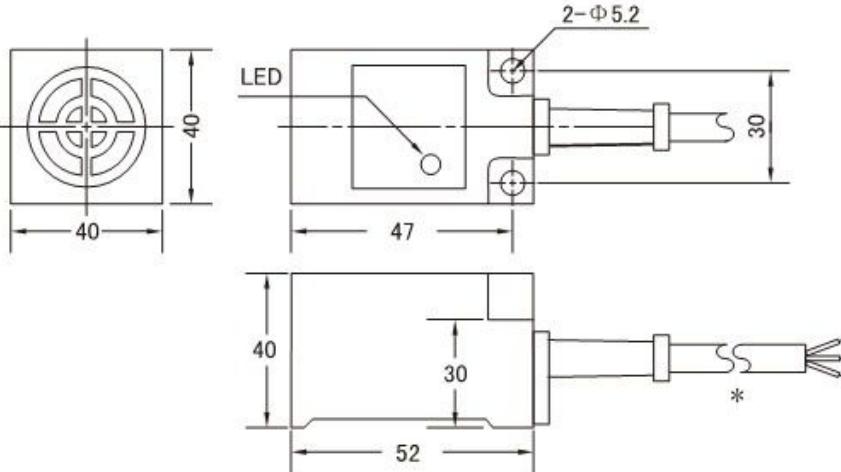


图 1.3 一号台—运料机构 (传感器)

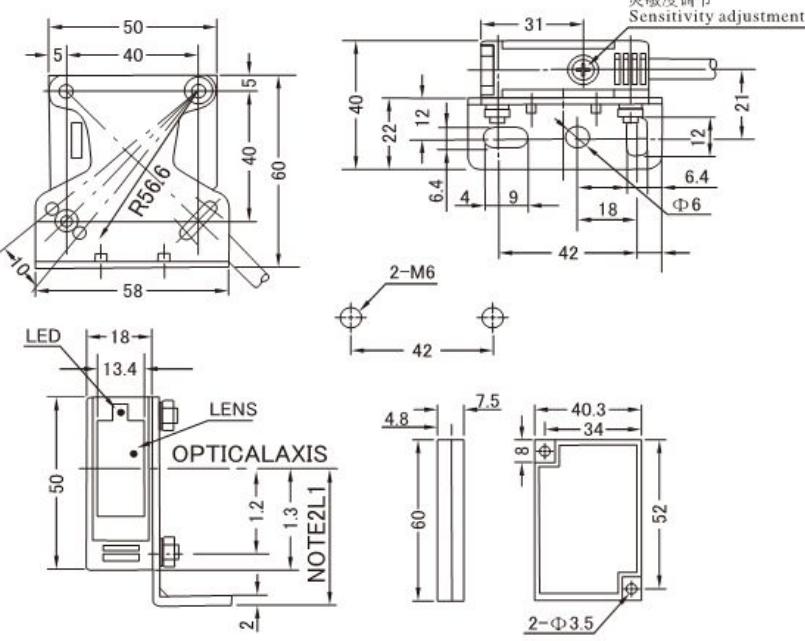
如图 1.3 所示，①是电容式接近开关具体参数如下表所示。

型号	LJC18A3-B
尺寸	
安装方式	非屏蔽式
检测距离	10mm ± 10%
设定距离	0—8mm
滞后距离	检测距离的10%以下
检测物体	任何介电物质
标准检测物体	铁 SPCC 50*50*1mm
响应频率	DC:0.5kHz
电源电压	直流型: DC12—24V(6-36V) 脉动(P-P) 10%以下
耐电压	AC1000V 50/60Hz 1min 充电部分与外壳间
电压的影响	额定电源电压范围±15%以内、额定电源电压值时±10%检测距离以内
消耗电流	N.P型: 13mA 以下, D型: 0.8mA 以下, A型: 1.7mA 以下
控制输出	N.P型: 300mA 以下, D型: 200mA 以下, A型: 400mA 以下
回路保护	N.P.D型: 逆连接保护、浪涌吸收、负载短路保护, A型: 浪涌吸收
环境温、湿度	动作时、保存时: 各-30—+65℃, (不结冰、不结霜), 动作时、保存时: 各35—95%RH
温度的影响	温度范围-30—+65℃, +23℃时、±15%检测距离以内, 温度范围-25—+60℃, +23℃时、±10%检测距离以内
绝缘阻抗	50MΩ 以上 (DC500兆欧表) 充电部分与外壳间
材质	外壳: 黄铜镀镍; 检测面: ABS
保护构造	IP67 (IEC 规格)

如图 1.3 所示，②是电涡流传感器具体参数如下表所示。

型号	TL-N20M
尺寸	
安装方式	非屏蔽式
检测距离	20mm ± 10%
设定距离	0—17mm
滞后距离	检测距离的10%以下
检测物体	磁性金属（非磁性金属时检测距离减小）
标准检测物体	铁50*50*1mm
响应频率	DC: 0.5kHz AC: 25Hz
电源电压	直流型: DC12—24V(6-36V) 脉动 (P-P) 10%以下
耐电压	AC1000V 50/60Hz 1min 充电部分与外壳间
电压的影响	额定电源电压范围±15%以内、额定电源电压值时±10%检测距离以内
消耗电流	N.P 型: 13mA 以下, D 型: 0.8mA 以下, A 型: 1.7mA 以下
控制输出	N.P 型: 300mA 以下, D 型: 200mA 以下, A 型: 400mA 以下
回路保护	N.P. D 型: 逆连接保护、浪涌吸收、负载短路保护, A 型: 浪涌吸收
环境温、湿度	动作时、保存时: 各-30—+65℃, (不结冰、不结霜), 动作时、保存时: 各35—95%RH
温度的影响	温度范围-30—+65℃, +23℃时、±15%检测距离以内, 温度范围-25—+60℃, +23℃时、±10%检测距离以内
绝缘阻抗	50MΩ 以上 (DC500兆欧表) 充电部分与外壳间
材质	外壳: ABS; 检测面: ABS
保护构造	IP67 (IEC 规格)

如图 1.3 所示，③是光电漫射传感器具体参数如下表所示。

型号	E3JK-R4
尺寸	
检测方式	反馈反射式
检测范围	4m ± 10%
检测目标	不透明物体
检测范围调节	灵敏度调节器
响应时间	30ms
接通延时	1.5ms
光源	红外光660nm
电源电压	直流型: DC12—24V(6-36V)脉动(P-P) 10%以下
耐电压	AC1000V 50/60Hz 1min 充电部分与外壳间
电压的影响	额定电源电压范围±15%以内、额定电源电压值时、±10%检测距离以内
功率电流	3VA 以下
控制输出	2A 以下(触点寿命: 10万次)
允许冲动和震动	B≤30g, T≤11ms, F≤55Hz, A≤1mm
环境温、湿度	动作时、保存时: 各-30—+65℃ (不结冰、不结露), 动作时、保存时: 各35—95%RH
温度的影响	温度范围-30—+65℃, +23℃时、±15%检测距离以内, 温度范围-25—+60℃, +23℃时、±10%检测距离以内
绝缘阻抗	50MΩ以上(DC500兆欧表) 充电部分与外壳间
材质	外壳: 铝压铸 ABS, 检测面(透镜): PMMA
保护构造	IP67 (IEC 规格)

一号台捡取物料机构如下图 1.4 所示，其中①②是两个双向气缸，③为单向气缸。①气缸可以驱动气缸末端连接的机构横向运动，而②气缸则可以驱动气缸末端连接的机构纵向运动，③号标识位置是由气缸控制的吸盘，将气缸打开的时候产生吸力将物体与吸盘紧紧吸住。

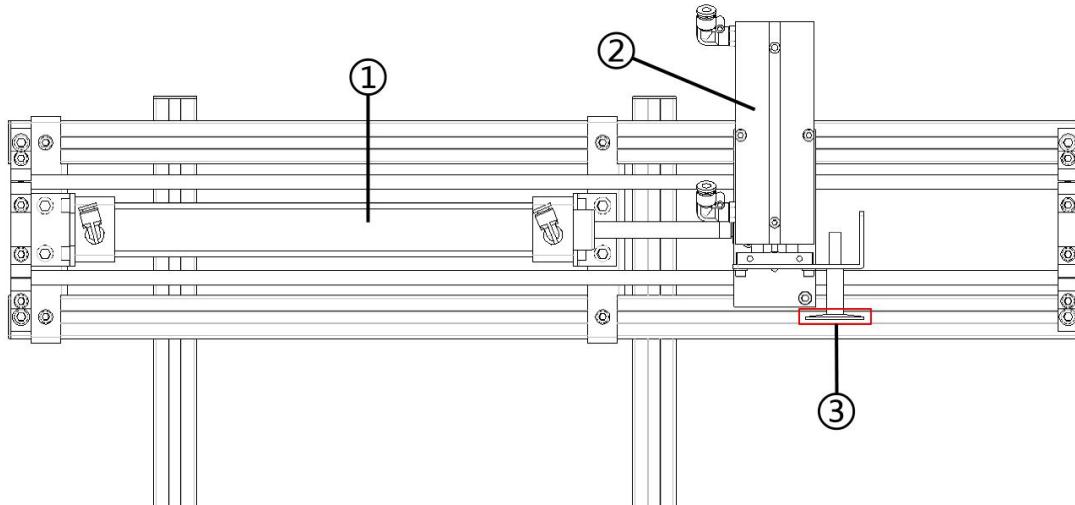


图 1.4 一号台—捡取物料机构示意图

1.2 二号台结构

二号台整体机构 3D 图如下图所示。

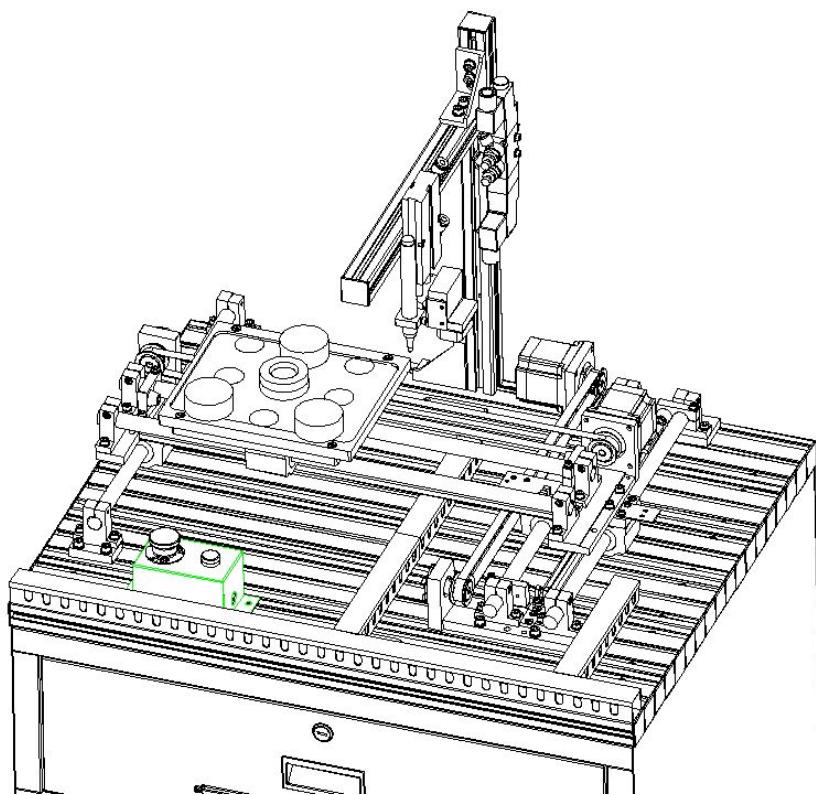


图 1.5 二号台—3D 形体图

如图 2.5 所示为二号台上的二维运动平台。

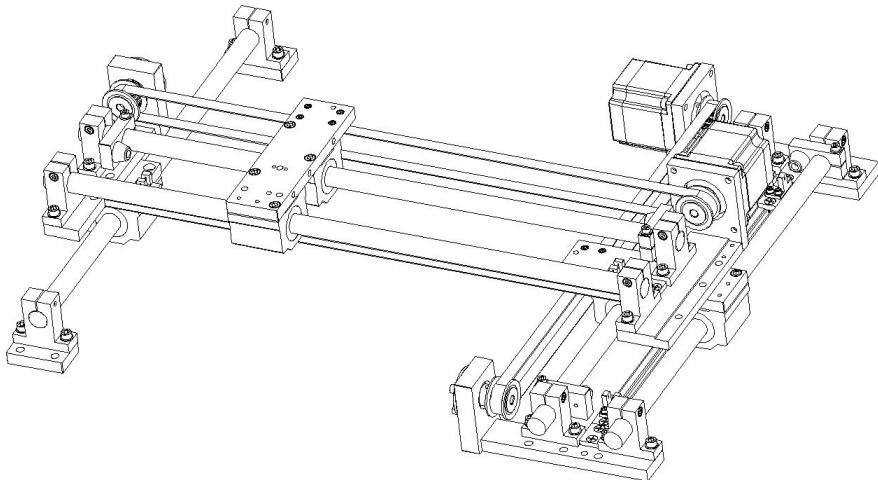


图 1.6 二号台—二维运动台

在上面图 1.5 与图 1.6 中，整个二维运动机构是由两个运动轴组成每个轴分别使用步进电机带动。在连接机构上面安装了放置物料的载物台完成特定的送料工作。载物台作为两用的操作平台还可以用于绘制运动轨迹的一些应用。

在二维运动机构的正上方为一个双向气缸，气缸可以控制机构纵向运动可以让气缸连接杆上的笔向下运动以达到绘制二维运动机构的运动轨迹的工作。

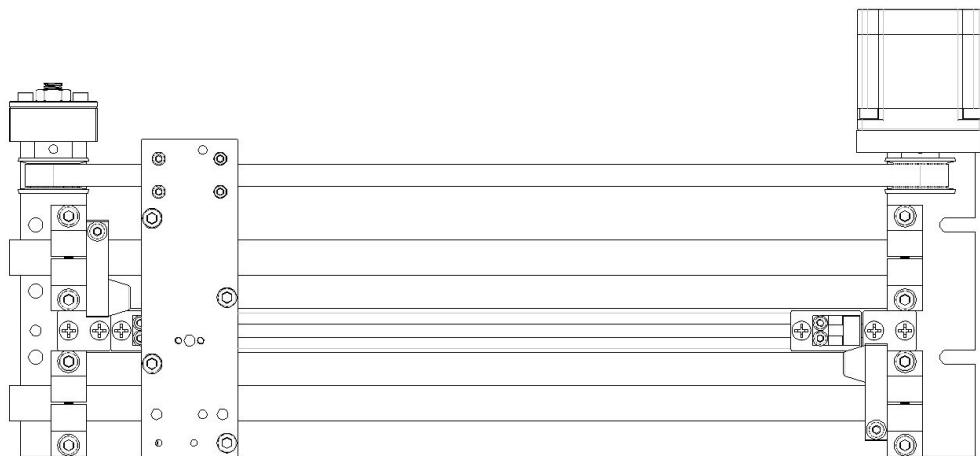


图 1.7 二号台一直线运动机构示意图

如图 1.7 所示，二维运动机构是由 2 个直线运动机构组成的。其中运动是由步进电机带动同步带进行传动的一个机构。

1.3 三号台结构

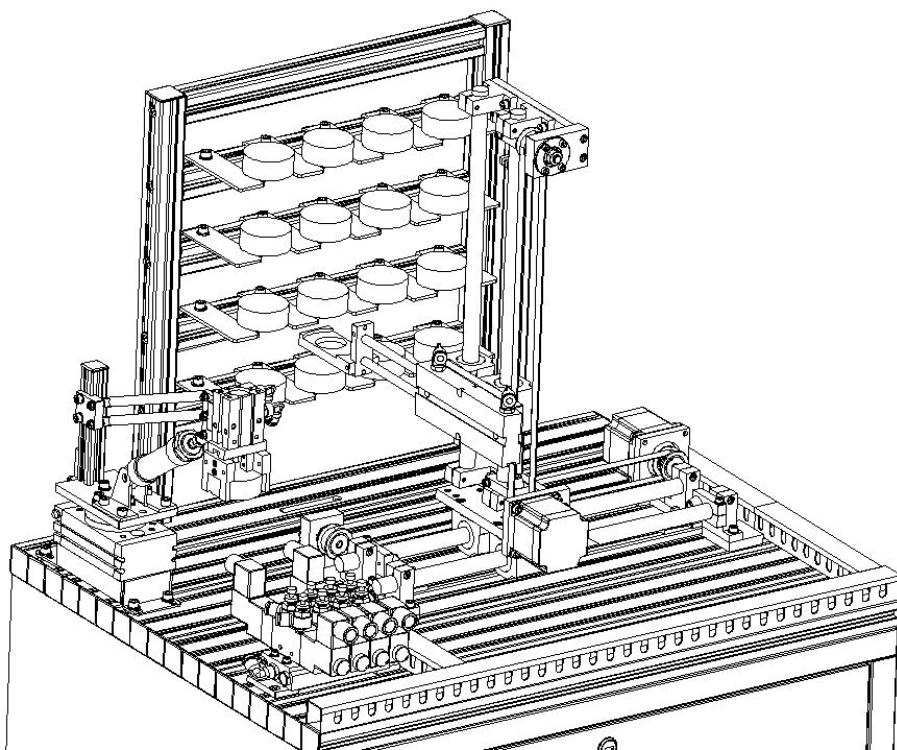


图 1.8 三号台—3D 形体图

如图 1.8 所示为三号台的 3D 形体图左部分的机械手可以从 2 号台将物体捡取并放置到 3 号台上完成 2 个平台之间的连接运送工作。而右部分的机构则是立体仓库机构，用户根据程序设定放置物料到仓库当中。

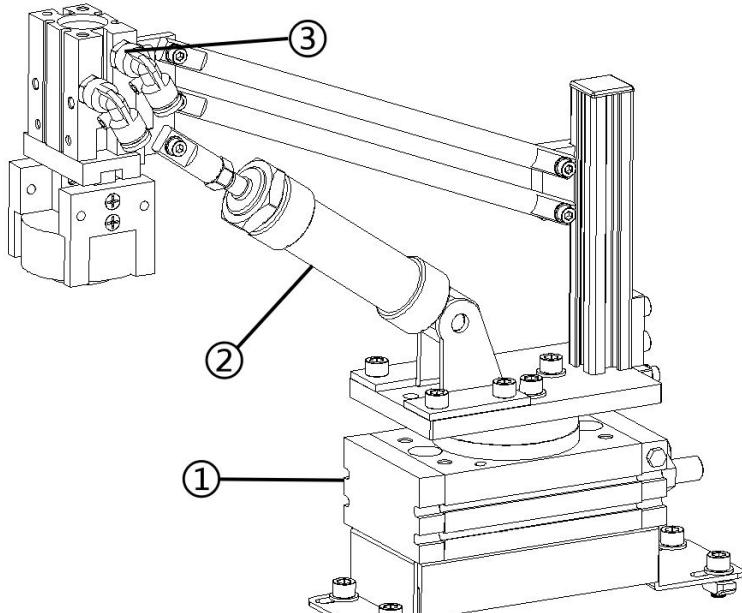


图 1.9 三号台—机械手结构

如图 1.9 所示为三号台的机械手结构①是旋转气缸控制气缸连接的机构整体旋转，②是一个双向气缸控制通过一个连杆机构控制机械手向上向下运动，③是一个单向气缸控制机械手抓爪抓紧物料。

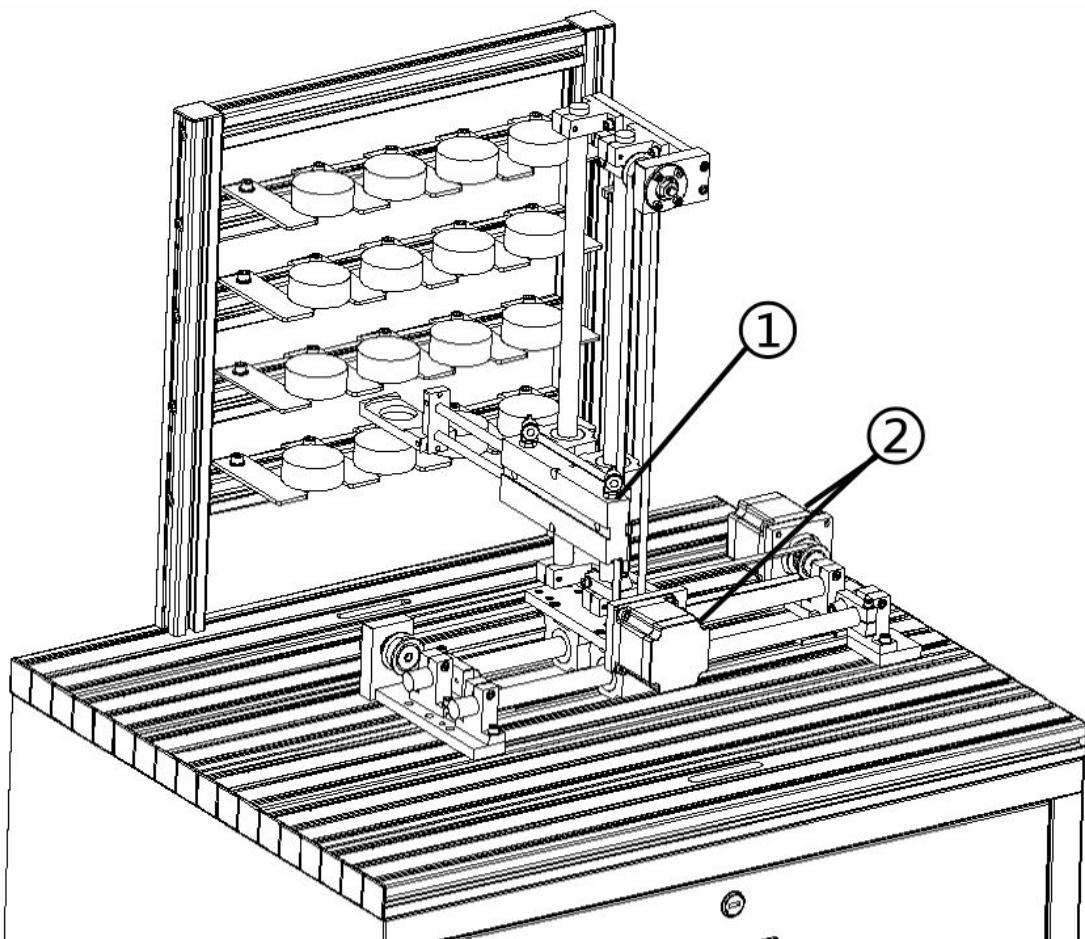


图 1.10 三号台—立体仓库结构

如图 1.10 所示，三号台立体仓库部分①是一个双向气缸控制气缸连接末端的物料槽向前推送物料。②是两个步进电机控制物料槽的二维运动。

第二章 电气连接部分

2.1 接口板定义

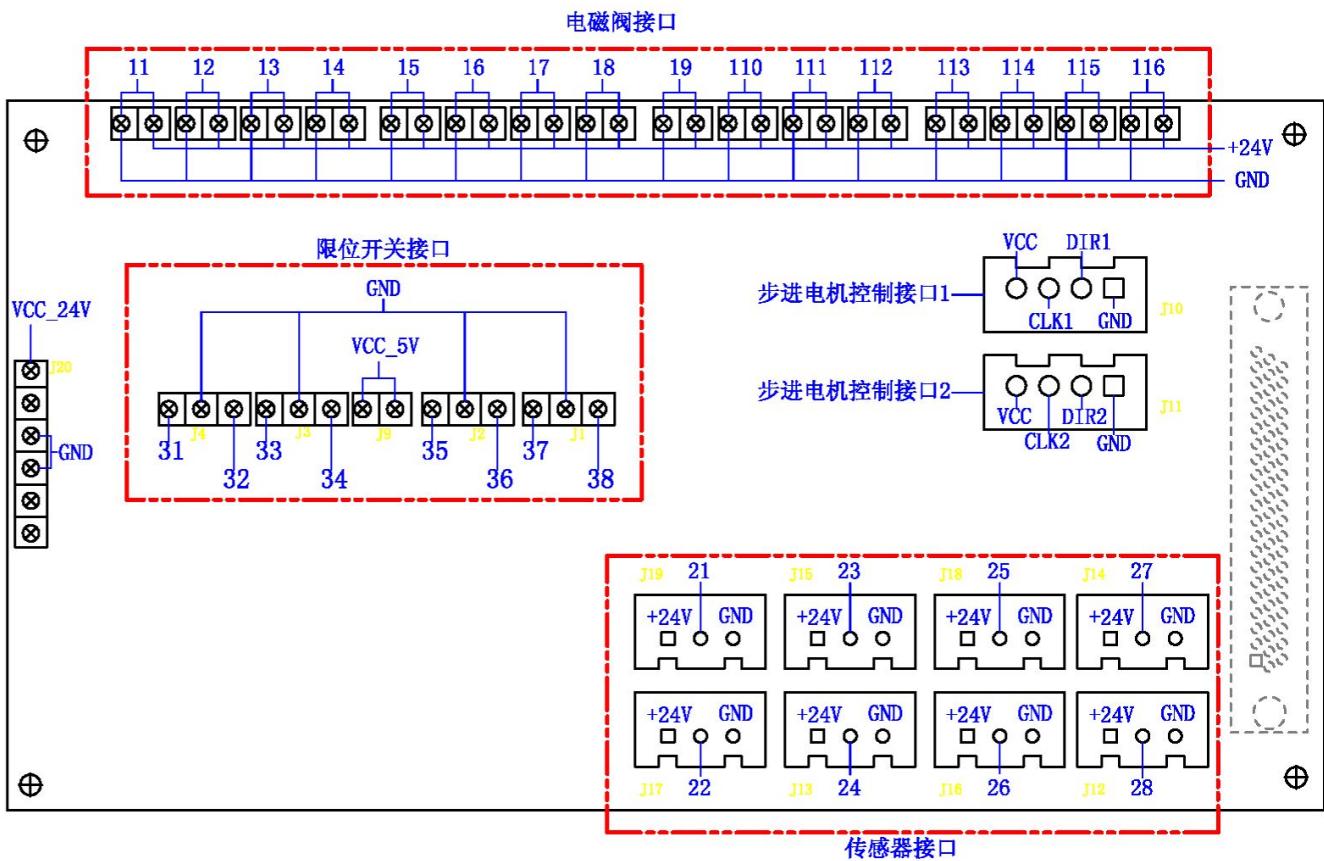


图 2.1 电气连接—接口板定义

- 每套机电平台有三个部分组成：输送线分拣部分、XY 二维平台部分、立体仓库部分，每个部分用一个接口板，三个接口板都分开使用 DC24V 供电。
- 每个接口板的 DC24V 供电从 J20 输入，J9 上的 “VCC_5V” 从主控板上引入，给限位开关供电。
- 如上图所示：31~38 是限位开关接口，J10~J11 是步进电机控制器接口，接步进电机驱动器，21~28 是传感器接口，11~116 是气动电磁阀接口。

2.2 限位开关连线

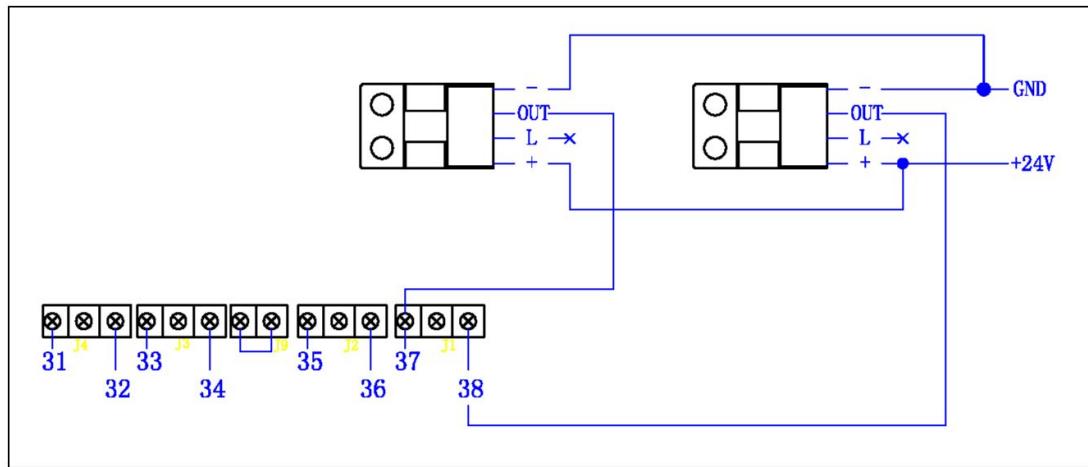
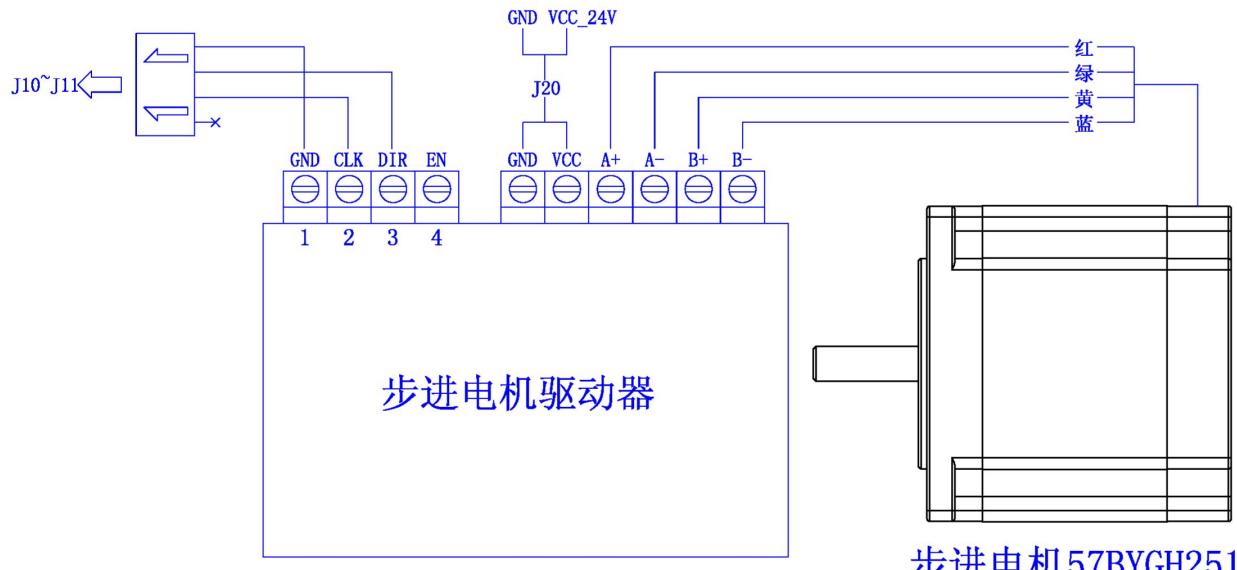


图 2.2 电气连接—限位开关接线定义

如图 2.2 所示是限位开关的接线定义，限位开关的信号脚默认接到 37,38 号口的 IO 上面。

2.3 驱动器与电机连线



注：工作台具体接线说明请参看[附表文件](#)。

第三章 软件部分

3.1 版权信息与命名规定

3.1.1 版权信息

本产品软件、驱动程序以及相关套件均由 武汉德普施科技有限公司 自主研发，驱动程序以及演示程序遵循 **GNU GPL v2.1**（协议内容请查看[附表文件](#)）开放源代码可自由发布以及应用。程序遵循协议约定所有代码托管在代码仓库 GitHub(<https://github.com/DepushTechnology/DYSYS>)。

3.1.2 命名规定

一、为了简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，为与其它函数名分离其统一的前缀为“**API_**”，如：初始化函数“**Init**”命名为“**API_Init**”。

二、函数名以及关键字缩写

缩写	全称	中文意义	缩写	全称	中文意义
Sys	System	系统	gv_	Global Variable	全局变量前缀
AD	Analog convert to Digital	模数转换	m_	Member Variable	成员变量前缀
Rcv	receive	接收	t_	Temporary Variable	临时变量前缀

3.2 使用纲要

3.2.1 使用上层用户接口函数，高效，简单

如果您只关心可重构机电平台操作应用并使用在**Windows**平台上，您不必了解复杂的硬件接口程序命令结构，我们推荐您使用上层用户接口函数，他们就是几个简单的**API**接口函数模式，具有灵活性、可靠性、高效性。如：**DYMECHNICALPLATFORM_API bool API_SysInit(void)**、**DYMECHNICALPLATFORM_API void API_SysQuit(void)**等。您使用这些函数您不需要硬件通讯打交道，其复杂的控制与通讯细节完全封装到了上层用户函数中。

3.2.2 函数库源文件中哪些对您不是必须的

公共类文件**CnComm.h**文件中包含的函数可以直接操作底层串口设备并具备安全线程功能这个类文件可以方便您直接与串口通讯，它是对我们公司的驱动接口程序的一种补充，也是驱动程序的核心。这个文件您不需要去了解，封装好的接口函数可以直接使用。您可能使用的不是普通**Windows**

系统或者您需要在嵌入式的Windows设备上面的扩展应用设备这时候您就需要加入CnComm.h文件，进行重新编译。

3.3 设备操作接口函数介绍

3.3.1 设备驱动接口函数总列表

一、VC++函数列表(DyMechanicalPlatform.dll)

函数名	函数功能
bool API_SysInit(void)	系统初始化
void API_SysQuit(void)	系统关闭
bool API_FeedMotor_Start(void)	进料电机持续转动
bool API_FeedMotor_Stop(void)	进料电机停止转动
bool API_MotorStart(int motorNum,int direction,int stepNum)	指定电机运动
bool API_MotorStop(int motorNum)	指定电机停止
bool API_MotorReset(int motorNum)	指定电机复位
bool API_SetMotorSpeed(int motorNum,int speed)	指定电机速度
bool API_solenAction(int solenNum,int statue)	指定电磁阀动作
bool API_solenAdvace(int platformNum,int statue)	指定工作台电磁阀全部动作
void API_getSolenStatue(int* SolenStatue)	查询继电器状态
bool API_getMotorStatue(int* motorStatue)	获取电机运动状态
bool API_getPlatformStatue(int PlatformNum,int platformStatue[8])	查询指定工作台的输入口状态
bool API_getSensorStatue(int SensorStatue[8])	查询传感器状态
bool API_getADSensor(double& value)	查询传感器 AD 值
bool API_RcvInit(void)	数据接收初始化

二、VC++函数列表(ColorDis.dll)

函数名	函数功能
bool APIColor_Init(int ComPort)	初始化设备打开句柄
void APIColor_Quit(void)	关闭设备端口释放句柄
void APIColor_GetColor(int& Red,int& Green,int& Blue)	获取采样 RGB 值

三、LabView库子VI列表(API DYM.lvlib)

子VI名称	子VI功能
API SYSINIT	系统初始化
API SYSQUIT	系统关闭
API Feed Motor START	进料电机持续转动
API Feed Motor Stop	进料电机停止转动
API Motor Start	指定电机运动
API Motor Stop	指定电机停止
API Motor Restart	指定电机复位
API Set Motor Speed	指定电机速度
API Solen Action	指定电磁阀动作
API Solen Advance	指定工作台全部电磁阀动作
API get solen statue	查询继电器状态
API get motor statue	获取电机运动状态
API get platform statue	查询指定工作台的输入口状态
API get Sensor Statue	查询传感器状态
API get Sensor AD	查询传感器 AD 值
API Rcv Data Init	数据接收初始化

四、LabView库子VI列表(ColorSensorlib.lvlib)

函数名	函数功能
ColorAPI_GetData	初始华设备打开句柄
ColorAPI_Init	关闭设备端口释放句柄
ColorAPI_Quit	获取采样 RGB 值

3.3.2 设备功能函数原型说明

使用须知:

Visual C++ &C++ Builder

要使用如下函数关键的问题是:

首先, 必须在您的源程序中包含如下语句:

```
#include "DYSYSAPI.h"
```

注: 以上语句采用的头文件地址您应根据您的头文件实际情况定义书写。

LabView

要使用如下函数关键问题是：

需要您将我们开发包中提供给您的API文件夹完整的拷贝到您的项目中。

Visual Studio IDE & C++ Builder IDE

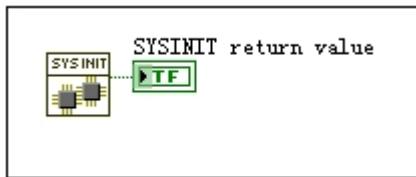
在IDE环境下使用如下函数的一个关键问题是首先必须将我们提供的头文件（DYSYSAPI.h）写入您的源文件头部。如：#include “DYSYSAPI.h”，然后在将”DyMechanicalPlatform.lib”库文件加入您的工程当中。

- 打开设备端口与设备进行连接

Visual C++ &C++ Builder:

```
bool API_SysInit(void)
```

LabVIEW:



功能：打开设备端口与设备进行连接。

参数：空

返回值：布尔类型，如果找到设备并成功打开设备返回为“真”。

例子：

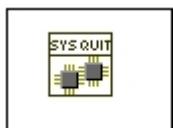
```
void Func(){
    if(!API_SysInit()) { return; }
}
```

- 关闭设备端口释放设备句柄

Visual C++ &C++ Builder:

```
void API_SysQuit(void)
```

LabVIEW:



功能：关闭设备端口释放设备句柄。

参数：空

返回值：空

例子：

```
void Func(){
    API_SysQuit();
}
```

- 进料电机开始运行

Visual C++ &C++ Builder:

```
bool API_FeedMotor_Start(void)
```

LabVIEW:



功能：进料电机开始运行。

参数：空。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_FeedMotor_Start();

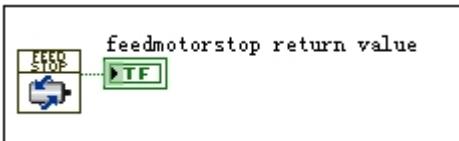
}
```

- 进料电机停止运行

Visual C++ &C++ Builder:

```
bool API_FeedMotor_Stop(void)
```

LabVIEW:



功能：进料电机停止运行。

参数：空。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_FeedMotor_Stop();

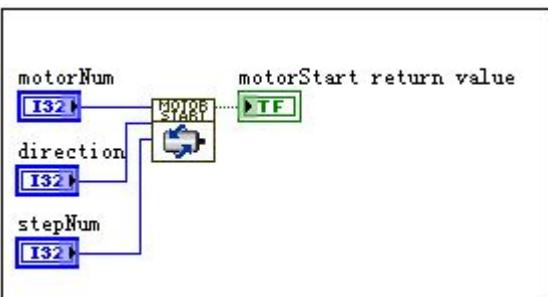
}
```

- 指定电机开始运行

Visual C++ &C++ Builder:

```
bool API_MotorStart(int motorNum,int direction,int stepNum)
```

LabVIEW:



功能：指定电机开始运行。

参数：**motorNum**:电机编号；

Direction:电机方向(取值为0,1);

stepNum:电机运行步数。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_MotorStart(3,1,1000); //3号电机正方向移动1000个步长

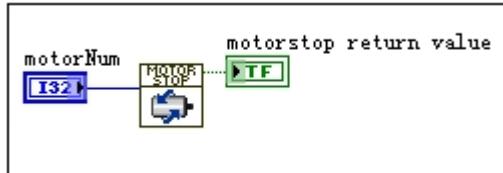
}
```

- 指定电机停止运行

Visual C++ &C++ Builder:

```
bool API_MotorStop(int motorNum)
```

LabVIEW:



功能：指定电机停止运行。

参数：motorNum:电机编号。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_MotorStop(3); //3号电机停止运动

}
```

- 指定电机复位

Visual C++ &C++ Builder:

```
bool API_MotorReset(int motorNum)
```

LabVIEW:



功能：指定电机复位。3,4,5,6回到复位位置。

参数：motorNum:电机编号。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_MotorReset(3); //3号电机复位
```

}

- 设定电机速度

Visual C++ &C++ Builder:

```
bool API_SetMotorSpeed(int motorNum,int speed)
```

LabVIEW:



功能：设定电机速度。

参数： motorNum:电机编号；

Speed:速度值 。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

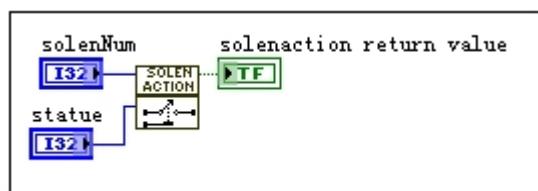
    //设定3号电机以1300Hz的PWM脉冲驱动运行
    API_SetMotorSpeed(3,6000000/(2*1300));
}
```

- 指定继电器动作

Visual C++ &C++ Builder:

```
bool API_solenAction(int solenNum,int statue)
```

LabVIEW:



功能：指定继电器动作。

参数： solenNum:继电器编号；

statue: 状态值(0:关;1:开)。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_solenAction(0,1); //打开1号继电器

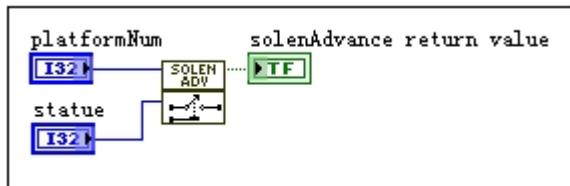
}
```

- 打开或关闭指定平台上的所有继电器

Visual C++ &C++ Builder:

```
bool API_solenAdvace(int platformNum,int statue)
```

LabVIEW:



功能：打开或关闭指定平台上的所有继电器。

参数：**motorNum:**电机编号；

Speed:速度值。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_solenAdvace(0,1); //打开1号平台所有继电器

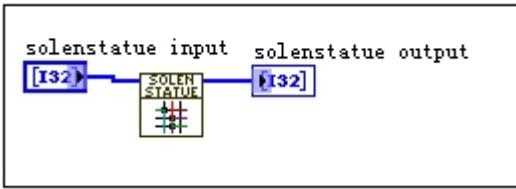
}
```

- 获取继电器状态

Visual C++ &C++ Builder:

```
void API_getSolenStatue(int* SolenStatue)
```

LabVIEW:



功能：获取继电器状态。

参数： **SolenStatue**：传入类型为int长度为48的数组指针，用于保存继电器状态数据。

SolenStatue数组值为1继电器处于打开状态，0则继电器处于关闭状态。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    int tSolenStatue[48];

    memset(tSolenStatue,0,sizeof(int)*48); //初始化数组

    API_RcvInit(); //通知设备完成一次数据采样

    API_getSolenStatue(tSolenStatue); //打开1号平台所有继电器

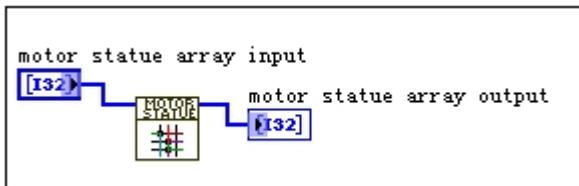
}
```

● 获取电机状态

Visual C++ &C++ Builder:

```
bool API_getMotorStatue(int* motorStatue)
```

LabVIEW:



功能：获取电机状态。

参数： **motorStatue**：传入类型为int长度为6的数组指针，用于保存电机运行状态数据。

motorStatue数组值为1电机处于运行状态，0则电机处于停止状态。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
```

```

if(!API_SysInit()) { return; }

int tmotorStatue[6];

memset(tmotorStatue,0,sizeof(int)*6);

API_RcvInit(); //通知设备完成一次数据采样

API_getMotorStatue(tmotorStatue);

}

}

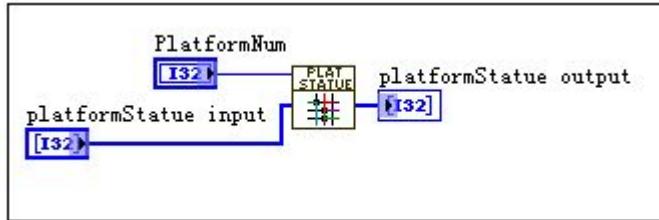
```

- 获取平台IO输入口状态

Visual C++ &C++ Builder:

```
bool API_getPlatformStatue(int PlatformNum,int platformStatue[8])
```

LabVIEW:



功能：获取平台IO输入口状态。

参数：**PlatformNum**: 平台号；

platformStatue: 传入类型为int长度为8的数组指针，用于保存平 台 IO 输入口状态。

platformStatue数组值为1输入口为高电平，数值为0输入口为低电平。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```

void Func(){

if(!API_SysInit()) { return; }

int tplatformStatue[8];

memset(tplatformStatue,0,sizeof(int)*8);

API_RcvInit(); //通知设备完成一次数据采样

API_getPlatformStatue(0,tplatformStatue);

}

}

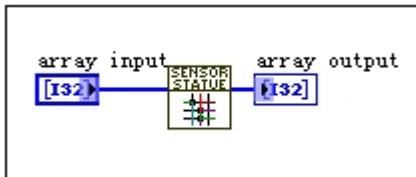
```

- 获取传感器状态

Visual C++ &C++ Builder:

```
bool API_getSensorStatue(int SensorStatue[8])
```

LabVIEW:



功能： 获取传感器状态。

参数： **SensorStatue**： 传入类型为int长度为8的数组指针，用于保存平台 传感 器 状 态 。

SensorStatue数组值为1接收到传感器的电平为高电平，数 值为0传感器的电平为低电平。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    int tSensorStatue[8];
    memset(tSensorStatue,0,sizeof(int)*8);

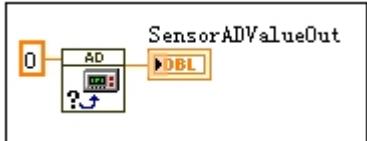
    API_RcvInit(); //通知设备完成一次数据采样
    API_getSensorStatue(tSensorStatue);
}
```

- 获取AD端口电平

Visual C++ &C++ Builder:

```
bool API_getADSensor(double& value)
```

LabVIEW:



功能： 获取AD端口电平。

参数： **value**： 传入类型为double型的引用。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    double tADValue = 0;

    API_RcvInit(); //通知设备完成一次数据采样

    API_getADSensor(tADValue);

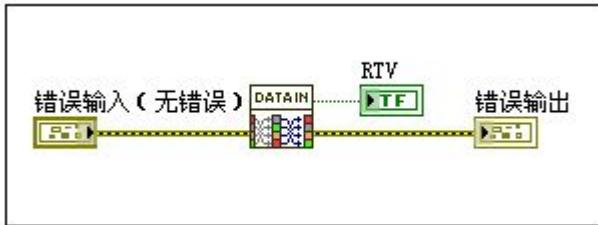
}
```

- 通知设备完成一次数据采样

Visual C++ &C++ Builder:

```
bool API_RcvInit(void)
```

LabVIEW:



功能：通知设备完成一次数据采样。

参数：空

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_RcvInit(); //通知设备完成一次数据采样

}
```

使用须知：

Visual C++ &C++ Builder

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

```
#include "ColorDisAPI.h"
```

注：以上语句采用的头文件地址您应根据您的头文件实际情况定义书写。

LabView

要使用如下函数关键问题是：

需要您将我们开发包中提供给您的API文件夹完整的拷贝到您的项目中。

Visual Studio IDE & C++ Builder IDE

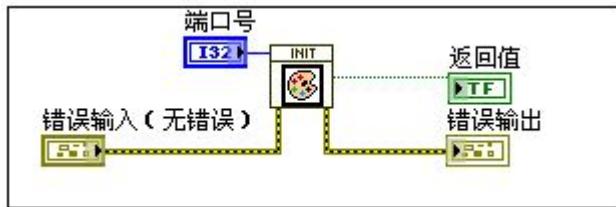
在IDE环境下使用如下函数的一个关键问题是首先必须将我们提供的头文件（ColorDisAPI.h）写入您的源文件头部。如：#include “ColorDisAPI.h”，然后在将” ColorDis.lib” 库文件加入您的工程当中。

- 打开颜色采集设备端口

Visual C++ &C++ Builder:

```
bool APIColor_Init(int ComPort)
```

LabVIEW:



功能：打开颜色采集设备端口。

参数：ComPort：传入类型为int的物理端口号。

返回值：布尔类型，如果打开设备成功返回 真。

例子：

```
void Func(){
    if(!APIColor_Init(10)) { return; }
}
```

- 关闭颜色采集设备端口

Visual C++ &C++ Builder:

```
void APIColor_Quit(void)
```

LabVIEW:



功能：关闭颜色采集设备端口。

参数：空

返回值：空

例子：

```
void Func(){
```

```
    APIColor_Quit();
```

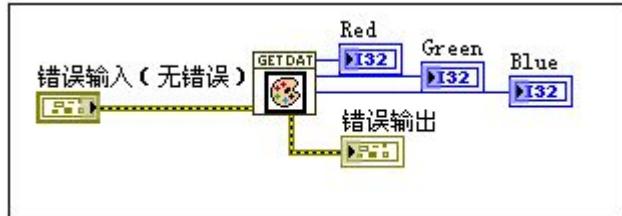
```
}
```

- 获取颜色采集设备当前采集到的RGB数值

Visual C++ &C++ Builder:

```
void APIColor_GetColor(int& Red,int& Green,int& Blue)
```

LabVIEW:



功能：获取颜色采集设备当前采集到的RGB数值。

参数：Red, Green, Blue: 传入int类型变量的引用。

返回值：空

例子：

```
void Func(){
```

```
    Int tRed,tGreen,tBlue;
```

```
    tRed = 0;tGreen = 0;tBlue = 0;
```

```
    APIColor_GetColor(tRed,tGreen,tBlue);
```

```
}
```

3.4 底层设备通讯协议

3.4.1 机电平台主控板通讯协议

通讯波特率： 57600；

底层协议基本通讯格式：

包头	长度	数据	包尾
0x55	数据长度+1	0xAA

- 通讯协议命令一览表

索引	数据	功能	返回
1	0xBB	握手指令	0x48 0x65 0x6c 0x6c 0x6f 0x4a 0x49 0x41 0x21
2	0xFB	进料电机持续转动	无
3	0xBF	进料电机停止转动	无
4	0xFA <u>电机编号(1byte)</u> <u>电机方向(1byte)</u> <u>运动步数(2byte)</u>	指定电机运动指定步数	无
5	0xFD <u>电机编号</u> -1 (1byte)	指定电机停止运动	无
6	0xFC <u>电机编号(1byte)</u>	指定电机复位	无
7	0xED <u>电机编号(1byte)</u> <u>速度值(2byte)</u>	指定电机速度值	无
8	0xFE <u>电磁阀编号</u> (1byte) <u>电磁阀状态</u> (1byte)	设定电磁阀状态	无
9	0xEF <u>工作台编号</u> (1byte) <u>电磁阀状态</u> (1byte)	设定指定工作台电磁阀全部动作	无
10	0xF6	查询状态数据	0x55 0x0e AD 数据(2byte) 传感器状态(1byte) I0 输入口状态(3byte) 电机运行状态(1byte) 继电器状态(6byte) 0xAA

注： 电机编号范围 1 - 6

工作台编号范围 0 - 2

电磁阀编号 0 - 47

电磁阀状态量 0: 关闭 ; 1: 打开;

3.4.2 颜色采集设备通讯协议

通讯波特率: 57600;

向设备查询指令:

指令	返回数据
0x54 0x41 0xd	Red 值(1byte) Green 值(1byte) Blue 值(1byte) 包尾(1byte)

3.5 样例程序使用介绍

3.5.1 样例程序功能简介

本产品样例程序作为设备套件的一部分我们提供完善的设备功能与实验展示。程序需要使用LabVIEW2011及以后的版本打开您可以自由修改发布我们的程序。程序由主界面代码、设备API以及功能子VI组成。

3.5.2 程序使用方法

一、可重构机电平台系统综合系统

本程序实现完成机电平台系统综合系统的单台控制、联动控制并能够单独控制每个机构。具有读写配置文件功能可以保存和读取配置参数。可完成教学、演示与设备调试功能。

步骤1: 首先打开“DYSYS.lvproj”项目文件, 会出现本程序的项目浏览器, 如下图3.1所示。

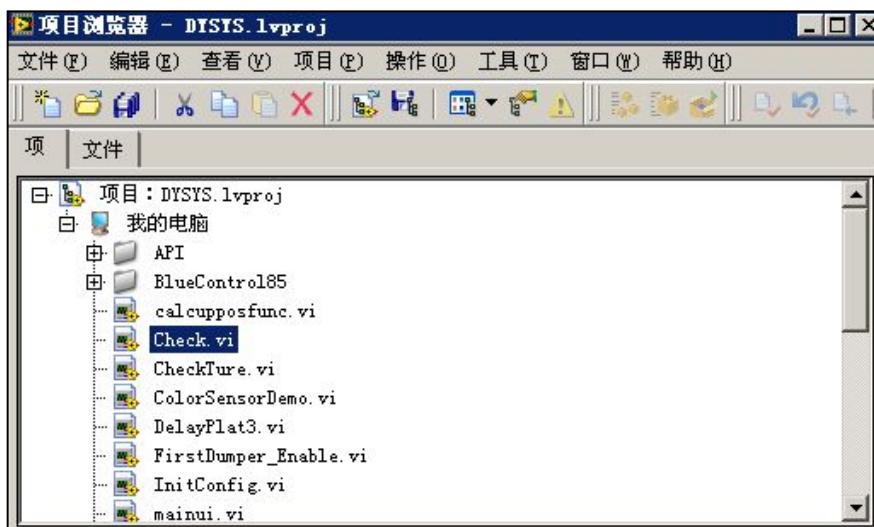


图3.1 DYSYS.lvproj—项目浏览器

步骤2：选择“mainui.vi”进入可重构机电平台前面板，如下图3.2所示。

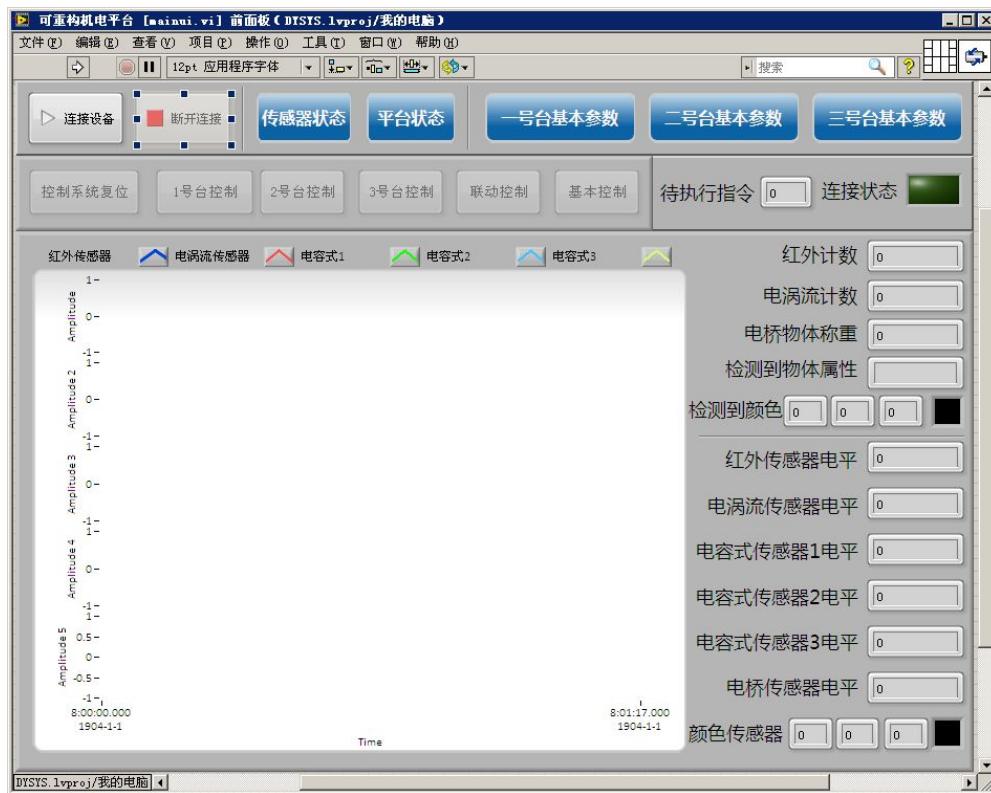


图3.2 可重构机电平台—前面板

步骤3：在LabVIEW中运行这个VI，然后点击连接设备程序将自动找到我们的设备并打开设备通讯端口。如果发送连接失败请您检查设备连接是否正常，控制主板是否上电。

步骤4：连接成功后在下面界面中您将看到所有传感器的电平状态，以及颜色采集设备传到主机上的采集数值，还有采集的电桥传感器的电平。

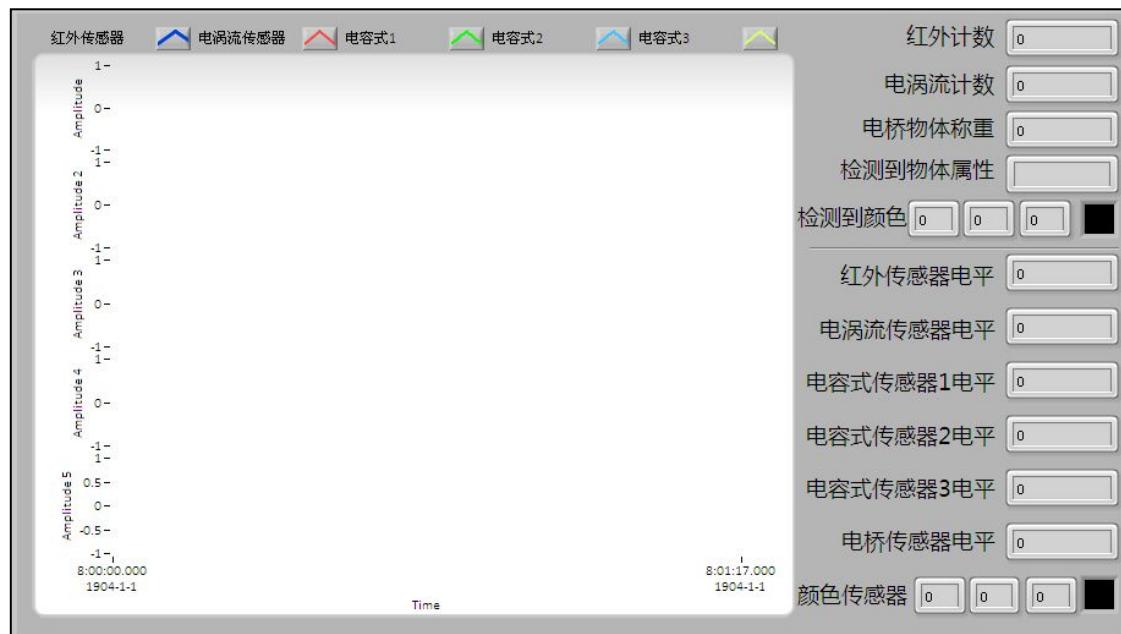


图3.3 可重构机电平台—传感器状态

步骤5：点击“平台状态”您将看到所有继电器状态，输入口状态，以及点击运行状态。如下图3.4所示。

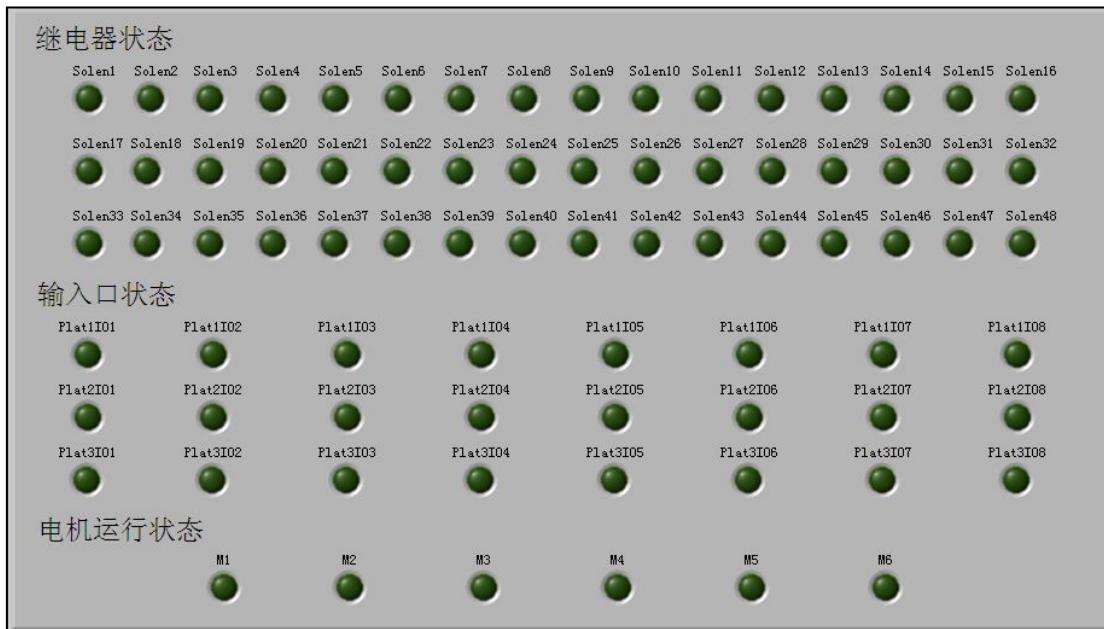


图3.4 可重构机电平台—平台状态

步骤6：点击“一号台基本参数”界面将切换到1号台基本参数设置界面中。如下图3.5所示。

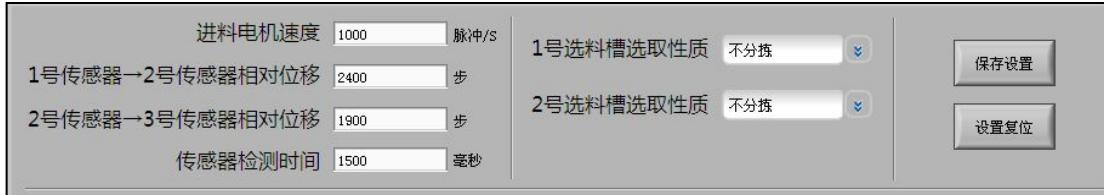


图3.5 可重构机电平台——号台基本参数设置界面

步骤7：在如上图3.4所示的界面中可以设定在物体进行一号台分拣的过程中的一些参数，包括进料电机的运行速度，1号传感器与2号传感器的相对位移，2号传感器与3号传感器的相对位移传感器的检测时间。1号、2号选料槽选取的物体性质。配置修改后电机保存设置该设置将被保存到配置文件中用于下一次执行时调用。

步骤8：点击“二号台基本参数”界面将切换到2号台基本参数设置界面中。如下图所示。

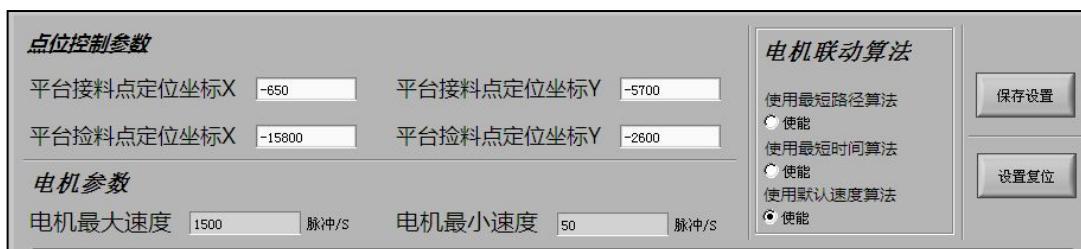


图3.6 可重构机电平台—二号台基本参数设置界面

步骤9：如图3.6所示的界面中可以设定二号平台接取一号平台物料的定位坐标以及三号平台捡取物料时二号平台的定位点坐标。在右侧的电机联动算法框中可以配置XY电机同时运动时使用的算法，当使用最短路径算法的时候2个电机同时运动同时停止以最短的路径到达目标位置，使用最短时间算法的时候2各电机以各自能够达到的最快速度到达目标位置，使用默认速度算法的时候2各电机使用各自的默认速度到达目标位置。同样的保存设置按钮将保存二号台的数据设置保存到配置文件当中以便下一次使用。

步骤10：点击“三号台基本参数”界面将切换到3号台基本参数设置界面中。如图3.7所示。



图3.7 可重构机电平台—三号台基本参数设置界面

步骤11：在如图3.7所示的界面中可以设定三号台的一些基本参数上半部分是三号台挂接的电桥传感器(选配)的标定，下半部分则是立体仓库的参数、电机参数以及放置策略的一些选择。同样的保存设置按钮可以将您的设置保存到配置文件中以便下一次打开程序的时候直接使用。在上半部分电桥传感器参数标定中根据您所放置砝码的重量分别填入砝码重量1以及砝码重量2获取电压后点击标定按钮点击计算将自动计算K、B参数值。程序将实时的转化电压值到重量值。在下半部分的立体仓库坐标参数中，放料点Z、X轴坐标为三号台机械手接料的坐标，在下面的坐标中填入相对于的坐标后在控制过程中三号台的机械手会自动将物料放入相应的坐标点中。电机联动算法与二号台的电机联动算法一致。放置策略指的是物料放置的顺序，如果是同规则一的放置策略那么运料机构的放置方法则是先放第一排第一排放满后再放第二排依次类推，而您选择使用规则二则是由上到下先放置第一列，第一列放满后再放置第二列。

步骤12：在上面的一些参数都配置完成了之后便可以开始进行各个平台的实验以及联动实验。

在与设备连接成功后下图中的按钮会由不可用变为可用。然后点击各个平台的控制软件根据程序脚本与上面设置的参数进行控制。



图3.8 可重构机电平台—控制按钮面板

步骤13：点击“基本控制”可进入基本控制面板，如图3.9所示。您可以点击相应的按钮对机电平台系统进行操作。



图3.9 可重构机电平台—基本控制面板

二、二号平台绘制控制程序

本程序使用脚本运行的开发方法，您可以写入脚本，控制程序将根据您的脚本实现二号平台的运动。

步骤1：打开“DYSYS_Part2.lvproj”项目文件，会出现本程序的项目浏览器，如图3.10所示。



图3.10 DYSYS_Part2.lvproj—项目浏览器

步骤2：点击“ncmainui.vi”进入系统主界面，如下图所示。

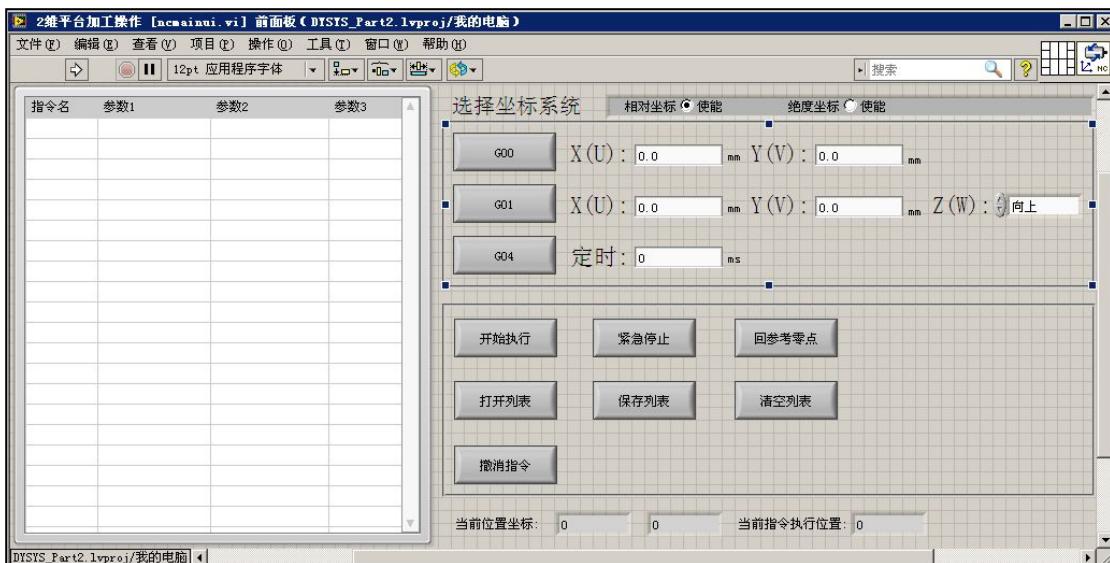


图3.11 ncmainui vi—前面板

步骤3：在LabVIEW中点击运行按钮，运行这个vi程序。整个程序分为2个部分左半部分作为指令的显示部分，您从文件加载的指令将会导入到这个表格当中。在程序的右半部分您可以添加指令到表格当中去，点击“开始执行”按钮二号台将会根据您的表格中的指令执行您的动作。

“开始执行”按钮：二号台根据脚本指令顺序执行动作。

“紧急停止”按钮：机构停止所有运动。

“回参考零点”按钮：机构回到参考零点。

“打开列表”按钮：读取GCode文件显示在列表当中。

“保存列表”按钮：保存列表中的脚本动作到脚本中。

“清空列表”按钮：清空当前列表当中的所有动作。

“撤销指令”按钮：从列表中撤销当前选定的动作。

“G00”按钮：快速定位。根据坐标系以及坐标参数进行定位。

“G01”按钮：直线插补。根据坐标系以及坐标参数进行插补操作。

“G04”按钮：延时操作。根据定时时间执行。

第四章 实训部分

本套系统的实训部分介绍包含系统控制程序的开发与实训。在使用系统时您需要对通用控制软件具有一定的了解。通过实训可以提高控制软件的使用技术以及通用上位机控制的开发方法，熟悉系统的控制流程。

4.1 与控制器“say hello”

在使用一套新设备时，人们习惯把“Hello World”作为第一个程序。渐渐的人们习惯用类似“Hello World”的程序作为一切程序的第一步，以此唤醒出人们对下边编写控制程序的乐观的一面。

如本节的题目一样，如何编写与设备“say hello”的程序是将要解决的问题。

通过前面几章的了解我们知道与我们控制器通讯可以直接采用封装好的API进行操作，那么我们看一下如何与设备“say hello”吧。

一、LabVIEW开发环境

如果您选用LabVIEW程序作为您的控制程序开发语言，那么首先打开LabVIEW编程语言开发环境。在这里我们使用LabVIEW2011为您演示。



图4.1 LabVIEW2011—启动窗口

点击图4.1中的项目，建立一个新的项目。接着LabVIEW为我们建立了一个未命名项目1我们先将这个项目保存到一个盘符下。在示例中我们将项目保存到了E盘下的LabViewPrj文件夹中并将项目重命名为DRPrj，如图4.2所示。

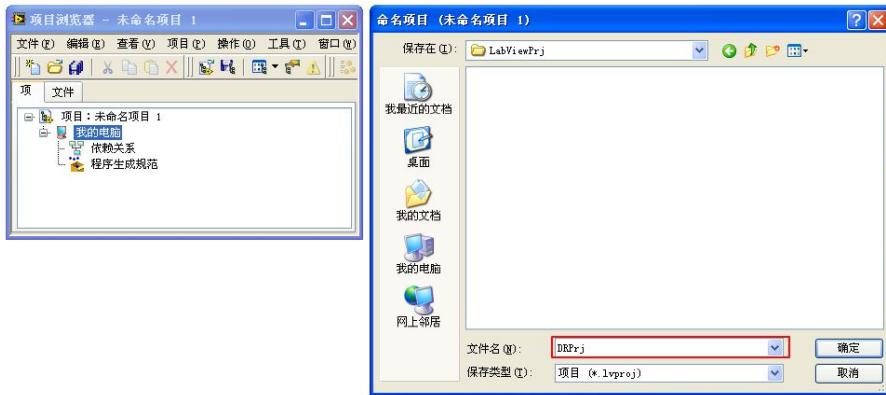


图4.2 LabVIEW2011—保存项目窗口

然后将光盘中的API文件夹复制到项目所在的文件夹中然后添加到项目中。添加文件夹首先在项目目录中“我的电脑”上点击右键选择“添加”，在添加的子选项卡选择“文件夹自动更新”选择刚才添加的API文件夹。

在完成上面项目建立的基础工作后，依照下面实训步骤进行我们第一个控制程序的设计。

步骤1：在菜单栏的文件中选择“新建VI”建立一个HelloDev的新VI。

步骤2：在前面板中放入“启动按钮”和“退出按钮”，再放入一个LED显示灯控件，并将其调整位置摆放整齐，如图4.3所示。



图4.3 HelloDev vi—前面板

步骤3：使用Ctrl+E键打开程序面板，放入事件结构基本框架。这里使用事件结构可以比较结构化的编写LabVIEW程序以完成程序流程。使用事件结构，可以使得包括一个或多个子程序框图或事件分支，结构执行时，仅有一个子程序框图或分支在执行。

步骤4：将2个按钮控件放到while循环中，并将While循环的条件接线端接为false，如图4.4所示。

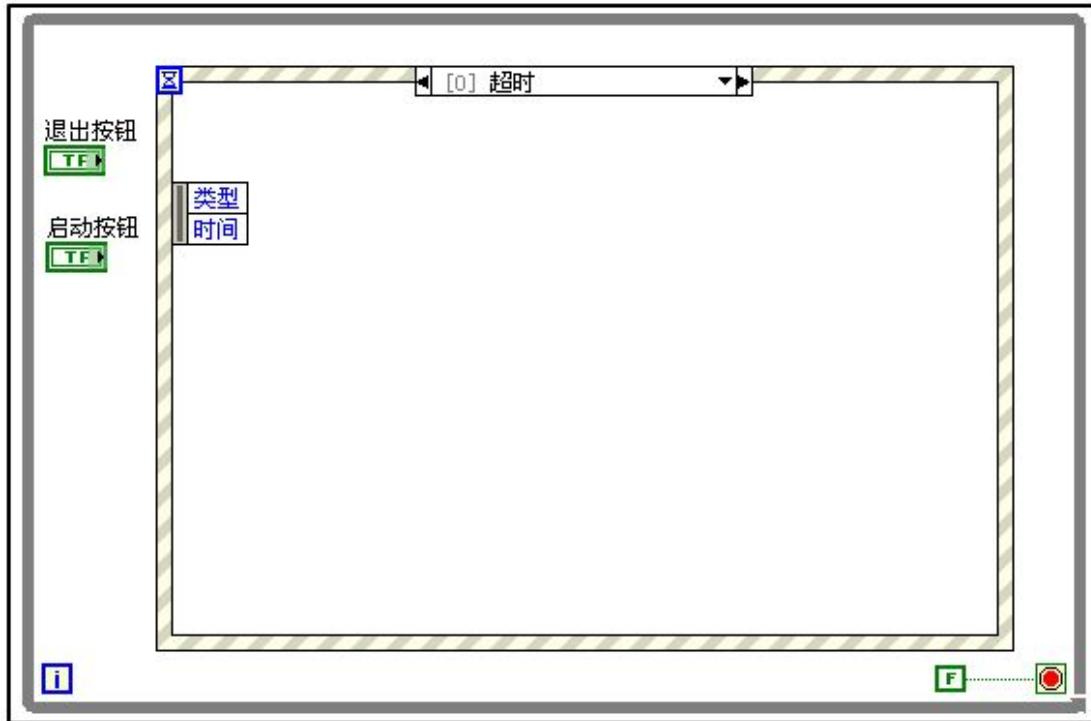


图4.4 HelloDev vi—程序面板

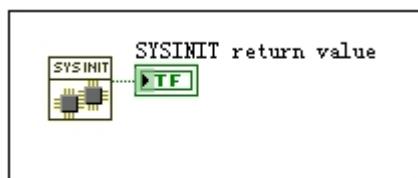
我们要完成与设备通讯的工作，这里需要使用2个API模块，在第三章中我们可以找到这2个API的原型。

- ① 打开设备端口与设备进行连接

Visual C++ &C++ Builder:

```
bool API_SysInit(void)
```

LabVIEW:



功能：打开设备端口与设备进行连接。

参数：空

返回值：布尔类型，如果找到设备并成功打开设备返回为“真”。

例子：

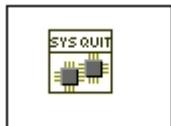
```
void Func(){
    if(!API_SysInit()) { return; }
}
```

② 关闭设备端口释放设备句柄

Visual C++ &C++ Builder:

```
void API_SysQuit(void)
```

LabVIEW:



功能：关闭设备端口释放设备句柄。

参数：空

返回值：空

例子：

```
void Func(){
    API_SysQuit();
}
```

步骤5：在程序面板中添加两个事件分支：①启动按钮：值改变；②退出按钮：值改变。

步骤6：在“启动按钮：值改变”的分支中我们通过打开项目浏览器中的API文件夹，找到“API SYSINIT”子VI将其拖入到程序框图中，其输出端与布尔控件连接，如图4.5所示。

步骤7：在“退出按钮：值改变”的分支中我们同样的打开API文件夹拖入“API SYSQUIT”子VI，如图4.5所示。

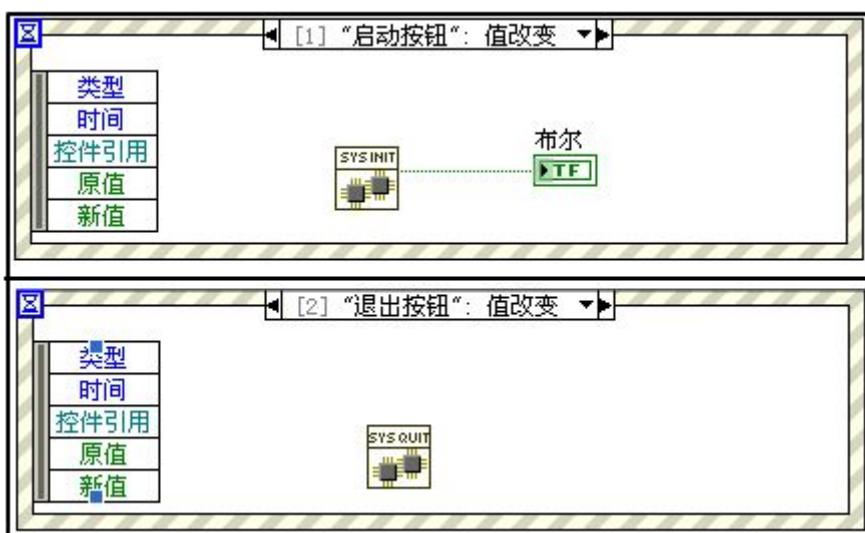


图4.5 HelloDev vi—事件分支

在这里我们使用“API SYSINIT”子VI其目的是使得程序自动查找获取我们控制器的设备端口

句柄，而在退出的时候我们则需要对端口句柄进行释放方便其他程序调用我们的设备资源，所以在退出的时候我们必须使用“API SYSQUIT”子VI。如果我们不执行“API SYSQUIT”子VI那么在我们再一次调用“API SYSINIT”子VI时就会发生错误，由于前一个执行过程中我们并没有释放资源导致资源一直处于占用状态。如果发生程序异常而没有调用“API SYSQUIT”子VI。那么需要您彻底关闭LabVIEW主程序，并重新运行它既可。

程序到这里编写完成，点击工具栏中的“运行”按钮后程序开始运行，然后点击前面板中的“启动按钮”发现前面板中LED灯点亮说明已经与设备通讯成功。退出时点击退出按钮后再终止整个LabVIEW程序。

二、实训习题

通过上面的讲解自己独立完成与控制器初始化通讯程序。

4.2 电机驱动

系统中共有5个步进电机，一号台上面有1个步进电机，二号台、三号台上分别有2个步进电机，系统中所有的传输与运动部分的大多数工作都是由步进电机来完成。经过上一个小结中我们已经对设备控制有了一个大概的了解，也成功的与设备通讯上了。那么这个小结我们开始控制系统中最核心部件“电机”。

一、LabVIEW开发环境

您如果选择LabVIEW语言作为您的开发工具。首先如上一个小结中我们讲到的方法先建立一个DRPrj的项目并将API文件夹添加到项目当中去。如果您还有保留上一小节中的项目请您直接使用上次的项目。项目建立完成之后，在项目中添加一个新的VI命名为“MotorRun”并添加到DRPrj项目中。如图4.6所示。



图4.6 DRPrj—项目结构

在控制程序中我们将会用到六个新的 API 函数。在第三章中我们可以找到这些 API 函数的原型。

① 进料电机开始运行

Visual C++ &C++ Builder:

```
bool API_FeedMotor_Start(void)
```

LabVIEW:



功能：进料电机开始运行。

参数：空。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_FeedMotor_Start();

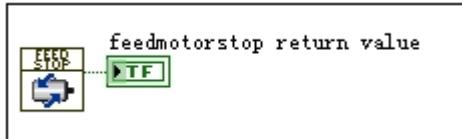
}
```

② 进料电机停止运行

Visual C++ &C++ Builder:

```
bool API_FeedMotor_Stop(void)
```

LabVIEW:



功能：进料电机停止运行。

参数：空。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_FeedMotor_Stop();

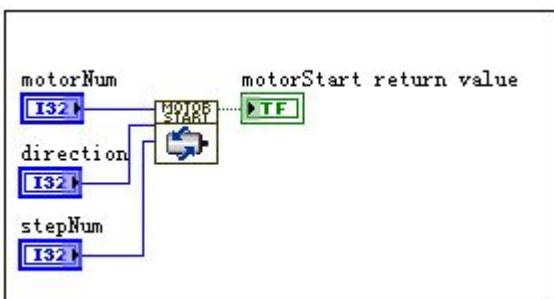
}
```

③ 指定电机开始运行

Visual C++ &C++ Builder:

```
bool API_MotorStart(int motorNum,int direction,int stepNum)
```

LabVIEW:



功能：指定电机开始运行。

参数： motorNum:电机编号；

Direction:电机方向(取值为0,1);

stepNum:电机运行步数。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
```

```

if(!API_SysInit()) { return; }

API_MotorStart(3,1,1000); //3号电机正方向移动1000个步长

}

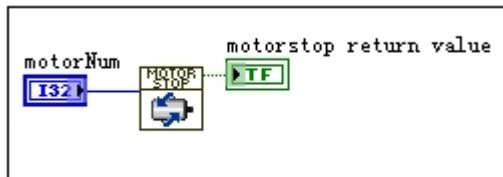
```

④ 指定电机停止运行

Visual C++ &C++ Builder:

```
bool API_MotorStop(int motorNum)
```

LabVIEW:



功能：指定电机停止运行。

参数： motorNum:电机编号。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```

void Func(){

    if(!API_SysInit()) { return; }

    API_MotorStop(3); //3号电机停止运动

}

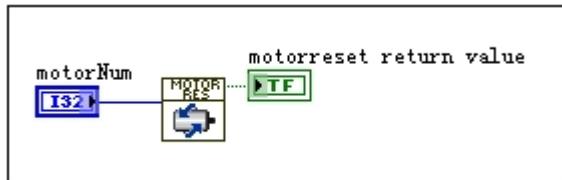
```

⑤ 指定电机复位

Visual C++ &C++ Builder:

```
bool API_MotorReset(int motorNum)
```

LabVIEW:



功能：指定电机复位。3,4,5,6回到复位位置。

参数： motorNum:电机编号。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_MotorReset(3); //3号电机复位
}
```

⑥ 设定电机速度

Visual C++ &C++ Builder:

```
bool API_SetMotorSpeed(int motorNum,int speed)
```

LabVIEW:



功能：设定电机速度。

参数：**motorNum**:电机编号；

Speed:速度值。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    //设定3号电机以1300Hz的PWM脉冲驱动运行
    API_SetMotorSpeed(3,6000000/(2*1300));

}
```

在第一个小节中我们已经成功与控制器通讯，在本次中我们依旧使用事件结构首先搭建，下面是实训步骤。

步骤 1：同第一小节中初始化设备端口的操作，在前面板上面添加“进料电机运行”按钮与“进料电机停止”按钮并摆放整齐。如图 4.7 所示。



图4.7 MotorRun vi—前面板

步骤 2：按住“**Ctrl+E**”切换到程序框图中，将“进料电机运行”按钮与“进料电机停止”按钮对应背面板控件加入 **While** 循环中，并为这两个按钮添加事件分支，分别为“进料电机运行按钮：值改变”与“进料电机停止按钮：值改变”。

步骤 3：打开项目浏览器将“**API Feed Motor START**”子 VI 拖入“进料电机运行按钮：值改变”的分支中，然后将“**API Feed Motor Stop**”子 VI 拖入“进料电机停止按钮：值改变”的分支中。如图 4.8 所示。



图4.8 MotorRun vi—程序面板

完成编写后，点击前面板中电机“运行”按钮开始编译运行程序。如果程序编译运行后没有出现任何错误，点击前面板中的启动按钮计算机开始与控制板开始通讯，通讯成功后前面板的 LED 灯被点亮，分别点击“进料电机运行”按钮与“进料电机停止”按钮观察一号台传送带的运行状况。

完成控制进料电机的程序编写调试后，同样的先点击退出按钮释放设备资源，然后停止 LabVIEW 程序；接下来编写二号台与三号台电机运动的程序。

步骤 1：首先在原有的程序前面板中加入“三号电机驱动”、“四号电机驱动”、“五号电机驱动”与“六号电机驱动”4 个按钮。

步骤 2：向前面板中添加 12 个数字输入框并排放整齐，如图 4.9 所示。



图4.9 MotorRun vi—前面板

步骤 3：按住“Ctrl+E”进入程序框图，分别为“三号电机驱动”、“四号电机驱动”、“五号电机驱动”与“六号电机驱动”4 个按钮添加事件分支。

步骤 4：在事件分支中拖入项目浏览器中 API 文件夹下的“API Motor Start”子 VI 并连接上对应数值输入控件，如图 4.10 所示。



图4.10 MotorRun vi—程序面板

在程序面板中完成了 4 个按钮事件分支的程序添加，同样点击前面板中电机“运行”按钮开始编译运行程序。如果程序编译运行后没有出现任何错误，点击前面板中的启动按钮计算机开始与控制板开始通讯，通讯成功后前面板的 LED 灯被点亮。如图 4.9 在电机编号中填入需要控制的电机编号。二号台的电机编号为 3 号与 4 号，三号台的电机编号为 5 号与 6 号。在运动方向中填入代表方向的数值，使用 0 或者 1 表示具体方向。在运行步数中填入电机运行的步数。在填写完成后，点击“驱动”按钮观察电机的实际运行与程序编写是否一致并记录下来。

完成上面的程序编写后，按照停止程序的步骤安全释放设备资源然后停止程序。接着开始编写电机的停止与复位的控制程序。

步骤 1：在前面板上添加“电机停止”与“电机复位”按钮，以及两个数值输入控件，分别代表要停止的电机号与需要执行复位动作的电机号，前面板布局如下图 4.11 所示。



图4.11 MotorRun vi—前面板

步骤 2：按住“Ctrl+E”组合键进入程序面板中，在事件分支中加入“电机停止按钮：值改变”与“电机复位按钮：值改变”的分支事件。

步骤 3：分别在“电机停止按钮：值改变”与“电机复位按钮：值改变”的分支事件中拖入项目浏览器 API 文件夹下的“API Motor Start”与“API Motor Restart”2个子 VI，并将相对应的电机编号控件与他们的输入参数连接。如图 4.12 所示。

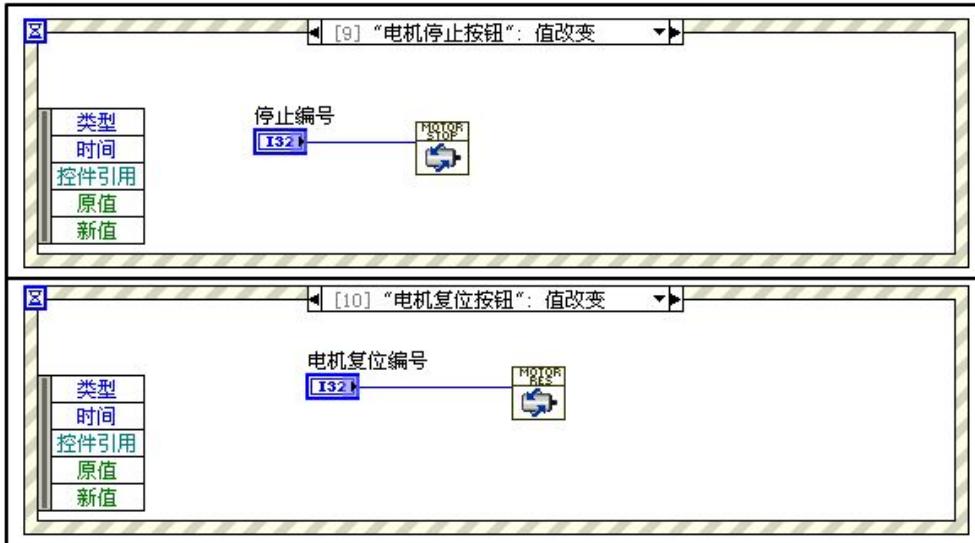


图4.12 MotorRun vi—程序面板

在完成了添加程序后，进入前面板点击 LabVIEW 中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。连接设备然后开始驱动电机，在电机停止的电机编号中填入想要停止运动的电机编号，然后点击“电机停止”按钮观察电机是否停止。接着点击“电机复位”按钮并记录电机复位的方向。如果上述操作都能够正常运行说明程序编写正确并能够顺利进行电机控制。

最后一个 API 的应用是设定电机的速度，设定电机速度这个 API 是可以在电机运行的过程中实时的改变电机速度或者是在电机运行之前改变这个电机的运行速度的功能模块。

步骤 1：在前面板上添加“设定电机速度”按钮，接着添加 2 个数值输入控件，分别用于输入电机编号和电机速度。前面板布局如图 4.13 所示。

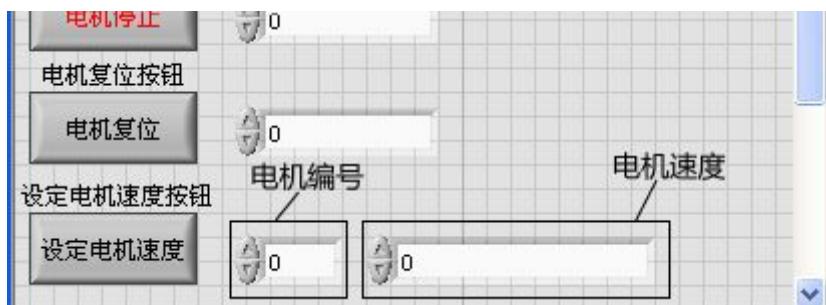


图4.13 MotorRun vi—前面板

步骤 2：点击“Ctrl+E”进入程序面板，在事件分支中添加“设定电机速度：值改变”的事件分支。并在分支中拖入项目浏览器 API 文件夹下“API Set Motor Speed”子 VI。将电机号与电机速度连接上对应的控件。在设定电机速度的时由于处理器时钟分频的关系需要对输入数据进行一定的计算。输入参数 = $6000000 / (2 * \text{电机速度 (Hz)})$ 如图 4.14 所示。

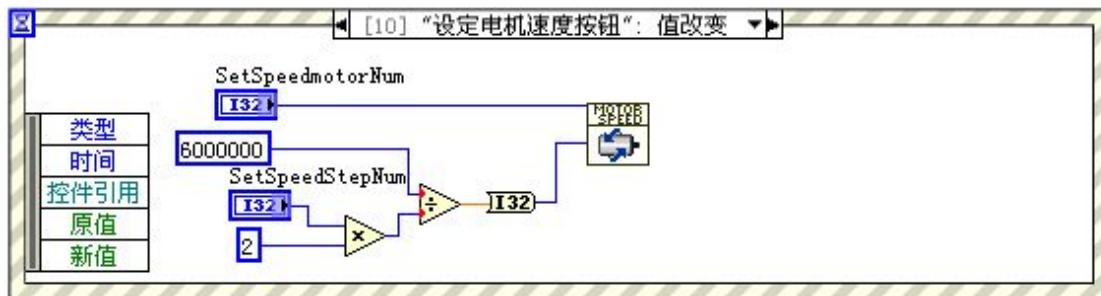


图4.14 MotorRun vi—程序面板

在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。设定不同的电机速度观察电机驱动的变化，并记录可以设定电机速度的范围。其中速度的单位为赫兹(Hz)代表了驱动器发送给步进电机的脉冲速度。

在实际的控制当中，我们需要知道电机的运动状态在。这里详细的介绍一下如何获取电机的运动状态。

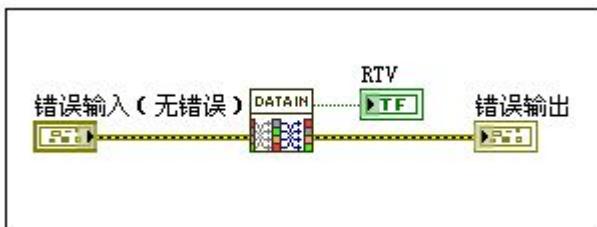
要获取电机运动状态我们会用到如下两个API模块。

① 通知设备完成一次数据采样

Visual C++ &C++ Builder:

```
bool API_RcvInit(void)
```

LabVIEW:



功能：通知设备完成一次数据采样。

参数：空

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){}
```

```

if(!API_SysInit()) { return; }

API_RcvInit(); //通知设备完成一次数据采样

}

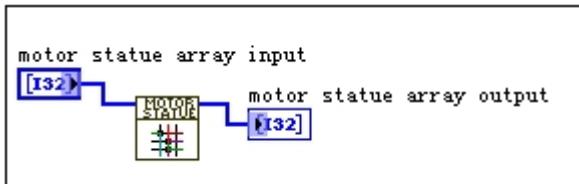
```

② 获取电机状态

Visual C++ &C++ Builder:

```
bool API_getMotorStatue(int* motorStatue)
```

LabVIEW:



功能：获取电机状态。

参数：**motorStatue**：传入类型为int长度为6的数组指针，用于保存电机运行状态数据。

motorStatue数组值为1电机处于运行状态，0则电机处于停止状态。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```

void Func(){

    if(!API_SysInit()) { return; }

    int tmotorStatue[6];

    memset(tmotorStatue,0,sizeof(int)*6);

    API_RcvInit(); //通知设备完成一次数据采样

    API_getMotorStatue(tmotorStatue);

}

```

从上面两个API模块可以看到如果想要获取电机的运行状态需要先调用通知设备完成一次数据采样的API，首先通知设备完成一个周期的数据采样。然后调用获取电机状态的API函数完成对电机状态的获取。下面是具体实现这个过程的方法和步骤。

步骤1：打开“MotorRun.vi”的前面板，然后在前面板中添加“获取电机运行状态按钮”。步

骤2：在程序面板中为它添加“获取电机运行状态按钮：值改变”的事件分支。

步骤3：“获取电机运行状态按钮：值改变”事件分支中我们放入两个API模块，分别是“API Rec Data Init”与“API get motor statue”。首先使用“API Rec Data Init”模块并获取它的返回值如果

返回为“真”那么在这个真的条件分支中去执行“API get motor statue”模块并将这个模块返回的数组显示到前面板上。需要注意的是在使用这个模块的时候我们需要提供一个初始化好的数组作为输入参数，我们知道电机状态是一个一维6个元素的数组所以在这里我们初始化一个维度为1元素个数为6的数组作为输入参数。详细程序框图如图4.15所示。

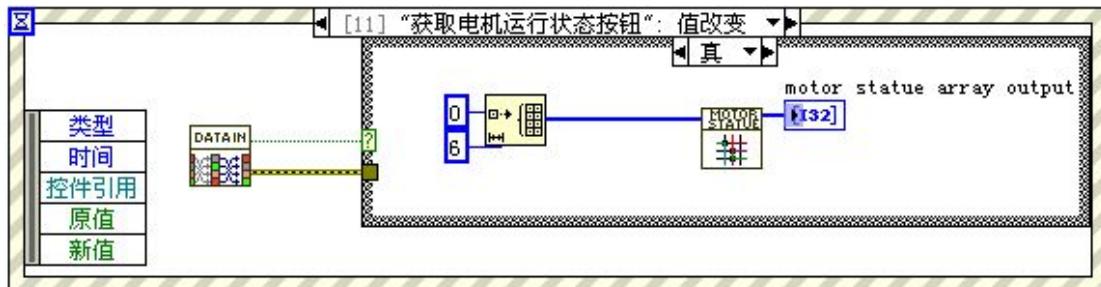


图4.15 MotorRun vi—程序面板

最后，在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。给定一个电机运动并点击获取电机运动状态观察数组里面的数据变化。在程序当中如果电机运动则数组对应元素的返回值为1没有运动则数组对应的返回值为0。

二、电机运动方向示意图

如图4.x与4.x所示分别为二号台与三号台电机运动方向示意图，如果您在使用过程中没有修改我们的接线，您可以参照图中所示的电机编号与方向编写您的控制程序。

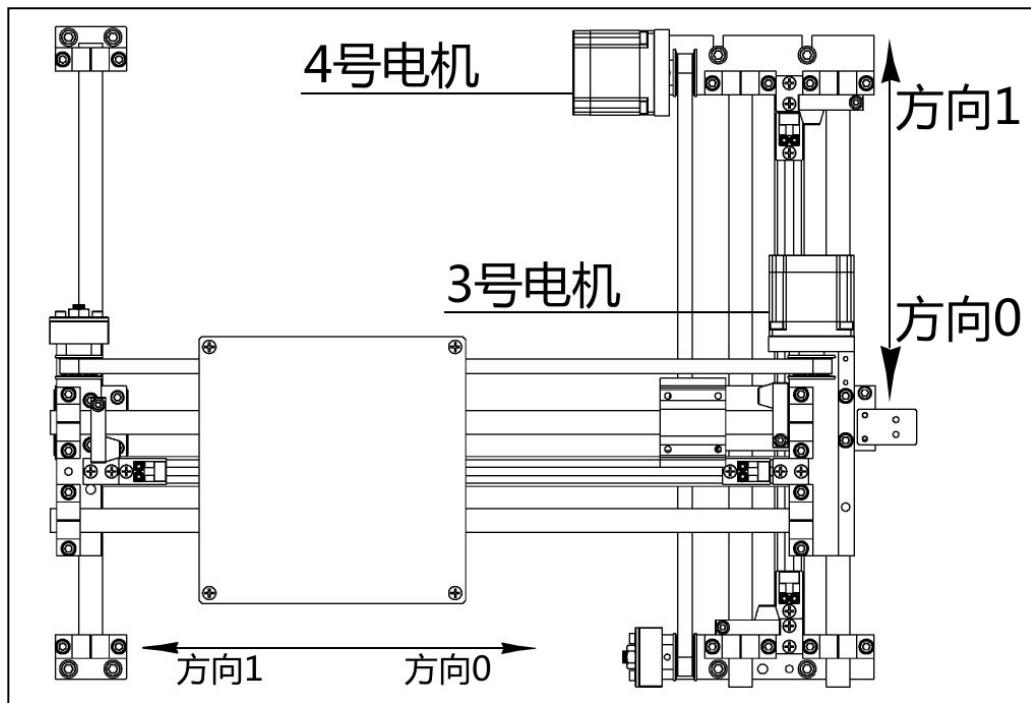


图4.16 二号台—机构运动反向示意图

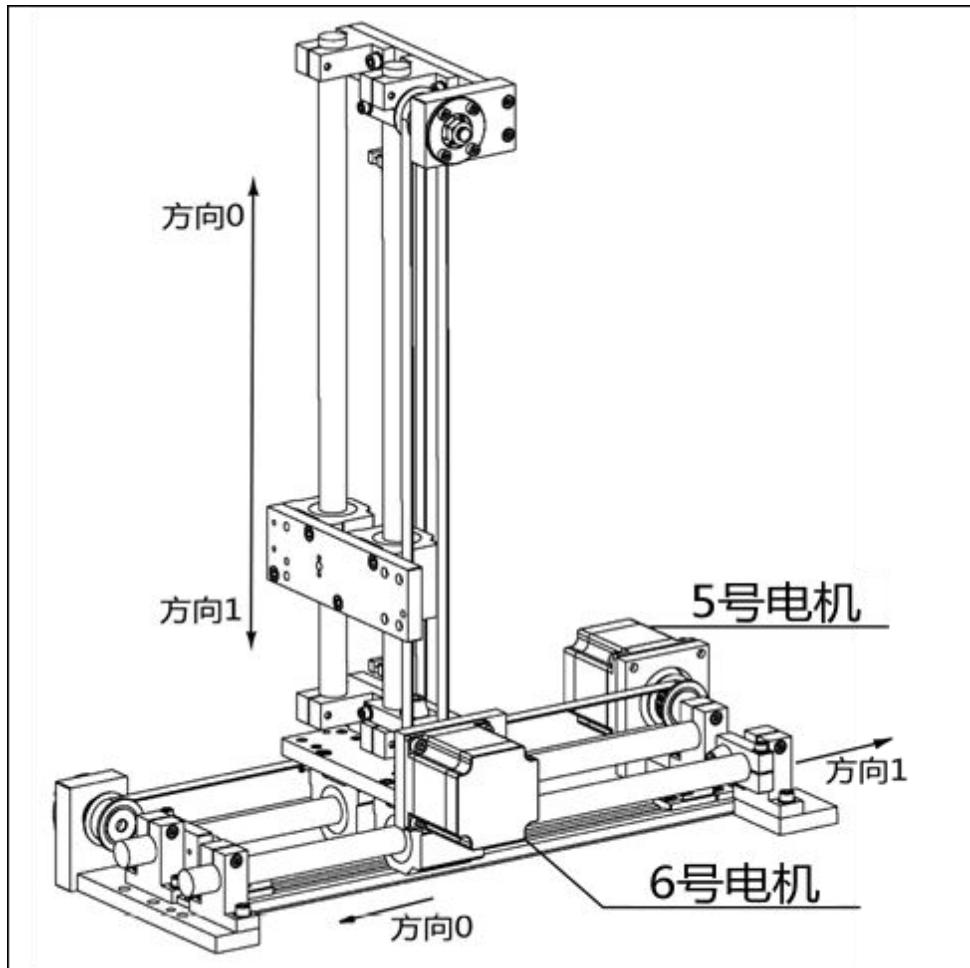


图4.17 三号台—机构运动反向示意图

三、实训习题

1. 通过上面的讲解自己独立完成电机的运动控制程序。
2. 通过设置电机速度驱动电机在二号台上完成电机的两轴联动控制程序。

4.3 IO 口控制

控制系统所有气动机构都是由 IO 口控制继电器的开关从而达到控制气缸的目的。我们知道气缸起着关键的作用，系统中所有平台之间的物料运输都是由气缸完成其工作。在上面两节中已经详细介绍了控制器的通讯和电机的驱动，这个小节我们将为您详细介绍如何使用 IO 口控制气缸的运动。

一、LabVIEW开发环境

您如果选择 LabVIEW 语言作为您的开发工具。首先如第一小结中我们讲到的方法首先建立一个 DRPrj 的项目并将 API 文件夹添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用

上次的项目。项目建立完成之后，在项目中添加一个新的 VI 命名为“IOControl”并添加到 DRPrj 项目中，如图 4.18 所示。

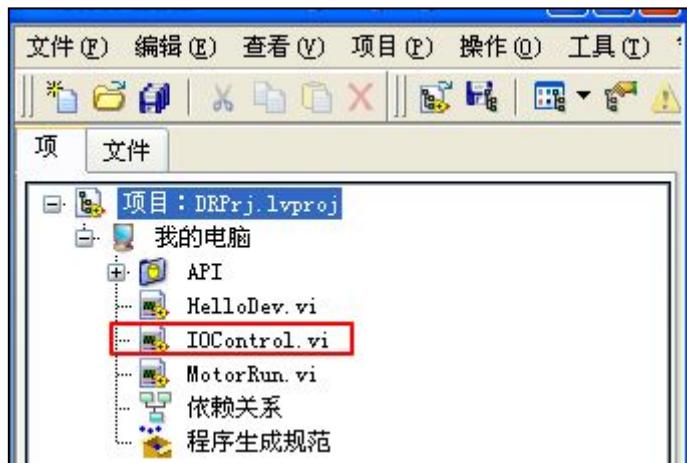


图4.18 DRPrj—项目结构

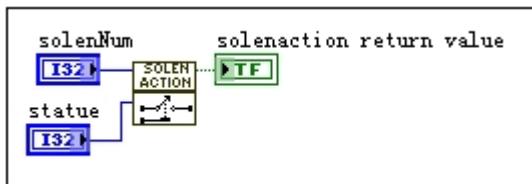
在这次的控制程序中会用到两个新的 API 函数。在第三章中我们可以找到这个 API 函数的原型。

① 指定继电器动作

Visual C++ &C++ Builder:

```
bool API_solenAction(int solenNum,int statue)
```

LabVIEW:



功能：指定继电器动作。

参数：**solenNum**:继电器编号；

statue:状态值(0:关;1:开)。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_solenAction(0,1); //打开1号继电器

}
```

注：在程序中继电器编号为0-47对应实际IO口IO1-IO48。

- ② 打开或关闭指定平台上的所有继电器

Visual C++ &C++ Builder:

```
bool API_solenAdvace(int platformNum,int statue)
```

LabVIEW:



功能：打开或关闭指定平台上的所有继电器。

参数： platformNum:平台编号；

statue:状态值(0:关;1:开)。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_solenAdvace(0,1); //打开1号平台所有继电器

}
```

注：在程序中平台编号 0-2 对应实际的平台编号 1-3。

在这次的实训中我们的程序结构依旧以事件结构作为最基本的程序框架。

步骤 1：将第一小节中的“启动”按钮、“退出”按钮以及 LED 灯等控件连同他们的程序框图复制到我们的程序中。

步骤 2：在程序前面板中添加这次用到的“单个继电器动作”按钮、“平台继电器动作”按钮。

步骤 3：在对应的按钮旁边分别添加 2 个输入参数的控件（数值输入控件），前面板布局如下图

4.所示。



图4.19 IOControl vi—前面板

步骤 4：按住“Ctrl+E”进入程序面板中，为两个按钮添加对应的两个事件分支“单个继电器动作按钮：值改变”与“平台继电器动作：值改变”。

步骤 5：在“单个继电器动作：值改变”的分支中加入项目浏览器 API 文件夹下的“API Solen Action”子 VI 并连接对应的参数输入控件而在“平台继电器动作：值改变”的分支中加入项目浏览器文件夹下的“API Solen Advance”子 VI 并连接对应的前面板输入控件，两个分支的程序框图如图 4.20 所示。

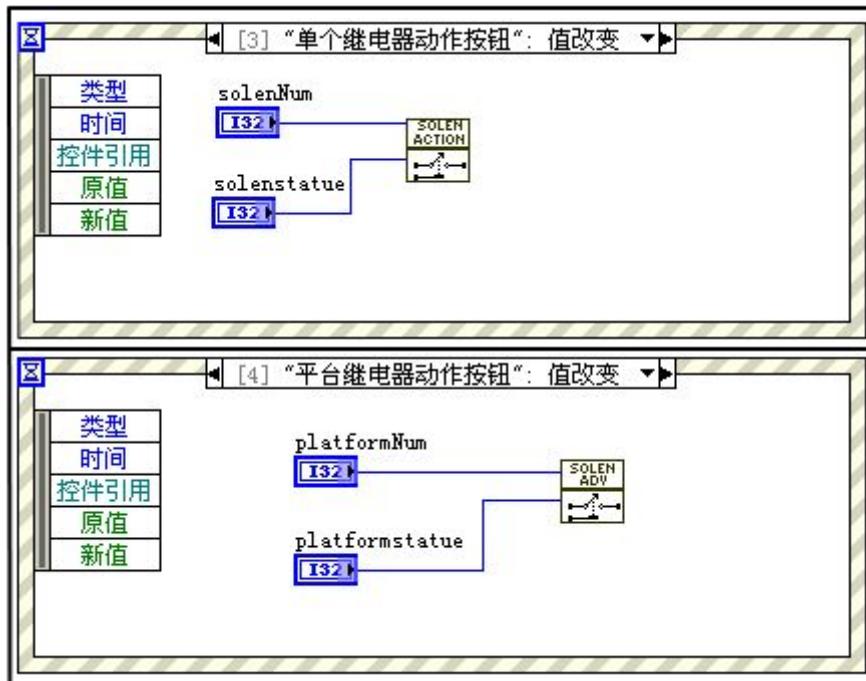


图4.20 IOControl vi—程序面板

在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误，在参数输入框中填入继电器编号以及继电器状态点击按钮观察不同的继电器状态气缸的动作变化并记录下来。

在实际的控制当中，有时还会需要知道IO口的当前状态。在这里我们将向您详细的介绍一下如何获取IO口的当前状态。

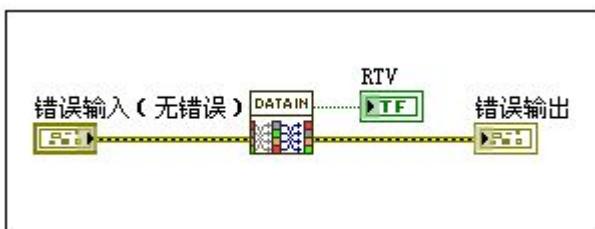
获取电机运动状态我们会用到如下两个API模块。

- ① 通知设备完成一次数据采样

Visual C++ &C++ Builder:

```
bool API_RcvInit(void)
```

LabVIEW:



功能：通知设备完成一次数据采样。

参数：空

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    API_RcvInit();           //通知设备完成一次数据采样

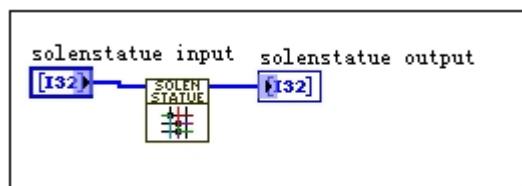
}
```

- ② 获取继电器状态

Visual C++ &C++ Builder:

```
void API_getSolenStatue(int* SolenStatue)
```

LabVIEW:



功能：获取继电器状态。

参数：**SolenStatue**：传入类型为int长度为48的数组指针，用于保存继电器状态数据。

SolenStatue数组值为1继电器处于打开状态，0则继电器处于关闭状态。

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    int tSolenStatue[48];
    memset(tSolenStatue,0,sizeof(int)*48); //初始化数组
    API_RcvInit(); //通知设备完成一次数据采样
    API_getSolenStatue(tSolenStatue); //打开1号平台所有继电器
}
```

从上面两个API模块我们可以看到如果想要获取继电器IO状态我们需要先调用“通知设备完成一次数据采样”的API模块完成一个周期的数据采样，然后调用“获取继电器状态”的API模块完成对电机状态的获取。下面是具体实现这个过程的方法和步骤。

步骤1：打开“IOControl.vi”的前面板，然后在前面板中添加“获取电机运行状态按钮”。并在对应在程序面板中为它添加“获取继电器状态按钮：值改变”的事件分支。

步骤2：在“获取继电器状态按钮：值改变”的事件分支中我们放入两个API模块，分别是“API Rec Data Init”与“API get solen statue”。首先使用“API Rec Data Init”模块并获取它的返回值如果返回为“真”那么在这个真的条件分支中去执行“API get solen statue”模块并将这个模块返回的数组显示到前面板上。需要注意的是在使用这个模块的时候我们需要提供一个初始化好的数组作为输入参数，我们知道IO状态是一个一维48个元素的数组所以在这里我们初始化一个维度为1元素个数为48的数组作为输入参数。详细程序框图如图4.21所示。

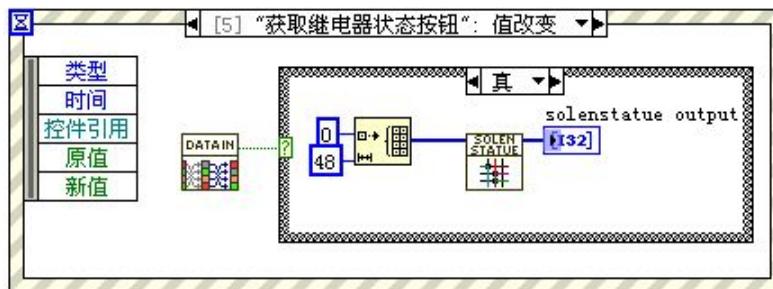


图4.21 IOControl vi—程序面板

最后，在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如

果程序编译没有任何的错误。打开几个继电器并点击获取继电器状态观察数组里面的数据变化。在程序当中如果继电器处于打开状态则返回对应数组的元素为1关闭状态则为零。

二、气动机构运动以及编号示意图

如下列组图所示，分别为一号台、二号台与三号台所有的气动器件运动示意图，如果您在使用过程中没有修改我们的IO口接线方法，您可以参照图中所示的气动器件编号与运动方式编写您的控制程序。

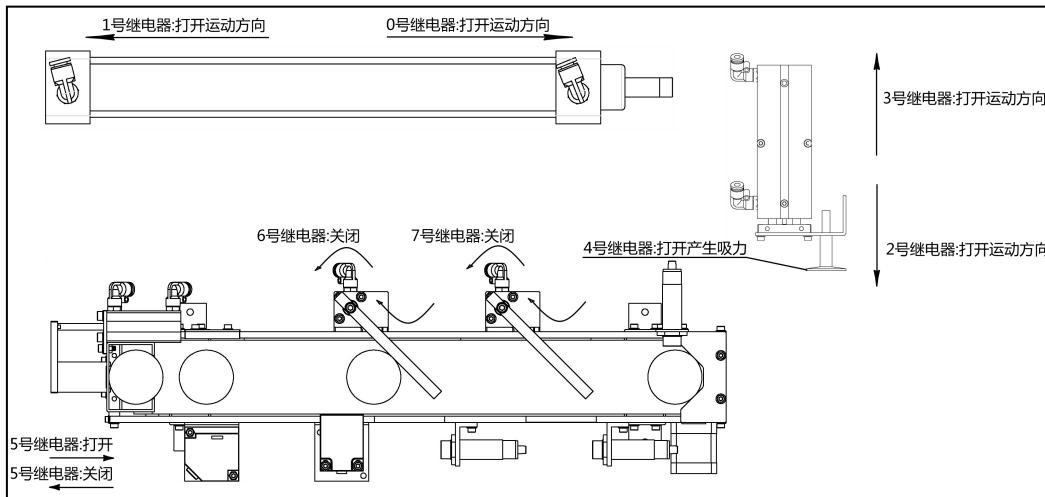


图4.22 一号台一气缸编号示意图

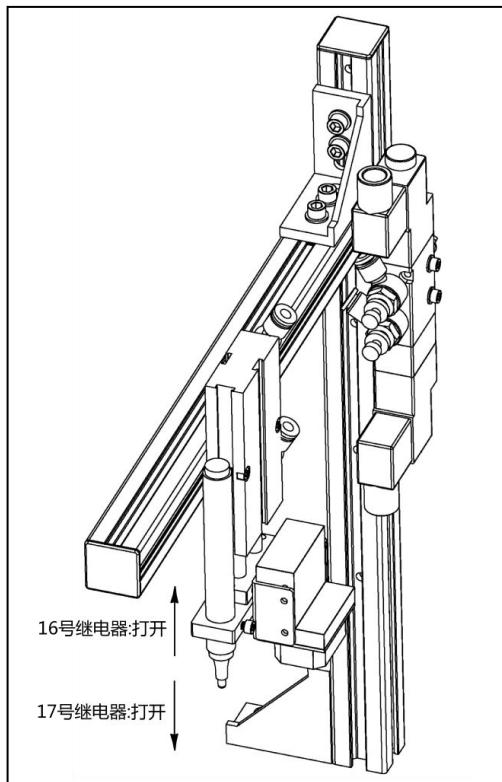


图4.23 二号台一气缸编号示意图

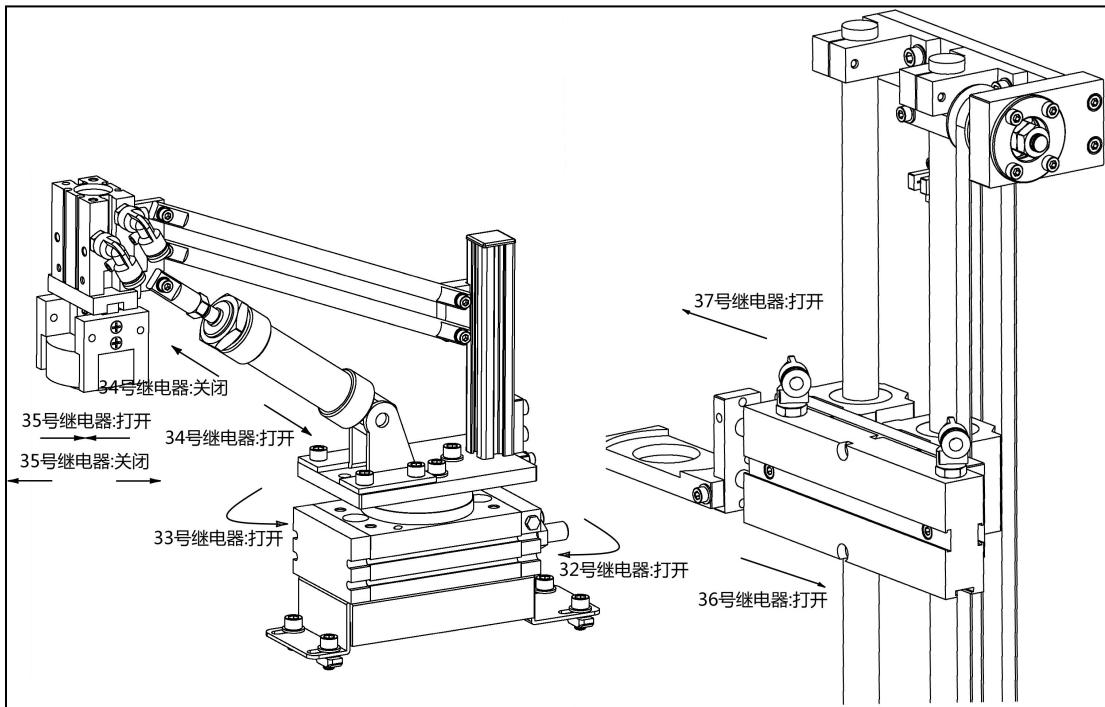


图4.24 三号台一气缸编号示意图

三、实训习题

1. 依据小节中讲解的步骤独立完成 IO 控制程序，并记录每个 IO 控制的气缸编号。
2. 编写控制程序完成一号台气缸运送物料工作。
3. 编写控制程序完成三号台气缸抓爪抓取物料工作。

4.4 获取平台输入口状态

在上面几节中我们已经对系统中运动部件的编程进行了初步的了解，在本节中我们将接着介绍系统中输入信号的获取。在系统中所有的电机的光电限位开关输出信号与气缸到达位置后磁环的输出信号全部接到每个平台的输入端口中。在下面将为您详细的介绍如何在系统中如何获取这些信号。

一、LabVIEW开发环境

您如果选择 LabVIEW 语言作为您的开发工具。首先如第一小结中我们讲到的方法首先建立一个 DRPrj 的项目并将 API 文件夹添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，在项目中添加一个新的 VI 命名为“GetInputStatue”并添加到 DRPrj 项目中。如图 4.25 所示。

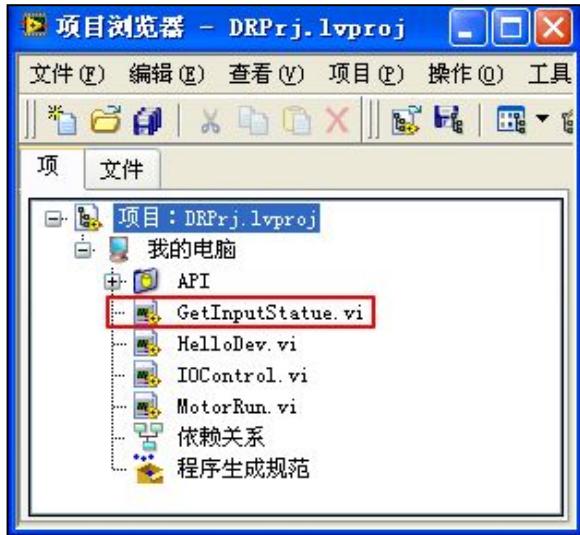


图4.25 DRPrj—项目结构

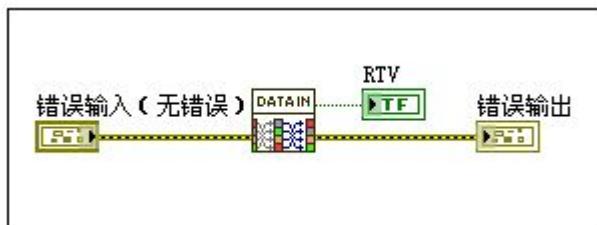
完成获取输入口状态程序需要使用下面 2 个主要的 API 模块。

- ① 通知设备完成一次数据采样

Visual C++ &C++ Builder:

```
bool API_RcvInit(void)
```

LabVIEW:



功能：通知设备完成一次数据采样。

参数：空

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

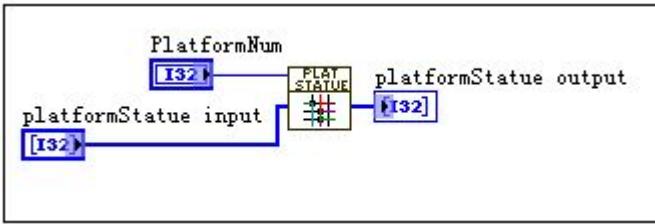
```
void Func(){
    if(!API_SysInit()) { return; }
    API_RcvInit(); //通知设备完成一次数据采样
}
```

- ② 获取平台IO输入口状态

Visual C++ &C++ Builder:

```
bool API_getPlatformStatue(int PlatformNum,int platformStatue[8])
```

LabVIEW:



功能：获取平台IO输入口状态。

参数：**PlatformNum**: 平台号；

platformStatue: 传入类型为int长度为8的数组指针，用于保存平 台 IO 输入口状态。

platformStatue数组值为1输入口为高电平，数值为0输入口为低电平。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```

void Func(){
    if(!API_SysInit()) { return; }

    int tplatformStatue[8];
    memset(tplatformStatue,0,sizeof(int)*8);

    API_RcvInit(); //通知设备完成一次数据采样
    API_getPlatformStatue(0,tplatformStatue);

}
  
```

注：在程序中平台编号 0-2 对应实际的平台编号 1-3。

在程序中获取平台IO输入口状态通过前几节的实训联系，知道首先需调用“通知设备完成一次数据采样” API模块接着调用“获取平台IO输入口状态” API模块完成程序流程，整体的程序框架沿用前几节使用的事件结构完成搭建，具体的步骤如下所示。

步骤1：打开建立的“**GetInputStatue.vi**”，然后在程序中加入第一节中与控制器通讯的模块。

步骤2：在前面板中加入“获取输入口状态”按钮、“数值输入控件”与“数组显示控件”。“数值输入控件”用于填入获取输入口状态的平台号，而“数组显示控件”用于显示输入口的状态数组。如下图4.26所示。



图4.26 GetInputStatue vi—前面板

步骤3：按住“Ctrl+E”进入程序面板中，首先添加事件分支“获取输入口状态按钮：值改变”。接着在“获取输入口状态按钮：值改变”分支中拖入项目浏览器API文件夹下的“API Rec Data Init”模块与“API get platform statue”模块。

步骤4：连接“API Rec Data Init”模块，由于“API get platform statue”模块具有一个返回值当“API Rec Data Init”模块执行成功后会返回true，在这里我们使用条件分支结构在“API Rec Data Init”模块执行成功后接着执行“API get platform statue”模块。“API get platform statue”模块需要连接两个输入控件，一个输入连接前面板上面代表平台号的数值输入控件而另外一个则需要连接一个初始化过的一维数组。我们知道这里是获取平台上8个输入口状态，所以我们在这里初始化一个一维数组里面含有8个元素。“API get platform statue”模块的输出口与显示控件连接。程序面板连接后如图4.27所示。

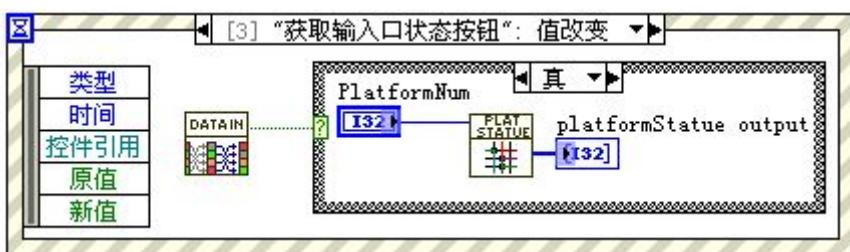


图4.27 GetInputStatue vi—程序面板

步骤5：在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。首先点击启动连接设备，在平台号输入控件中输入要查询的平台号。在机电平台上面用不透明的纸张遮住其中一个光电开关，然后再LabVIEW中点击获取输入口状态观察数组的改变并记录下来。

二、平台输入口编号示意图

如下列组图所示，分别为一号台、二号台与三号台平台上的输入IO示意图，如果您在使用过程中没有修改输入IO口的接线方法，您可以参照图中所示的输入IO的编号来编写验证您的控制程序。

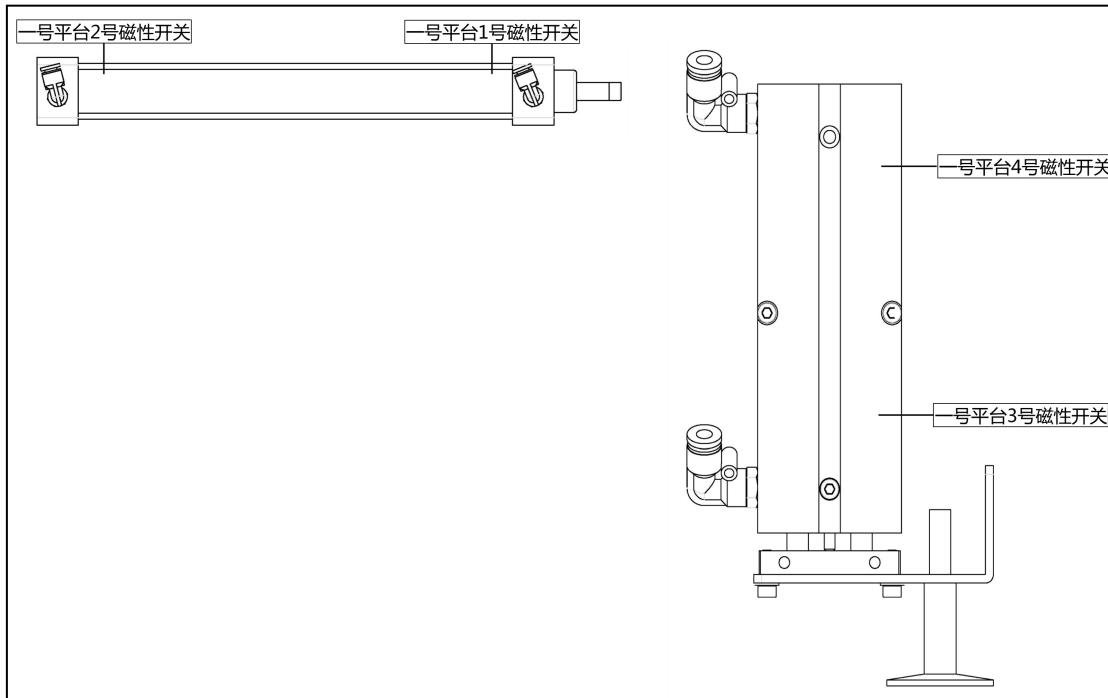


图4.28 一号台—输入口编号

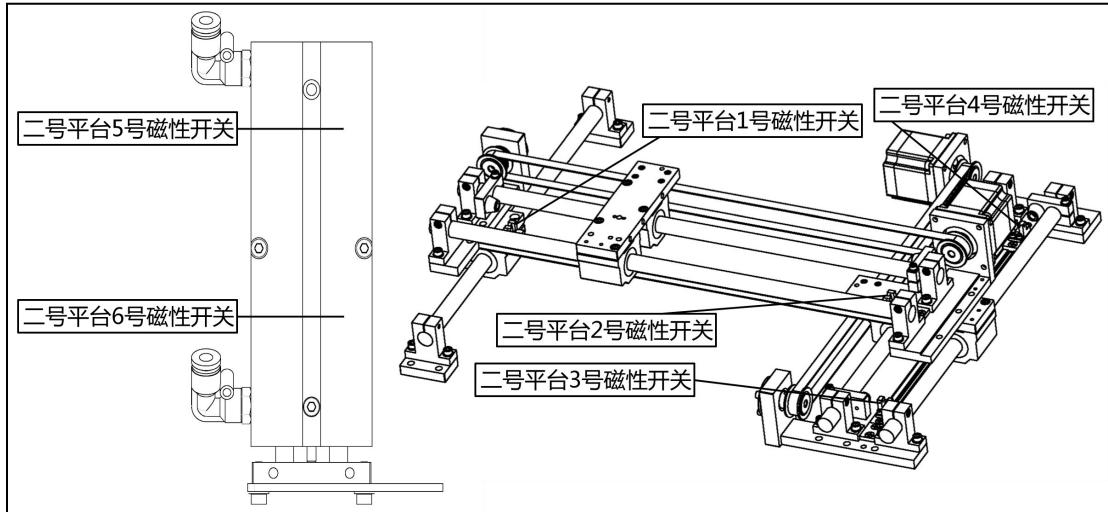


图4.29 二号台—输入口编号

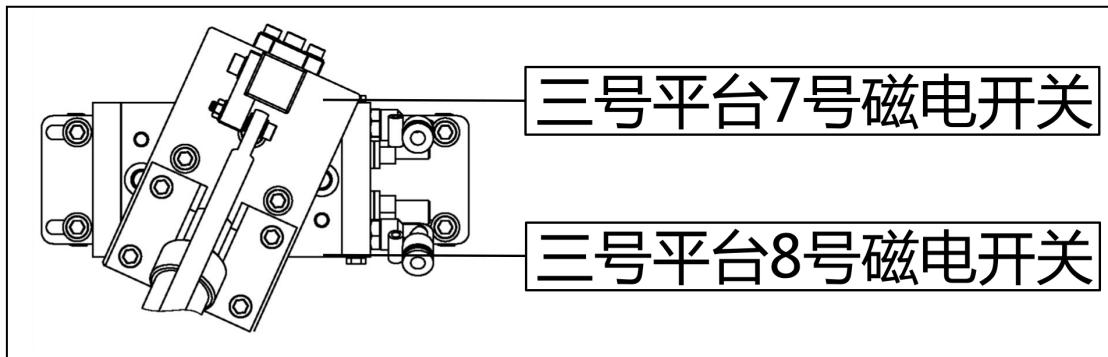


图4.30 三号台—输入口编号1

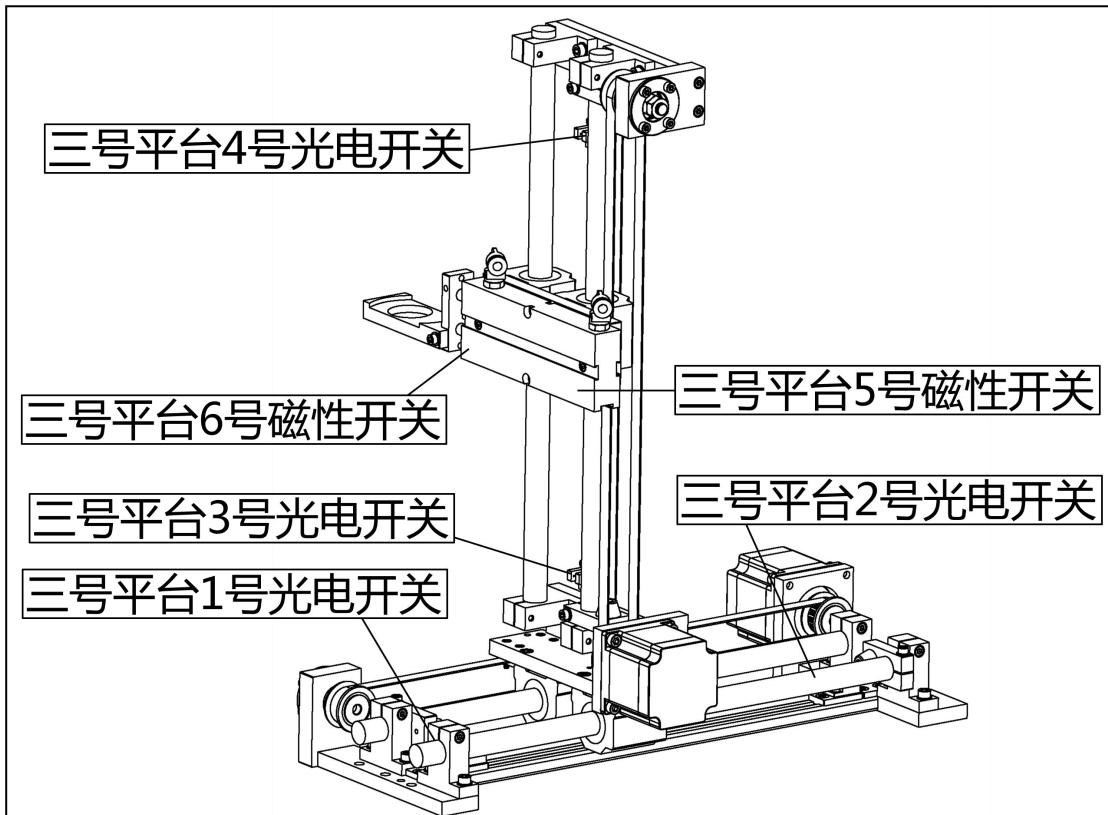


图4.31 三号台—输入口编号2

三、实训习题

依据小节中讲解的步骤独立完成查询输入口状态程序并记录每个输入口的状态量。

4.5 获取传感器电平

在本节中我们将接着介绍系统中传感器电平状态的获取。在系统中所有的传感器都连接到特定的 IO 口上。在下面将为您详细的介绍如何在系统中如何获取这些信号。

一、LabVIEW开发环境

您如果选择 LabVIEW 语言作为您的开发工具。首先如第一小结中我们讲到的方法首先建立一个 DRPrj 的项目并将 API 文件夹添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，在项目中添加一个新的 VI 命名为“GetSensorStatue”并添加到 DRPrj 项目中。如图 4.32 所示。

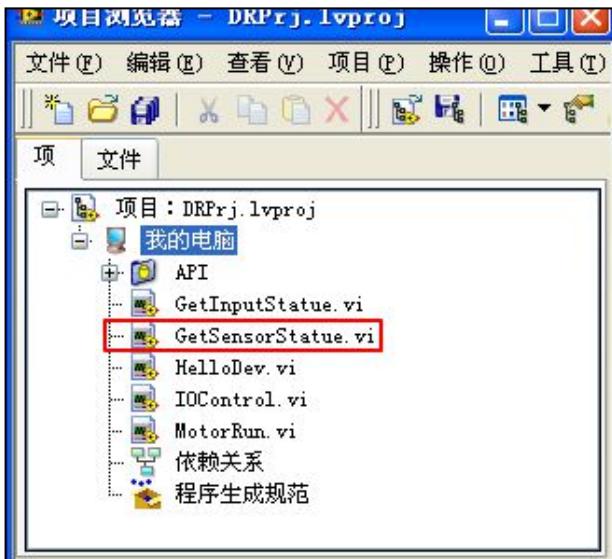


图4.32 DRPrj—项目结构

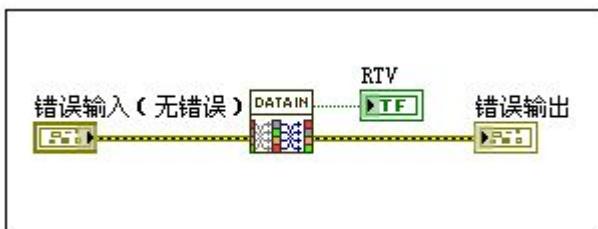
要完成获取输入口状态需要使用下面 2 个主要的 API 模块。

① 通知设备完成一次数据采样

Visual C++ &C++ Builder:

bool API_RcvInit(void)

LabVIEW:



功能：通知设备完成一次数据采样。

参数：空

返回值：布尔类型，如果设备指令发送成功返回为“真”。

例子：

```
void Func(){
```

```
    if(!API_SysInit()) { return; }
```

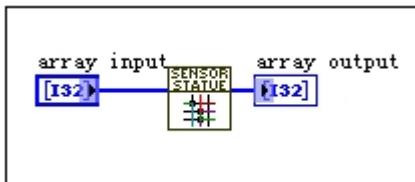
```
API_RcvInit(); //通知设备完成一次数据采样
}
```

② 获取传感器状态

Visual C++ &C++ Builder:

```
bool API_getSensorStatue(int SensorStatue[8])
```

LabVIEW:



功能： 获取传感器状态。

参数： **SensorStatue**: 传入类型为int长度为8的数组指针，用于保存平台 传感 器 状态 。

SensorStatue数组值为1接收到传感器的电平为高电平，数 值为0传感器的电平为低电平。

返回值：布尔类型，如果设备指令发送成功返回为 “真”。

例子：

```
void Func(){
    if(!API_SysInit()) { return; }

    int tSensorStatue[8];
    memset(tSensorStatue,0,sizeof(int)*8);

    API_RcvInit(); //通知设备完成一次数据采样

    API_getSensorStatue(tSensorStatue);

}
```

在程序中去获取平台IO输入口状态我们通过前几节的学习，我们知道需要先调用“通知设备完成一次数据采样” API模块接着调用“获取传感器状态” API模块完成程序流程，整体的程序框架沿用前几节使用的事件结构完成搭建。具体的步骤如下所示。

步骤1：打开建立的“**GetSensorStatue.vi**”，然后在程序中加入第一节中与控制器通讯的模块。

步骤2：在前面板中加入“获取传感器状态”按钮与“数组显示控件”。数组显示控件用于显示传感器的状态数组，如下图4.33所示。



图4.33 GetSensorStatue vi—前面板

步骤3：按住“Ctrl+E”进入程序面板中，首先添加事件分支“获取传感器状态按钮：值改变”。接着在“获取传感器状态按钮：值改变”分支中拖入项目浏览器API文件夹下的“API Rec Data Init”模块与“API get sensor status”模块。

步骤4：连接“API Rec Data Init”模块，由于“API Rec Data Init”模块具有一个返回值当“API Rec Data Init”模块执行成功后会返回true，在这里我们使用条件分支结构在“API Rec Data Init”模块执行成功后接着执行“API get sensor status”模块。“API get sensor status”模块需要连接一个输入控件，这个输入端需要连接一个初始化过的一维数组。我们知道这里是获取平台上8个传感器输入口的状态，所以我们在那里初始化一个一维数组里面含有8个元素。“API get platform statue”模块的输出口与显示控件连接。程序面板连接后如图4.34所示。

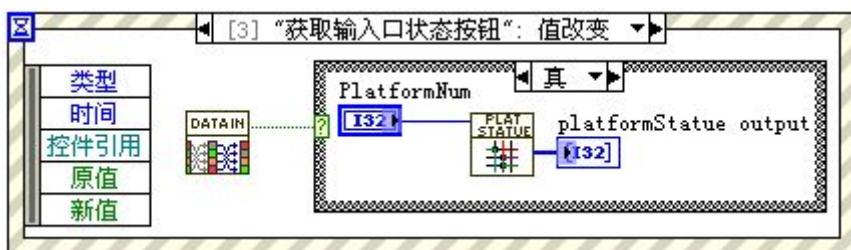


图4.34 GetSensorStatue vi—程序面板

在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。首先点击启动连接设备，根据传感器的原理给予传感器信号(例如：遮挡红外漫射传感器，在电涡流传感器前面放置磁性物体等)，然后再LabVIEW中点击获取传感器状态按钮观察数组的改变并记录下来。

二、传感器输入口编号示意图

如下图4.35所示，为传感器输入口接线，如果您在使用过程中没有修改传感器输入口的连线方

法，您可以参照图中所示的传感器输入口的编号来编写验证您的控制程序。

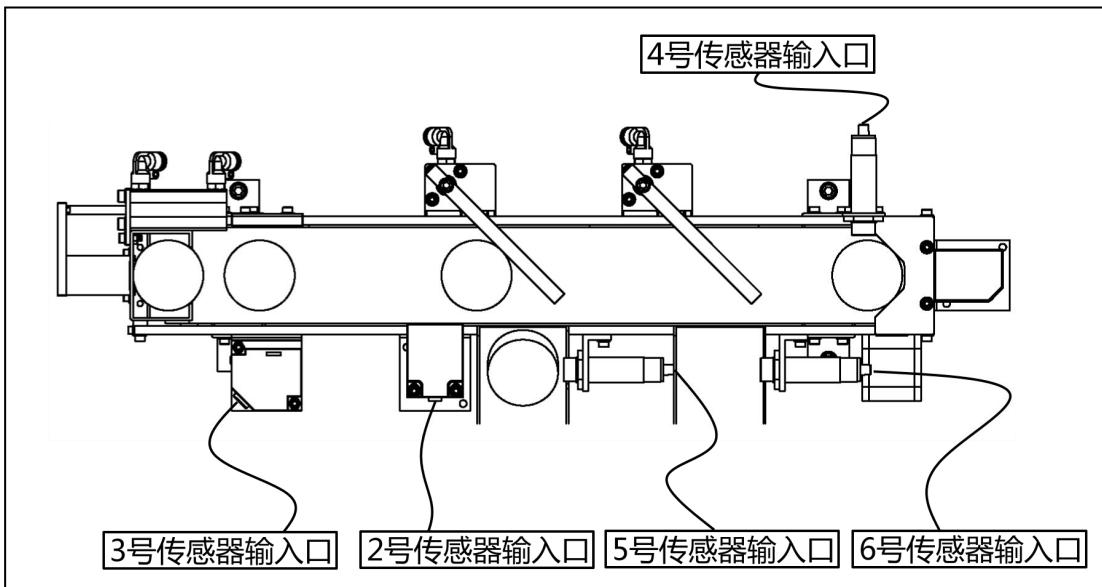


图4.35 传感器输入口—编号示意图

三、实训习题

依据小节中讲解的步骤独立完成查询传感器输入口电平状态程序并记录每个传感器在检测物体时反应的不同电平变化。

4.6 颜色传感器应用

在本节中我们将接着介绍如何使用颜色传感器采集颜色数据，本套系统中颜色传感器是唯一的一个直接挂载在 USB 总线上的传感器，其通过微控制器控制前端摄像头采集颜色数据。在本小节中您将知道如何采集这些数据到 PC 上位机系统中。

一、LabVIEW开发环境

您如果选择 LabVIEW 语言作为您的开发工具。首先如第一小结中我们讲到的方法首先建立一个 DRPrj 的项目并将 API 文件夹 添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，在项目中添加一个新的 VI 命名为“GetColorDis”并添加到 DRPrj 项目中。如图 4.36 所示。

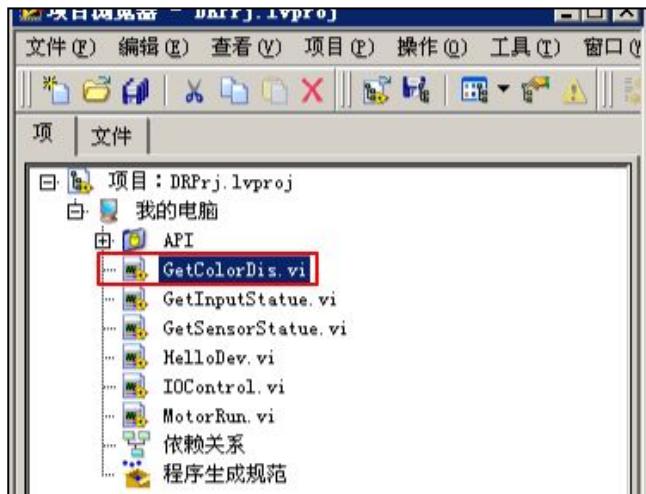


图4.36 DRPrj—项目结构

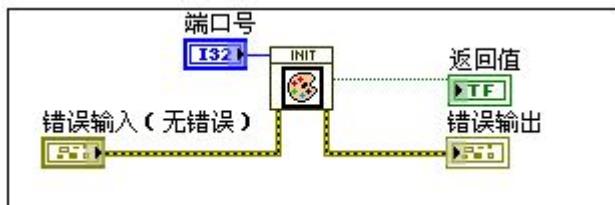
在这次的实训中，要完成获取颜色传感器数据需要使用下面的 API 模块：

- ① 打开颜色采集设备端口

Visual C++ &C++ Builder:

```
bool APIColor_Init(int ComPort)
```

LabVIEW:



功能：打开颜色采集设备端口。

参数： **ComPort**: 传入类型为int的物理端口号。

返回值：布尔类型，如果打开设备成功返回 真。

例子：

```
void Func(){
    if(!APIColor_Init(10)) { return; }
}
```

- ② 关闭颜色采集设备端口

Visual C++ &C++ Builder:

```
void APIColor_Quit(void)
```

LabVIEW:



功能：关闭颜色采集设备端口。

参数：空

返回值：空

例子：

```

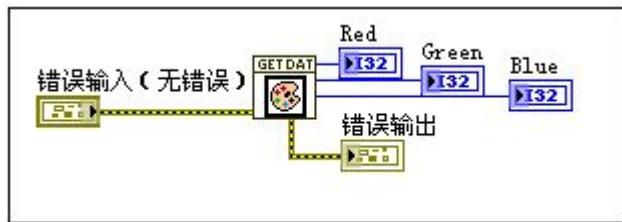
void Func(){
    APIColor_Quit();
}
  
```

③ 获取颜色采集设备当前采集到的RGB数值

Visual C++ &C++ Builder:

```
void APIColor_GetColor(int& Red,int& Green,int& Blue)
```

LabVIEW:



功能：获取颜色采集设备当前采集到的RGB数值。

参数：Red, Green, Blue：传入int类型变量的引用。

返回值：空

例子：

```

void Func(){
    Int tRed,tGreen,tBlue;
    tRed = 0;tGreen = 0;tBlue = 0;
    APIColor_GetColor(tRed,tGreen,tBlue);
}
  
```

注：在使用初始化模块的时候我们在程序中同样需要注意在整个程序运行完成后需要对句柄资源进行释放，如果没有释放句柄资源会导致下一次调用初始化模块失败。同样如果发送这种情况请您完全关闭LabVIEW进程，重新运行LabVIEW进行编程工作。

本次实训实验使用事件结构作为程序的整体框架，下面是本次实训的步骤：

步骤1：打开新建的“GetColorDis”程序进入前面板，先在前面板上添加“初始化”按钮、“退出”按钮以及“获取RGB”按钮。

步骤2：在前面板上添加3个数值显示控件分别显示通过颜色传感器获取的R值、G值与B值。

步骤3：在前面板上添加1个数值输入控件用于输入端口号，然后添加一个LED灯用于显示颜色传感器的初始化状态。如图4.37所示。



图4.37 GetColorDis vi—前面板

步骤4：按住“Ctrl+E”进入程序面板中，添加事件框架。分别为3个按钮添加对应的分支，“初始化按钮：值改变”、“退出按钮：值改变”以及“获取RGB按钮值改变”。

步骤5：进入“初始化按钮：值改变”的分支中拖入项目浏览器API文件夹下的“ColorAPI_Init”模块，在输入端上连接端口号控件，输出端返回值连接LED灯控件。

步骤6：进入“退出按钮：值改变”的分支中拖入项目浏览器API文件夹下的“ColorAPI_Quit”模块。

步骤7：进入“获取RGB按钮：值改变”的分支中拖入项目浏览器API文件夹下的“ColorAPI_GetData”模块，在输出端口中连接对应的显示控件。如图4.38所示。

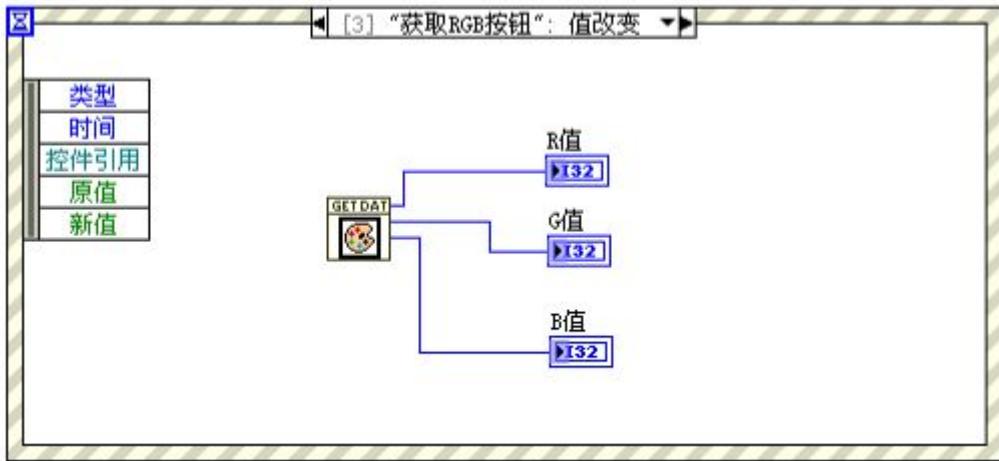


图4.38 GetColorDis vi—程序面板

在完成了添加程序后，进入前面板点击LabVIEW中的“运行”按钮编译并运行程序。如果程序编译没有任何的错误。点击初始化按钮连接设备，在颜色传感器下方放入不同的颜色的物块电机“获取RGB按钮”记录其在系统中反馈的RGB值。

二、实训习题

1. 依据上面的实训步骤自己搭建颜色采集程序。
2. 在程序中添加“颜色盒”控件，将颜色采集出的 RGB 值转化显示在颜色盒中。
3. 更改上面的程序结构实现颜色采集的实时显示。

4.7 一号平台控制编程

在前面的6个小节中我们具体的讲述了如何使用开发工具，操作系统中的每个部件以及获取系统中的信号，那么在下面的小节中我们将具体的讲述如何获取这些信号进行控制系统运行。在本小节中控制对象为一号平台，使用一号平台完成特定的送料与运输工作。

一、LabVIEW程序控制流程

控制流程图如图4.39所示，在流程图中可以看到控制一号台运动的过程中我们需要结合前面几个小节所认识到的API进行组合应用。

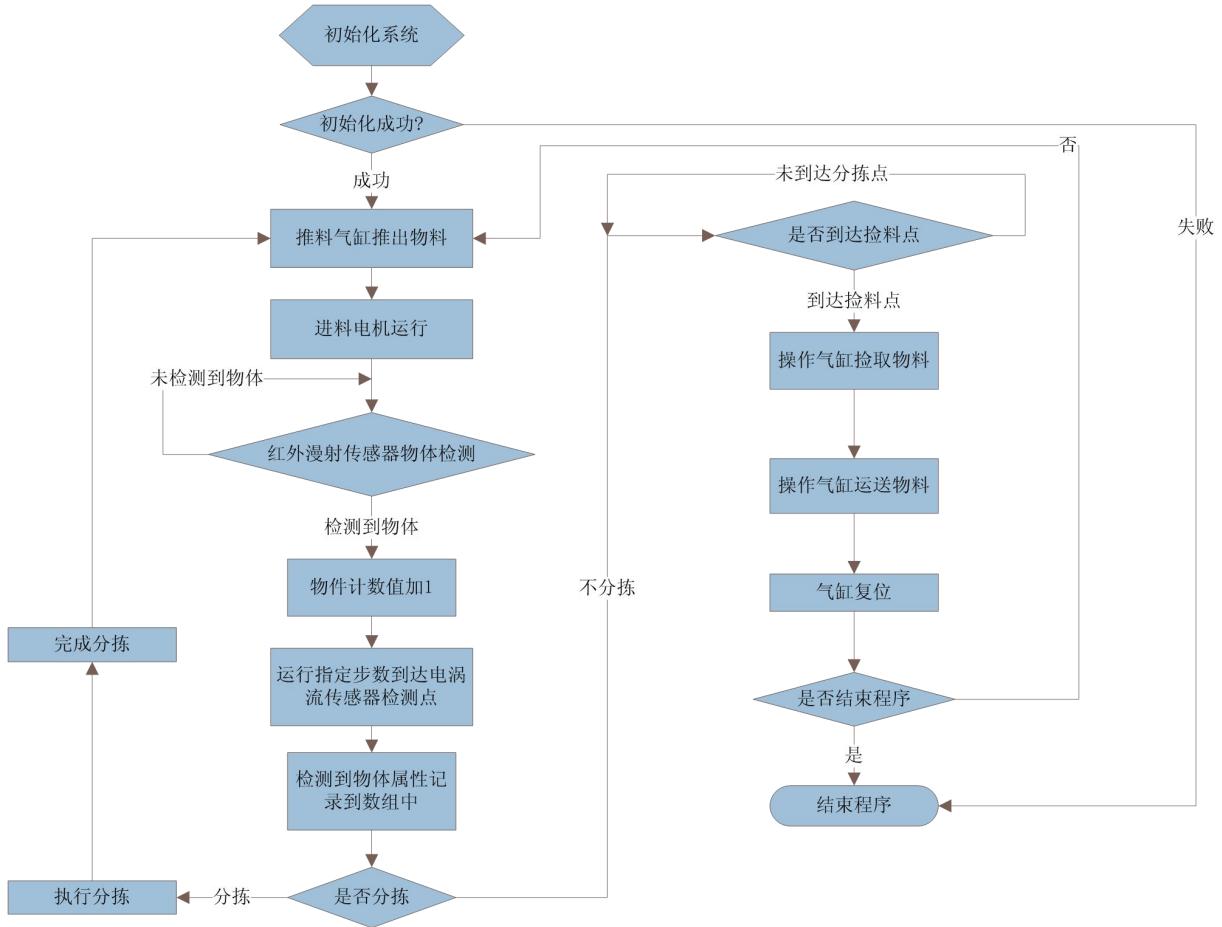


图4.39 一号平台编程—程序流程图

流程图中可以看出控制主要应用到：初始化系统、控制电机的运动、控制气缸的开关与查询传感器的状态等模块。这些模块在前面的小节中已经详细的讲解过，在我们在这次的实验中只需要将以前的知识进行一定的组合。

需要注意的是整个控制程序的设计框架不再和前面单独应用某一个模块相同，我们需要使用一个新的框架更好的去处理控制流程。我们的流程图可以非常好的转换成为一个有限的状态机，在这里我们就直接使用状态机作为控制程序的主要框架。

FAQ: 什么是状态机？

在LabVIEW中，顺序模式是我们最先接触，也是最基本的一种编程模式，程序按照固定的顺序依次执行，结束。如图4.40所示。

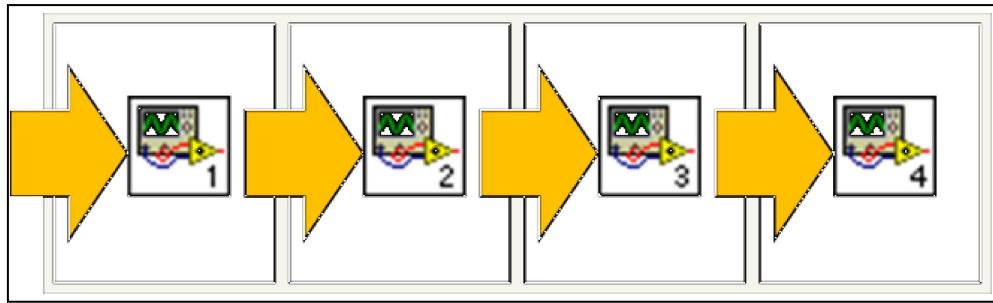


图4.40 一号平台编程—顺序流程图

但在很多情况下，静态的顺序模式并不能满足我们编程的要求，我们需要更有效地动态结构来实时改变程序的执行顺序。比如，一个自动的可乐贩售机，当然它可以实现简单的投币，取可乐，结束这样的顺序模式，但更多的情况下，需要经过多次投币，更或者是取消购买来结束一次操作。如图4.41所示。

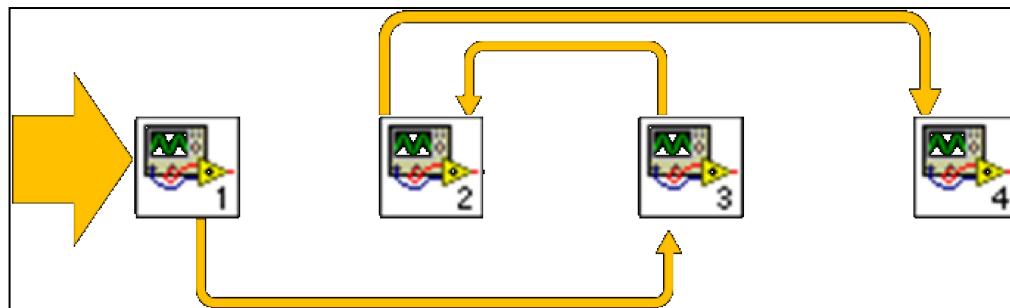


图4.41 一号平台编程—动态结构流程图

简单的说，状态机是对系统的一种描述，该类系统包含了有限的状态，并且在各个状态间可以通过一定的条件进行转换。一般可以用状态图来对一个状态机进行精确地描述。大家请看这个可乐机的状态图。如图4.42所示。

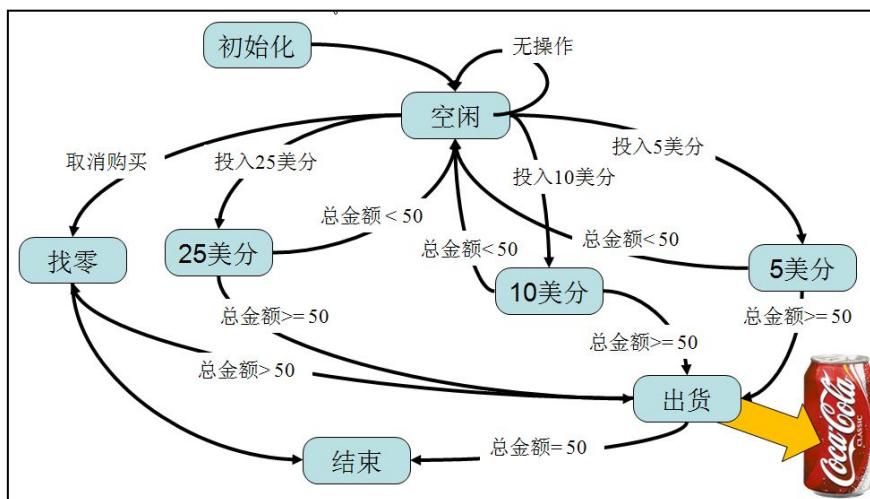


图4.42 一号平台编程—可乐机状态流程图

在LabVIEW中，任何一个状态机都是由三个基本部分构成的，首先外层是一个while循环，同时在while循环中包含有一个条件结构，while循环用于维持状态机的运行，条件结构用以对各个不同的状态进行判断，第三个基本部分是移位寄存器，用以将下一个状态传递到下一次循环状态判断中。另外在一个完整的状态机中，我们一般还会提供初始状态，每一个状态的执行步骤以及下一个状态切换代码等等。在实际的工程状态机中，很多时候会遇到一个状态可能往多个状态转换的情况，在这个时候，我们能够使用LabVIEW提供的附加的逻辑功能来实现下一个状态的判断。

在状态框图中有一个子VI，利用该子VI来对选择哪一个分支为下一状态进行判断。这个功能的实现分为两个子功能，一是子VI实现的逻辑运算，这一部分，大家可以根据程序实际的需要自行编写，第二部分就是状态变量的选择。一般来说，实现状态变量的选择有4个基本的方法。

一般来说，实现状态变量的选择有4个基本的方法：①默认转换，就是无论这个状态执行的情况怎么样，都会无条件地转换到指定的下一个状态中。②选择转换，利用真假选择函数来实现，在逻辑子VI输出为真时，跳入真分支的状态，为假时，跳入假分支的状态，这种方式在两个可能状态的情况下非常简单和好用，但如果下一个状态的有可能是三个或三个以上状态呢？后两种方式就是处理这个情况的。③条件结构转换：条件结构的多个分支分别对应了不同的下一状态值，根据逻辑判断子VI输出的结果，跳入不同的分支以选择不同的状态，当子VI输出1时，跳入State1。④转换数组转换，条件结构虽然解决了多个状态选择的问题，但从结构上来看不太直观，管理不够方便，我们可以把所有待选择的状态放入数组中，通过逻辑判断子VI输出不同的索引值直接从数组中索引出下一个状态输出到移位寄存器。

首先如第一小节中我们讲到的方法首先建立一个DRPrj的项目并将API文件夹添加到项目当中去。如果您还有保留上一小节中的项目请您直接使用上次的项目。项目建立完成之后，按照下边的步骤首先建立一个基于状态机的编程模型，添加到DRPrj项目中。

步骤1：在菜单栏中点击文件(F)→新建。如图4.43所示。



图4.43 一号平台编程—项目浏览器

步骤2：在弹出的对话框中选择VI→基于模板→设计模式→标准状态机。然后点击确定建立这个

程序并保存到项目中去。

通过上面的步骤我们完成了程序的一个基本的建立工作。下面我们在模板中的枚举选项中首先添加控制过程。

根据流程图分析，共有一下几个控制过程。1：初始化过程。2：结束过程。3：推送物料。4：推料气缸复位。5：物件到位检测-光电漫射。6：物体属性检测-电涡流。7：分拣物体。8：不分拣物体。9：物件到位检测-电容式接近开关。10：气缸机构运动。11：气缸机构复位。

下面通过上面的流程图完成接下来的程序。

步骤1：打开刚才创建的程序，在程序面板那的枚举变量中添加如上的过程，如图4.x所示。



图4.44 一号平台编程—枚举对话框

步骤2：在前面板中添加一个开关控件用于停止系统动作。

步骤3：在状态机的分支结构中添加所有的枚举结构。

步骤4：在初始化分支中，添加系统初始化过程，并使用条件结构转化的方法。判断系统初始化是否成功，在成功时将移位寄存器赋值为推送物料；失败时赋值为停止。如图4.x所示。

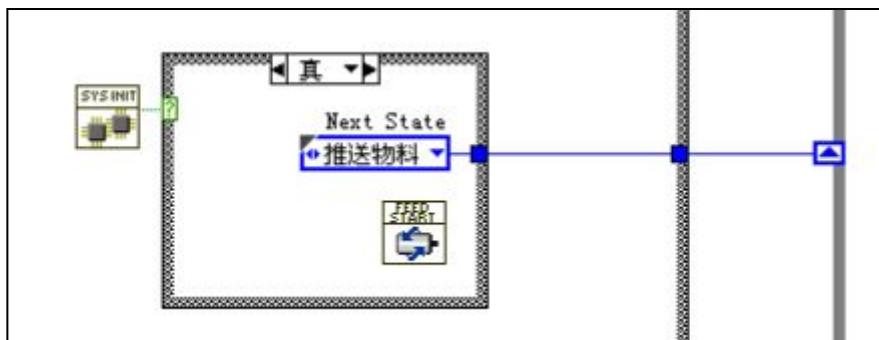


图4.45 一号平台编程—程序面板

步骤5：在停止分支中，添加系统退出过程。

步骤6：在推送物料的分支中添加打开5号继电器的动作。



图4.46 一号平台编程—程序面板

步骤7：在推料气缸复位的分支中添加关闭5号继电器的过程，以及1号电机运动4300步。并传送“物体属性检测-电涡流”状态到移位寄存器中。如图4.x所示。

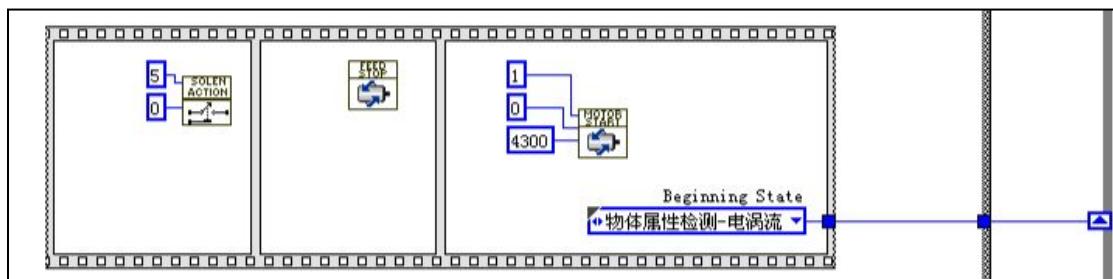


图4.47 一号平台编程—程序面板

步骤8：在物件到位检测-光电漫射分支中，添加查询光电传感器电平状态过程。查询如果检测到有物体到达传感器时传送状态“推料气缸复位”到达移位寄存器；如果没有到达则传送状态“物件到位检测-光电漫射”进行循环查询状态。如图4.x所示。

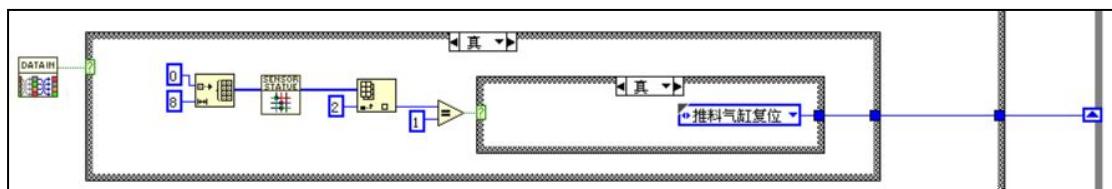


图4.48 一号平台编程—程序面板

步骤9：在物件到位检测-电涡流分支中，首先查询电机运动状态在电机停止的时候延时一段时间，添加查询电涡流传感器电平状态过程。查询到为磁性物体时传送状态“分拣物体”到移位寄存器中；为非磁性物体时传送状态“不分拣物体”到移位寄存器中。如图4.x所示。

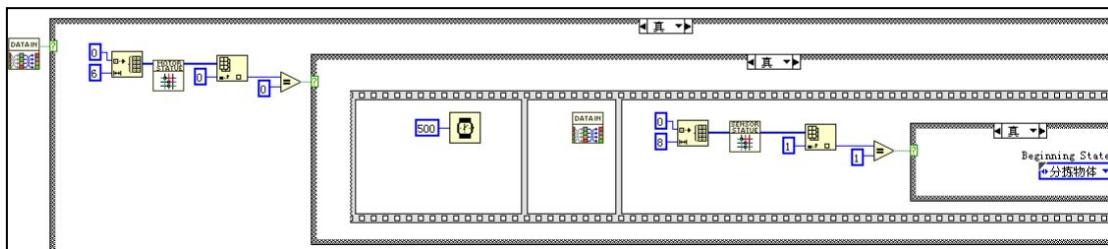


图4.49 一号平台编程—程序面板

步骤10：在分拣物体分支中，首先打开6号继电器将选料气缸打开使得物块可以滑下到选料槽中，接着循环检测物块是否滑下，如图4.x所示。在物块滑下后关闭气缸并让进料电机继续运行。

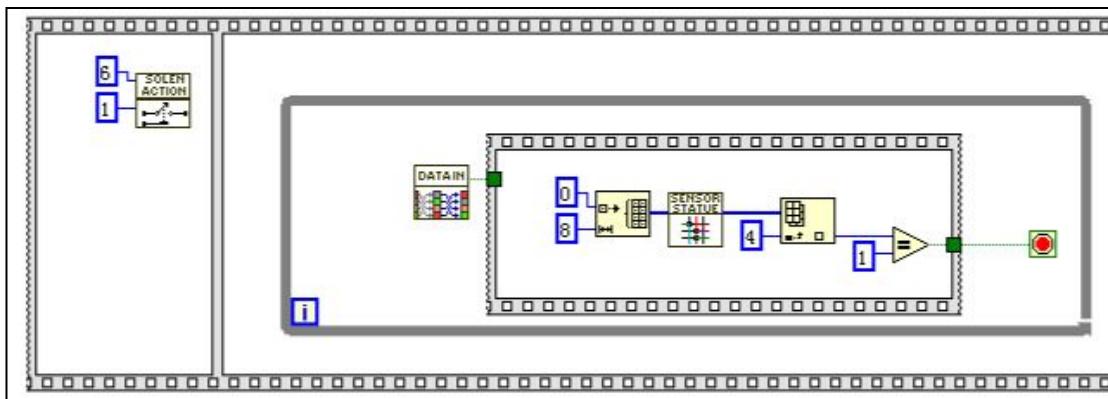


图4.50 一号平台编程—程序面板

步骤11：在不分拣物体分支中，添加进料电机继续运动动作并传送“物体到位检测-电容式接近开关”状态到移位寄存器中。

步骤12：在物体到位检测-电容式接近开关分支中，同步骤9查询电容式接近开关是否检测到物体。在检测到物体后传送“气缸机构运动”状态到移位寄存器中。

步骤13：在气缸运动分支中，添加气缸运行系统系列动作，完成后如果停止按钮为真则传送“停止”状态到移位寄存器中；为假则传送“推送物料”状态到移位寄存器中。以实现循环推料捡料的工作。

通过上面步骤完成一号台的推料检测简单的一个控制流程设计。上面的程序可以在光盘中的例子文件夹中找到相对应的程序。

二、实训习题

1. 绘制实例程序中的状态机控制流程图。
2. 依照上面的程序请您修改步骤13中的气缸运动流程使得气缸更为流畅快速的运动。例子中步骤13的气缸运动全部使用延时的方法控制运动完成时间，请您使用查询限位开关状态的方法完成气缸的连续运动。

3. 依照上面的程序请您选取非磁性的物体到选料槽中。
4. 依照上面的程序请您使用2号选料槽来选取指定性质的物料。

4.8 二号平台控制编程

在上面一个小节中通过实训完成了一号平台的选取运输物料的控制程序。在本小节的实训中将使用二号平台模拟工业生产中的雕刻动作。在系统的二号平台的唯一一个气缸上面装载上一支记号笔，实训中通过控制程序控制二号平台上的电机带动载物台使用记号笔记录下运动轨迹。

一、LabVIEW程序控制流程

控制流程图如图 4.51 所示，流程图中可以看到控制二号台运动的过程与一号台控制过程相比要单一得多，在二号台控制程序中程序基本结构更适合使用顺序执行的框架完成控制程序设计。

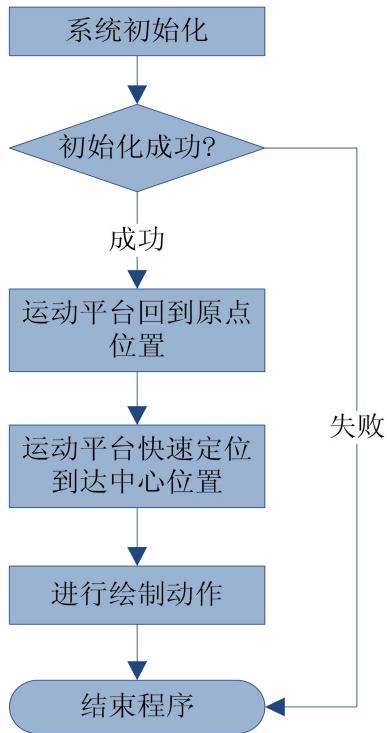


图4.51 二号平台编程—程序流程图

如图 4.51 所示，在执行二号平台控制程序编写的时候我们需要如流程图中所描述的顺序使用前面小节中详细介绍过的程序模块进行组合。

依据流程图分析，共有以下几个控制过程：①系统初始化②控制电机使载物台回到坐标轴原点位置③控制电机使载物台到达平台中央位置④完成绘制形状的动作⑥结束程序释放设备资源。

那么，首先如第一小结中我们讲到的方法首先建立一个 **DRPj** 的项目并将 **API文件夹** 添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，直接建

立一个空的vi程序，添加到DRPj项目中。

在完成了建立程序以及项目的工作后，根据上面的流程图完成接下来的控制程序。

步骤1：打开刚才创建的vi程序，在程序面板中首先添加系统初始化模块API。

步骤2：在添加的系统初始化模块API后面加入条件结构，如图4.52所示。



图4.52 二号平台编程—程序面板

步骤3：在假的条件分支中拖入项目浏览器 API 文件夹下的“API SYSQUIT”模块。

步骤4：在真的条件分支中放入一个顺序结构，参考图4.53中的电机方向与编号将3号电机与4号电机分别运行到达方向1的限位位置。程序框图如图4.54。

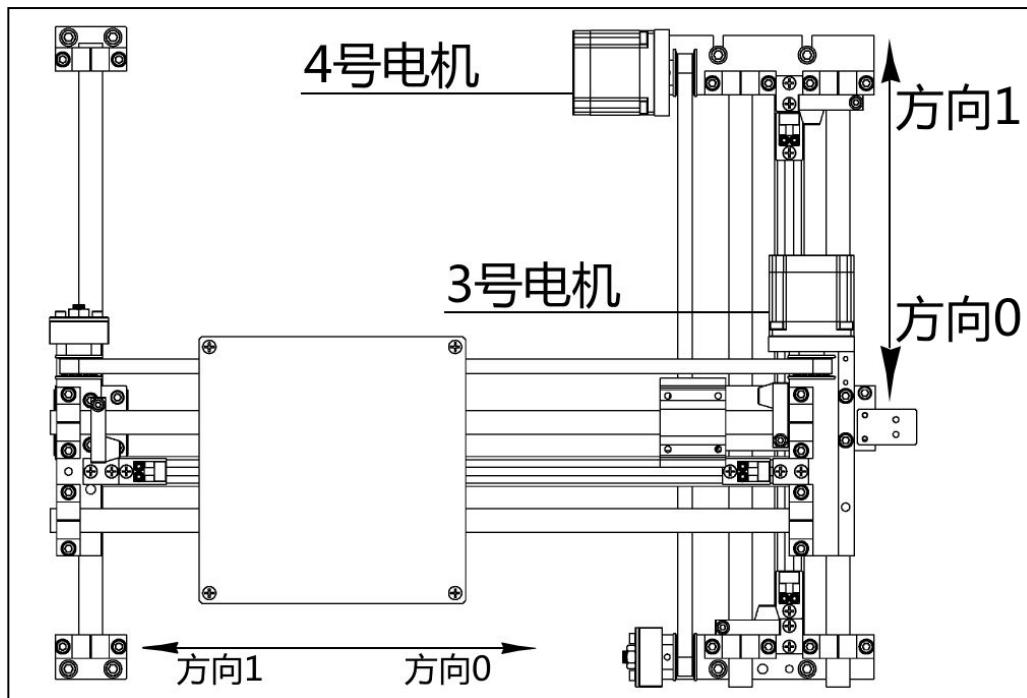


图4.53 二号台—电机编号与方向示意图

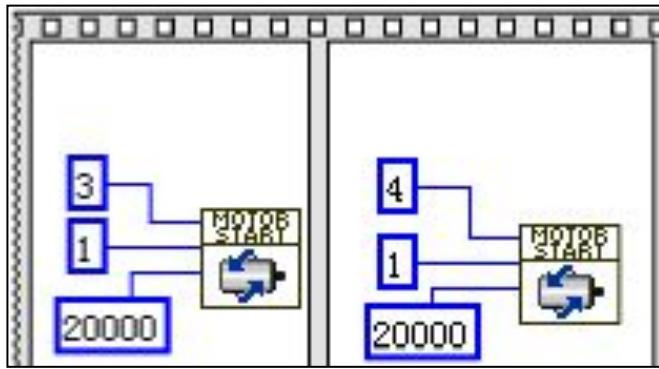


图 4.54 二号平台编程—程序面板

步骤5：在后面的顺序帧中使用循环获取电机状态在电机停止时我们认为电机已到达限位的位置而接着执行后面的动作，不然一直处于循环中，保持系统同步。程序框图如图4.55所示。

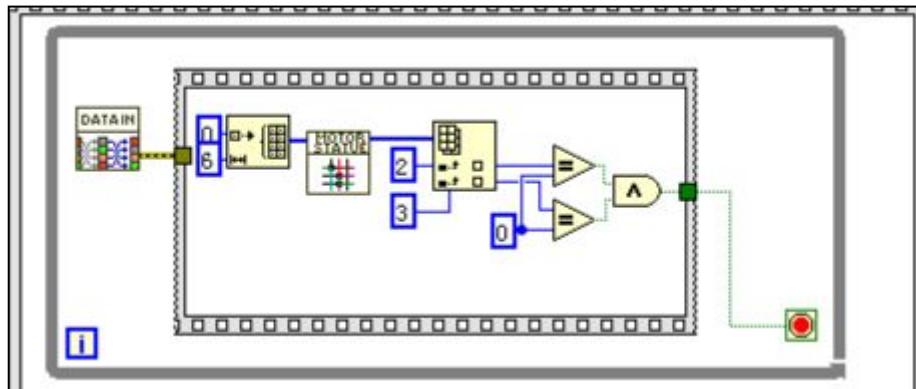


图4.55 二号平台编程—程序面板

步骤 6：同步骤 4 使 3 号电机向 0 方向移动 8100 步长，使 4 号电机向 0 方向移动 3700 步长让电机回到中心点，程序如图 4.56 所示。

步骤 7：在后面帧中加入如步骤 5 中的查询电机运动状态的模块等待电机运动完成。

通过上面的步骤载物台已经运动到达平台中心位置，在这次实训我们完成绘制正方形的一个路径。

步骤 8：将装载有记号笔的气缸向下运动，使记号笔向下接触到载物台上。直接打开 17 号气缸完成该动作。

步骤 9：驱动 3 号电机向 1 方向移动 2000 步长，接着如步骤 5 中的查询电机运动状态的模块等待电机运动完成。

步骤 10：驱动 4 号电机向 1 方向移动 2000 步长，接着如步骤 5 中的查询电机运动状态的模块等待电机运动完成。

步骤 11：驱动 3 号电机向 0 方向移动 2000 步长，接着如步骤 5 中的查询电机运动状态的模块等待电机运动完成。

步骤 12：驱动 4 号电机向 0 方向移动 2000 步长，接着如步骤 5 中的查询电机运动状态的模块等待电机运动完成。

步骤 13：将装载有记号笔的气缸向上运动，使记号笔向上收起。直接打开 16 号气缸完成该动作。

步骤 14：退出系统释放资源。

通过上面的步骤完成成程序后，点击LabVIEW软件上的运行按钮编译运行程序。如果没有错误观察并记录下台面的运行轨迹。上面的程序可以在光盘中的例子文件夹中找到相对应的程序。

二、实训习题

- 1.绘制实例程序中的控制流程图。
- 2.自行完成封装步骤5的过程成为子VI。
- 2.依照上面的程序修改步骤9~步骤13在载物台上面绘制不同的图形。
- 3.请自行完成一条斜线的绘制控制程序。（注：绘制斜线时使用双轴联动的方法，双轴联动时需对电机速度进行规划）。

4.9 三号平台控制编程

在上面的两个小节中我们完成了一号平台与二号平台的控制程序，而系统中三号平台是一个立体仓库系统，是用来堆放物料的仓库。控制程序需要运输物料进入仓库内。

一、LabVIEW程序控制流程

控制流程图如图 4.56 所示，在三号台的控制上面我们分 2 个控制过程进行展开实训。在控制流程图中左边的过程为气动机械手抓取并放置物料到指定位置的过程，在右边部分则是控制电机放置物料进入对应仓库的过程。

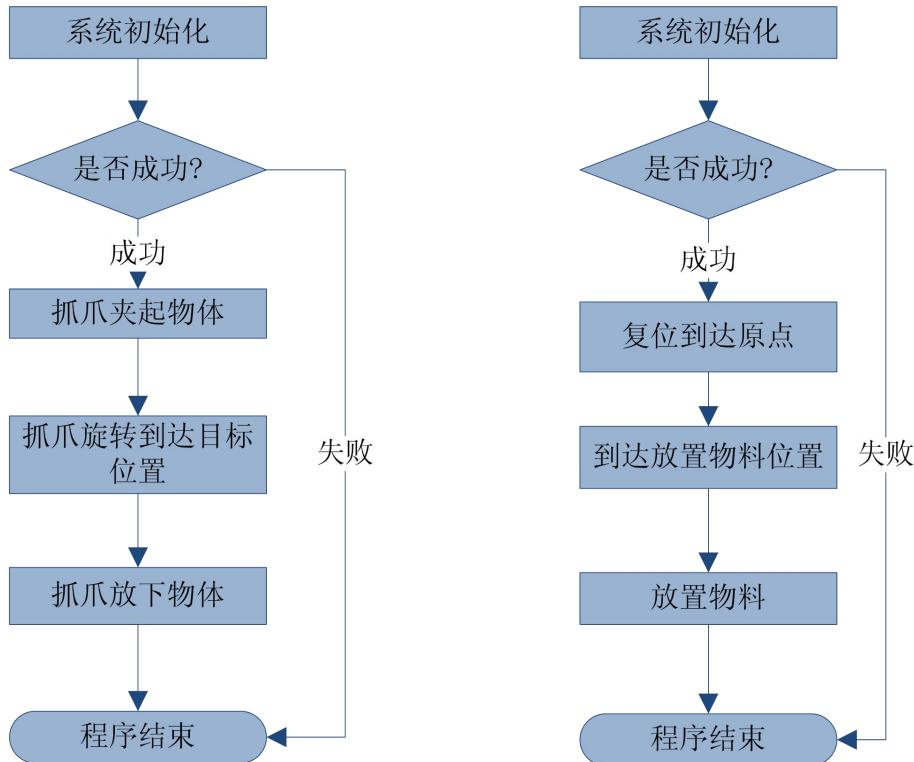


图 4.56 三号平台编程—程序流程图

如图 4.56 流程图中所示，控制程序的编写须如同流程图中描述的模块使用顺序结构进行组合。

本小节的第一个实训内容是完成气动机械手抓取并放置物料到指定位置的控制程序，如同图 4.56 左边的流程图所示，共有以下几个控制过程：①系统初始化②气动抓爪夹起物体③抓爪旋转④气动抓爪放下物体。

那么，首先如第一小结中我们讲到的方法首先建立一个 **DRPj** 的项目并将 **API 文件夹** 添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，直接建立一个空的 vi 程序，添加到 **DRPj** 项目中。

在完成了建立程序以及项目的工作后，根据上面的流程图完成接下来的控制程序。

步骤 1：打开刚才创建的 vi 程序，在程序面板中添加系统初始化模块，并在模块后面的连接上条件结构判断初始化步骤是否成功。

步骤 2：依照图 4.57 中的继电器编号打开 34 号继电器使抓爪向下运动并等待 1000ms，确保向下运动完成。

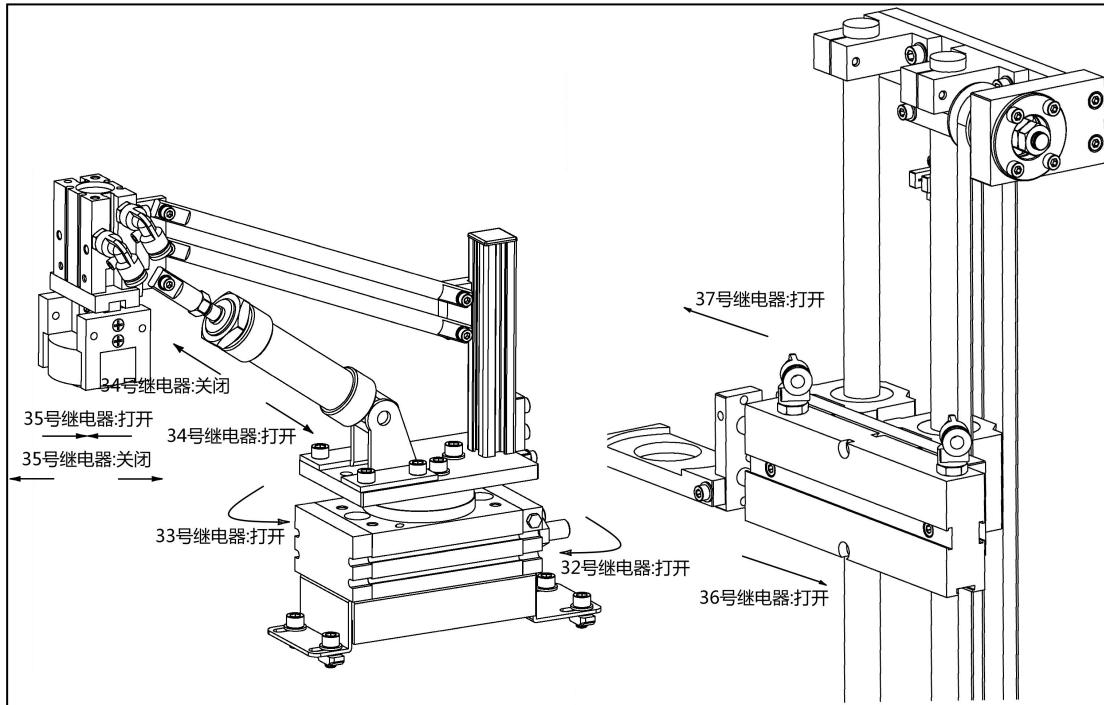


图 4.57 三号平台—继电器编号

- 步骤 3：打开 35 号继电器，使得机械手抓爪夹起物体。
- 步骤 4：关闭 34 号继电器，使得机械手抬起并延时 1000ms，确保机械手运动完成。
- 步骤 5：打开 33 号继电器，使得机械手开始逆时针旋转，循环查询平台 3 的 7 号输入口状态，以确保抓爪到达指定位置，如图 4.58。

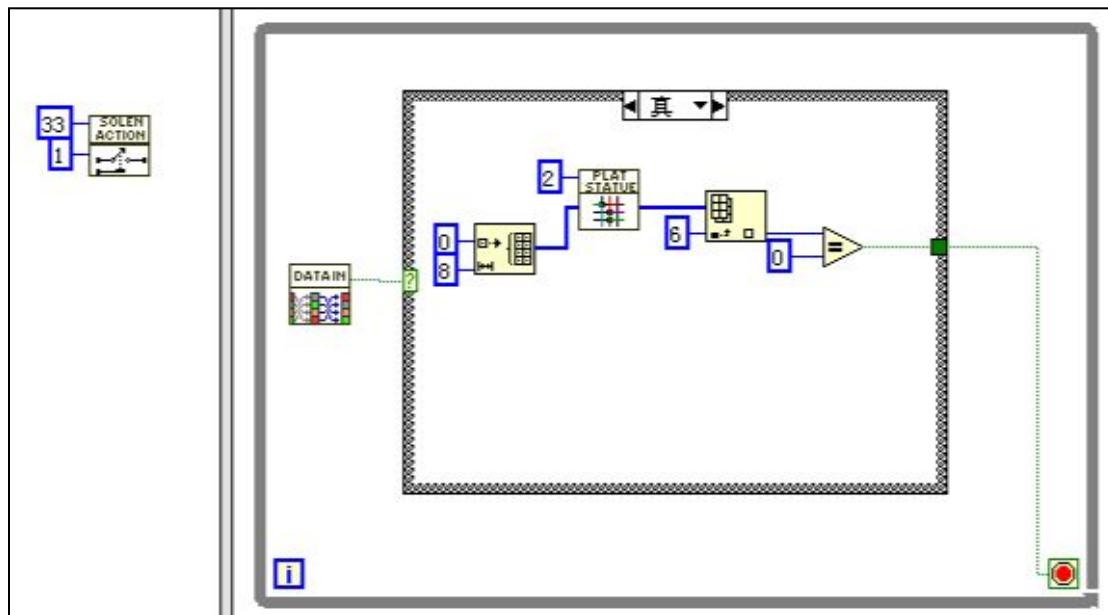


图 4.58 三号平台—程序框图

- 步骤 6：重复步骤 2 中的过程。使的抓爪向下运动。
- 步骤 7：关闭 35 号继电器，使得机械手放下物体。

通过上面的步骤完成成程序后，点击LabVIEW软件上的运行按钮编译运行程序。如果没有错误观察并记录下机械手的运行状态。上面的程序可以在光盘中的例子文件夹中找到相对应的程序。

本小节的第二个实训内容是完成三号平台运料系统放置物料到达物料仓库的控制程序，如同图 4.56 右边的流程图所示，共有以下几个控制过程：①系统初始化②电机复位到达原点位置③电机驱动到达放置物料位置④放置物料进入仓库。

同样如中我们上面讲到的步骤首先建立一个 DRPrj 的项目并将 API 文件夹 添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，直接建立一个空的 vi 程序，添加到 DRPrj 项目中。

在完成了建立程序以及项目的工作后，根据上面的流程图完成接下来的控制程序。

步骤 1：打开刚才创建的 vi 程序，在程序面板中添加系统初始化模块，并在模块后面的连接上条件结构判断初始化步骤是否成功。

步骤 2：依据图 4.59 中的电机编号与运动方向，将 5 号电机与 6 号电机分别以方向 0 与方向 1 运动 20000 步长回到原点位置，并循环查询运动状态确保电机运动完成后执行下一个步骤。

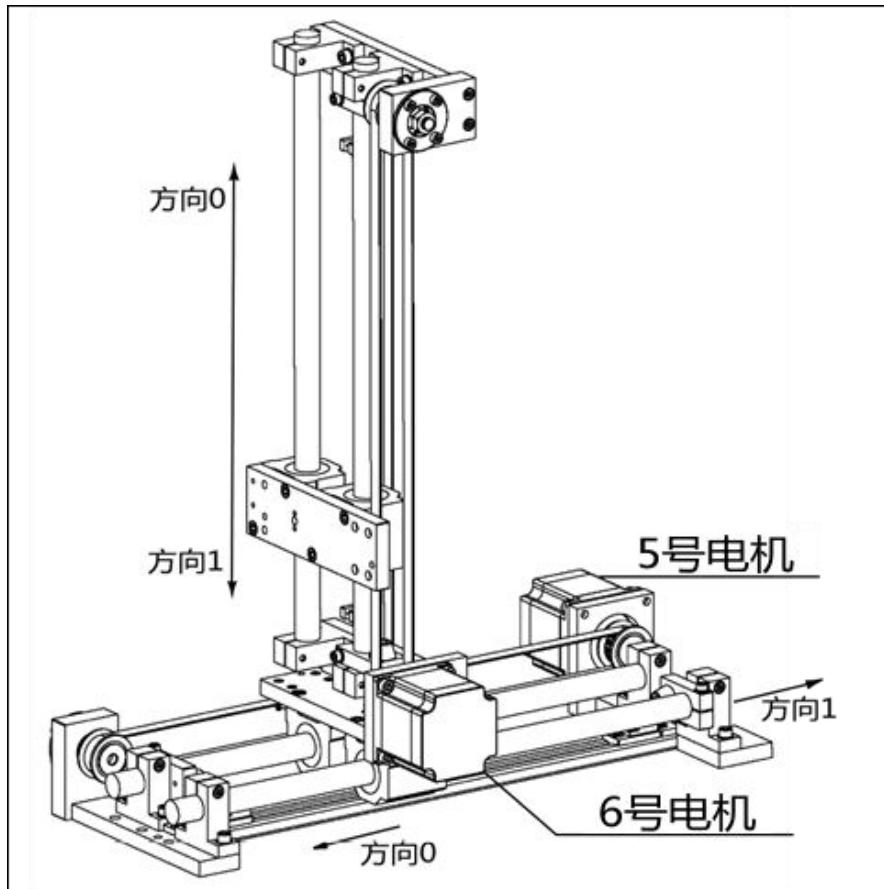


图 4.59 三号平台—电机编号与运动方向

步骤3：控制5号电机向方向1运行1350步长，6号电机向方向0运行10500步长，并循环查询运动状态确保电机运动完成后执行下一个步骤。

步骤4：依据图4.57打开37号继电器，使送料气缸前伸，循环查询6号磁性开关状态确保气缸动作完成后执行下一个步骤。

步骤5：控制6号电机向方向1运行500步长，并循环查询运动状态确保电机运动完成后执行下一个步骤。

步骤6：依据图4.57打开36号继电器，使送料气缸收回，循环查询5号磁性开关状态确保气缸动作完成后执行下一个步骤。

步骤7：结束程序，释放设备资源。

通过上面的步骤完成成程序后，点击LabVIEW软件上的运行按钮编译运行程序。如果没有错误，执行程序观察并记录下放置物料的整个控制流程，在程序中如果没法确定电机运行步骤请运行测试程序确定具体电机运行步骤以免发生撞车等不当操作。上面的程序可以在光盘中的例子文件夹中找到相对应的程序。

二、实训习题

- 1.绘制实例程序中的控制流程图。
- 2.独立完成放置物块的控制程序。
- 3.合并机械手控制流程与运料系统控制流程并绘制控制流程图，思考控制方案。

4.10 综合控制编程

前面的3个小节中分别完成了单个平台的控制程序，在本小节中将结合三个平台，对物料进行分拣入库。

一、LabVIEW程序控制流程

控制流程图如图4.60所示，流程图中可以看到控制可以分成4个部分，从一号平台开始，一号平台负责物料的分拣，二号平台负责物料的运输，三号平台的气动机械手负责物料的捡取，三号平台的运料机构负责放置物料进入仓库。

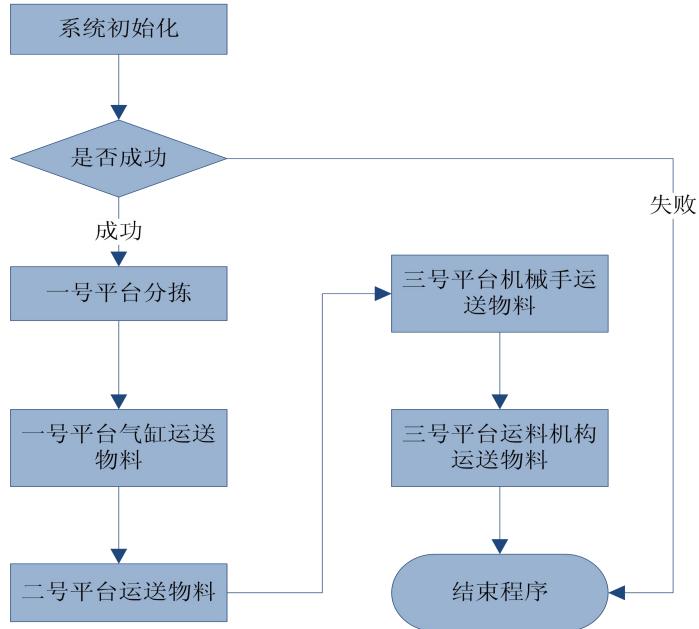


图 4.60 综合控制编程—控制流程图

如图 4.60 中的流程图，在这次实训中我们需要结合前面 3 节中的控制程序进行完成整套控制系统的程序。

那么，首先如第一小结中我们讲到的方法首先建立一个 DRPrj 的项目并将 API 文件夹 添加到项目当中去。如果您还有保留上一小结中的项目请您直接使用上次的项目。项目建立完成之后，依照下面的步骤建立一个基于状态机模板的程序，添加到 DRPrj 项目中。

步骤1：在菜单栏中点击文件 (F) → 新建。如图4.43所示。



图4.61 综合控制编程—项目浏览器

步骤2：在弹出的对话框中选择 VI→基于模板→设计模式→标准状态机。然后点击确定建立这个程序并保存到项目中。

在完成了建立程序以及项目的工作后，根据上面的流程图完成接下来的控制程序。

步骤 1：打开建立的 vi 程序，在枚举中添加程序过程。①Initialize②一号平台分拣运输③二号平台运输物料④三号平台气动机械手运输物料⑤三号平台运输机构运输物料⑥Stop，如图 4.62。



图 4.61 综合控制编程一枚举选项编辑

步骤 2：在条件分支中添加上面步骤 1 中添加的枚举的过程。

步骤 3：在初始化的分支中添加系统初始化模块，初始化模块后面接入条件分支以判断初始化是否成功。

步骤 4：在一号平台分拣运输的分支中添加一号平台控制编程中的控制流程，如 4.7 小节中的状态机程序。

步骤 5：在二号平台运输物料的分支中添加气缸放下物料的过程并将物料运输到平台有段的过程。

步骤 6：在三号平台气动机械手运输物料的分支中添加气动机械手抓取物块的过程。

步骤 7：在三号平台运输机构运输物料的分支中添加气动机械手放下物料过程与如 4.9 小节中的运料机构的放料过程。

步骤 8：在停止分支中加入释放设备资源的过程。

通过上面的步骤完成成程序后，点击 LabVIEW 软件上的运行按钮编译运行程序。如果没有错误，执行程序观察并记录下综合控制程序中的整体流程。上面的程序可以在光盘中的例子文件夹中找到相对应的程序。

二、实训习题

1. 绘制实例程序中的控制流程图。
2. 独立完成整个控制程序。
3. 依照上面的程序重新规划新的控制方案并完成控制程序。

注意和警告提示

	警告	操作不当将会导致人身伤亡或设备损坏
	注意	操作不当将会导致人员受伤或其他物品损失
	禁止	禁止操作
	必须	必须进行的操作

储运常识

1. 搬运安全

**注意**

搬运时，特别注意要相互配合好，轻抬轻放，严防将人压伤，碰伤，砸伤！

2. 近途运输

近途运输(省内)设备，可以不必包装，但要用塑料薄膜封盖好，并且注意：

- (1) 严防剧烈颠簸，遇到不平坦的路面，一定要放慢行驶！
- (2) 严禁急刹车！以防止设备滑动，并与车体碰撞而损坏。
- (3) 严防设备被雨水淋湿或水浸湿！

**必须**

必须将设备平稳地放于车厢底部，并且，平台的支脚下要垫上较好的减震物（如海绵或泡沫塑料）。以防止在运输中将设备震坏。

3. 远途运输

设备进行远途运输必须包装。

- (4) 要轻抬轻放，严防剧烈颠簸，碰撞，敲击！
- (5) “轻拿轻放”，“严防潮湿”，“严禁倒置”，“贵重设备”等标识一定要明确。
- (6) 一定要将设备的向上放置方向在木箱（纸箱）上标识正确，严禁倒置。
- (7) 将设备整体用塑料密封好以后再放于木箱（纸箱）内进行包装，以防潮湿和水淋。

**必须**

包装时，根据设备大小，订制木箱（纸箱）包装。木箱与设备之间一定要充塞好减震物（如海绵或泡沫塑料），不许设备与木箱直接碰撞！

安全事项

安全生产涉及到人身安全及财产损失，事关重大，所以特别强调说明。



注意

请一定要注意安全用电！请不要在通电状态下触碰或操作电箱和系统的电线电缆及器件，以防造成人身伤亡和设备损坏。



禁止

本设备属于贵重设备，严禁用力敲击或撞击！

维护和保养

1. 注意事项:

- 要远离强电磁辐射源，防止电磁干扰。
- 避免在空气太潮湿和温差太大的地方使用。
- 通风要良好。长时间使用时，如果发现系统某个部件过热，应当停机休息，待冷却后再用；
- 要远离金属粉屑污染源，以避免金属粉屑进入电控柜的箱体内，造成短路，甚至使电控柜受到严重损坏而无法正常使用。
- 防止机械及电控柜意外震动。

2. 正常维护

● 润滑油

定期加润滑油（一般三个月加油一次），减小机械部件间的摩擦力，把机械磨损值降到最低标准。长期不加润滑油，会使机械部件之间的磨擦系数增大，造成摩擦力增大，进而造成机器部件的磨损，直接影响加工精度，严重时会因为摩擦力过大，马达无法带动丝杠运动而导致偏差。

● 防尘

为了防止灰尘造成破坏，需要用户每天及时用吸尘器吸去工作台和机械部件上的灰尘。

3. 定期检查

提请用户定期检查机械及控制系统，有问题请与我们联系。