AX系列可编程控制器软件手册

深圳市英威腾电气股份有限公司
SHENZHEN INVT ELECTRIC CO., LTD.

前 言

非常感谢您使用 AX 系列可编程控制器。

本手册记载了使用 AX 系列可编程控制器所必需的信息。使用前请详细阅读本手册，充分理解其功能和性能，完成系统构建，发挥其优越性能。

阅读对象

本手册适用于具有电工知识的人员、电气工程师或具有同等知识的人员阅读。

适用产品

AX70 可编程控制器

AX71 可编程控制器

AX 系列可编程控制器扩展模块

在线支持

除本手册外，还可以通过登录英威腾官方网站获取产品资料和技术支持。

网址：http://www.invt.com.cn

终端用户为军事单位，或将本产品用于兵器制造等用途时，请遵守《中华人民共和国对外贸易法》有关出口管制的相关规定，办理相应手续。

目录

第1章 控制器及编程平台简介

1.1 AX70 系列可编程控制器概述

1.1.1 产品简介

AX70 系列可编程控制器是一款采用模块化结构设计的高性能可编程控制器，为用户提供智能化的自动化解决方案。采用 IEC61131-3 编程语言体系，支持 IL、LD、FBD、ST、SFC、CFC 六种标准编程语言。通过 EtherCAT 总线可实现电子凸轮、电子齿轮、同步控制、定位等高阶运动控制功能；支持 200 kHz 高速 I/O，可实现直线插补、圆弧插补等运动控制功能。

AX70 可编程控制器采用机架式布局，每个机架支持本地扩展 16 个扩展模块，支持数字量输入/输出模块、模拟量输入/输出模块、温度模块、通信模块等多种功能扩展模块，并可通过 EtherCAT 现场总线进行远程 I/O 扩展。

此外，AX70 系列可编程控制器支持 EtherCAT、CANopen、RS485、以太网多种通信接口，满足用户多样化的应用需求。

1.1.2 产品配置及模块说明

AX70-C-1608P 可编程控制器 CPU 支持以下模块：电源模块、数字量输入模块、数字量输出模块、模拟量输入模块、模拟量输出模块、温度模块和通信模块，系统组合示意图如下。

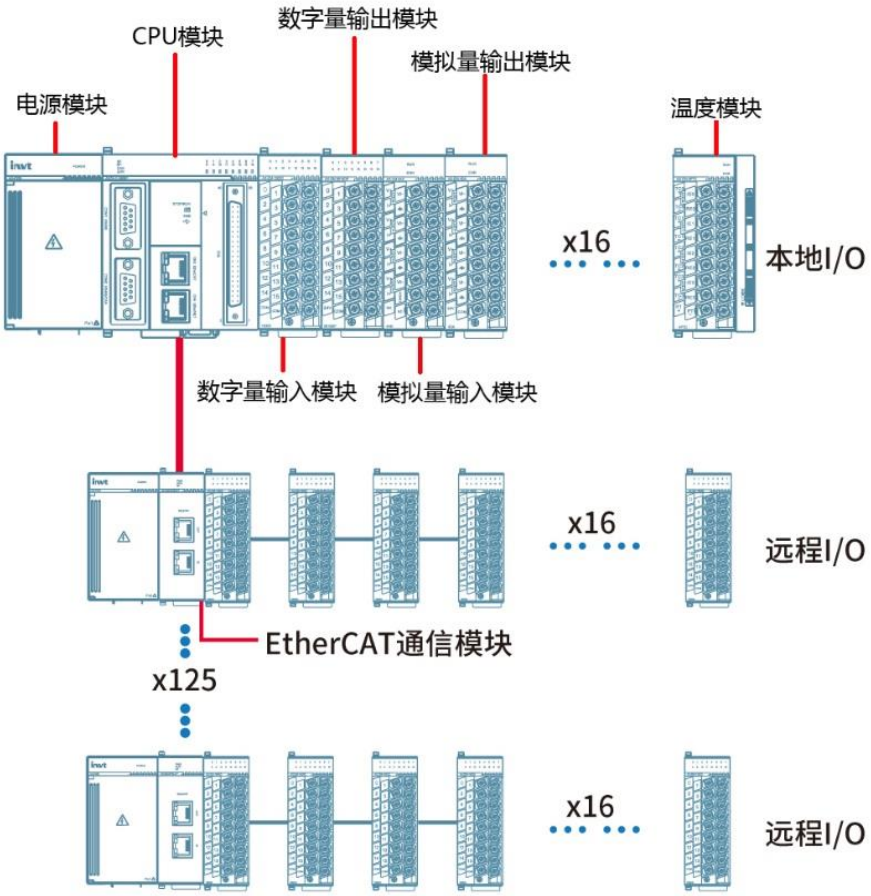


图 1-1 系统集成示意图

1.1.3 系统应用流程

1.2 编程平台概述

1.2.1 Invtmatic Studio 简介

Invtmatic Studio 是深圳英威腾电气股份有限公司开发的编程平台，全面支持 IEC61131-3 编程语言体系，支持 IL、LAD、FBD、SFC、ST、CFC 六种标准编程语言。

1.2.2 软件编程界面简介

Invtmatic Studio 软件创建完应用工程后的界面如下图所示。

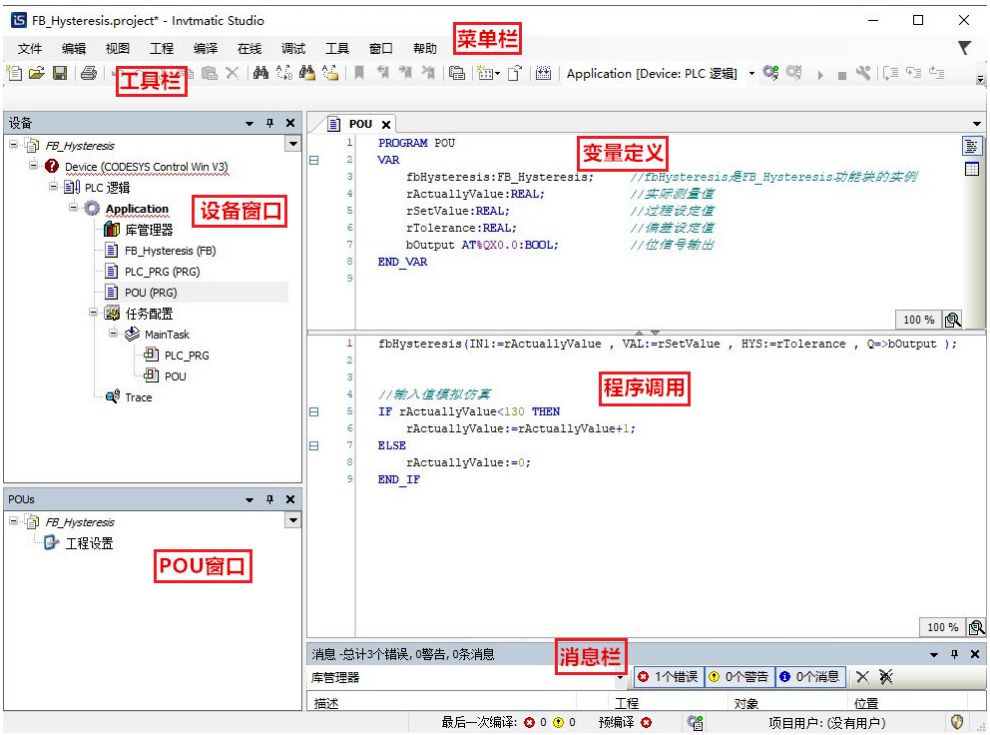


图 1-2 Invtmatic Studio 软件应用工程界面

1.3 PLCopen 规范介绍

PLCopen 国际组织成立于 1992 年，是一个独立于生产商和产品的全球性协会，其主要一项活动就是致力于 IEC61131-3 的推广，它是全球工控界编程的唯一标准。标准的编程接口允许不同背景和技能的人们在软件生命周期的不同阶段创造不同元素的程序：技术规范、设计、实现、测试、安装和维护。然而它们都遵守一个共同的结构并且和谐地一起工作。该标准定义了六种编程语言 CFC（顺序功能块）、SFC（顺序功能图）、IL（指令表）、LD（梯形图）、FBD（功能块图）和 ST（结构文本）。通过分解成逻辑元素、模块化以及现代软件技术来组成每个程序，从而提高了其重复使用性，同时对于编程人员，基于 IEC61131-3 的编程技术可在整个工业控制领域中广泛的使用。

AX 系列控制器选用的 Invtmatic Studio 编程平台，该平台完整支持 PLCopen 规范，用户可以引用许多标准的功能函数库；高级语言的编程方式，易于控制器厂家和用户开发自己专有的功能块和指令库，借用已有的类似控制程序，形成行业特点的“工艺包”，可显著提高用户编程效率。

第2章 入门指引

2.1 软件安装与卸载

2.1.1 软件获取

英威腾 AX 系列可编程控制器用户编程软件采用 Invtmatic Studio 平台，安装文件以及相关参考资料等，用户可通过以下途径获取：

✧ 访问英威腾官网（www.invt.com.cn）“服务与支持 > 自助服务 > 资源下载”页面免费下载软件安装包。

✧ 从英威腾各级经销商处获得软件安装光盘。

2.1.2 软件安装要求

具备以下条件的台式 PC 或便携式 PC 机。

✧ Windows XP/ Windows 7/ Windows 8/ Windows 10 操作系统

✧ CPU 主频：2GHz 以上

✧ 内存：2GB 以上

✧ 空间：可用硬盘空间 5G 以上

2.1.3 安装准备

若为首次安装 **Invtmatic Studio**，请首先查看个人电脑的硬件条件是否具备上述的软件安装要求，确认满足安装条件后，直接安装即可。

若想安装最新版本 **Invtmatic Studio**，可先查看本机安装的 **Invtmatic Studio** 版本信息，查看“帮助 > 关于”，若不是最新版本可采用在线升级模式升级软件。

图 2-1 版本信息

2.1.4 开始安装

1、 打开安装文件所在位置，双击打开“Invtmatic Studio Setup 64 V1.0.2.exe”文件。

2、 双击打开后，启动安装，可以看到如下界面，进入安装准备阶段。

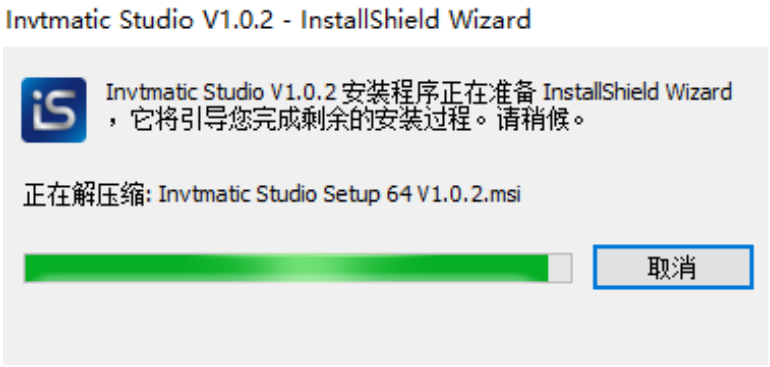


图 2-2 安装准备

3、 出现如下提示界面，点击“下一步”开始安装。

图 2-3 安装向导

4、 选入许可证协议界面，勾选“我接受该许可证协议中的条款(A)”，然后点击“下一步”。

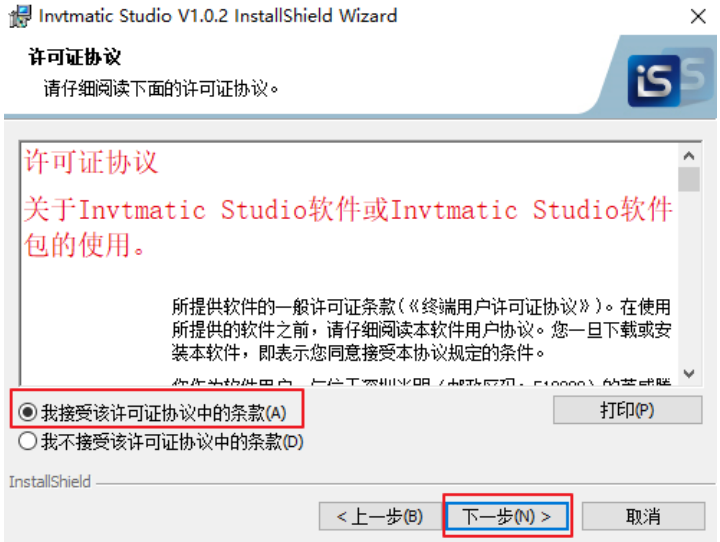


图 2-4 许可证协议

5、 设置好软件安装路径后，点击“下一步”。

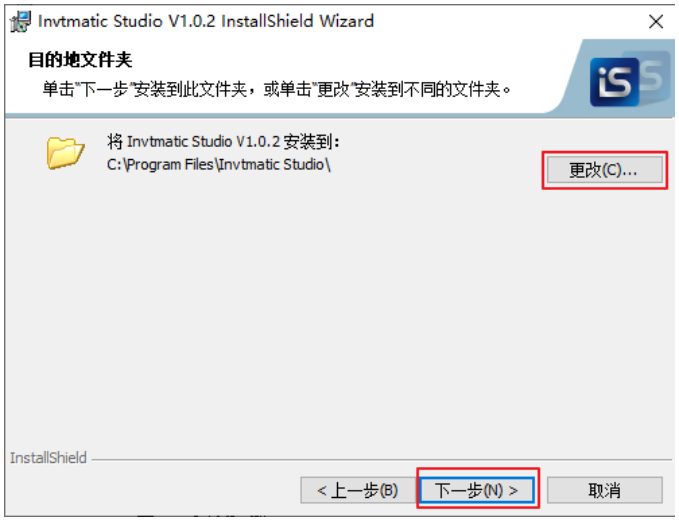


图 2-5 安装路径

6、 进入安装组件选择界面，可选择自定义进行勾选，若无特殊需求，按默认勾选即可，点击“下一步”。

图 2-6 安装类型

7、 出现如下提示界面，点击“安装”。

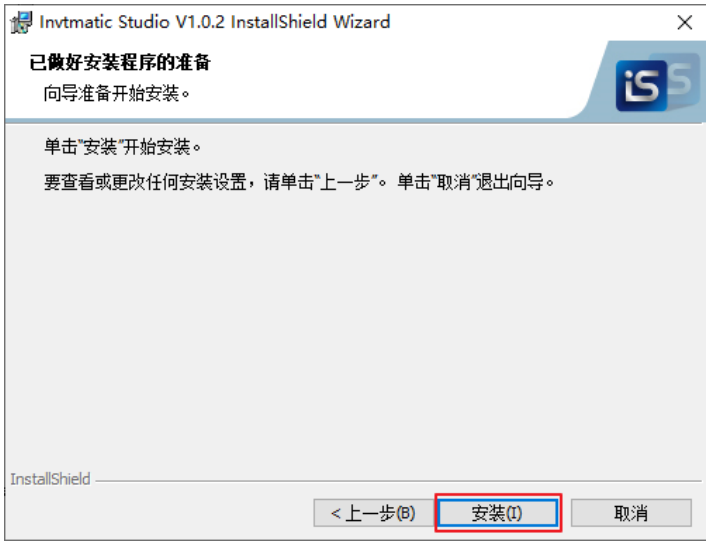


图 2-7 安装步骤

8、 出现如下界面，等待安装进度条，直到出现下面所示提示，点击“完成”，完成 **Invtmatic Studio** 的安装。

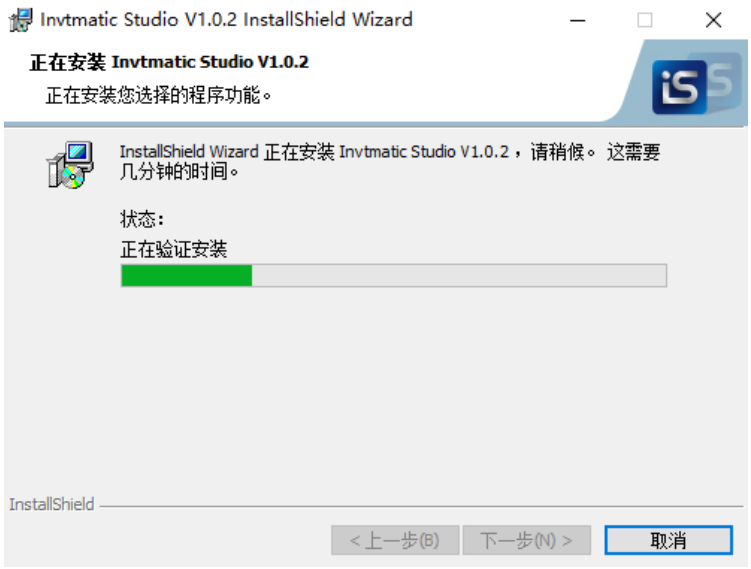


图 2-8 安装进度

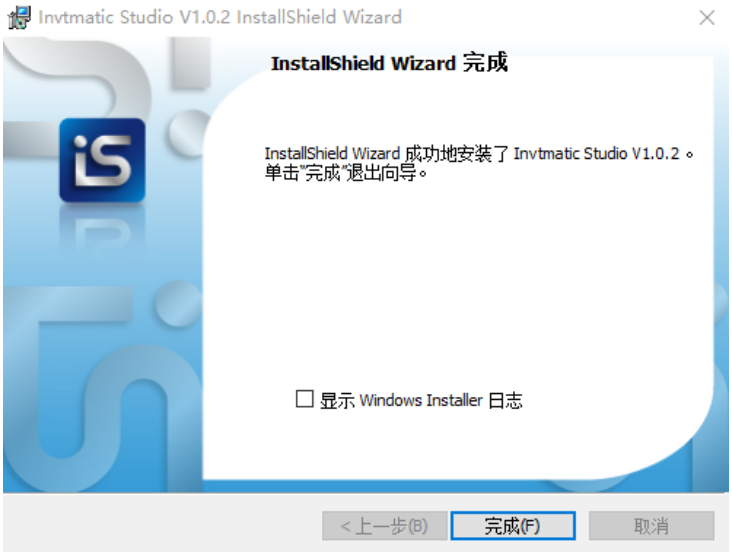


图 2-9 安装完成

2.1.5 卸载 Invtmatic Studio

通过使用标准的 Windows 系统卸载软件方法卸载 **Invtmatic Studio** 即可，具体步骤如下：

1、 关闭 **Invtmatic Studio** 运行程序，包括后台运行程序。

2、 进入控制面板，找到 **Invtmatic Studio** 程序，右键单击，选择“卸载”。

3、 确认并等待程序卸载完毕。

2.2 AX70 系列硬件连接

上位机与控制器的硬件连接方式：

A：采用 LAN 网络电缆连接

B：采用 Mini USB 线缆连接

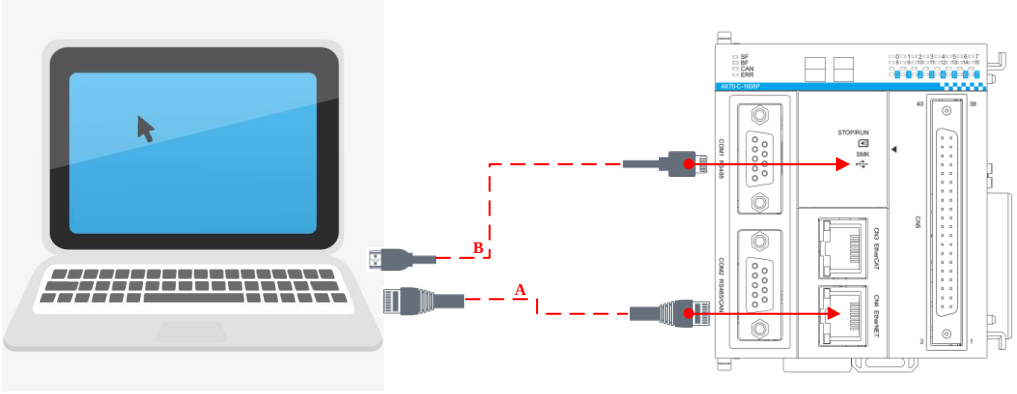


图 2-10 上位机与控制器的硬件连接示意图

2.3 PC 端通信配置

1、 采用 LAN 网络电缆连接，需保证 PC 端 IP 地址和控制器的 IP 地址位于同一网段，AX 系列的出厂默认 IP 地址为 192.168.1.10，PC 端 IP 地址应设置为 192.168.1.xxx。（xxx 表示除控制器端 IP 末尾地址外的 1~254 范围内任一整数值）

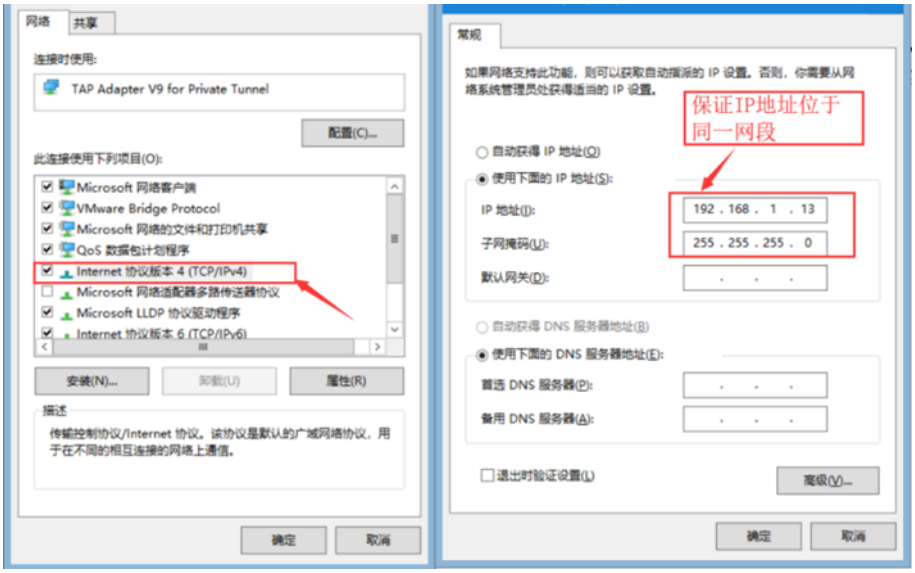


图 2-11 采用 LAN 网络电缆连接 PC 端通信配置

2、 采用 Mini USB 线缆连接，PC 端配置方法如下：

✧ USB 驱动安装：

1、 在计算机管理中选中设备管理器，找到“RNDIS/Ethernet Gadget”，点击右键选择“更新驱动程序软件(P)…”。

图 2-12 RNDIS/Ethernet Gadget 设备

2、 选择“浏览计算机以查找驱动程序软件（R） > 从计算机的设备驱动程序列表中选择（L） > 网络适配器 > c 厂商 > Microsoft Corporation > Remote NDIS Compatible Device”。

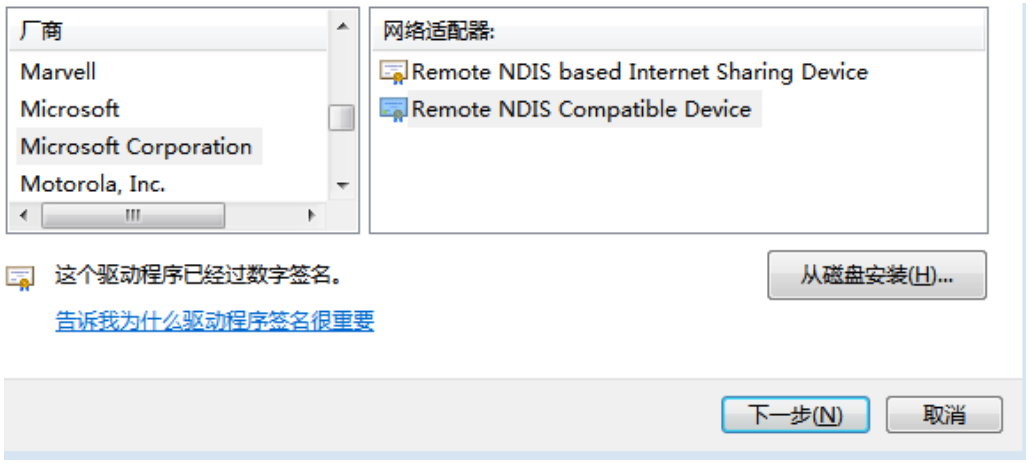


图 2-13 选择驱动软件

3、 安装完成后，启动控制器，使用 Mini USB 线缆将其连接至 PC 机，可在计算机设备管理器中看到该 USB 驱动已经成功安装。

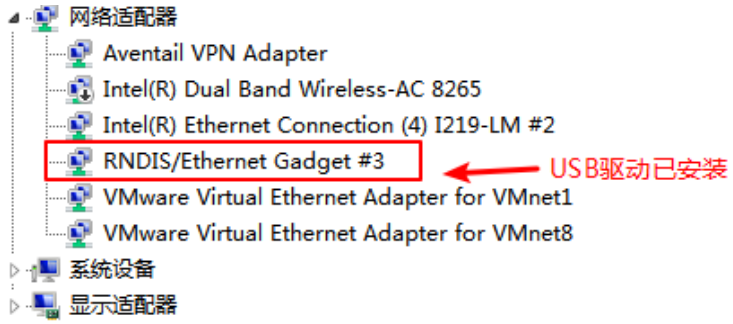


图 2-14 安装驱动

✧ USB IP 地址配置：

1、 在控制面板找到网络连接，找到 RNDIS 的本地连接，右键选择“属性 > Internet 协议版本 4”。

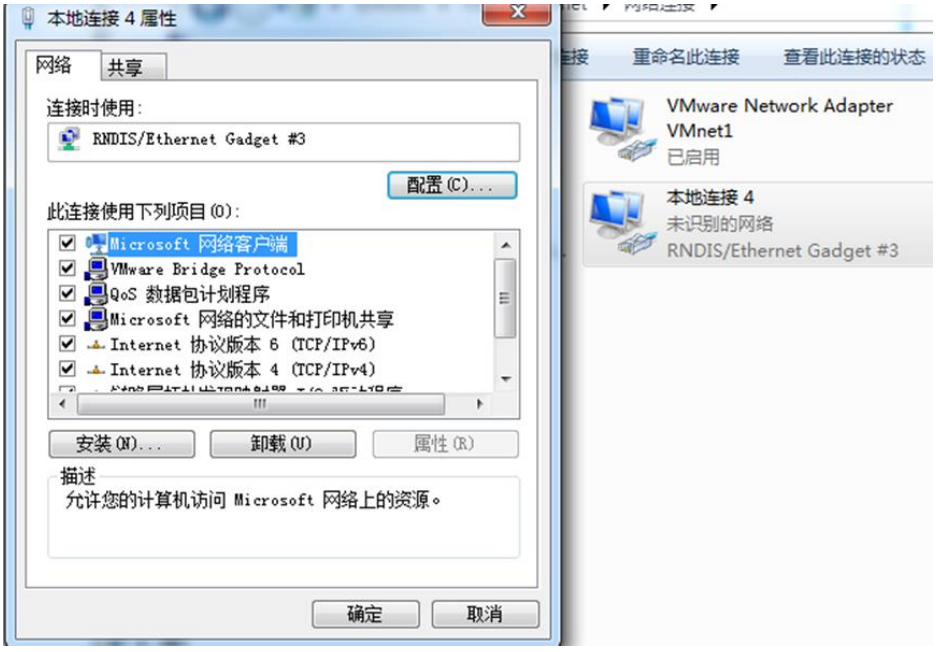


图 2-15 选择 RNDIS 的本地连接

2、 手动配置 IP 地址，网段为 192.168.2.xxx，xxx 为 1-255。点击“确定”，IP 地址配置完成。

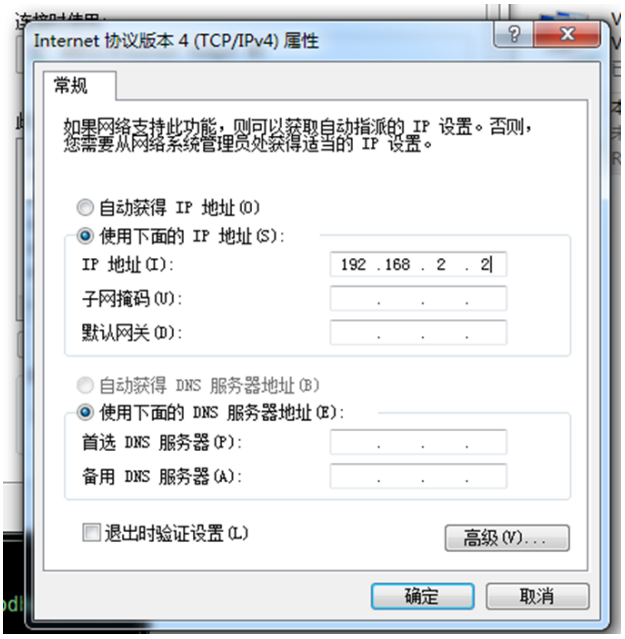


图 2-16 IP 地址配置

2.4 工程创建

2.4.1 启动编程环境

1、 双击桌面 Invtmatic Studio V1.0.2 编程软件图标，启动后的 **Invtmatic Studio** 编程环境如下图：

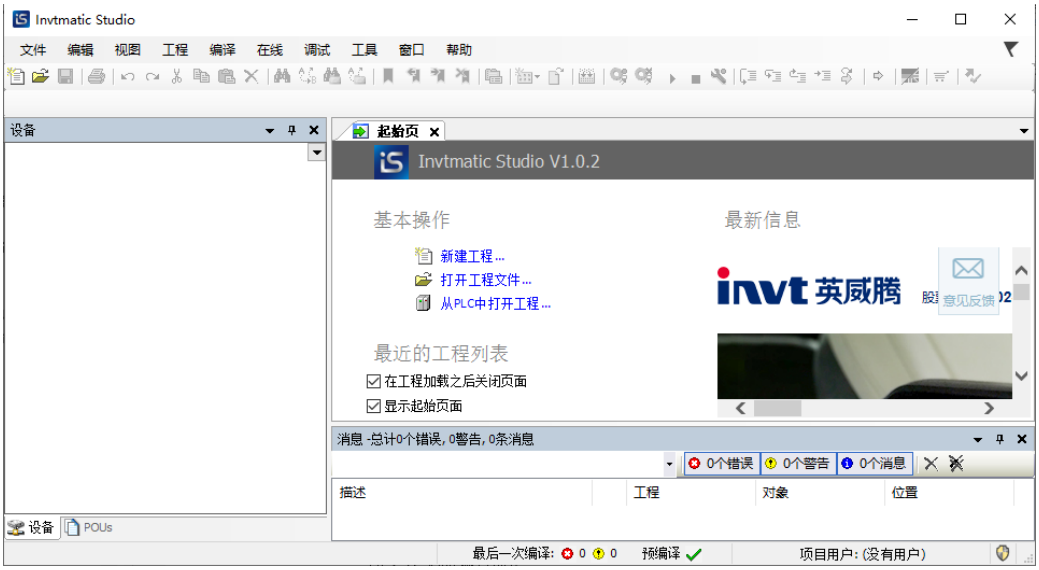


图 2-17 nvtmatic Studio 首页

2、 添加设备描述文件，选择菜单栏中的“工具 > 设备存储库”。

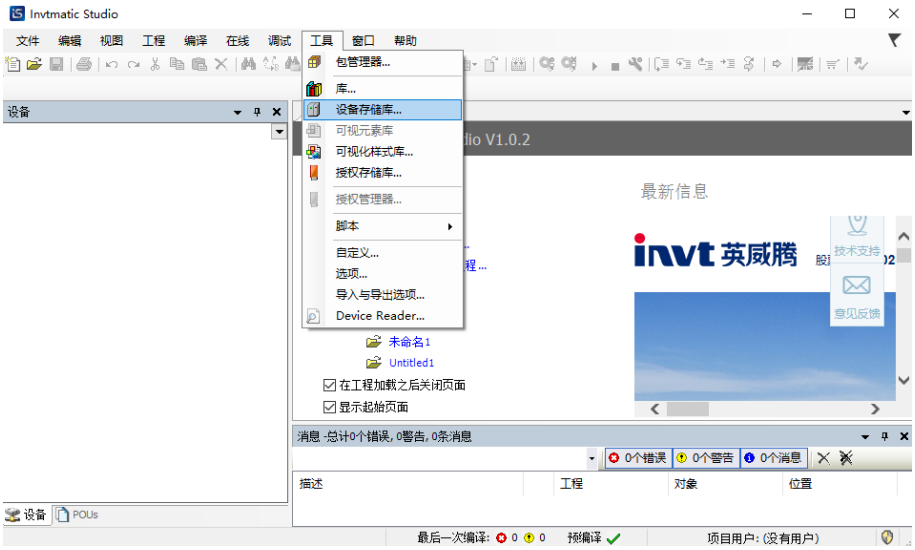


图 2-18 添加设备描述文件

3、 在弹出的“设备存储库”窗口中点击“安装”。

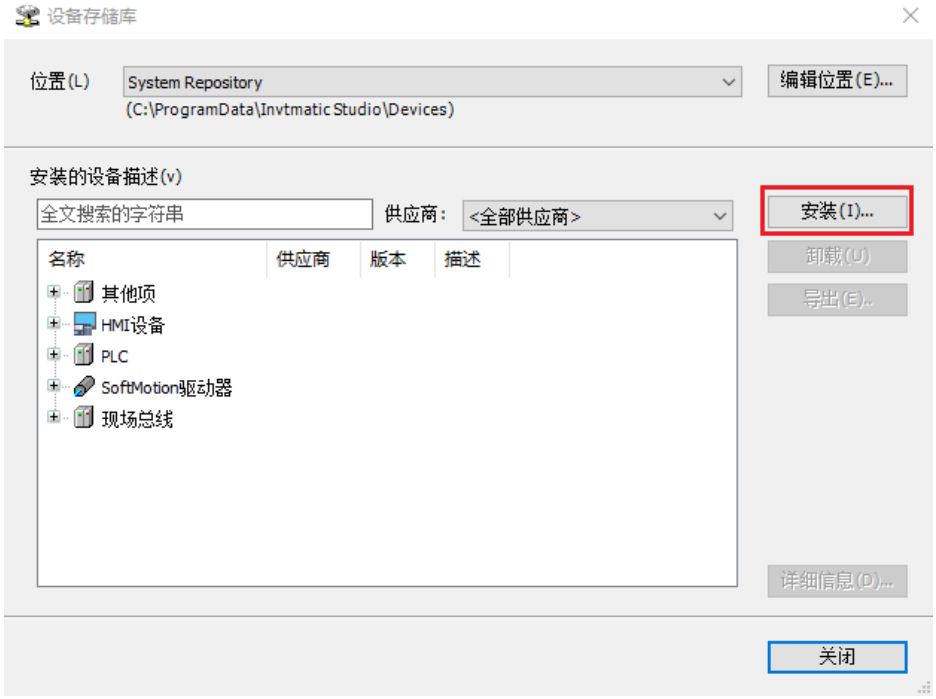


图 2-19 设备安装

4、 在弹出的窗口中找到本地文件夹中待安装的设备描述文件，将其选中并点击“打开”。

图 2-20 安装设备描述文件

注：此处可按上述步骤添加英威腾所提供的所有设备描述文件。

2.4.2 新建工程

1、 单击左上角新建工程图标或“文件 > 新建工程”，也可直接单击窗口中的“新建工程”快速建立工程，其中选择工程类型、工程模板、工程保存路径以及工程文件名称，如下图所示。

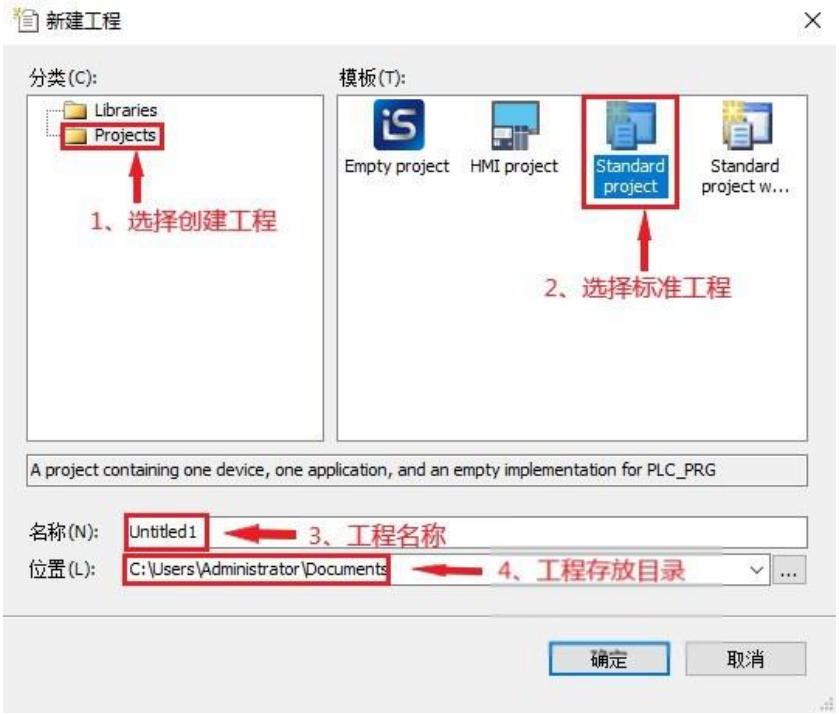


图 2-21 新建工程

2、 单击“确定”键后，进入标准工程设置界面，用户可以选择设备类型和编程语言。

图 2-22 标准工程设置界面

3、 完成上述步骤后，进入 **Invtmatic Studio** 组态配置与编程界面，双击设备中的“PLC_PRG(PRG)”编写应用程序。

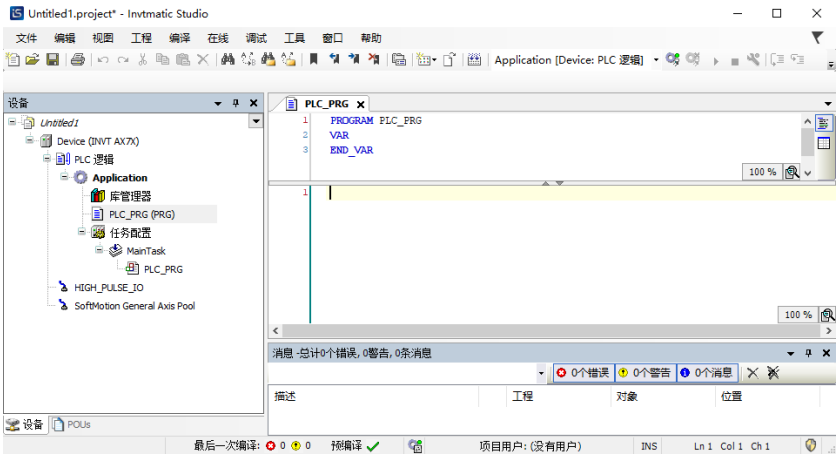


图 2-23 Invtmatic Studio 组态配置与编程界面

2.5 工程编写典型步骤

从上面的举例来看，编写具有 MC 运动控制功能的用户程序，一般需要如下几个步骤：

1、 应用系统硬件配置

根据所使用的主控制器、扩展模块、网络类型、伺服从站等，进行网络配置。

2、 用户程序编写

根据所需实现的控制功能，将运动控制用一个 POU 编写（如 POU1），将普通逻辑控制用一个 POU 编写（如 POU2）。

3、 伺服驱动器参数配置

根据硬件配置中的伺服名称，伺服的运行模式，来配置 SDO、PDO 的对象，保证用户程序的 MC 功能块与伺服之间所需的通讯对象都填在配置表中。

4、 伺服电机参数配置

要准确填写伺服电机的编码器分辨率、机械结构的传动比、轴运动范围特点等，使得控制对象的位移指令与实际位移准确对应。

5、 任务安排

按控制的实时性要求，将运动控制功能 POU1 放在 EtherCAT 任务中执行，周期可设为 4ms，优先级为 0；将普通逻辑控制 POU2 放在普通任务下，周期可设为 20ms，优先级为 16。

6、 在线调试

将 AX70 控制器通过 LAN 网络与 PC 相连接，接线无误后上电，下载调试用户程序，排除用户程序 Bug（若有条件，可将伺服驱动系统接入 AX70 控制器，再进行调试。若手头没有伺服系统，可以将伺服设为虚轴；若手头没有 AX70 控制器，还可以在 PC 上仿真调试用户程序，先排除用户程序中可能的错误，直到满意为止）。

2.6 程序编写与调试实例

在说明编程系统原理与运控程序编写方法之前，先举例介绍一个基本的伺服控制程序，以便对编程过程有一个初步的了解。

要求编写一个简单的程序，让 AX7x CPU 控制器实现如下功能：

让伺服电机正转 50 转，再反转 50 转，如此反复。

例程的编程方法与步骤如下：

1、 添加相应的设备：EtherCAT 主站、伺服驱动器、电机轴。

2、 伺服的运动控制，需要放在高实时 EtherCAT 任务周期中处理。

3、 进行相关参数设置。

4、 编写程序。

2.6.1 添加设备

1、 右键点击设备树中的 Device，选择“添加设备”，然后选择相应的 EtherCAT 主站。

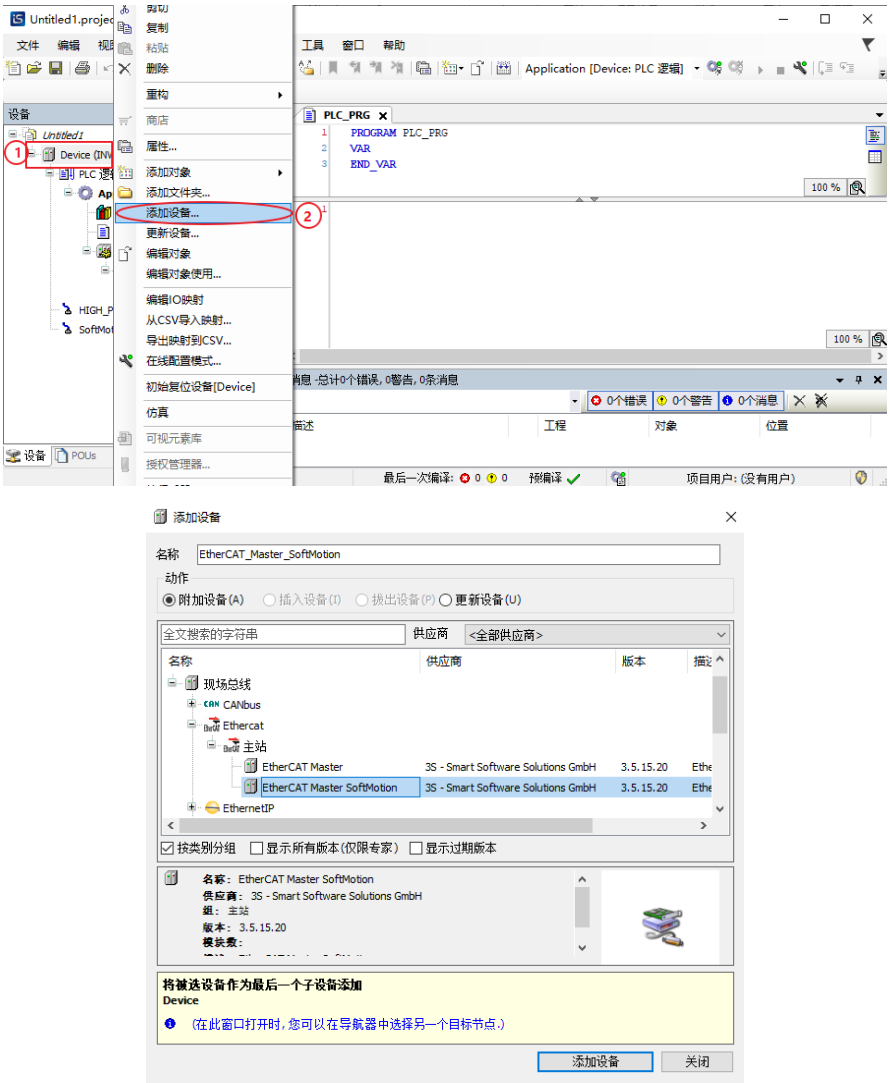


图 2-24 添加 EtherCAT 主站

2、 右键点击设备树中的 Device，选择“添加设备”，然后选择相应的 EtherCAT 从站设备。

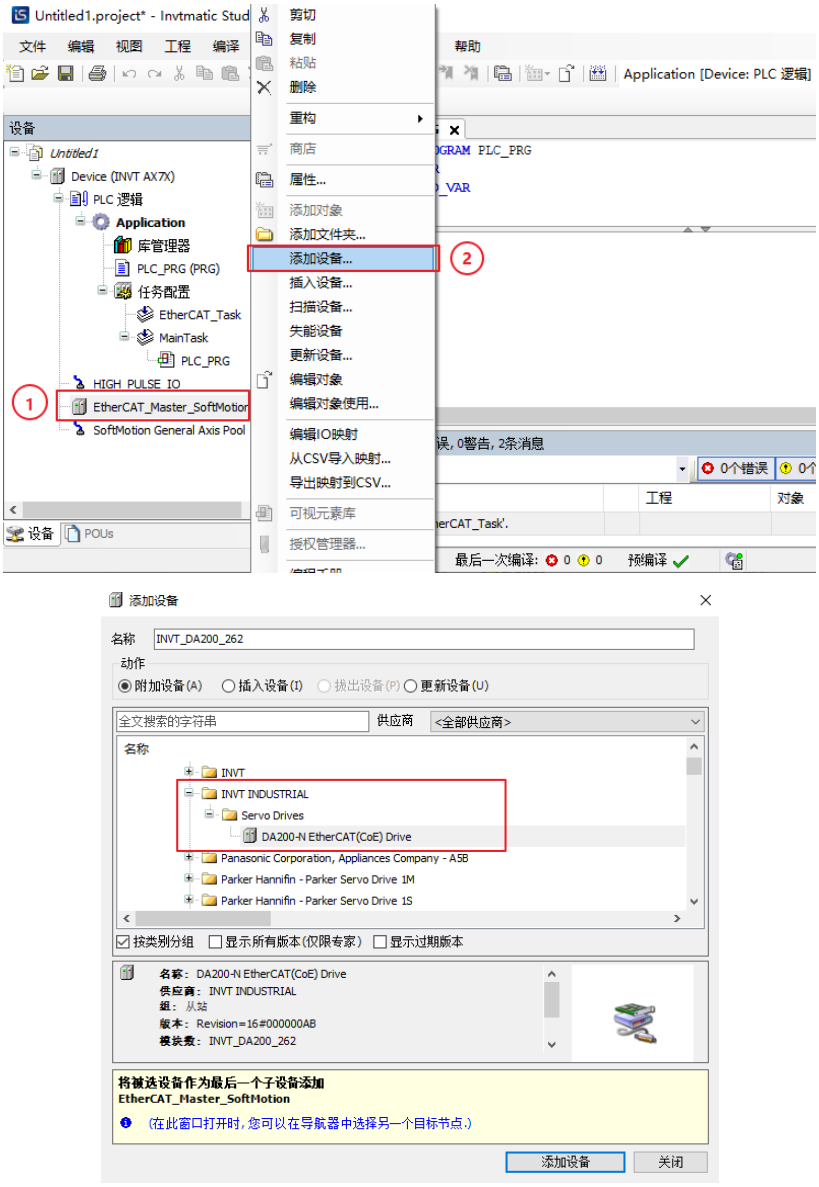


图 2-25 添加 EtherCAT 从站

3、 右键点击“添加 SoftMotion CiA402 轴”，添加伺服轴。

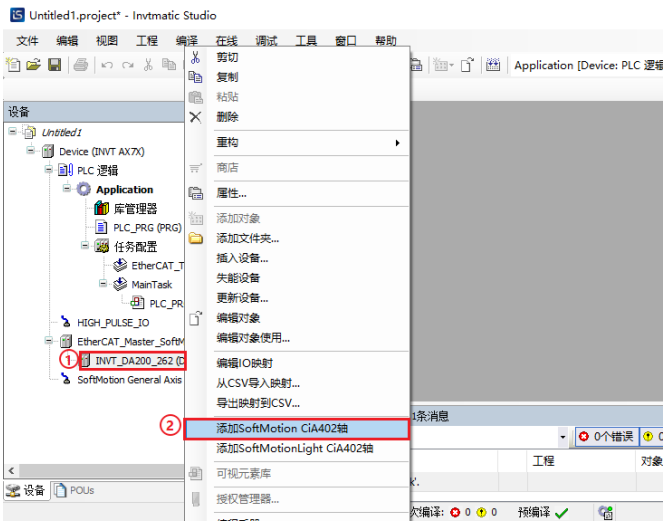


图 2-26 添加伺服轴

2.6.2 编写功能处理 POU

先看一下 Invtmatic Studio 编程环境中的默认任务配置，默认有一个 EtherCAT_Task 任务和 MainTask 任务，其中 MainTask 任务下有一个名称为 PLC_PRG 的 POU，在新建工程的同时创建了，我们希望的伺服控制程序代码就可以在 PLC_PRG 中来编写。

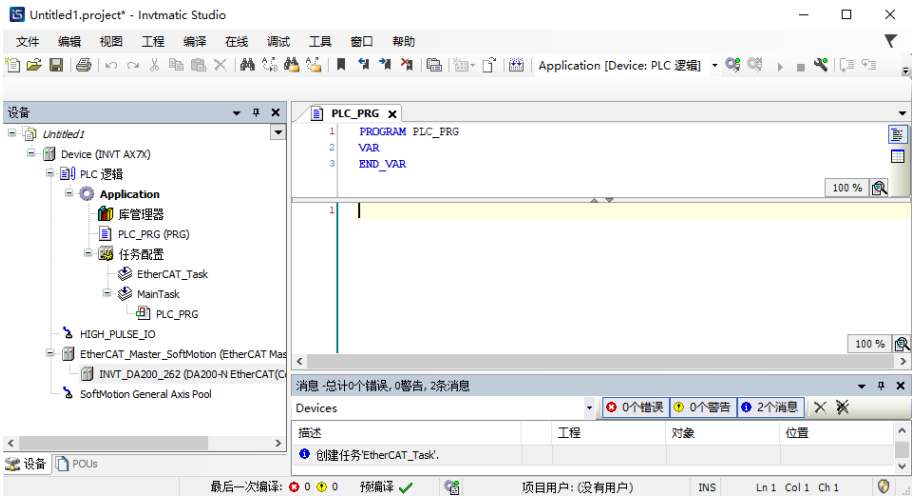


图 2-27 PLC_PRG 编程界面

2.6.3 电机参数的设置

为了精确地控制运动位置，控制器必需准确计算伺服电机的位置，根据应用系统的运转特性、行程特点，选择“轴类型与限位”，以便控制器内部对读电机编码器反馈信息进行计算，得到准确位置，避免编码器脉冲数累积溢出造成错误。

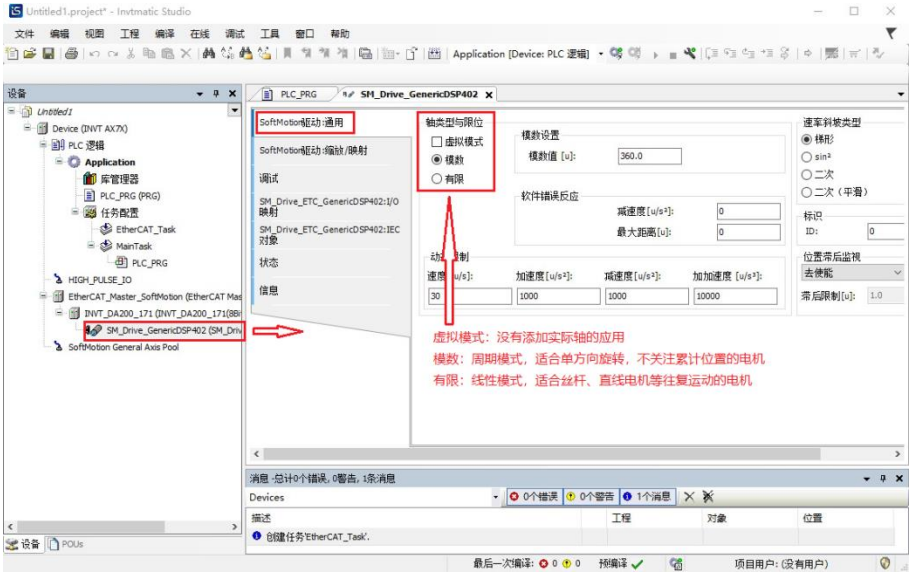


图 2-28 电机参数设置

对于丝杆类型的往复运行机构，其行程是有限的，我们往往需要知道其在丝杆行程范围内的绝对位置，此时选择“有限”比较好。

若是单方向运转类型的转轴，采用线性模式容易出现位置计数溢出，导致位置计算错误，则选择“模数”比较好。

电机的编码器参数（如分辨率），应用系统的机械减速比可能各不相同，在编程时也需要根据实际情况进行设定，如下图所示。

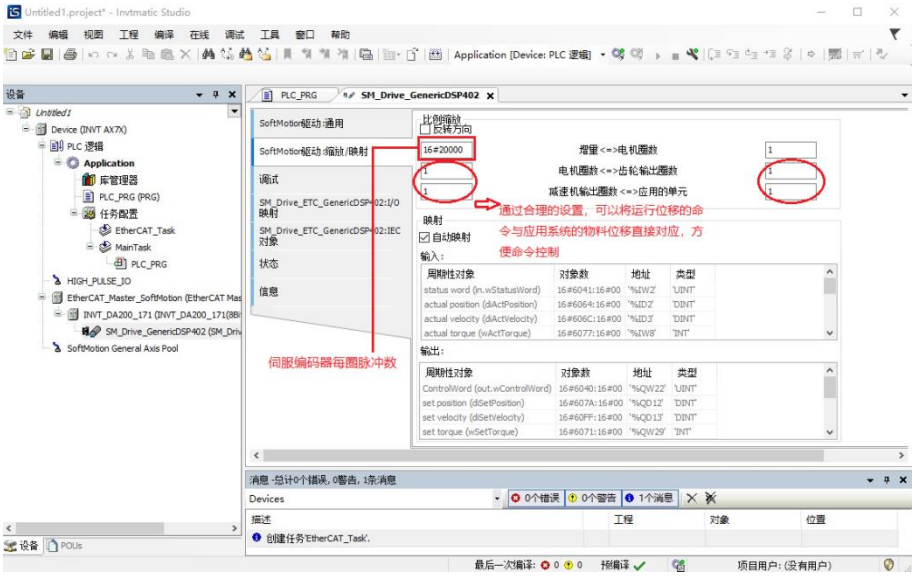


图 2-29 电机编码器参数设置

DA200 伺服配套的电机有两种典型分辨率，普通增量式编码器为 20bit 分辨率，即每圈有 1048576 个脉冲数；而绝对编码器为 23bit 分辨率，每圈 8388608 个脉冲数。实际运行时，控制器以 EtherCAT 通信方式向伺服驱动器发送所需要运行的脉冲数，来控制伺服运行，因此编码器分辨率，需要根据实际情况准确设定，如上图。例如上图中为 20bit 编码器，没有减速机的情况，当命令伺服运行 1 个单位时，伺服将会选择 1 圈（轴运动 360°）。如果将上图中圆圈内“应用的单元”参数输入栏填写为 360，当命令伺服运行 1 个单位时，伺服将会选择 1/360 圈（轴运动 1°）。依此类推，按照实际机械结构的设定对应参数（俗称电子齿轮比）之后，就可以按照应用系统的运动距离物理单位输入 distance 命令了，使控制参数直观易懂。

另外还需注意，上图中圆圈内的参数输入栏中只能输入整形数，因左右两边对应行中的参数之比为有效比例值，可以通过在左右两边对应行输入合适的整数值。例如伺服电机经过变比为 4：1 机械减速机构后，驱动导程为 6.8mm 的丝杆（即丝杆转动 1 圈，丝杆滑块运动 6.8mm）运动，设定如下图所示。

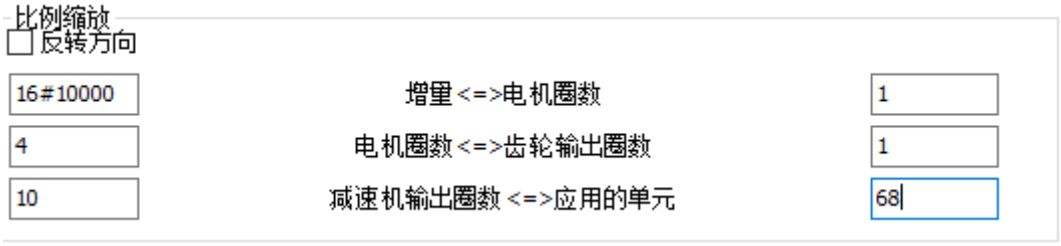


图 2-30 参数举例

图中圆圈栏中参数的量纲，后续就可以作为 MC 控制命令中 distance 的参数量纲了。上面说明的伺服驱动器、电机设置内容，在伺服轴的对应项中均需设置和核实，否则不会按所希望的特性运转。

2.6.4 电机正反转编写

对于伺服轴的运动控制，默认的同步周期为 4ms，用户可以根据实际需要进行选择，如下图所示。

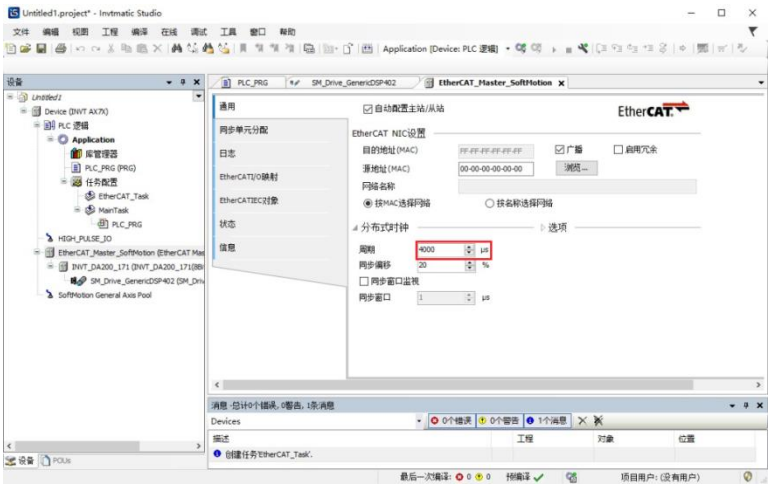


图 2-31 伺服轴运动控制周期设置

上图的程序采用 ST 语言编写，相关代码如下图所示：

```
PLC_PRG
PROGRAM PLC_PRG
VAR
    MC_Power : MC_Power;
    MC_MoveAbsolute: MC_MoveAbsolute;
    iStatus: INT:=0;
    i:UINT:=1000;  //力矩限制
END_VAR

CASE iStatus OF
0:
MC_Power(Axis:= SM_Drive_GenericDSP402, Enable:= TRUE, bRegulatorOn:= TRUE, bDriveStart:=TRUE , );
IF MC_Power.Status
THEN
    iStatus:=iStatus+1;
END_IF
1:
MC_MoveAbsolute(Axis:=SM_Drive_GenericDSP402 , Execute:= TRUE, Position:=200 , Velocity:=5 , Acceleration:= 5, Deceleration:= 5,);
IF MC_MoveAbsolute.Done
THEN
MC_MoveAbsolute(Axis:=SM_Drive_GenericDSP402 , Execute:= FALSE,);
iStatus:=iStatus+1;
END_IF
2:
MC_MoveAbsolute(Axis:=SM_Drive_GenericDSP402 , Execute:= TRUE, Position:=0 , Velocity:=4, Acceleration:= 5, Deceleration:= 5,);
IF MC_MoveAbsolute.Done
THEN
 MC_MoveAbsolute(Axis:=SM_Drive_GenericDSP402 , Execute:= FALSE,);
   iStatus:=1;
END_IF
END_CASE
```

图 2-32 ST 相关代码

2.6.5 用户程序编译

若有编写错误，上图中会列出错误类型与原因，双击其中的错误描述，光标会跳转到对应的程序编辑窗口，便于修订；逐一处理后，再进行编译，直到所有编译问题排除。

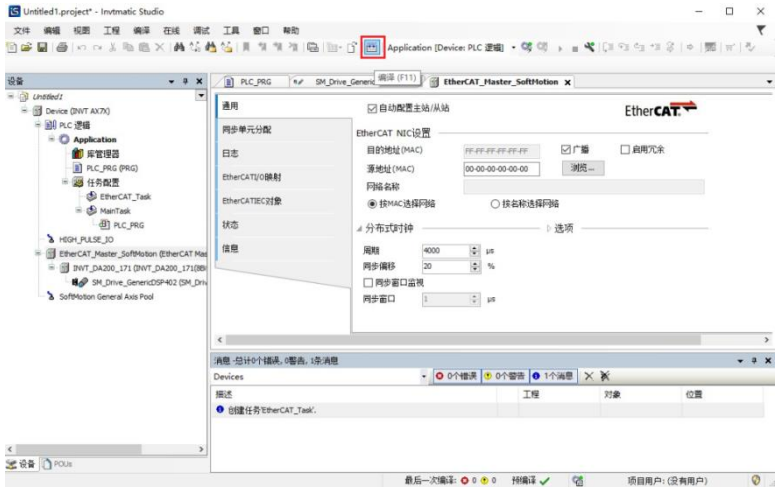


图 2-33 程序编译

最后将用户程序下载到 AX7x CPU 模块中。

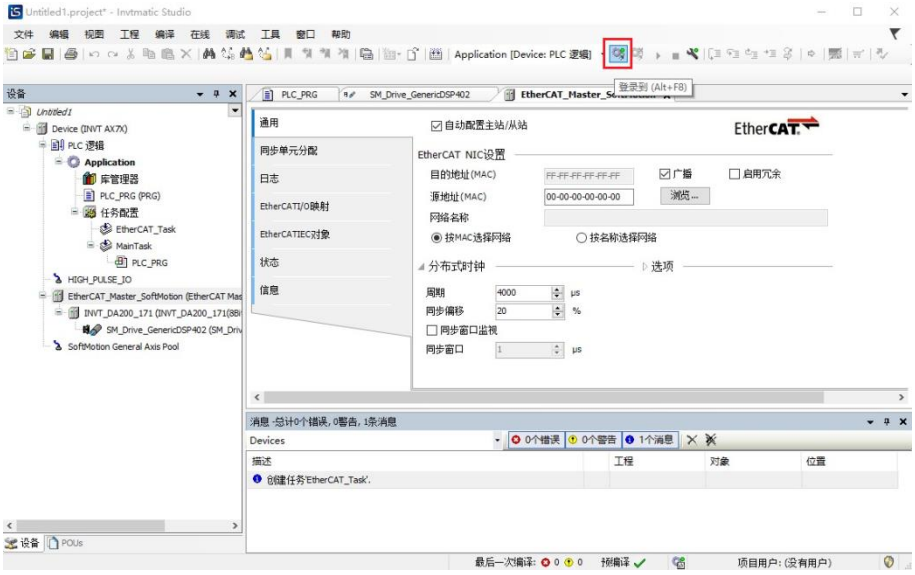


图 2-34 用户程序下载

2.6.6 监控程序运行

如图 2-34 登录到设备后，可以通过观测伺服的实际运行情况或者查看上位机伺服轴的 position 值，就可以看到程序的运行，至此，编程所需的伺服点动、触发运行 2 圈的功能都已实现，一个简单的编程过程就完成了。

第3章 网络配置

AX70 系列可编程控制器网络配置主要包含以下网络：ModbusTCP、ModbusRTU、EtherCAT 和 CANopen。

3.1 ModbusTCP

3.1.1 ModbusTCP_Master 主站

ModbusTCP 可以访问的变量数量定义如下：

- ✧ 读线圈（0x01）线圈数量 1~2000（0x7D0）
- ✧ 读离散线圈（0x02）线圈数量 1~2000（0x7D0）
- ✧ 读保持寄存器（0x03）寄存器数量 1~125（0x7D）
- ✧ 读输入寄存器（0x04）寄存器数量 1~125（0x7D）
- ✧ 写单个线圈（0x05）
- ✧ 写单个寄存器（0x06）
- ✧ 写多个线圈（0x0F）线圈数量 1~1968（0x7B0）
- ✧ 写多个寄存器（0x10）寄存器数量 1~120（0x78）

ModbusTCP_Master 主站作为 ModbusTCP_Master 功能模块的一个重要组成，其使前应先添加相应库文件：

创建 ModbusTCP_Master 主站的应用工程；添加本模块需要的库文件“CmpModbusTCP_Master_1.0.0.0.library”。

3.1.2 ModbusTCP_Slave 从站

创建 ModbusTCP_Slave 从站应用工程，添加本模块需要的库文件“ModbusTCP_Slave_1.1.0.0.library”。

ModbusTCP_Slave 从站定义了可供外部访问的存储区域，其详细区域如下表：

表 3-1 ModbusTCP_Slave 功能码

TCP 主站功能码	地址名称	范围	偏移量
01	%QX	0.0-511.7	无
05	%QX	0.0-511.7	无
02	%IX	0.0-511.7	无
04	%IW	0-511	无
03/06	%MW	0-8192	5000
03/06	%QW	0-511	无
01	%MX	0.0-8191.7	5000
05	%MX	0.0-8191.7	5000

3.2 ModbusRTU

AX70-C-1608P 支持两路 Modbus 串口通信，分别是 COM1 和 COM2，均支持标准的 ModbusRTU 协议，可独立配置为主站或从站，支持 2400、4800、9600、19200、38400、57600、115200 这 7 种波特率。

ModbusRTU 可以访问的变量数量定义如下：

- ✧ 读线圈（0x01）线圈数量 1~2000（0x7D0）
- ✧ 读离散线圈（0x02）线圈数量 1~2000（0x7D0）
- ✧ 读保持寄存器（0x03）寄存器数量 1~125（0x7D）

- ✧ 读输入寄存器（0x04）寄存器数量 1~125（0x7D）
- ✧ 写单个线圈（0x05）
- ✧ 写单个寄存器（0x06）
- ✧ 写多个线圈（0x0F）线圈数量 1~1968（0x7B0）
- ✧ 写多个寄存器（0x10）寄存器数量 1~120（0x78）

3.2.1 ModbusRTU_Master 主站

创建 modbusRTU_master 主站应用工程，AX70 包含两个串口，添加 ModbusRTU_Master 主站模块需要对应其相应的库文件“ModbusRTU_Master1_1.0.0.0.library”、“ModbusRTU_Master 2_1.0.0.0.library”，（ModbusRTU_Master1_1.0.0.0.library 对应硬件 COM1 口，ModbusRTU_Master2_1.0.0.0.library 对应硬件 COM2 口）。

3.2.2 ModbusRTU_Slave 从站

创建 ModbusRTU_slave 从站应用工程，AX70 包含两个串口，添加 ModbusRTU_Slave 从站模块需要对应其相应的库文件“ModbusRTU_Slave1_1.1.0.0.library”、“ModbusRTU_Slave2_1.1.0.0.library”（ModbusRTU_Slave1_1.1.0.0.library 对应硬件 COM1 口，ModbusRTU_Slave2_1.1.0.0.library 对应硬件 COM2 口）。

ModbusRTU_Slave 从站定义了可供外部访问的存储区域，其详细区域如下表：

表 3-2 ModbusRTU_Slave 功能码

RTU 主站功能码	地址名称	范围	偏移量
01	%QX	0.0-511.7	无
05	%QX	0.0-511.7	无
02	%IX	0.0-511.7	无
04	%IW	0-511	无
03/06	%MW	0-8192	5000
03/06	%QW	0-511	无
01	%MX	0.0-8191.7	5000
05	%MX	0.0-8191.7	5000

3.3 EtherCAT 主站

有关 EtherCAT 主站相应参数配置可请参见 Invtmatic Studio 相关帮助文档的内容介绍，此处以 EtherCAT 主站连接 DA200 伺服驱动器从站使用为案例，以供参考使用。

（1）创建 DA200 伺服应用工程

添加本模块需要的库文件“INVT_DA200_171.devdesc.xml”。

注意：

1、在创建 EtherCAT Master SoftMotion 工程时其任务优先级推荐使用最高优先级。

2、同时同步周期和任务周期此处推荐保持一致设置 4ms 或者以上。

3、创建 EtherCAT Master SoftMotion 推荐使用单独任务，例如 I/O、模拟量输入输出、modbus 通信等任务和 EtherCAT Master SoftMotion 任务分开）。

（2）选中设备树中的运动控制器设备描述文件，单击鼠标右键，添加 EtherCAT Master SoftMotion 具体操作流程如下图所示。

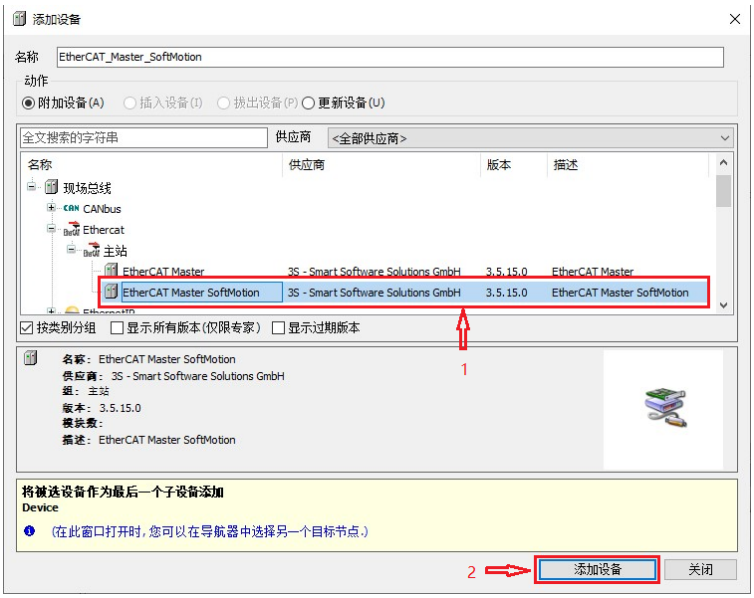


图 3-1 添加 EtherCAT 运动控制主站过程

（3）选中设备树中的 EtherCAT_Master_SoftMotion，单击鼠标右键添加 INVT DA200 伺服驱动器具体操作流程如下图所示。

图 3-2 DA200 伺服驱动器添加步骤

（4）选中设备树中的 INVT_DA200_171，单击鼠标右键添加电机轴（此处选用 SoftMotion 的 CiA 402 轴）。添加调用程序如下图所示。

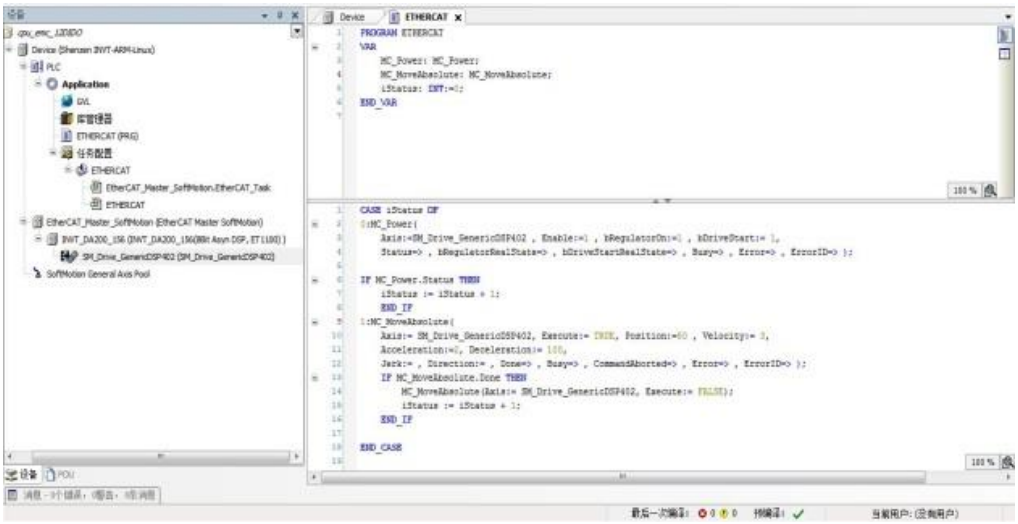


图 3-3 DA200 伺服驱动应用范例

3.4 CANopen

CANopen 是一种架构在控制局域网路（Controller Area Network, CAN）上的基于 CAL 协议扩展的高层通讯协定，包括通讯子协议及设备子协议。

通讯模型定义了 4 种报文（通讯对象）：

管理报文

层管理，网络管理和 ID 分配服务：如初始化，配置和网络管理（包括：节点保护）。

服务和协议符合 CAL 中的 LMT，NMT 和 DBT 服务部分。这些服务都是基于主从通讯模式：在 CAN 网络中，只能有一个 LMT，NMT 或 DBT 主节点以及一个或多个从节点。

服务数据对象 SDO（Service Data）

通过使用索引和子索引（在 CAN 报文的前几个字节），SDO 使客户机能够访问设备（服务器）对象字典中的项（对象）。

SDO 通过 CAL 中多元域的 CMS 对象来实现，允许传送任何长度的数据（当数据超过 4 个字节时分拆成几个报文）。

协议是确认服务类型：为每个消息生成一个应答（一个 SDO 需要两个 ID）。SDO 请求和应答报文总是包含 8 个字节（没有意义的数据长度在第一个字节中表示，第一个字节携带协议信息）。SDO 通讯有较多的协议规定。

过程数据对象 PDO（Process Data Object）

用来传输实时数据，数据从一个创建者传到一个或多个接收者。数据传送限制在 1 到 8 个字节（例如，一个 PDO 可以传输最多 64 个数字 I/O 值，或者 4 个 16 位的 AD 值）。

PDO 通讯没有协议规定。PDO 数据内容只由它的 CAN ID 定义，假定创建者和接收者知道这个 PDO 的数据内容。

每个 PDO 在对象字典中用 2 个对象描述：

1）PDO 通讯参数：包含哪个 COB-ID 将被 PDO 使用，传输类型，禁止时间和定时器周期。

2）PDO 映射参数：包含一个对象字典中对象的列表，这些对象映射到 PDO 里，包括它们的数据长度（in bits）。创建者和接收者必须知道这个映射，以解释 PDO 内容。

PDO 消息的内容是预定义的（或者在网络启动时配置的）：

映射应用对象到 PDO 中是在设备对象字典中描述的。如果设备（创建者和接收者）支持可变 PDO 映射，那么使用 SDO 报文可以配置 PDO 映射参数。

PDO 可以有多种传送方式：

1）同步（通过接收 SYNC 对象实现同步）

非周期：由远程帧预触发传送，或者由设备子协议中规定的对象特定事件预触发传送。

周期：传送在每 1 到 240 个 SYNC 消息后触发。

2）异步

由远程帧触发传送。

由设备子协议中规定的对象特定事件触发传送。

预定义报文或者特殊功能对象

同步（SYNC）

时间标记对象（Time Stamp）

紧急事件（Emergency）

节点保护（Node guarding）

3.4.1 CANopen 主站配置

3.4.1.1 主站的使用流程

安装相应的 CANopen 从站设备

相关的 CANopen 从站设备描述文件必须首先被安装到系统中，这里提到的设备描述文件可以是*.devdesc.xml 文件或者是制造商专用的 EDS（电子数据表）文件。

在设备树种添加“CAN 总线”

CANopen 的基本节点（在 CAN 总线配置树种最上层条目）必须是 CAN 总线对象。一个 CAN 总线可以插到 AX70-C-1608P 设备节点下方，添加 CAN 总线后的设备树结构图如下：

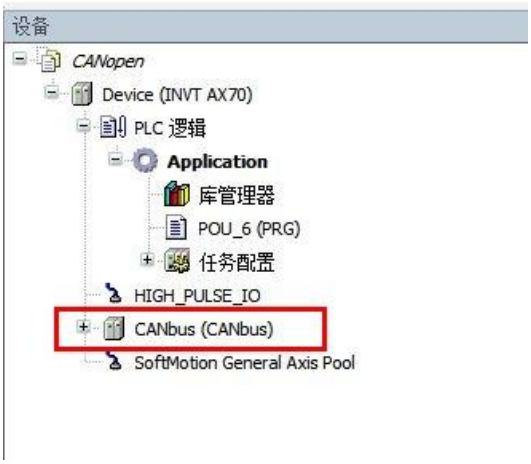


图 3-4 添加 CAN 总线的设备树结构图

3.4.1.2 添加 CANopen 管理设备

在 CAN 总线下方添加“CANopen 管理”设备，本设备可以作为一个 CANopen 主站，添加后的设备树结构图如下：

图 3-5 添加 CANopen 主站的设备树图

3.4.1.3 添加 CANopen 从站

此处以我司 TC-TX105 CANopen 通讯卡为例，在已完成本通讯卡 EDS 文件添加的基础上，在 CANopen Manager 下添加本从站通讯卡，添加效果图如下：

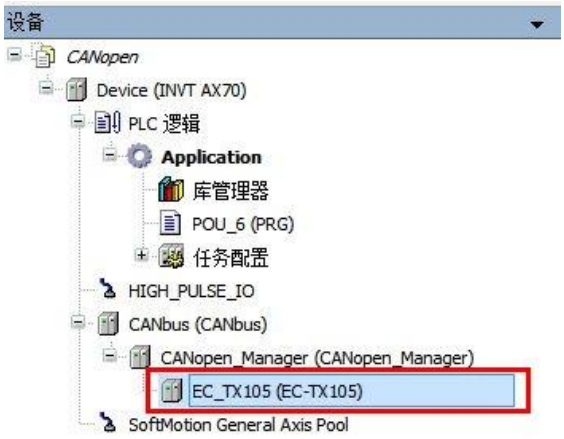


图 3-6 添加 CANopen 从站的设备树图

至此完成 CANopen 主站的软件组态工装。

3.4.2 CANopen 主站相关参数配置

首先配置 CAN 总线“网络”和“波特率”。

“网络”：通过 CAN 总线连接的 CAN 网络数量，范围 0~100。

“波特率”：总线上用于传输的波特率，可以设置以下的波特率：10000、20000、50000、100000、125000、250000、500000、800000 以及 1000000。

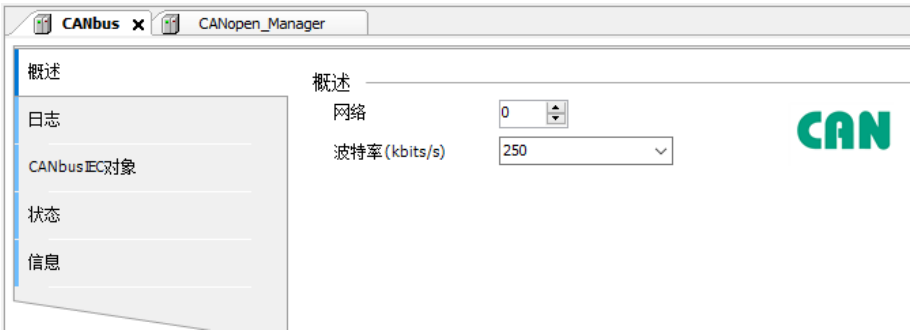


图 3-7 CAN 总线参数配置

“CANopen 管理”位于 CANbus 节点下的一个节点，用于通过内部函数支持 CANbus 配置，一般作为 CAN 总线的主站，其部分配置参数如下图所示。

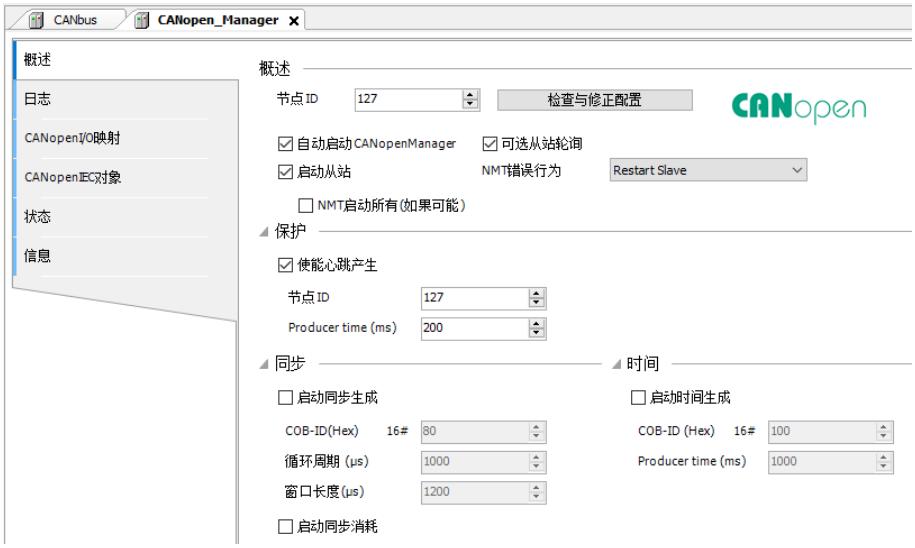


图 3-8 CANopen 主站参数配置

“节点 ID”：节点 ID 提供 CANopen 管理器能一一对应的组数对模块，ID 值 1~127（必须为十进制整数）。

“保护”：心跳方式是一种传统的保护机制，不同于节点保护功能，此功能可以被主站以及从站模块进行处理，通常情况下配置主站发送心跳到从站设备。

“激活心跳产生”：如果这个选项被激活，主站将会根据内部定义的“心跳时间”连续的发送心跳。如果添加一个新的从站心跳功能，他们的心跳动作将会自动被激活并进行配置，也就是说，节点-ID 在管理配置中会自动被设置，同时心跳间隔会自动被乘因子 1,2。如果 CANopen 管理中的心跳创建没有被激活，那么从站中将会激活节点保护 (具有生命时间因子 10 以及一个 100ms 保护时间)。

“节点 ID”：总线上心跳产生 (1 - 127) 的唯一标识符。

“产生时间 (ms)”：以毫秒定义内部心跳时间。

第4章 模块配置

4.1 CPU 模块

AX70 运动控制器实时时间和 IP 地址配置。

（1）创建 PlcCfg 工程

添加本模块需要的库文件“CmpPlcCfg_1.0.0.2.library”，创建标准。

（2）变量定义及使用

表 4-1 变量定义

变量名称	类型		功能	注释
setEnable	INPUT	BOOL	时间设置功能激活	0：非激活 1：激活
getEnable		BOOL	时间读取功能激活	0：非激活 1：激活
inTime		ARRAY OF UINT	待输入设置时间时分秒	例：2018 12 26
inDate		ARRAY OF UINT	待输入设置时间年月日	例：14 48 56
rEnable		BOOL	IP 设置功能激活	0：非激活 1：激活
wEnable		BOOL	IP 读取功能激活	0：非激活 1：激活
new_IP		STRING	设置新的 IP	例：’192.168.1.16’
new_netmask		STRING	设置新的子网掩码	例：’255. 255. 255.0’
setDone	OUTPUT	BOOL	时间设置完成标志	0：指令正在执行 1：执行指令完成
getDone		BOOL	时间获取完成标志	0：指令正在执行 1：执行指令完成
PLC		INT	错误标志	见 PlcCfg 错误码表
ErrorID		INT	错误码	见 PlcCfg 错误码表
outTime		ARRAY OF UINT	读取本机的时分秒信息	例：14 48 56
outDate		ARRAY OF UINT	读取本机的年月日信息	例：2018 12 26
Done		BOOL	完成标志	0：指令正在执行 1：执行指令完成
read_IP		STRING	读取的 IP	例：’192.168.1.16’
read_netmask		STRING	读取的子网掩码	例：’ 255. 255. 255.0’

表 4-2 AX70 本机时间配置

设置项	功能	示例
setEnable	时间设置功能激活	0：非激活 1：激活
getEnable	时间读取功能激活	0：非激活 1：激活
inTime	待输入设置时间年月日	例：2018 12 16
inDate	待输入设置时间时分秒	例：14 48 56

按照时间格式 inTime 和 inDate 中的时间数组 inTime[0]为时，inTime[1]为分，inTime[2]为秒，inDate[0]为年，inDate [1]为月，inDate [2]为日输入要设置的时间（需要全部输入，不能为空），完成设置时间输入后，使能 setEnable 将上述时间置为

AX70 当前时间。

使能 getEnable 后可获得 AX70 的实时时间，时间显示在 outTime 和 outDate 数组中。

表 4-3 AX70 本机 IP 配置

设置项	功能	示例
rEnable	IP 设置功能激活	0：非激活 1：激活
wEnable	IP 读取功能激活	0：非激活 1：激活
new_IP	设置新的 IP	例：’192.168.1.16’
new_netmask	设置新的子网掩码	例：’ 255. 255. 255.0’

按照 IP 和子网掩码格式输入要设置的内容，完成设置时间输入后，使能 wEnable 将上述 IP 或子网掩码置为 AX70 当前的 IP 或子网掩码。

注意：USB 虚拟网口与 EtherNET 网口相互独立，当用户使用 USB 连接设备时，CmpPlcCfg_1.0.0.2.library 修改的 IP 或子网掩码依旧是 EtherNET 网口的 IP 或子网掩码。修改 IP 或子网掩码后，AX70 连接 PC 端 Invtmatic Studio 需等待一段时间。

使能 rEnable 后可获得控制器的 IP 地址以及子网掩码，二者分别显示在 read_IP 和 read_netmask 字符串中。

4.2 高速 I/O 模块

4.2.1 创建高速 I/O 模块使用工程

创建高速 I/O 模块应用，直接添加相应的应用程序代码，并在 HIGH_PULSE_IO 设备树中添加相应的变量映射。

HSIO 是 High Speed Input and Output 的英文缩写。HSIO 可以用于高速计数，高速脉冲输出，具有三种中断功能，根据需要配置中断。HSIO 包括设备描述文件 Shenzen INVT-AX70-CPU_1.x.x.x.devdesc，高速计数功能块库 CmpHSIO_C.library 和运动控制功能块库 CmpHSIO_M.library 三个部分。

HSIO 的设备描述文件用于高速 IO 各种功能的配置，包括输入输出端口功能配置、计数器配置、高速脉冲输出配置、滤波参数配置、中断配置等。

高速计数功能块库 CmpHSIO_C.library 包含计数器设置、计数值读取、锁存、预设值、脉宽测量、定时采样、计数值比较等多个功能块，通过调用这些功能块来完成计数所需的应用。

运动控制功能块库 CmpHSIO_M.library 通过专用说明详细介绍。

目前 AX70&AX71 可编程控制器集成 16 路 200kHz 脉冲输入，8 路 200kHz 脉冲输出，脉冲输出支持脉冲+方向、正/反转脉冲、正交脉冲三种模式，每路端口可以配置不同的功能。配置表如下：

输入端口	普通输入功能（默认）	计数功能	触发锁存和 Z 信号功能	正负限位零点功能	脉宽测量功能	输出端口	普通输出功能（默认）	高速脉冲输出功能	比较输出功能
	功能值为 0	功能值为 1	功能值为 2	功能值为 3	功能值为 4		功能值为 1	功能值为 2	功能值为 3
X0	普通输入	C0A/CW0		CH0N		Y0	普通输出	CH0CW/PULS0	CMP0
X1	普通输入	C0B/CWW0		CH1N		Y1	普通输出	CH0CCW/SIGN0	CMP1
X2	普通输入	C1A/CW1		CH2N		Y2	普通输出	CH1CW/PULS1	CMP2

输入端口	普通输入功能（默认）	计数功能	触发锁存和 Z 信号功能	正负限位零点功能	脉宽测量功能	输出端口	普通输出功能（默认）	高速脉冲输出功能	比较输出功能
	功能值为 0	功能值为 1	功能值为 2	功能值为 3	功能值为 4		功能值为 1	功能值为 2	功能值为 3
X3	普通输入	C1B/CWW1		CH3N		Y3	普通输出	CH1CCW/SIGN1	CMP3
X4	普通输入	C4A/CW4	C0Z	CH0P		Y4	普通输出	CH2CW/PULS2	CMP4
X5	普通输入	C4B/CWW4	C1Z	CH1P		Y5	普通输出	CH2CCW/SIGN2	CMP5
X6	普通输入	C5A/CW5	C2Z	CH2P		Y6	普通输出	CH3CW/PULS3	CMP6
X7	普通输入	C5B/CWW5	C3Z	CH3P		Y7	普通输出	CH3CCW/SIGN3	CMP7
X8	普通输入	C2A/CW2	C0T		PWC0				
X9	普通输入	C2B/CWW2	C1T		PWC1				
XA	普通输入	C3A/CW3	C2T		PWC2				
XB	普通输入	C3B/CWW3	C3T		PWC3				
XC	普通输入	C6A/CW6		CH0Z					
XD	普通输入	C6B/CWW6		CH1Z					
XE	普通输入	C7A/CW7		CH2Z					
XF	普通输入	C7B/CWW7		CH3Z					

说明：

X0-XF 是输入端口，Y0-Y7 是输出端口。

普通输入、普通输出指普通 I/O 信号，通常是开关信号。

CxA、CxB、CxZ 分别是编码器 A、B、Z 信号。

CW 指顺时针，CCW 指逆时针。

CxT 指触发、锁存功能通道，支持 4 通道 C0T~C3T。

CHxP、CHxN 指正、负限位信号，N 是负方向，P 是正方向；CHxZ 指零点信号。

PWCx 指脉宽检测信号(pulse width check)。

CHxCW 是顺时针信号、CHxCCW 是逆时针信号。

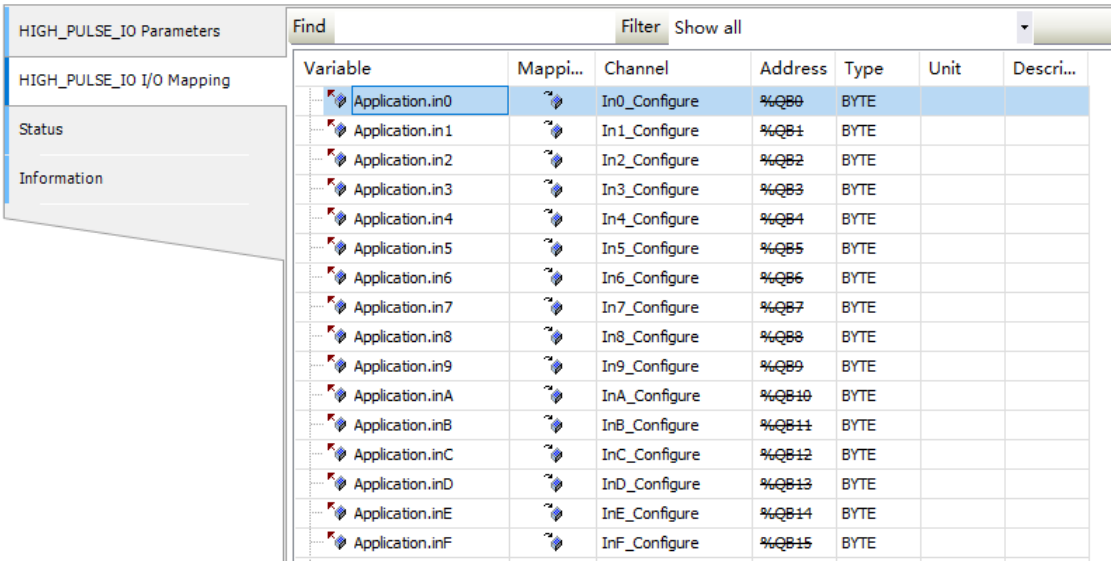
PULSx 指脉冲。

SIGNx 指脉冲方向。

CMPx 指比较输出。

4.2.2 输入端口功能说明

输入端口可以设置为 5 种功能，分别是：普通输入功能、计数功能、触发锁存和 Z 信号功能、正负限位零点功能和脉宽测量功能。如下为配置输入功能的映射表对应 Inx_Configure 的参数，x 范围为 0~F。



Variable	Mappi...	Channel	Address	Type	Unit	Descri...
Application.in0		In0_Configure	%QB0	BYTE		
Application.in1		In1_Configure	%QB1	BYTE		
Application.in2		In2_Configure	%QB2	BYTE		
Application.in3		In3_Configure	%QB3	BYTE		
Application.in4		In4_Configure	%QB4	BYTE		
Application.in5		In5_Configure	%QB5	BYTE		
Application.in6		In6_Configure	%QB6	BYTE		
Application.in7		In7_Configure	%QB7	BYTE		
Application.in8		In8_Configure	%QB8	BYTE		
Application.in9		In9_Configure	%QB9	BYTE		
Application.inA		InA_Configure	%QB10	BYTE		
Application.inB		InB_Configure	%QB11	BYTE		
Application.inC		InC_Configure	%QB12	BYTE		
Application.inD		InD_Configure	%QB13	BYTE		
Application.inE		InE_Configure	%QB14	BYTE		
Application.inF		InF_Configure	%QB15	BYTE		

4.2.2.1 普通输入功能

功能值为 0，则信号端口配置为普通输入端口，可以作为普通输入使用。

普通输入端口接线

普通输入：

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
24VDC	X0	普通输入	40	39	普通输入	X1	24VDC
			38	37			
	COM	输入公共端	36	35	输入公共端	COM	
24VDC	X2	普通输入	34	33	普通输入	X3	24VDC
			32	31			
	COM	输入公共端	30	29	输入公共端	COM	
24VDC	X4	普通输入	28	27	普通输入	X5	24VDC
			26	25			
	COM	输入公共端	24	23	输入公共端	COM	
24VDC	SS1	输入公共端	22	21	输入公共端	SS2	24VDC
	X6	普通输入	20	19	普通输入	X7	
	X8	普通输入	18	17	普通输入	X9	
	X10	普通输入	16	15	普通输入	X11	
	X12	普通输入	14	13	普通输入	X13	
	X14	普通输入	12	11	普通输入	X15	

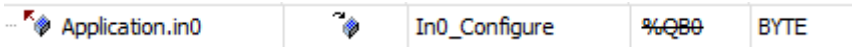
普通输入端口配置

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

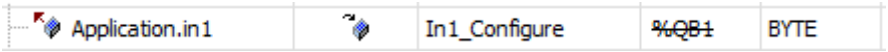
1：配置 X0 为普通输入端口

in0:=0；



2：配置 X1 为普通输入端口

in1:=0；



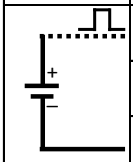
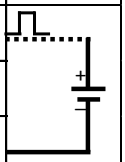
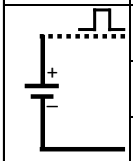
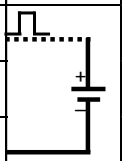
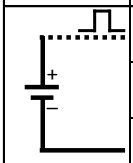
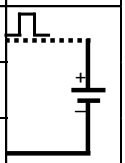
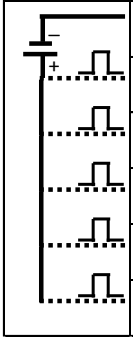
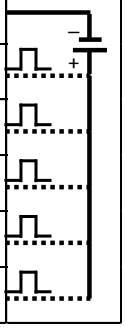
4.2.2.2 计数功能

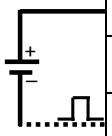
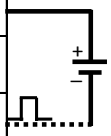
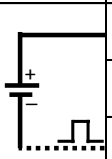
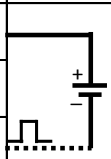
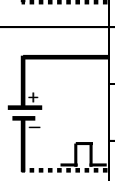
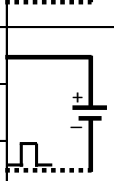
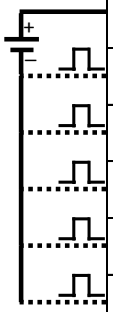
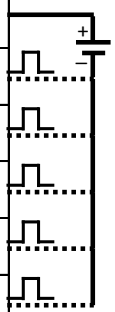
功能值为 1，则信号端口配置为计数功能，16 个输入端口均可作为计数输入。

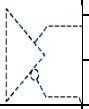
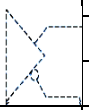
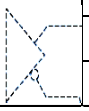
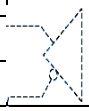
计数功能模块可以对输入脉冲进行计数和计算，可实现位置、速度、频率等检测。输入脉冲最大频率 200Khz。

计数功能端口接线

计数功能（单端源型）：

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	C0A	A相脉冲输入	40	39	B相脉冲输入	C0B	
			38	37			
	COM	输入公共端	36	35	输入公共端	COM	
	C1A	A相脉冲输入	34	33	B相脉冲输入	C1B	
			32	31			
	COM	输入公共端	30	29	输入公共端	COM	
	C4A	A相脉冲输入	28	27	B相脉冲输入	C4B	
			26	25			
	COM	输入公共端	24	23	输入公共端	COM	
	SS1	输入公共端	22	21	输入公共端	SS2	
	C5A	A相脉冲输入	20	19	B相脉冲输入	C5B	
	C2A	A相脉冲输入	18	17	B相脉冲输入	C2B	
	C3A	A相脉冲输入	16	15	B相脉冲输入	C3B	
	C6A	A相脉冲输入	14	13	B相脉冲输入	C6B	
	C7A	A相脉冲输入	12	11	B相脉冲输入	C7B	

计数功能（单端漏型）：							
外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	COM	输入公共端	40	39	输入公共端	COM	
			38	37			
	C0A	A相脉冲输入	36	35	B相脉冲输入	C0B	
	COM	输入公共端	34	33	输入公共端	COM	
			32	31			
	C1A	A相脉冲输入	30	29	B相脉冲输入	C1B	
	COM	输入公共端	28	27	输入公共端	COM	
			26	25			
	C4A	A相脉冲输入	24	23	B相脉冲输入	C4B	
	SS1	输入公共端	22	21	输入公共端	SS2	
	C5A	A相脉冲输入	20	19	B相脉冲输入	C5B	
	C2A	A相脉冲输入	18	17	B相脉冲输入	C2B	
	C3A	A相脉冲输入	16	15	B相脉冲输入	C3B	
	C6A	A相脉冲输入	14	13	B相脉冲输入	C6B	
	C7A	A相脉冲输入	12	11	B相脉冲输入	C7B	

计数功能（差分信号）：							
外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
			40	39			
	C0A+	A相差分+	38	37	B相差分+	C0B+	
	C0A-	A相差分-	36	35	B相差分-	C0B-	
			34	33			
	C1A+	A相差分+	32	31	B相差分+	C1B+	
	C1A-	A相差分-	30	29	B相差分-	C1B-	
			28	27			
	C4A+	A相差分+	26	25	B相差分+	C4B+	
	C4A-	A相差分-	24	23	B相差分-	C4B-	

计数端口配置

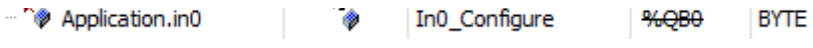
功能值配置：

先定义配置端口的变量，数据类型为 BYTE，并映射到高速脉冲映射表中。

配置例程：

1：配置 X0 为计数端口

in0:=1；



2：配置 X1 为计数端口

in1:=1；

Application.in1		In1_Configure	%QB1	BYTE

其他端口以此类推。

4.2.2.3 触发、锁存和 Z 信号功能

功能值为 2，则信号端口配置为触发、锁存和 Z 信号功能。

触发功能，可以给计数器预设计数值，触发信号上升沿有效，当该信号有效则预设值被写入计数器。计数器预设值写入通常有三种方法：软件写入、外部触发写入、比较一致触发写入，本产品触发功能是外部触发写入。

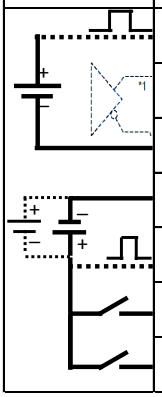
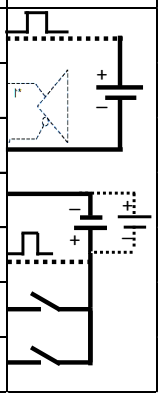
锁存功能，指瞬间锁定计数器值，供上位机读取。

触发、锁存功能支持 4 通道，C0T~C3T(对应端口为 X8，X9，XA，XB)。

Z 信号功能，Z 信号编码器每转动一圈产生一个脉冲，Z 信号功能用于 Z 清零和 Z 补偿功能。

Z 信号功能支持 4 通道， C0Z~C3Z(对应的端口为 X4，X5，X6，X7)。

触发、锁存和 Z 信号端口接线

输入功能3：（CnT配线参照普通输入；CnZ配线参照计数脉冲输入）							
外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	C0Z	Z相单端输入	28	27	Z相单端输入	C1Z	
	C0Z+	Z相差分输入	26	25	Z相差分输入	C1Z+	
	COM	输入公共端	24	23	输入公共端	COM	
	SS1	输入公共端	22	21	输入公共端	SS2	
	C2Z	Z信号输入	20	19	Z信号输入	C3Z	
	C0T	探针信号输入	18	17	探针信号输入	C1T	
	C2T	探针信号输入	16	15	探针信号输入	C3T	

触发、锁存和 Z 信号端口配置

功能值配置：先定义配置端口的变量，数据类型为 BYTE，并映射到高速脉冲映射表中。

配置例程：

1：配置 X8 为触发、锁存端口

in8:=2；

Application.in8		In8_Configure	%QB8	BYTE

2：配置 X4 为 Z 信号端口

in4:=2；

Application.in4		In4_Configure	%QB4	BYTE

4.2.2.4 正负限位零点功能

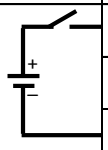
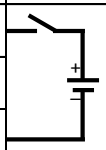
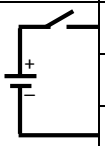
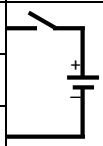
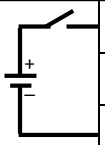
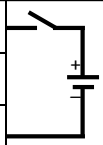
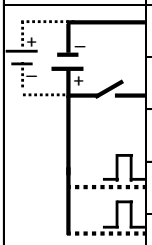
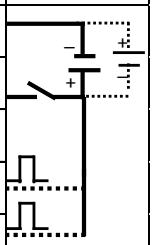
功能值为 3，则信号端口配置为为正负限位零点功能。

只有端口 X0~X7 可以作为 CHxP/CHxN 在 x 通道正负限位信号功能，x 范围 0~3。正限位起到正方向限定的作用，电机运

动至此需要停下或者反向运动；负限限位起到负方向限定作用，电机运动至此需要停下或者反向运动。

只有端口 XC~XF 可以作为 CHyZ 的 y 通道零点信号，y 范围 0~3。

正负限位零点端口接线

输入功能4：（CHnN、CHnP配线参照普通输入；CHnZ配线参照计数脉冲输入）							
外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	CH0N	负限位输入	40	39	负限位输入	CH1N	
			38	37			
	COM	输入公共端	36	35	输入公共端	COM	
	CH2N	负限位输入	34	33	负限位输入	CH3N	
			32	31			
	COM	输入公共端	30	29	输入公共端	COM	
	CH0P	正限位输入	28	27	正限位输入	CH1P	
			26	25			
	COM	输入公共端	24	23	输入公共端	COM	
	SS1	输入公共端	22	21	输入公共端	SS2	
	CH2P	正限位输入	20	19	正限位输入	CH3P	
	CH0Z	回零原点信号	14	13	回零原点信号	CH1Z	
	CH2Z	回零原点信号	12	11	回零原点信号	CH3Z	

正负限位零点端口配置

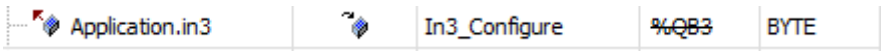
功能值配置：

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

1：配置 X3 为正负限位端口

in3:=3；



2：配置 XC 为零点端口

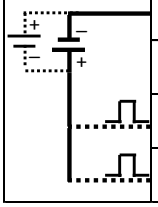
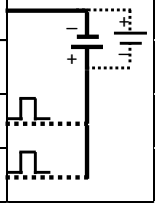
inC:=3；

4.2.2.5 脉宽测量功能

功能值为 4，则信号端口配置为脉宽测量功能。

PWCx 是脉宽测量输入通道 x，x 范围 0~3，对应的端口为 X8、X9、XA、XB。

脉宽测量端口接线

输入功能5：（PWCn配线参照计数脉冲输入）							
外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	SS1	输入公共端	22	21	输入公共端	SS2	
	PWC0	脉宽测量信号	18	17	脉宽测量信号	PWC1	
	PWC2	脉宽测量信号	16	15	脉宽测量信号	PWC3	

脉宽测量端口配置

功能值配置：

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

1：配置 X8 为脉宽测量端口

in8:=4；

Application.in8		In8_Configure	%QB8	BYTE

2：配置 X9 为脉宽测量端口

in9:=4；

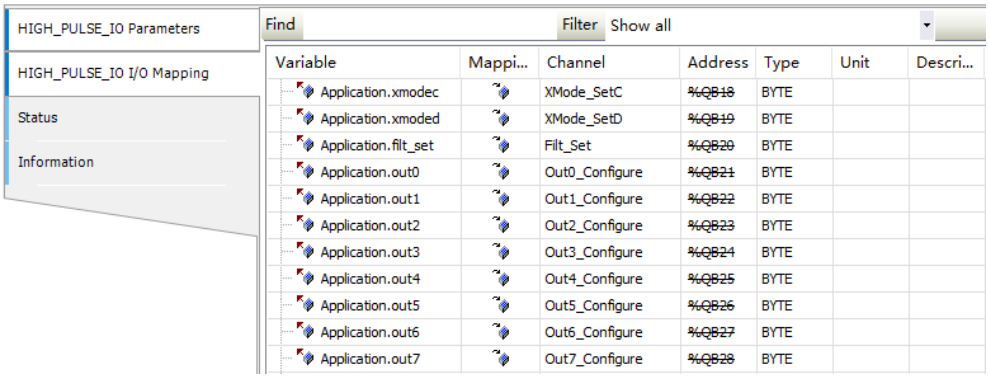
Application.in9		In9_Configure	%QB9	BYTE

4.2.3 输出端口功能说明

输出端口可以设置为 3 种功能，分别是：普通输出功能、高速脉冲输出功能和比较输出功能。

4.2.3.1 普通输出功能

功能值为 0，则信号端口配置为普通输出端口，可以作为普通输出使用。如下为配置输出功能的映射表对应 Outx_Configure 的参数，x 范围为 0~7。



Variable	Mappi...	Channel	Address	Type	Unit	Descri...
Application.xmodec		XMode_SetC	%QB18	BYTE		
Application.xmoded		XMode_SetD	%QB19	BYTE		
Application.filt_set		Filt_Set	%QB20	BYTE		
Application.out0		Out0_Configure	%QB21	BYTE		
Application.out1		Out1_Configure	%QB22	BYTE		
Application.out2		Out2_Configure	%QB23	BYTE		
Application.out3		Out3_Configure	%QB24	BYTE		
Application.out4		Out4_Configure	%QB25	BYTE		
Application.out5		Out5_Configure	%QB26	BYTE		
Application.out6		Out6_Configure	%QB27	BYTE		
Application.out7		Out7_Configure	%QB28	BYTE		

普通输出端口接线

普通输出：

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
负载	Y0	普通输出	10	9	普通输出	Y1	负载
负载	Y2	普通输出	8	7	普通输出	Y3	负载
负载	Y4	普通输出	6	5	普通输出	Y5	负载
负载	Y6	普通输出	4	3	普通输出	Y7	负载
24VDC 保险丝	COM	输出公共端	2	1	输出公共端	COM	保险丝 24VDC

输出端口共有 8 路输出信号，仅支持单端输出，信号类型为源型输出。Y0、Y2、Y4、Y6 共用公共端 COM1，Y1、Y3、Y5、Y7 共用公共端 COM2。

普通输出端口配置

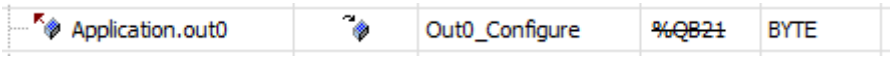
功能值参数配置：

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

1：配置 Y0 为普通输出端口

out0:=0；



2：配置 Y1 为普通输出端口

out1:=0；

4.2.3.2 高速脉冲输出功能

功能值为 1，则信号端口配置为高速脉冲输出功能，8 个输出端口均可配置为高速脉冲输出。

高速脉冲输出支持脉冲+方向、正反转脉冲、正交脉冲三种脉冲模式。

高速脉冲输出端口接线

输出脉冲：

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
驱动器0 脉冲+ 脉冲-	Plus0	脉冲输出	10	9	方向输出	Sign0	方向+ 方向- 驱动器0
驱动器1 脉冲+ 脉冲-	Plus1	脉冲输出	8	7	方向输出	Sign1	方向+ 方向- 驱动器1
驱动器2 脉冲+ 脉冲-	Plus2	脉冲输出	6	5	方向输出	Sign2	方向+ 方向- 驱动器2
驱动器3 脉冲+ 脉冲-	Plus3	脉冲输出	4	3	方向输出	Sign3	方向+ 方向- 驱动器3
	COM	输出公共端	2	1	输出公共端	COM	

24VDC

高速脉冲输出端口配置

功能值配置：

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

1：配置 Y0 为高速脉冲输出端口

out0:=1；

Application.out0		Out0_Configure	%QB21	BYTE

2：配置 Y1 为高速脉冲输出端口

out1:=1；

Application.out1		Out1_Configure	%QB22	BYTE

4.2.3.3 比较输出功能

功能值为 2，则信号端口配置为比较输出功能，共 8 通道。

比较输出是输出计数器单值比较的结果，每个计数通道都有比较输出功能。如果计数器的值与设定的比较值相等，则输出高电平，不相等则输出低电平。

比较输出端口接线

比较一致输出：

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
负载	Y0	普通输出	10	9	普通输出	Y1	负载
负载	Y2	普通输出	8	7	普通输出	Y3	负载
负载	Y4	普通输出	6	5	普通输出	Y5	负载
负载	Y6	普通输出	4	3	普通输出	Y7	负载
24VDC +	COM	输出公共端	2	1	输出公共端	COM	24VDC +

比较输出端口配置

功能值配置：

先定义配置端口的变量，并映射到高速脉冲映射表中。

配置例程：

1：配置 Y0 为比较输出端口

out0:=2；

Application.out0		Out0_Configure	%QB21	BYTE

2：配置 Y1 为比较输出端口

out1:=2；

Application.out1		Out1_Configure	%QB22	BYTE

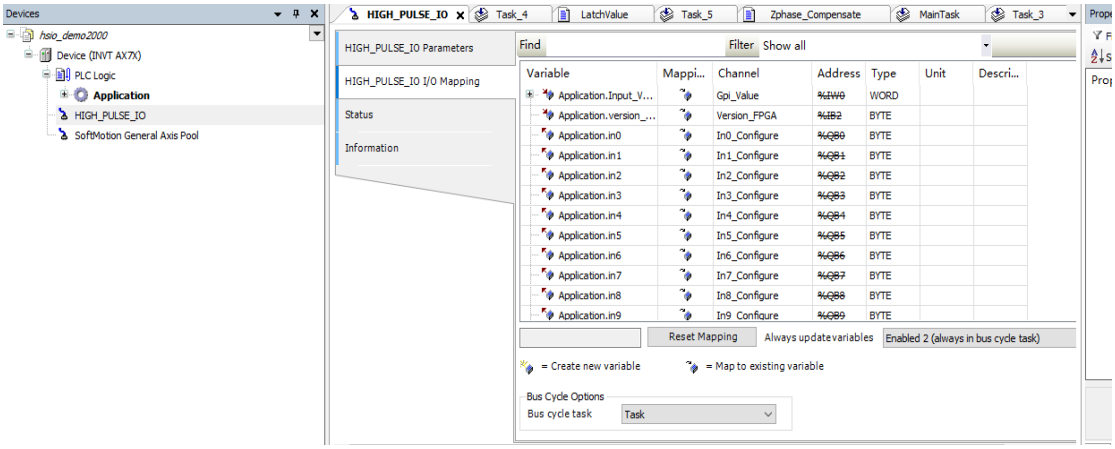
4.2.4 高速 I/O 映射表

Shenzen INVT-AX70-CPU_1.x.x.x.devdes 这个设备描述文件为 CPU 设备描述文件，包含了高速计数功能的设备描述，**主要**用于对输入输出端口进行功能配置以及中断功能的使用和配置。如下表所示：

序号	变量名	输入输出类型	数据类型	含义
1	Gpi_Value	IN	Word	16 路通用输入反馈
2	Version_FPGA	IN	BYTE	FPGA 版本号 bit6-bit7：主版本号 bit3-bit5：次版本号 bit0-bit2：修订号
3	In0_Configure	IN	BYTE	输入端口功能配置 0：标准输入功能 1：计数功能 2：触发锁存和 Z 信号功能 3：正负限位零点功能 4：脉宽测量功能
4	In1_Configure	IN	BYTE	
5	In2_Configure	IN	BYTE	
6	In3_Configure	IN	BYTE	
7	In4_Configure	IN	BYTE	
8	In5_Configure	IN	BYTE	
9	In6_Configure	IN	BYTE	
10	In7_Configure	IN	BYTE	
11	In8_Configure	IN	BYTE	
12	In9_Configure	IN	BYTE	
13	InA_Configure	IN	BYTE	
14	InB_Configure	IN	BYTE	
15	InC_Configure	IN	BYTE	
16	InD_Configure	IN	BYTE	
17	InE_Configure	IN	BYTE	
18	InF_Configure	IN	BYTE	
19	XMode_SetA	OUT	BYTE	通道 0（bit0-bit3）、通道 1(bit4-bit7)计数功能配置： 0：单脉冲 1：正交（QEP） 2：计时 3：SIGN+PULS
20	XMode_SetB	OUT	BYTE	通道 2（bit0-bit3）、3(bit4-bit7) 计数功能配置 0：单脉冲 1：正交（QEP） 2：计时 3：SIGN+PULS
21	XMode_SetC	OUT	BYTE	通道 4（bit0-bit3）、5(bit4-bit7) 计数功能配置 0：单脉冲 1：正交（QEP） 2：计时 3：SIGN+PULS
22	XMode_SetD	OUT	BYTE	通道 6（bit0-bit3）、7(bit4-bit7) 计数功能配置 0：单脉冲 1：正交（QEP） 2：计时 3：SIGN+PULS

序号	变量名	输入输出类型	数据类型	含义
23	Filt_Set	OUT	BYTE	输入信号滤波参数设置（单位：0.25us）
24	Out0_Configure	OUT	BYTE	输出端口功能配置 0：普通输出功能 1：高速脉冲输出功能 2：比较输出功能 3-255：保留
25	Out1_Configure	OUT	BYTE	
26	Out2_Configure	OUT	BYTE	
27	Out3_Configure	OUT	BYTE	
28	Out4_Configure	OUT	BYTE	
29	Out5_Configure	OUT	BYTE	
30	Out6_Configure	OUT	BYTE	
31	Out7_Configure	OUT	BYTE	
32	GPO_Set	OUT	BYTE	普通输出信号值设置 bit0-bit7
33	Run_Enable	OUT	BYTE	bit0：输出通道 0 使能(1 运行，0 不运行) bit1：输出通道 1 使能(1 运行，0 不运行) bit2：输出通道 2 使能(1 运行，0 不运行) bit3：输出通道 3 使能(1 运行，0 不运行) Bit4-bit7：保留。
34	Interrupt	OUT	BOOL	全局中断使能
35	Interrupt_Enable	OUT	DWORD	中断使能 bit0：中断 0 使能 bit1：中断 1 使能 …… bit19: 中断 19 使能
36	Interrupt_Mode	OUT	DWORD	中断模式 bit0-bit1：X0 中断模式 bit2-bit3：X1 中断模式 bit4-bit5：X2 中断模式 bit6-bit7：X3 中断模式 bit8-bit9：X4 中断模式 bit10-bit11：X5 中断模式 bit12-bit13：X6 中断模式 bit14-bit15：X7 中断模式 bit16-bit17：探针 0 中断模式 bit18-bit19：探针 1 中断模式 bit20-bit21：探针 2 中断模式 bit22-bit23：探针 3 中断模式 0：上升沿 1：下降沿 2：双沿

在 Invtmatic Studio 操作界面显示如下：



4.2.4.1 通用输入值

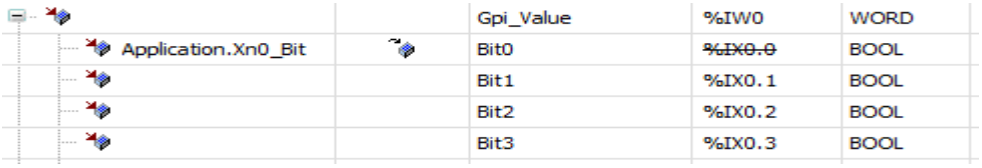
对应设备描述文件的变量为 Gpi_Value，变量的数据类型为 WORD，这个参数在输入信号设置为普通输入功能时使用。变量 Gpi_Value 的 bit 位对应的输入信号如下表所示：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0

如果需要读取普通输入信号值，可以采用变量类型为 WORD 映射或 bit 位映射方式，只能选其中一种方式进行映射。

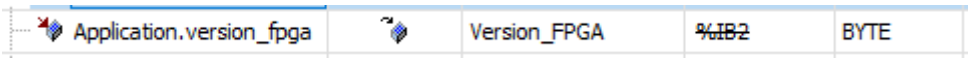
WORD 类型变量映射方式，可以同时读取 16 个输入信号值。

Bit 位映射方式，一个变量只能读取一个信号值，变量类型为 BOOL 型



4.2.4.2 版本

对应设备描述文件的变量为 Version_FPGA，读取 FPGA 的版本号，数据类型为 BYTE， bit6-bit7：主版本号；bit3-bit5：次版本号；bit0-bit2：修订号。



4.2.4.3 输入端口功能配置

配置输入端口的功能，数据类型为 BYTE。有 16 个输入端口可以配置，输入端口可以配置 5 种功能。包括标准输入功能，计数功能，触发锁存和 Z 信号功能，正负限位零点功能，脉宽测量功能。

Application.in0		In0_Configure	%QB0	BYTE
Application.in1		In1_Configure	%QB1	BYTE
Application.in2		In2_Configure	%QB2	BYTE
Application.in3		In3_Configure	%QB3	BYTE
Application.in4		In4_Configure	%QB4	BYTE
Application.in5		In5_Configure	%QB5	BYTE
Application.in6		In6_Configure	%QB6	BYTE
Application.in7		In7_Configure	%QB7	BYTE
Application.in8		In8_Configure	%QB8	BYTE
Application.in9		In9_Configure	%QB9	BYTE
Application.inA		InA_Configure	%QB10	BYTE
Application.inB		InB_Configure	%QB11	BYTE
Application.inC		InC_Configure	%QB12	BYTE
Application.inD		InD_Configure	%QB13	BYTE
Application.inE		InE_Configure	%QB14	BYTE
Application.inF		InF_Configure	%QB15	BYTE

4.2.4.4 计数模式配置

配置计数的模式，有 4 个变量进行配置，数据类型为 BYTE。每个变量可以配置 2 个通道的计数模式。总共可以配置 8 个计数器的模式。如下所示：

Application.xmodea		XMode_SetA	%QB16	BYTE
Application.xmodeb		XMode_SetB	%QB17	BYTE
Application.xmodec		XMode_SetC	%QB18	BYTE
Application.xmoded		XMode_SetD	%QB19	BYTE

用 4 个 bit 位来设置计数器模式，其值如下表：

计数模式配置值	计数模式
0	单脉冲计数
1	正交计数
2	计时计数
3	脉冲加方向计数

XMode_SetA 的 bit 位设置不同计数器的模式

7	6	5	4	3	2	1	0
计数器 1				计数器 0			

XMode_SetB 的 bit 位设置不同计数器的模式

7	6	5	4	3	2	1	0
计数器 3				计数器 2			

XMode_SetC 的 bit 位设置不同计数器的模式

7	6	5	4	3	2	1	0
计数器 5				计数器 4			

XMode_SetD 的 bit 位设置不同计数器的模式

7	6	5	4	3	2	1	0
计数器 7				计数器 6			

4.2.4.5 滤波参数

对应设备描述文件的变量为 Filt_Set，单位为 0.25us，设置输入输出信号的滤波参数，数据类型为 BYTE，最大滤波宽度 64us。调整这个参数来提高信号的抗干扰性。

如果信号干扰较强可设大一些，如果干扰较弱可设小一些。一般来讲，以高脉冲和低脉冲宽度两者中小者为基准，滤波参数设为基准宽度的 1/4 到 1/3 为宜，一般不会超过 1/2，最大不超过 64us。滤波参数过大会滤除有效脉冲，过小则可能无法有效滤除杂波。

Application.filt_set		Filt_Set	%QB20	BYTE

4.2.4.6 输出端口功能配置

配置输出端口的功能，数据类型为 BYTE。有 8 个输出端口可以配置，输出端口可以配置 3 种功能。具体可查看输出端口功能说明。

Application.out0		Out0_Configure	%QB21	BYTE
Application.out1		Out1_Configure	%QB22	BYTE
Application.out2		Out2_Configure	%QB23	BYTE
Application.out3		Out3_Configure	%QB24	BYTE
Application.out4		Out4_Configure	%QB25	BYTE
Application.out5		Out5_Configure	%QB26	BYTE
Application.out6		Out6_Configure	%QB27	BYTE
Application.out7		Out7_Configure	%QB28	BYTE

4.2.4.7 普通输出值

普通就是普通功能输出。对应设备描述文件的变量为 GPO_Set，变量的数据类型为 BYTE，这个参数在输出信号设置为标准输出功能时使用。变量 GPO_Set 的 bit 位对应的输出信号如下表所示：

7	6	5	4	3	2	1	0
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

如果需要设置普通输出信号值，可以采用变量类型为 BYTE 映射或 bit 位映射方式，只能选其中一种方式进行映射。

BYTE 类型变量映射方式，可以同时设置 8 个输出信号值。

Application.OutPut_Byte		Gpo_Set	%QB29	BYTE

Bit 位映射方式，一个变量只能设置一个信号值，变量类型为 BOOL 型

		Gpo_Set	%QB29	BYTE
Application.Yn0_Bit		Bit0	%QX29.0	BOOL
		Bit1	%QX29.1	BOOL
		Bit2	%QX29.2	BOOL

4.2.4.8 高速脉冲输出通道使能

对应设备描述文件的变量为 Run_Enable，变量的数据类型为 BYTE，这个参数在高速脉冲输出时用于通道使能。变量 Run_Enable 的 bit 位对应的通道使能，1 表示使能，0 表示不使能。如下表所示通道与 bit 位的对应关系：

7	6	5	4	3	2	1	0
保留				通道 3	通道 2	通道 1	通道 0

4.2.4.9 全局中断使能

对应设备描述文件的变量为 Interrupt，是使能所有中断的总开关，数据类型为 BOOL，总中断使能为 1，不使能为 0。

序号	变量名	输入输出类型	数据类型	含义
35	Interrupt	OUT	BOOL	全局中断使能

4.2.4.10 中断使能

对应设备描述文件的变量为 Interrupt_Enable，变量的数据类型为 DWORD。HSIO 支持 20 种中断，分别为 8 个外部输入中断、8 个计数比较一致中断和 4 个探针中断，每一个中断可以用 Interrupt_Enable 的 bit 来使能，中断与位的对应关系如下：

19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
探针中断使能位				比较中断使能位								外部中断使能位							

Bit0~bit7 分别对应外部中断 0~7

Bit8~bit15 分别对应比较中断 0~7

Bit16~bit19 分别对应探针中断 0~3

4.2.4.11 中断模式

对应设备描述文件的变量为 Interrupt_Mode，变量的数据类型为 DWORD，只有外部中断和探针中断需要设置中断模式，每种中断模式由 2 位 bit 组成。中断模式与位的对应关系如下所示：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
外部中断 7		外部中断 6		外部中断 5		外部中断 4		外部中断 3		外部中断 2		外部中断 1		外部中断 0	

23	22	21	20	19	18	17	16
探针中断 3		探针中断 2		探针中断 1		探针中断 0	

用 2 个 bit 位来设置中断模式，其值如下表：

运动模式配置值	运动模式
0	上升沿
1	下降沿
2	双沿

4.2.5 中断使用说明

HSIO 支持 20 种中断，分别为 8 个外部输入中断、8 个计数比较一致中断和 4 个探针中断，在使用中断功能时需对对应的 IO 口功能进行配置。然后使能全局中断和使能需要的中断位，如果使用外部输入中断或探针中断还需设置对应的中断的模式。

4.2.5.1 外部中断说明

外部中断对应的输入端口号为 X0~X7。这些端口需配置为普通输入端口，配置中断模式，使能中断，配置中断任务，则在中断任务可以执行相应的操作。

外部中断配置

实现外部中断功能步骤：

1：设置输入端口为标准输入功能

详见输入端口功能说明

2：设置全局中断使能

设置 Interrupt 为 true，详见设备描述文件参数说明中的全局中断使能。

序号	变量名	输入输出类型	数据类型	含义
35	Interrupt	OUT	BOOL	全局中断使能

3：设置输入端口中断使能

设置设备描述文件 Interrupt_Enable 字的以下 8 个输入端口对应位，输入端口 x 对应 Gpix 设置为 true。哪个需要中断使能哪一位。

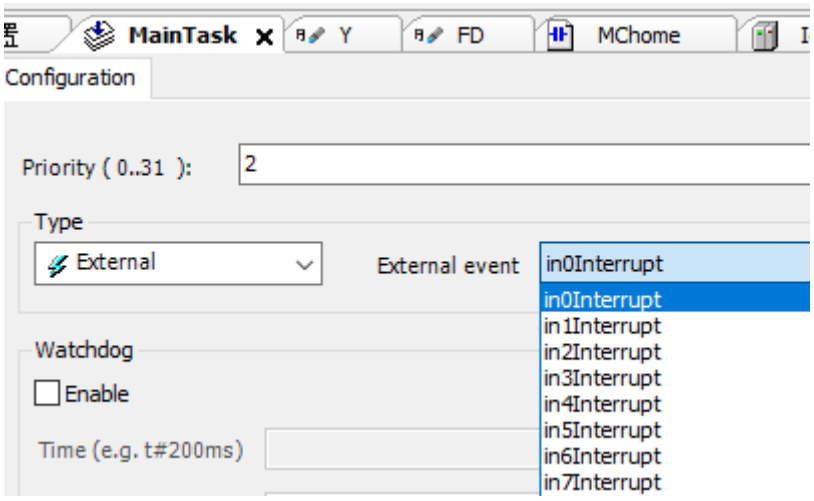
		Interrupt_Enable	%QD9	DWORD
		Gpi0	%QX36.0	BOOL
		Gpi1	%QX36.1	BOOL
		Gpi2	%QX36.2	BOOL
		Gpi3	%QX36.3	BOOL
		Gpi4	%QX36.4	BOOL
		Gpi5	%QX36.5	BOOL
		Gpi6	%QX36.6	BOOL
		Gpi7	%QX36.7	BOOL

4：设置中断模式

中断模式设置由 2 个 bit 组成，不同的中断对应不同位。详见设备描述文件参数说明中的中断模式。

5：选择中断任务

在 Invtmatic Studio 任务中选择任务类型为 External，输入端口 X0~X7 对应中断 inxInterrupt，x 范围为 0~7。



外部信号根据中断模式产生中断，调用对应的任务执行。

外部中断时序

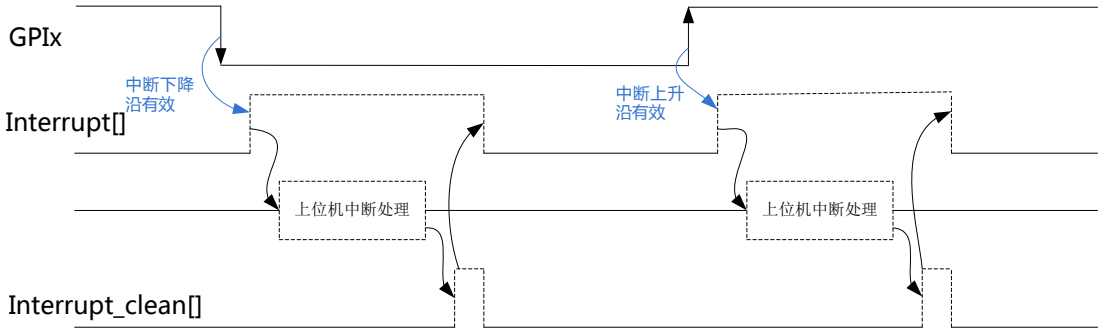


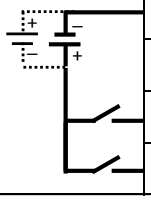
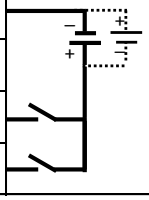
图 4-1 外部输入中断时序示意图

说明：GPIx 代表第 x 外部通用输入通道，0 =< x <=7，Interrupt[]为与 GPIx 对应的中断状态输出。Interrupt[]输出的高电平脉冲采用虚线，表示只有中断模式有效，且中断使能有效，方可输出中断。上位机中断处理过程和中断清除信号 Interrupt_clean[]是在中断 Interrupt[]输出后才出现的，所以也以虚线呈现。Interrupt_clean[]是上位机对中断 Interrupt[]响应后给出的清除信号，该信号将把 Interrupt[]清零。

4.2.5.2 探针中断说明

探针中断对应的输入端口号为 X8~XB（即 CxT，0 =< x <= 3）。需要配置输入端口信号功能为锁存功能。

探针中断接线

外部配线	端口名称	端口功能	CN5端子编号		端口功能	端口名称	外部配线
	SS1	输入公共端	22	21	输入公共端	SS2	
	C0T	探针信号输入	18	17	探针信号输入	C1T	
	C2T	探针信号输入	16	15	探针信号输入	C3T	

探针中断配置

实现外部中断功能步骤：

1：设置输入端口为锁存功能

详见输入端口功能说明

2：设置全局中断使能

设置 Interrupt 为 true，详见设备描述文件参数说明中的全局中断使能。

序号	变量名	输入输出类型	数据类型	含义
35	Interrupt	OUT	BOOL	全局中断使能

3：设置输入端口中断使能

设置设备描述文件 Interrupt_Enable 字的以下 4 个输入端口对应位，输入端口 x 对应 Trigx 设置为 true。哪个需要中断使能哪一位。

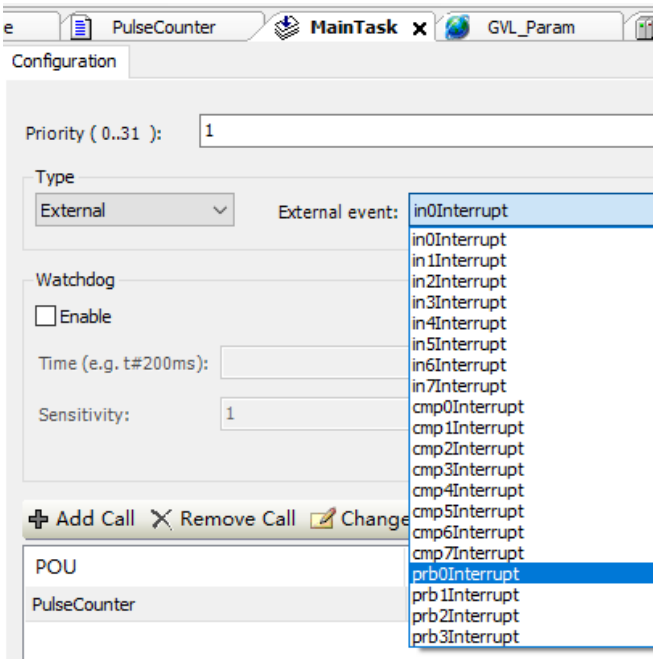
Application.P...		Trig0	~~%QX38.0~~	BOOL
Application.P...		Trig1	~~%QX38.1~~	BOOL
Application.P...		Trig2	~~%QX38.2~~	BOOL
Application.P...		Trig3	~~%QX38.3~~	BOOL

4：设置中断模式

中断模式设置由 2 个 bit 组成，不同的中断对应不同位。详见设备描述文件参数说明中的中断模式

5：选择中断任务

在 Invtmatic Studio 任务中选择任务类型为 External，输入端口 X8~XB 对应中断 prbxInterrupt，x 范围为 0~3。通过中断任务标志，在 *LatchValue_HP 功能块读取探针锁存值。*



外部信号根据中断模式产生中断，调用对应的任务执行。

探针中断时序

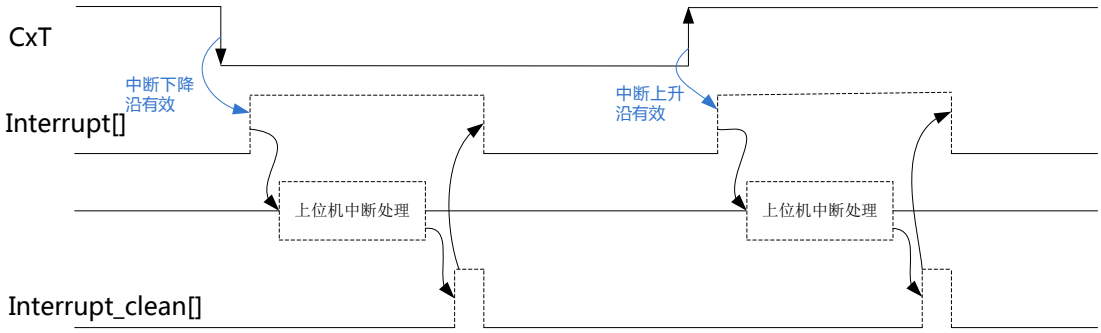


图 4-2 探针输入中断时序示意图

说明：CxT 代表第 x 探针输入通道，0 =< x <= 3，Interrupt[]为与 CxT 对应的中断状态输出。Interrupt[]输出的高电平脉冲采用虚线，表示只有中断模式有效，且中断使能有效，方可输出中断。上位机中断处理过程和中断清除信号 Interrupt_clean[]是在中断 Interrupt[]输出后才出现的，所以也以虚线呈现。Interrupt_clean[]是上位机对中断 Interrupt[]响应后给出的清除信号，该信号将把 Interrupt[]清零。

4.2.5.3 比较中断说明

比较中断分单值比较中断和多值比较中断，单值比较中断需要调用功能块 CompareSingleValue_HP 产生，多值比较中断需要调用 CompareMoreValue_HP 产生。下面分别按步骤说明产生单值中断和多值中断。

比较中断配置

- 单值比较中断：

  1：设置输入端口计数功能

详见输入端口功能说明。

2：设置全局中断使能

设置 Interrupt 为 true，详见设备描述文件参数说明中的全局中断使能。

序号	变量名	输入输出类型	数据类型	含义
35	Interrupt	OUT	BOOL	全局中断使能

3：设置输入端口中断使能

设置设备描述文件 Interrupt_Enable 字的以下 8 个输入端口对应位，输入端口 x 对应 Compx 设置为 true。哪个需要中断使能哪一位。

Variable	Mapping	Channel	Address	Type
Application.P...		Comp0	%QX37.0	BOOL
Application.P...		Comp1	%QX37.1	BOOL
Application.P...		Comp2	%QX37.2	BOOL
Application.P...		Comp3	%QX37.3	BOOL
Application.P...		Comp4	%QX37.4	BOOL
Application.P...		Comp5	%QX37.5	BOOL
Application.P...		Comp6	%QX37.6	BOOL
Application.P...		Comp7	%QX37.7	BOOL

4：设置比较中断输出

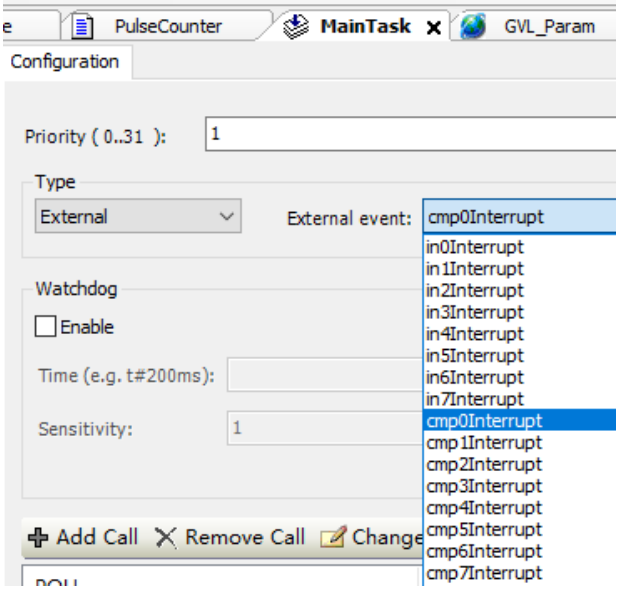
如果不需要比较中断输出，这步可以忽略。

选择需要输出的端口，在设备描述文件对应端口设置为比较输出功能，通过单值比较功能块 CompareSingleValue_HP 参数 OutChannel 选择以下 8 个通道中的任何一个，OutChanne 的值范围为 0 到 7。一个输出通道 OutChannel 值只能对应一个 CMP 通道。

输出端口	标准输出功能	高速脉冲输出功能	比较输出功能
Y0	普通 0	CH0CW/PULS0	CMP0
Y1	普通 1	CH0CCW/SIGN0	CMP1
Y2	普通 2	CH1CW/PULS1	CMP2
Y3	普通 3	CH1CCW/SIGN1	CMP3
Y4	普通 4	CH2CW/PULS2	CMP4
Y5	普通 5	CH2CCW/SIGN2	CMP5
Y6	普通 6	CH3CW/PULS3	CMP6
Y7	普通 7	CH3CCW/SIGN3	CMP7

5：选择中断任务

在 Invtmatic Studio 任务中选择任务类型为 External，cmpxInterrupt，x 范围为 0~7。



比较值相等产生中断，调用对应的任务执行。通道 x 对应 cmpxInterrupt 比较中断任务，不能随意修改。

6：调用功能块产生中断

单值比较调用功能块 CompareSingleValue_HP 产生中断，设置比较值与计数值一样就会产生中断输出。

- 多值比较中断：

1：设置输入端口计数功能

详见输入端口功能说明。

2：设置全局中断使能

设置 Interrupt 为 true，详见设备描述文件参数说明中的全局中断使能。

序号	变量名	输入输出类型	数据类型	含义
35	Interrupt	OUT	BOOL	全局中断使能

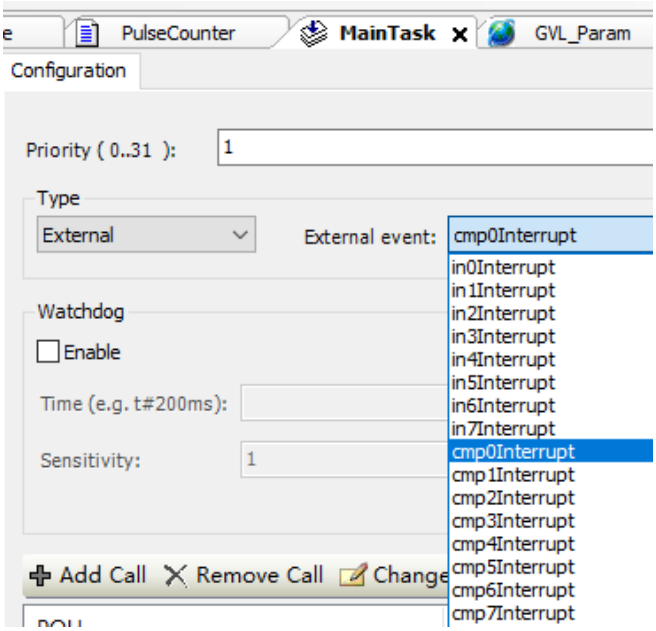
3：设置输入端口中断使能

设置设备描述文件 Interrupt_Enable 字的以下 8 个端口对应位，端口 x 对应 Compx 设置为 true。因为一个多值比较功能块可以对应多个中断，则多值比较第 1 个比较值对应 Cmp0 中断使能位，第 2 个比较值对应 Cmp1 中断使能位，以此类推第 8 个比较值对应 Cmp7 中断使能位。用户不能随意修改。

Variable	Mapping	Channel	Address	Type
Application.P...		Comp0	%QX37.0	BOOL
Application.P...		Comp1	%QX37.1	BOOL
Application.P...		Comp2	%QX37.2	BOOL
Application.P...		Comp3	%QX37.3	BOOL
Application.P...		Comp4	%QX37.4	BOOL
Application.P...		Comp5	%QX37.5	BOOL
Application.P...		Comp6	%QX37.6	BOOL
Application.P...		Comp7	%QX37.7	BOOL

4：选择中断任务

在 Invtmatic Studio 任务中选择任务类型为 External， cmpxInterrupt，x 范围为 0~7。



多值比较功能块有多个比较值，每个比较值对应的中断使能位 Compx。与中断任务 cmpxInterrupt 是一一对应的，x 范围 0~7，不能随意修改。

5：调用功能块产生中断

多值比较调用功能块 CompareMoreValue_HP 产生中断，设置比较值与计数值一样产生中断。暂时多值比较只支持 8 个比较值产生中断，即多值比较的前面 8 个值可以产生中断。

比较中断时序

- 单值比较中断

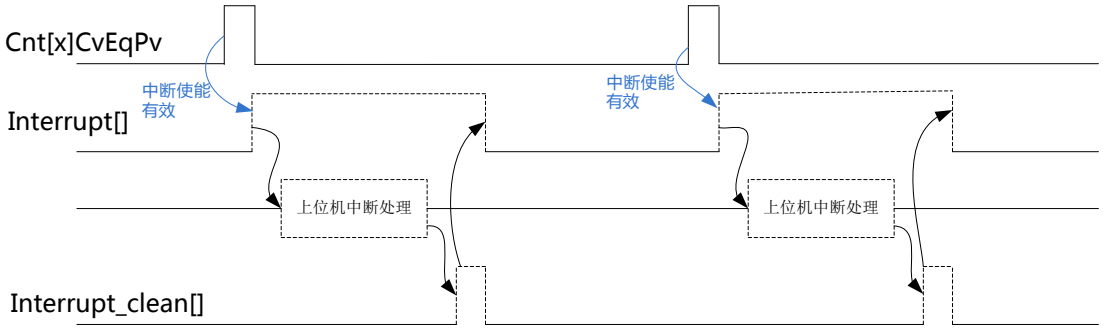


图 4-3 单值比较中断时序示意图

Cnt[x]CvEqPv 代表第 x 计数通道单值比较信号，高脉冲表示 cv 和 pv 相等，0 =< x <= 7，Interrupt[]为与 Cnt[x]CvEqPv 对应的中断状态输出。Interrupt[]输出的高电平脉冲采用虚线，表示只有中断使能有效方可输出中断。上位机中断处理过程和中断清除信号 Interrupt_clean[]是在中断 Interrupt[]输出后才出现的，所以也以虚线呈现。Interrupt_clean[]是上位机对中断 Interrupt[]响应后给出的清除信号，该信号将把 Interrupt[]清零。

- 多值比较中断

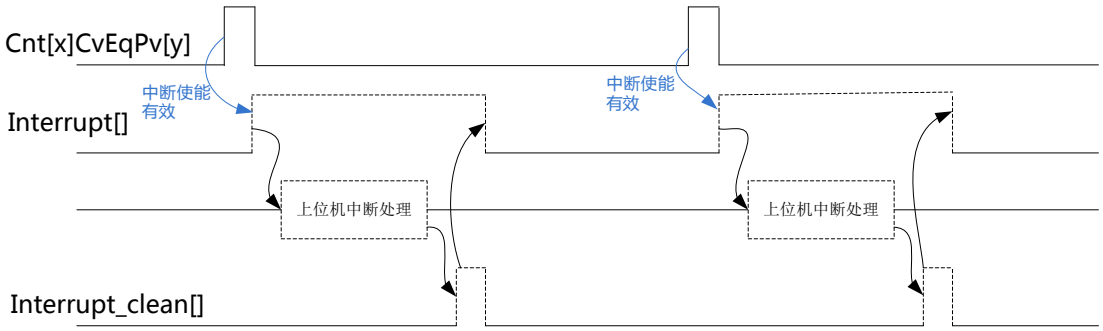


图 4-4 多值比较中断时序示意图

Cnt[x]CvEqPv[y]代表第 x 计数通道第 y 个比较值比较信号，高脉冲表示 cv 和 pv 相等，0 =< x <= 3，0 =< y <= 7，Interrupt[]为与 Cnt[x]CvEqPv[y]对应的中断状态输出。Interrupt[]输出的高电平脉冲采用虚线，表示只有中断使能有效方可输出中断。上位机中断处理过程和中断清除信号 Interrupt_clean[]是在中断 Interrupt[]输出后才出现的，所以也以虚线呈现。Interrupt_clean[]是上位机对中断 Interrupt[]响应后给出的清除信号，该信号将把 Interrupt[]清零。

单值比较中断每个计数通道只有一个中断信号输出，所有计数通道（0-7）都可以输出单值比较中断信号。多值比较中断，只有计数通道 0-3 可以输出多值中断，每个计数器可以输出 8 个（0-7）中断信号，当选定某个多值计数通道后，其第 y 个比较值与中断信号一一对应。多值比较中断每次只能有一个计数通道有效。

4.3 数字量输入输出模块

4.3.1 创建数字量输入输出模块使用工程

创建数字量输入输出应用，添加模块需要的库文件“IoDrvDI16_1.1.0.0.devdesc.xml”、“IoDrvDO16_1.1.0.0.devdesc.xml”。

4.3.2 变量定义及使用

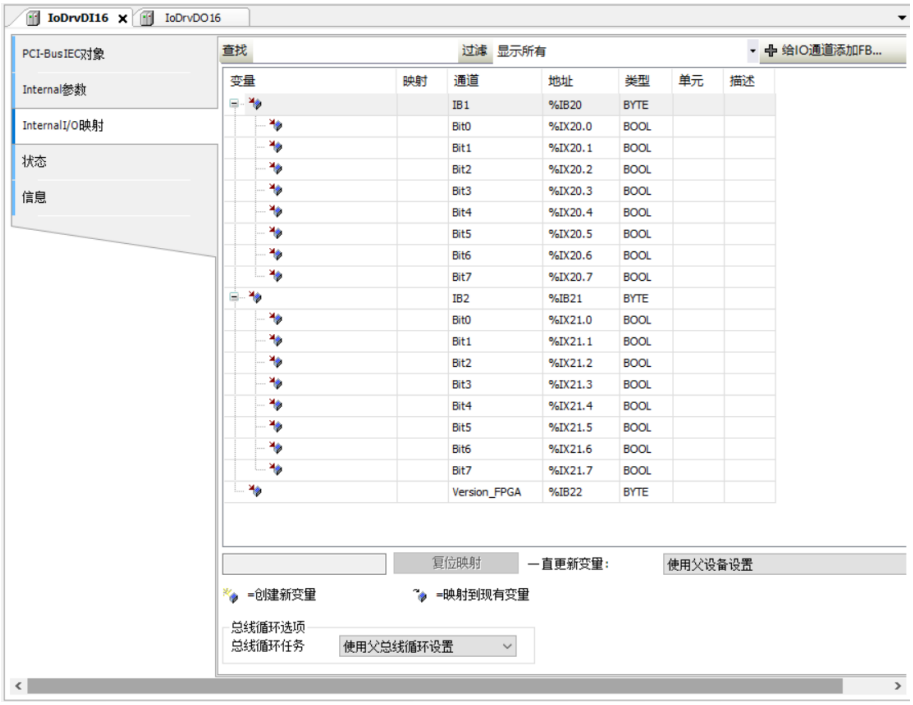


图 4-5 输入模块变量映射表

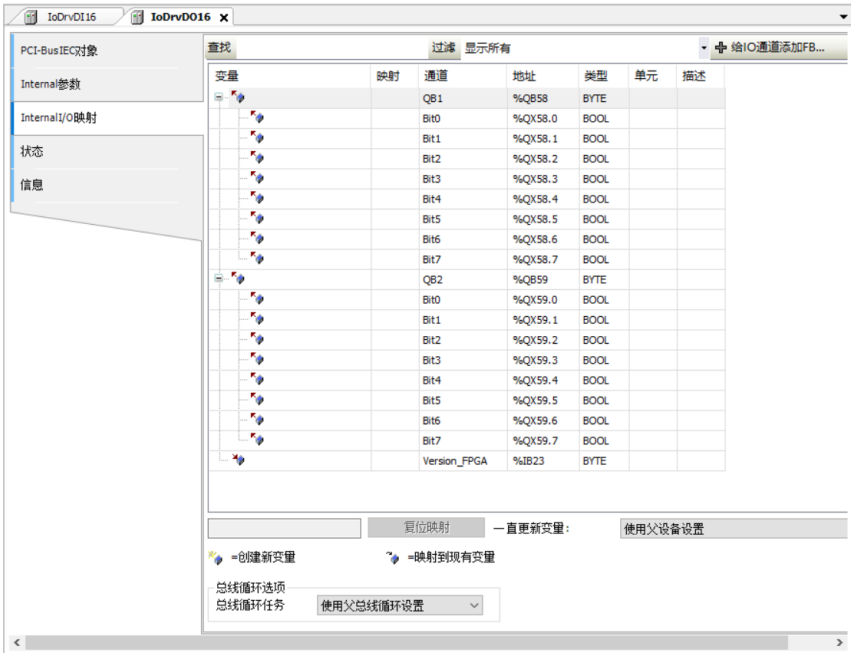


图 4-6 输出模块变量映射表

4.4 模拟量输入输出模块

4.4.1 创建模拟量输入输出模块使用工程

创建模拟量输入输出应用工程，添加模块需要的库文件“IoDrv4AD_1.1.0.0.devdesc.xml”、“IoDrv4DA_1.1.0.0.devdesc.xml”。

4.4.2 变量定义及使用

图 4-7 模拟量输入变量映射表

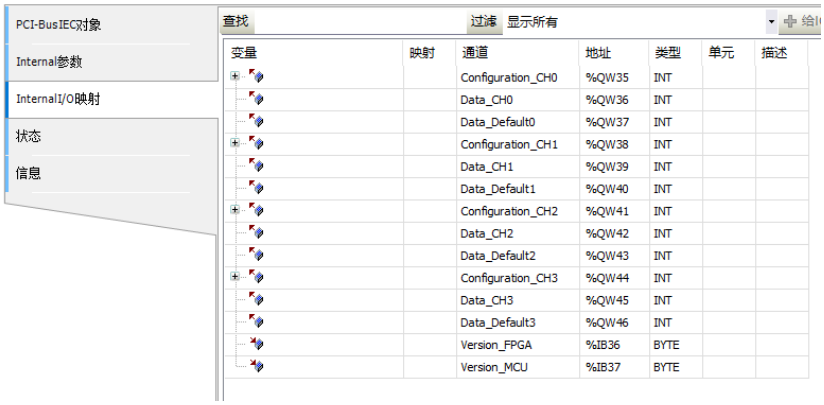


图 4-8 模拟量输出变量映射

4.5 温度模块

4.5.1 创建温度模块使用工程

创建温度模块应用，添加本模块需要的库文件“IoDrvTemperature_1.1.0.0.devdesc.xml”。

4.5.2 变量定义及使用

变量	映射	通道	地址	类型	单元	描述
		Temperature0	%ID10	REAL		
		Breakup0	%IB44	BYTE		
		Overrun0	%IB45	BYTE		
		Temperature1	%ID12	REAL		
		Breakup1	%IB52	BYTE		
		Overrun1	%IB53	BYTE		
		Temperature2	%ID14	REAL		
		Breakup2	%IB60	BYTE		
		Overrun2	%IB61	BYTE		
		Temperature3	%ID16	REAL		
		Breakup3	%IB68	BYTE		
		Overrun3	%IB69	BYTE		
		Version_FPGA	%IB70	BYTE		
		Version_MCU	%IB71	BYTE		
		In_CJC	%ID18	REAL		
		Out_CJC	%ID19	REAL		
		Basic_Set_0	%QB94	BYTE		
		Sampling_Period_0	%QB95	BYTE		
		Sensor_Type_0	%QB96	BYTE		
		Filtering_Time_0	%QB97	BYTE		
		Upper_Value_0	%QW49	INT		
		Lower_Value_0	%QW50	INT		
		Basic_Set_1	%QB102	BYTE		
		Sampling_Period_1	%QB103	BYTE		
		Sensor_Type_1	%QB104	BYTE		
		Filtering_Time_1	%QB105	BYTE		
		Upper_Value_1	%QW53	INT		
		Lower_Value_1	%QW54	INT		
		Basic_Set_2	%QB110	BYTE		
		Sampling_Period_2	%QB111	BYTE		
		Sensor_Type_2	%QB112	BYTE		
		Filtering_Time_2	%QB113	BYTE		
		Upper_Value_2	%QW57	INT		
		Lower_Value_2	%QW58	INT		
		Basic_Set_3	%QB118	BYTE		
		Sampling_Period_3	%QB119	BYTE		
		Sensor_Type_3	%QB120	BYTE		
		Filtering_Time_3	%QB121	BYTE		
		Upper_Value_3	%QW61	INT		
		Lower_Value_3	%QW62	INT		

图 4-9 温度模块变量映射

4.5.3 温度模块

采用 EtherCAT 远程扩展模块（AX-EM-RCM-ET），通过背板扩展温度模块（AX-EM-4PTC），使用说明如下：

1) 在设备栏点击 AX-EM-ECM-ET 点击右键，添加温度模块（AX_EM_4PTC）；通过 Module/IO 映射选项卡里面的多组变量实现对模块的控制，如下图所示：

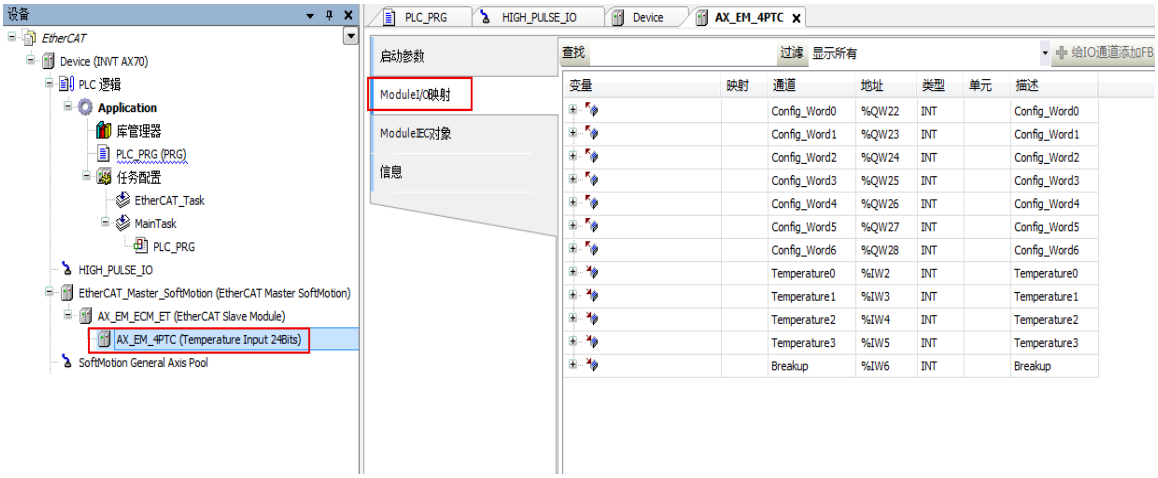


图 4-10 温度模块变量映射

2) 编译通过后，登录下载工程并运行即可；

3) 变量说明：以下说明四个通道所有变量的使用方法。

表 4-4 变量说明

参数		值	有效位	变量名
通道 0 温度值			[15：0]	Temperature0
通道 1 温度值			[15：0]	Temperature1
通道 2 温度值			[15：0]	Temperature2
通道 3 温度值			[15：0]	Temperature3
通道 0 断线检测结果	正常	00	[1：0]	Breakup
	断线	01		
通道 1 断线检测结果	正常	00	[3：2]	
	断线	01		
通道 2 断线检测结果	正常	00	[9：8]	
	断线	01		
通道 3 断线检测结果	正常	00	[11：10]	
	断线	01		
通道 0 使能	使能	1	[0]	Config_Word0
	禁用	0		
显示模式	℃	0	[1]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[2]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[3]	
	禁用	0		
超限检测	使能	1	[4]	
	禁用	0		
传感器类型	B E J	000 001 010	[11：8]	

参数		值	有效位	变量名
	K N R S T PT100 PT500 PT1000 CU500	011 100 101 110 111 1000 1001 1010 1011		
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[13：12]	
滤波时间	0~100	0~100	[6：0]	Config_Word1
通道 1 使能	使能	1	[8]	
	禁用	0		
显示模式	℃	0	[9]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[10]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[11]	
	禁用	0		
超限检测	使能	1	[12]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[3：0]	Config_Word2
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[5：4]	
滤波时间	0~100	0~100	[14：8]	
通道 2 使能	使能	1	[0]	Config_Word3
	禁用	0		
显示模式	℃	0	[1]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[2]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[3]	
	禁用	0		

参数		值	有效位	变量名
超限检测	使能	1	[4]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[11：8]	
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[13：12]	
滤波时间	0~100	0~100	[6：0]	Config_Word4
通道 3 使能	使能	1	[8]	
	禁用	0		
显示模式	℃	0	[9]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[10]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[11]	
	禁用	0		
超限检测	使能	1	[12]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[3：0]	Config_Word5
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[5：4]	
滤波时间	0~100	0~100	[14：8]	
通道 0 采样周期	250ms 500ms 1000ms	01 10 11	[1：0]	Config_Word6

参数		值	有效位	变量名
通道 1 采样周期	250ms 500ms 1000ms	01 10 11	[3：2]	
通道 2 采样周期	250ms 500ms 1000ms	01 10 11	[5：4]	
通道 3 采样周期	250ms 500ms 1000ms	01 10 11	[7：6]	

表 4-5 支持的传感器类型和测量范围

项目	传感器名称	摄氏度温度范围	华氏度温度范围
热电阻类型	PT100	-200.0℃~850℃	-328.0℉~1562.0℉
	PT500	-200.0℃~850℃	-328.0℉~1562.0℉
	PT1000	-200.0℃~850℃	-328.0℉~1562.0℉
	CU100	-50.0℃~150℃	-58.0℉~302.0℉
热电偶类型	B	200.0℃~1800℃	392.0℉~3272.0℉
	E	-270.0℃~1000℃	-454.0℉~1832.0℉
	N	-200.0℃~1300℃	-328.0℉~2372.0℉
	J	-210.0℃~1200℃	-346.0℉~2192.0℉
	K	-270.0℃~1370℃	-454.0℉~2498.0℉
	R	-50.0℃~1765℃	-58.0℉~3209.0℉
	S	-50.0℃~1765℃	-58.0℉~3209.0℉
	T	-270.0℃~400℃	-454.0℉~752.0℉

4.6 通信模块

EtherCAT 通信模块作为 EtherCAT 从站使用，使用时需添加设备描述文件“INVT_ECAT_SLAVE_FOR_Invtmatic Studio_V1.07.xml”。具体使用方法可参考 EtherCAT 主站添加 DA200 伺服驱动器案例。

1、 在 Invtmatic Studio 上位机新建工程，点击 Device 右键添加设备，添加 EtherCAT Master SoftMotion 模块，如下图所示：

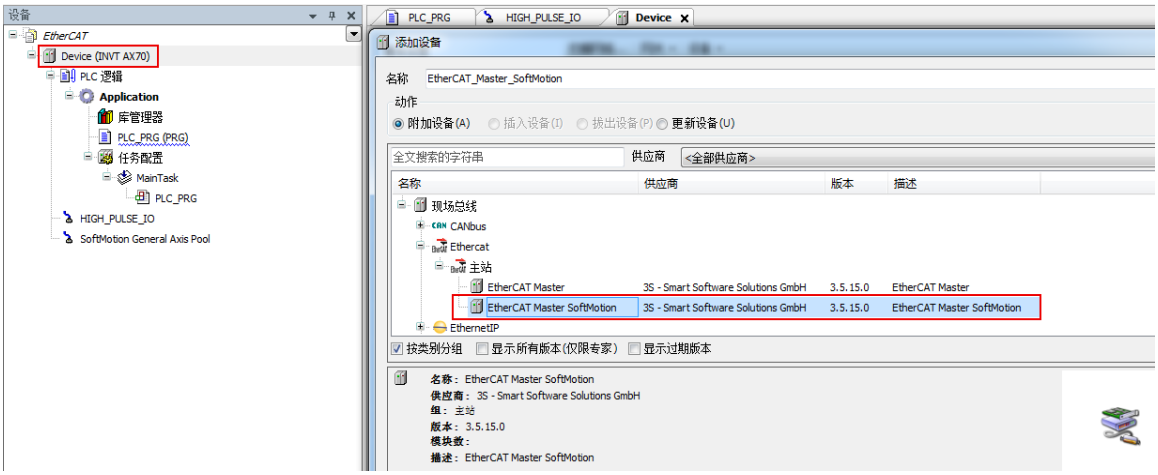


图 4-11 添加 EtherCAT Master SoftMotion 模块

2、 在工程 EtherCAT Master SoftMotion 模块中右键添加设备，添加 EtherCAT Slave Module 模块（AX-EM-RCM-ET），如下图所示：

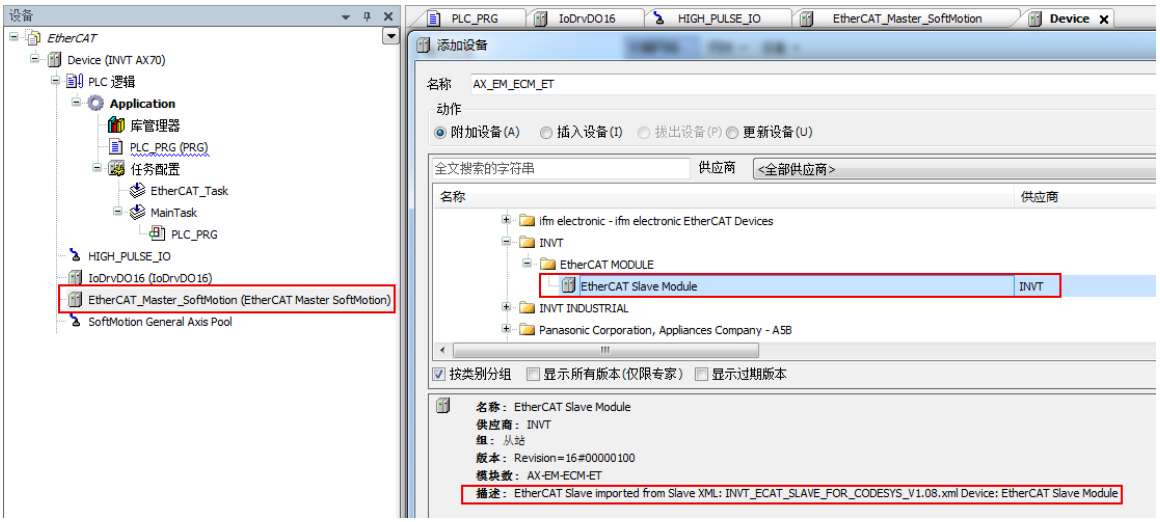


图 4-12 添加 EtherCAT 远程扩展模块

下文就 EtherCAT 远程扩展模块扩展我司现有 IO 的使用方法进行说明。

4.6.1 数字量输入模块

采用 EtherCAT 远程扩展模块（AX-EM-RCM-ET），通过背板扩展数字量输入模块(AX-EM-1600D)，使用说明如下：

1、 在设备栏点击 AX-EM-ECM-ET 点击右键，添加数字量输入模块（AX_EM_1600D）；通过 Module/IO 映射选项卡里面的 InByte0 和 InByte1 两组变量，实现对 16 路通道的控制，如下图所示：

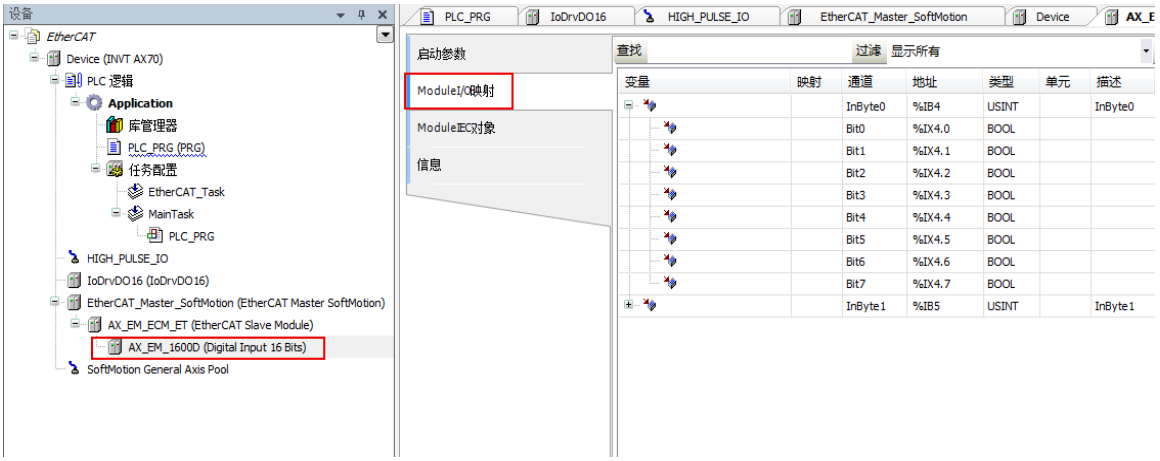


图 4-13 数字量输入模块变量映射

2、 编译通过后，登录下载工程并运行即可。

4.6.2 数字量输出模块

采用 EtherCAT 远程扩展模块（AX-EM-RCM-ET），通过背板扩展数字量输出模块（AX-EM-0016DP/ AX-EM-0016DN），使用说明如下：

1、 在设备栏点击 AX-EM-ECM-ET 点击右键，以添加数字量输出模块（AX_EM_0016DP）为例；通过 Module/IO 映射选项卡里面的 OutByte0 和 OutByte1 两组变量，实现对 16 路通道的控制，如下图所示：

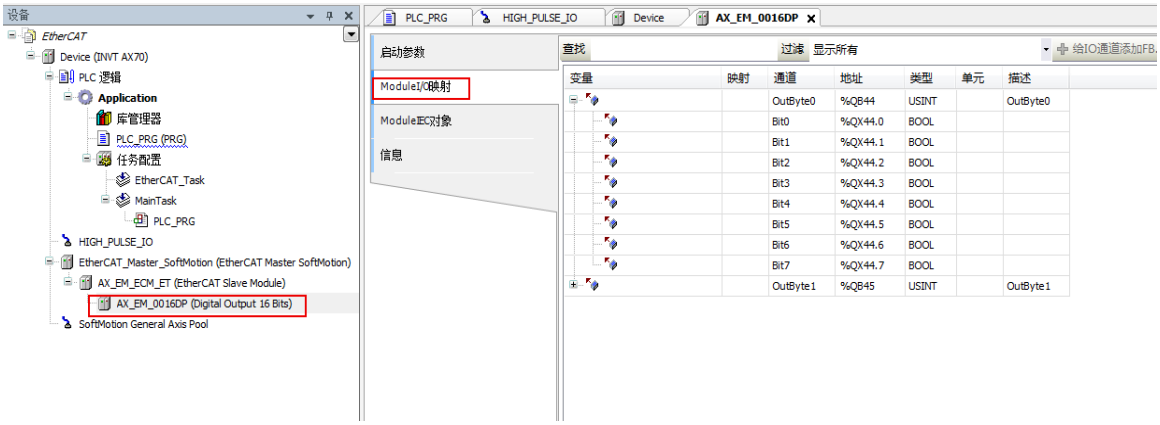


图 4-14 数字量输出模块变量映射

2、 编译通过后，登录下载工程并运行即可。

4.6.3 模拟量输入模块

采用 EtherCAT 远程扩展模块（AX-EM-RCM-ET），通过背板扩展模拟量输入模块（AX-EM-4AD），使用说明如下：

1、 在设备栏点击 AX-EM-ECM-ET 点击右键，添加模拟量输入模块(AX_EM_4AD)；通过 Module/IO 映射选项卡里面的多组变量实现对模块的控制，如下图所示：

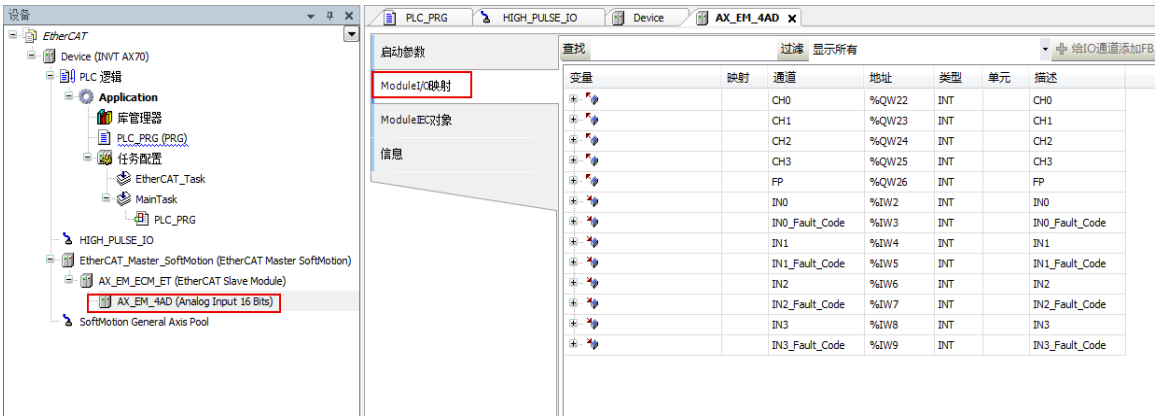


图 4-15 模拟量输入模块变量映射

2、 编译通过后，登录下载工程并运行即可。

3、 变量说明：以下以通道 0 为例，通过表格说明通道 0 所有变量的使用方法。

表 4-6 通道 0 变量说明

参数			值	有效位	变量名	变量类型
滤波器选用	sinc5+sinc1		00	[1:0]	FP	WORD
	sinc5+sinc1+enhance 50/60		01			
	sinc3		10			
	预留					
通道 0 配置项	通道使能	使能	1	[0]	CH0	
		禁用	0			
	断线检测	使能	1	[1]		
		禁用	0			
	转换模式	0V~5V	000	[4:2]		
		0V~10V	001			
		-5~5V	010			
		-10V~10V	011			
		-20mA~20mA	100			

参数			值	有效位	变量名	变量类型
		0mA~20mA	101			
		4mA~20mA	111			
	超限标志	使能	1	[5]		
		禁用	0			
	超量程检测使能位	使能	1	[6]		
		禁用	0			
	预留			[15:7]		
通道 0 数据项	数据			[15:0]	IN0	
通道 0 故障码（具体见表 4-8）	指示模块当前的故障信息			[15:0]	IN0_Fault_Code	

表 4-7 映射与实际输入模拟量值对应关系

类型	输入额定范围	额定对应数字量
模拟电压输入	-10V~10V	-10000~+10000
	0V~10V	0~10000
	-5V~+5V	-5000~+5000
	0V~5V	0~5000
模拟电流输入	-20mA~20mA	-20000~20000
	0mA~20mA	0~20000
	4mA~20mA	4000~20000

表 4-8 通道 0 故障码含义

通道 0	含义
A0	通道 0 断线
A1	通道 0 超极限（即超过-25V~+25V 的范围）
A2	通道 0 超量程上限（即超过当前所选择的电压范围的上限值）
A3	通道 0 超量程下限（即超过当前所选择的电压范围的下限值）

通道 1	含义
A4	通道 1 断线
A5	通道 1 超极限（即超过-25V~+25V 的范围）
A6	通道 1 超量程上限（即超过当前所选择的电压范围的上限值）
A7	通道 1 超量程下限（即超过当前所选择的电压范围的下限值）

通道 2	含义
A8	通道 2 断线
A9	通道 2 超极限（即超过-25V~+25V 的范围）
AA	通道 2 超量程上限（即超过当前所选择的电压范围的上限值）
Ab	通道 2 超量程下限（即超过当前所选择的电压范围的下限值）

通道 3	含义
AC	通道 3 断线
Ad	通道 3 超极限（即超过-25V~+25V 的范围）
AE	通道 3 超量程上限（即超过当前所选择的电压范围的上限值）
AF	通道 3 超量程下限（即超过当前所选择的电压范围的下限值）

4.6.4 模拟量输出模块

采用 EtherCAT 远程扩展模块(AX-EM-RCM-ET)，通过背板扩展模拟量输出模块(AX-EM-4DA)，使用说明如下：

1、 在设备栏点击 AX-EM-ECM-ET 点击右键，添加模拟量输出模块(AX_EM_4DA)；通过 Module/IO 映射选项卡里面的多组变量实现对模块的控制，如下图所示：

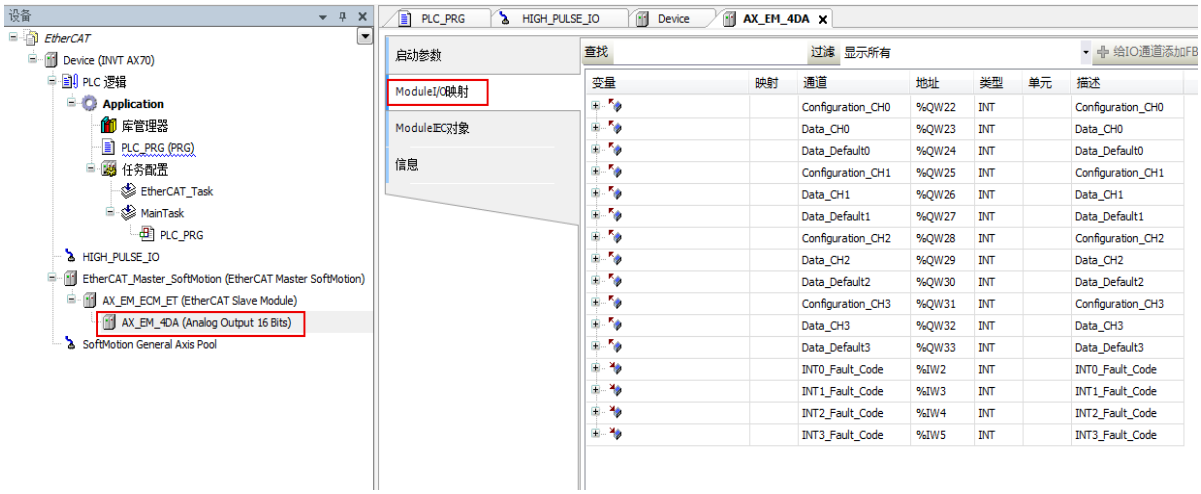


图 4-16 模拟量输出模块变量映射

2、 编译通过后，登录下载工程并运行即可；

3、 变量说明：以下以通道 0 为例，通过表格说明通道 0 所有变量的使用方法。

表 4-9 通道 0 变量说明

参数			值	有效位	变量名
通道 0 配置项	通道使能	使能	1	[0]	Configuration_CH0
		禁用	0		
	断线检测	预留		[1]	
	转换模式	0V~5V	000	[4：2]	
		0V~10V	001		
		-5V~5V	010		
		-10V~10V	011		
		4mA~20mA	100		
		0mA~20mA	101		
	停止后输出状态	输出清零	00	[6：5]	
		输出保持	01		
		输出预设值	10		
	预留			[15：7]	
通道 0 码值	数据			[15：0]	Data_CH0
通道 0 输出预设值	输出预设值			[15：0]	Data_Default0
通道 0 故障码（具体见 表 4-11）	指示模块当前的故障信息			[15：0]	INT0_Fault_Code

表 4-10 映射与实际输入模拟量值对应关系

类型	输入额定范围	额定对应数字量
模拟电压输出	-10V~10V	-10000~+10000
	0V~10V	0~10000
	-5V~+5V	- 5000~+5000

类型	输入额定范围	额定对应数字量
	0V~5V	0~5000
模拟电流输出	4mA~20mA	4000~20000
	0mA~20mA	0~20000

表 4-11 通道 0 故障码含义

通道 0	含义
B0	通道 0 的电流输出断线
B1	通道 0 的电压输出短路

通道 1	含义
B2	通道 1 的电流输出断线
B3	通道 1 的电压输出短路

通道 2	含义
B4	通道 2 的电流输出断线
B5	通道 2 的电压输出短路

通道 3	含义
B6	通道 3 的电流输出断线
B7	通道 3 的电压输出短路

输出模块断电故障	含义
B8	输出模块的电源板 24V 断电

4.6.5 温度模块

采用 EtherCAT 远程扩展模块（AX-EM-RCM-ET），通过背板扩展温度模块(AX-EM-4PTC)，使用说明如下：

1、 在设备栏点击 AX-EM-ECM-ET 点击右键，添加温度模块(AX_EM_4PTC)；通过 Module/IO 映射选项卡里面的多组变量实现对模块的控制，如下图所示。

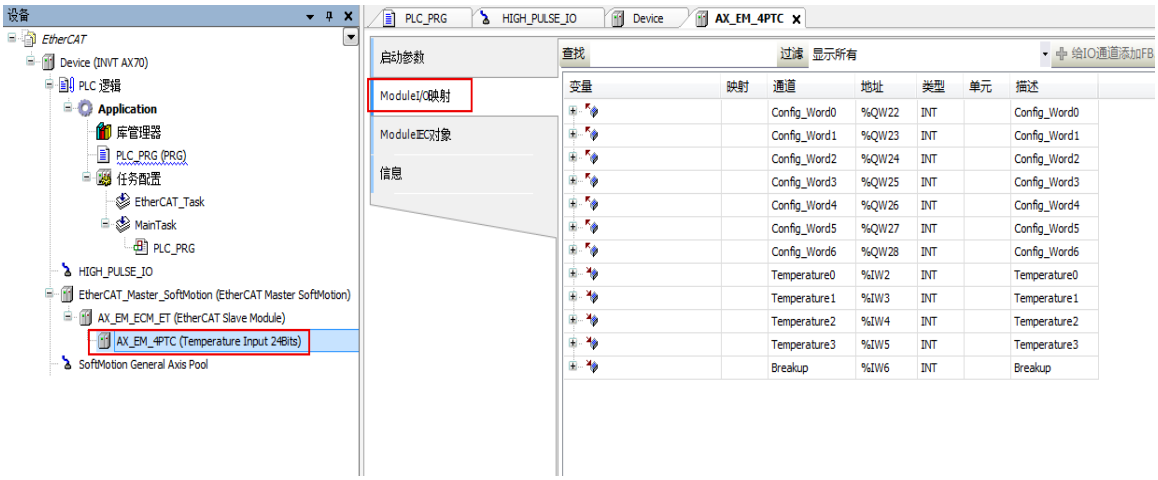


图 4-17 温度模块变量映射

2、 编译通过后，登录下载工程并运行即可。

3、 变量说明：以下说明四个通道所有变量的使用方法。

表 4-12 变量说明

参数		值	有效位	变量名
通道 0 温度值			[15：0]	Temperature0
通道 1 温度值			[15：0]	Temperature1
通道 2 温度值			[15：0]	Temperature2
通道 3 温度值			[15：0]	Temperature3
通道 0 断线检测结果	正常	00	[1：0]	Breakup
	断线	01		
通道 1 断线检测结果	正常	00	[3：2]	
	断线	01		
通道 2 断线检测结果	正常	00	[9：8]	
	断线	01		
通道 3 断线检测结果	正常	00	[11：10]	
	断线	01		
通道 0 使能	使能	1	[0]	Config_Word0
	禁用	0		
显示模式	℃	0	[1]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[2]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[3]	
	禁用	0		
超限检测	使能	1	[4]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[11：8]	
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[13：12]	
滤波时间	0~100	0~100	[6：0]	Config_Word1
通道 1 使能	使能	1	[8]	
	禁用	0		
显示模式	℃	0	[9]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[10]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[11]	
	禁用	0		
超限检测	使能	1	[12]	

参数		值	有效位	变量名
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[3：0]	Config_Word2
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[5：4]	
滤波时间	0~100	0~100	[14：8]	
通道 2 使能	使能	1	[0]	Config_Word3
	禁用	0		
显示模式	℃	0	[1]	
	℉	1		
冷端补偿方式	内部冷端补偿	0	[2]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[3]	
	禁用	0		
超限检测	使能	1	[4]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[11：8]	
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[13：12]	
滤波时间	0~100	0~100	[6：0]	Config_Word4
通道 3 使能	使能	1	[8]	
	禁用	0		
显示模式	℃	0	[9]	
	℉	1		

参数		值	有效位	变量名
冷端补偿方式	内部冷端补偿	0	[10]	
	外部冷端补偿	1		
传感器断线检测	使能	1	[11]	
	禁用	0		
超限检测	使能	1	[12]	
	禁用	0		
传感器类型	B E J K N R S T PT100 PT500 PT1000 CU500	000 001 010 011 100 101 110 111 1000 1001 1010 1011	[3：0]	Config_Word5
	2 线 3 线 4 线 （针对 RTD）	00 01 10	[5：4]	
滤波时间	0~100	0~100	[14：8]	
通道 0 采样周期	250ms 500ms 1000ms	01 10 11	[1：0]	Config_Word6
通道 1 采样周期	250ms 500ms 1000ms	01 10 11	[3：2]	
通道 2 采样周期	250ms 500ms 1000ms	01 10 11	[5：4]	
通道 3 采样周期	250ms 500ms 1000ms	01 10 11	[7：6]	

表 4-13 支持的传感器类型和测量范围

项目	传感器名称	摄氏度温度范围	华氏度温度范围
热电阻类型	PT100	-200.0℃~850℃	-328.0℉~1562.0℉
	PT500	-200.0℃~850℃	-328.0℉~1562.0℉
	PT1000	-200.0℃~850℃	-328.0℉~1562.0℉
	CU100	-50.0℃~150℃	-58.0℉~302.0℉
热电偶类型	B	200.0℃~1800℃	392.0℉~3272.0℉
	E	-270.0℃~1000℃	-454.0℉~1832.0℉
	N	-200.0℃~1300℃	-328.0℉~2372.0℉
	J	-210.0℃~1200℃	-346.0℉~2192.0℉
	K	-270.0℃~1370℃	-454.0℉~2498.0℉
	R	-50.0℃~1765℃	-58.0℉~3209.0℉

项目	传感器名称	摄氏度温度范围	华氏度温度范围
	S	-50.0℃~1765℃	-58.0℉~3209.0℉
	T	-270.0℃~400℃	-454.0℉~752.0℉

4.7 各模块优先级设置（推荐值）

4.7.1 优先级设置注意事项

若创建的工程中包含多个功能模块，创建多个任务，任务优先级设置如下图所示，任务优先级推荐使用值如下表所示。

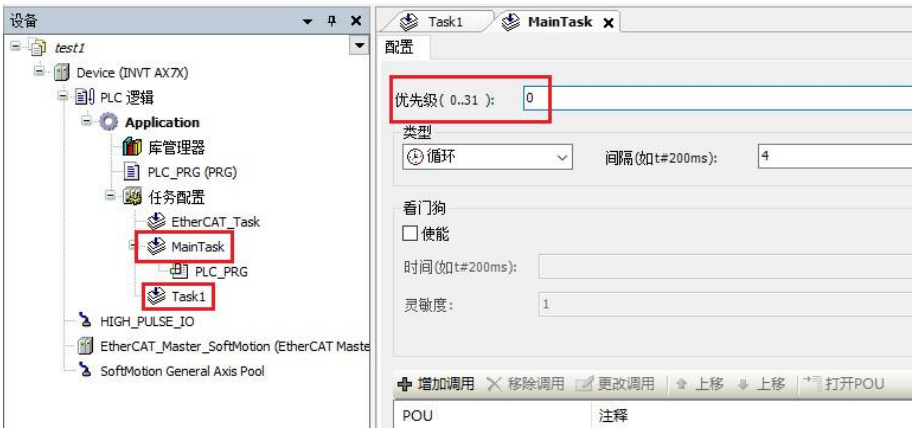


图 4-18 任务工程优先级设置参考示例

表 4-14 优先级设置

功能模块	推荐优先级
PlcCfg 模块	31
ModbusTCP	15~30
ModbusRTU	15~30
高速 I/O	1~15
模拟量输入输出	1~15
温度模块	1~15
EtherCAT	0

4.7.2 子设备总线循环任务（Bus Cycle Options）配置注意事项

在 AX7x 设备“PLC 设置 > 总线周期 > 总线周期任务”选项中，**总线周期任务：**选项列表提供了当前有效工程中任务配置里已定义的任务（例如"MainTask"，"EtherCAT Master"等）。选择其中一个任务作为当前工程的总线周期，或者选择选项<未标明的>，该选项意味着将应用最短的任务周期时间，也就是最快的执行周期。您可以切换到另一种设置，**但是请务必注意下列事项：**

注意：在修改“**<unspecified>**”设置之前，您已经清楚其影响。“<unspecified>”是指将应用由设备描述定义的缺省行为。所以请检查这方面的相关描述：缺省方式下，该任务可能被定义为具有最短周期时间，但也可能具有最长周期时间。

因此在使用扩展模块以及 EtherCAT 模块时，为提高系统运行稳定性（尤其在使用 EtherCAT_Master_SoftMotion 模块时），在“EtherCAT I/O 映射->总线循环选项（EthreCAT I/O Mapping->Bus Cycle Options）”中选择各模块对应的任务，参考例程如下图：

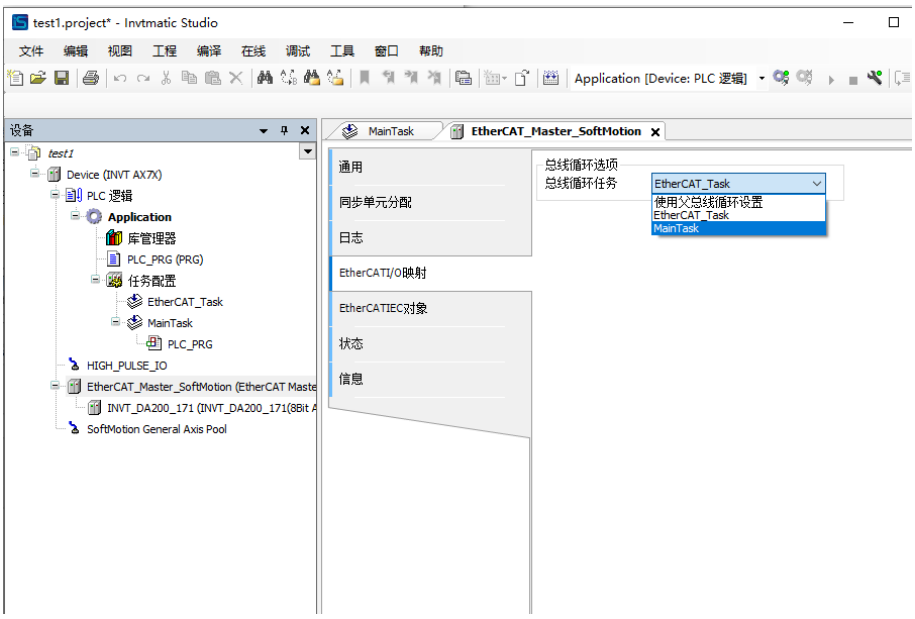


图 4-19 扩展模块总线循环任务设置示例

第5章 设备诊断

AX7x 设备诊断信息通过三种方式体现，分别是故障灯、数码管和诊断码。其中故障灯表示系统和总线错误，数码管显示具体某功能模块的错误代码；诊断码进一步详细指示错误的具体类型，一般通过上位机软件查寻。

5.1 故障指示灯

AX7x 故障灯部分主要由两个部分组成，第一部分主要为系统和总线指示灯；第二部分主要为高速输入输出指示灯。

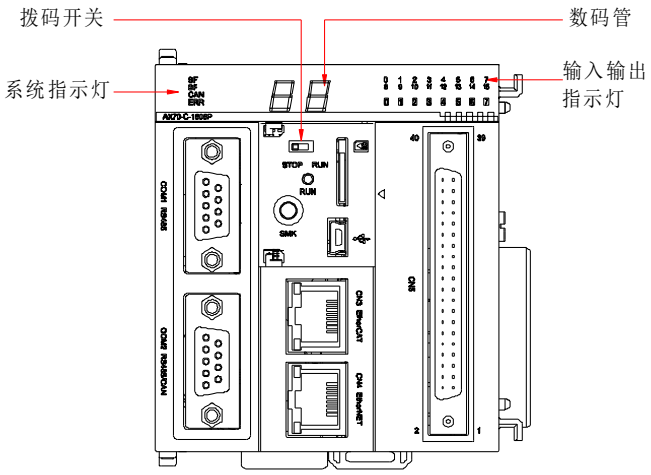


图 5-1 故障指示灯分布图

5.1.1 系统与总线故障灯

表 5-1 系统与指示灯

故障指示灯名称	错误类型
SF	系统故障
BF	总线通讯故障
CAN	CAN 总线故障
ERR	模块故障

说明：当连接多台可编程控制器时，可通过点击软件平台“闪烁”按钮，观察 SF、BF、CAN、ERR 四灯同时闪烁，识别设备。

5.1.2 高速输入输出指示灯

对应端口若输出/输入为高电平则相应的指示灯点亮，若输出/输入为低电平则相应的指示灯熄灭。

5.2 错误码

5.2.1 PlcCfg 错误码表

表 5-2 PlcCfg 错误码表

错误码	出错类型	解决办法
16#10	设置本机新 IP 错误	检查底层网络配置文件
16#11	设置本机新子网掩码错误	检查底层网络配置文件
16#12	读取本机 IP 及子网掩码失败	检查底层网络配置文件
16#13	时间设置格式不符合常规	检查相应的时间设置格式

错误码	出错类型	解决办法
16#14	设置运动控制器时间错误	检查底层相应代码
16#15	获取运动控制器实时时间错误	检查底层相应代码

5.2.2 ModbusRTU 错误码

（1）COM1 口实现 RTU 模块功能时错误码如下表：

表 5-3 COM1 口 RTU 错误码

模块	错误码	出错类型	解决办法
COM1 485 ModbusRTU_Slave1	16#20	打开串口 COM1 失败	底层串口编号是否与硬件对应
	16#21	波特率设置失败	检查从站波特率设置
	16#22	数据位、停止位或者校验位设置失败	查看 **Invtmatic Studio** ErrorID 具体错误码；数据位：ErrorID=3，校验位 ErrorID=4，停止位 ErrorID=5
	16#23	从站功能使能失败	系统出错 Err_Sym，或者从站使能为打开
	16#24	从站读写出错	查看具体参数设置
COM1 485 ModbusRTU_Master 1	16#25	打开串口 COM1 失败	底层串口编号是否与硬件对应
	16#26	SlaveID 设置失败	检查主站 SlaveID 号的设置
	16#27	数据位、停止位或者校验位设置失败	查看数据位设置值是否为 7 或 8，校验位是否为 0、1、2，停止位是否为 1 或 2
	16#28	主站功能使能失败	系统出错 Err_Sym，或者主站使能为打开
	16#29	主站读写线圈、读保持寄存器、写单个寄存器、写多个寄存器，其中一个出错	检查主从站初始化参数配置是否一致，查看硬件连接是否有误

（2）COM2 口实现 RTU 模块功能时错误码如下表：

表 5-4 COM2 口 RTU 错误码

模块	错误码	出错类型	解决办法
COM2 485 ModbusRTU_Slave2	16#30	打开串口 COM2 失败	底层串口编号是否与硬件对应
	16#31	波特率设置失败	检查从站波特率设置
	16#32	数据位、停止位或者校验位设置失败	查看 **Invtmatic Studio** ErrorID 具体错误码；数据位：ErrorID=3，校验位 ErrorID=4，停止位 ErrorID=5
	16#33	从站功能使能失败	系统出错 Err_Sym，或者从站使能为打开
	16#34	从站读写出错	查看具体参数设置
COM2 485 ModbusRTU_Master 2	16#35	打开串口 COM2 失败	底层串口编号是否与硬件对应
	16#36	SlaveID 设置失败	检查主站 SlaveID 号的设置
	16#37	数据位、停止位或者校验位设置失败	查看数据位设置值是否为 7 或 8，校验位是否为 0、1、2，停止位是否为 1 或 2
	16#38	主站功能使能失败	系统出错 Err_Sym，或者主站使能为打开
	16#39	主站读写线圈、读保持寄存器、写单个寄存器、写多个寄存器，其中一个出错	检查主从站初始化参数配置是否一致，查看硬件连接是否有误

5.2.3 ModbusTCP 错误码表

表 5-5 Modbus TCP 错误码

模块	错误码	出错类型	解决办法
modbusTCP_Slave	16#60	配置从站 IP 错误	查看底层相应配置
	16#61	端口设置错误	检查端口设置
	16#62	监听 socket 失败（建立 socket 失败，绑定 socket 失败，监听 socket 失败）	查看相应配置
	16#63	接受客户端失败	查看相应配置
	16#64	接受客户端数据失败	查看相应配置
	16#65	modbus 回复出错（modbus_reply）	查看相应配置
modbusTCP_Master	16#66	设置从站 IP 或端口错误	检查 IP 设置或是否为默认单元号
	16#67	设置从站失败	检查相应参数设置
	16#68	连接从站失败	检查相应参数如从站 IP 设置或端口设置
	16#69	写从站寄存器失败	检查相应参数设置
	16#6A	读从站寄存器失败	检查相应参数设置

5.2.4 模拟量模块

（1）模拟量输入模块功能时错误码如下表：

表 5-6 模拟量输入模块错误码

错误码	出错类型	解决办法
16#A0	通道 0 断线	检查线路是否连接正常
16#A1	通道 0 超极限（即电压超过-25V~+25V，电流超过-104mA~104mA 的范围）	检查输入的电压（电流）是否超过范围
16#A2	通道 0 超量程上限（即超过当前所选择的电压范围的上限值）	降低所输入的电压（电流）值，或者选用更大范围的转换模式
16#A3	通道 0 超量程下限（即超过当前所选择的电压范围的下限值）	增大所输入的电压（电流）值，或者选用更大范围的转换模式
16#A4	通道 1 断线	检查线路是否连接正常
16#A5	通道 1 超极限（即电压超过-25V~+25V，电流超过-104mA~104mA 的范围）	检查输入的电压（电流）是否超过范围
16#A6	通道 1 超量程上限（即超过当前所选择的电压范围的上限值）	降低所输入的电压（电流）值，或者选用更大范围的转换模式
16#A7	通道 1 超量程下限（即超过当前所选择的电压范围的下限值）	增大所输入的电压（电流）值，或者选用更大范围的转换模式
16#A8	通道 2 断线	检查线路是否连接正常
16#A9	通道 2 超极限（即电压超过-25V~+25V，电流超过-104mA~104mA 的范围）	检查输入的电压（电流）是否超过范围
16#AA	通道 2 超量程上限（即超过当前所选择的电压范围的上限值）	降低所输入的电压（电流）值，或者选用更大范围的转换模式
16#Ab	通道 2 超量程下限（即超过当前所选择的电压范围的下限值）	增大所输入的电压（电流）值，或者选用更大范围的转换模式
16#AC	通道 3 断线	检查线路是否连接正常
16#Ad	通道 3 超极限（即电压超过-25V~+25V，电流超过-104mA~104mA 的范围）	检查输入的电压（电流）是否超过范围
16#AE	通道 3 超量程上限（即超过当前所选择的电压范围的上限值）	降低所输入的电压（电流）值，或者选用更大范围的转换模式

错误码	出错类型	解决办法
16#AF	通道 3 超量程下限（即超过当前所选择的电压范围的下限值）	增大所输入的电压（电流）值，或者选用更大范围的转换模式

（2）模拟量输出模块功能时错误码如下表：

表 5-7 模拟量输出模块错误码

错误码	出错类型	解决办法
16#b0	通道 0 的电流输出断线	检查电流通道是否断线，若断线则将其重新连接
16#b1	通道 0 的电压输出短路	检查电压通道是否短路，若短路则并将其恢复正常
16#b2	通道 1 的电流输出断线	检查电流通道是否断线，若断线则将其重新连接
16#b3	通道 1 的电压输出短路	检查电压通道是否短路，若短路则并将其恢复正常
16#b4	通道 2 的电流输出断线	检查电流通道是否断线，若断线则将其重新连接
16#b5	通道 2 的电压输出短路	检查电压通道是否短路，若短路则并将其恢复正常
16#b6	通道 3 的电流输出断线	检查电流通道是否断线，若断线则将其重新连接
16#b7	通道 3 的电压输出短路	检查电压通道是否短路，若短路则并将其恢复正常
16#b8	输出模块的电源板 24V 断电	检查 24V 供电是否正常，有没有反接。

5.2.5 温度模块

温度模块功能运行时错误码类型如下表：

表 5-8 温度模块错误码

错误码	出错类型	解决办法
16#C0	通道 0 超量程上限（即温度实际值超过设定的上限值）	检查设定的温度上限值是否大于实际值
16#C1	通道 0 超量程下限（即温度实际值低于设定的下限值）	检查设定的温度下限值是否小于实际值
16#C2	通道 1 超量程上限（即温度实际值超过设定的上限值）	检查设定的温度上限值是否大于实际值
16#C3	通道 1 超量程下限（即温度实际值低于设定的下限值）	检查设定的温度下限值是否小于实际值
16#C4	通道 2 超量程上限（即温度实际值超过设定的上限值）	检查设定的温度上限值是否大于实际值
16#C5	通道 2 超量程下限（即温度实际值低于设定的下限值）	检查设定的温度下限值是否小于实际值
16#C6	通道 3 超量程上限（即温度实际值超过设定的上限值）	检查设定的温度上限值是否大于实际值
16#C7	通道 3 超量程下限（即温度实际值低于设定的下限值）	检查设定的温度下限值是否小于实际值
16#C8	超限设置错误（设定的上限值低于下限值）	检查设定的温度上限值是否大于温度下限值
16#C9	通道 0 断线（保留）	
16#CA	通道 1 断线（保留）	
16#CB	通道 2 断线（保留）	
16#CC	通道 3 断线（保留）	

第6章 控制器程序结构与执行

6.1 程序结构

软件模型用分层结构表示。每一层隐含其下面层的许多特性。软件模型描述基本的软件元素及其相互关系。这些软件元素包含：设备、应用、任务、全局变量、访问路径和应用对象，其内部结构如图 6-1 所示，该软件模型与 IEC 61131-3 标准的软件模型保持一致。

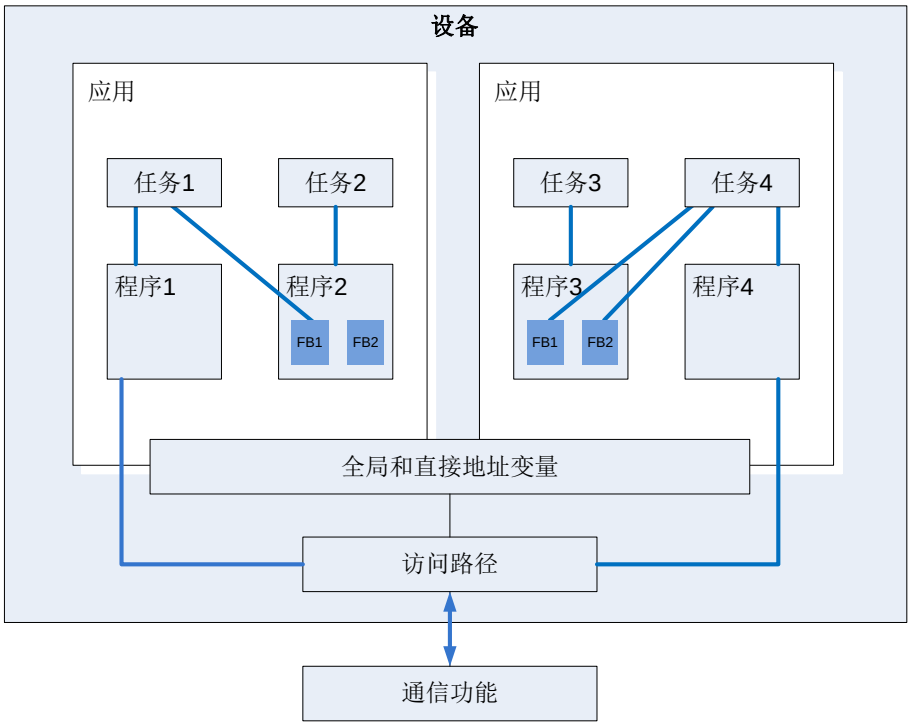


图 6-1 程序分层结构图

6.2 任务

一个程序可以用不同的编程语言来编写。典型的程序由许多互连的功能块组成，各功能块之间可互相交换数据。在一个程序中不同部分的执行通过“任务”来控制。“任务”被配置以后，可以使一系列程序或功能块周期性地执行或由一个特定的事件触发开始执行。

在设备树中有“任务管理器”选项卡，使用它除了声明特定的 PLC_PRG 程序外，还可以控制工程内其他子程序的执行处理。任务是用于规定程序组织单元在运行时的属性，它是一个执行控制元素，具有调用的能力。在一个任务配置中可以建立多个任务，而一个任务中，可以调用多个程序组织单元，一旦任务被设置，它就可以控制程序周期执行或者通过特定的事件触发开始执行。

在任务配置中，用名称、优先级和任务的启动类型来定义它。这启动类型可以通过时间（周期的，随机的）或通过内部或外部的触发任务时间来定义，例如使用一个布尔型全局变量的上升沿或系统中的某一特定事件。对每个任务，可以设定一串由任务启动的程序。如果在当前周期内执行此任务，那么这些程序会在一个周期的长度内被处理。优先权和条件的结合将决定任务执行的时序，任务设置界面如图 6-2 所示。

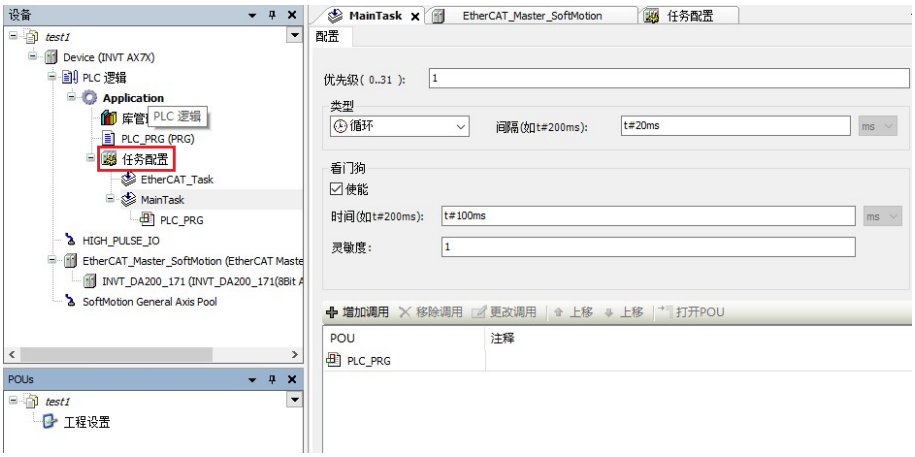


图 6-2 任务配置界面

编程者需遵循如下规则：

✧ 循环任务的最大个数为 100。

✧ 自由运行任务的最大个数为 100。

✧ 事件触发任务的最大个数为 100。

✧ 根据目标系统，PLC_PRG 可能会在任何情况下作为一个自由程序执行，而不用手动插入任务配置中。

✧ 处理和调用程序是根据任务编辑器内自上而下的顺序所执行的。

6.3 程序执行过程

下图详细的描述了在 AX7x 系列可编程控制器内部执行程序的完整流程，主要的流程有输入采样、程序执行和输出刷新这三个部分组成。

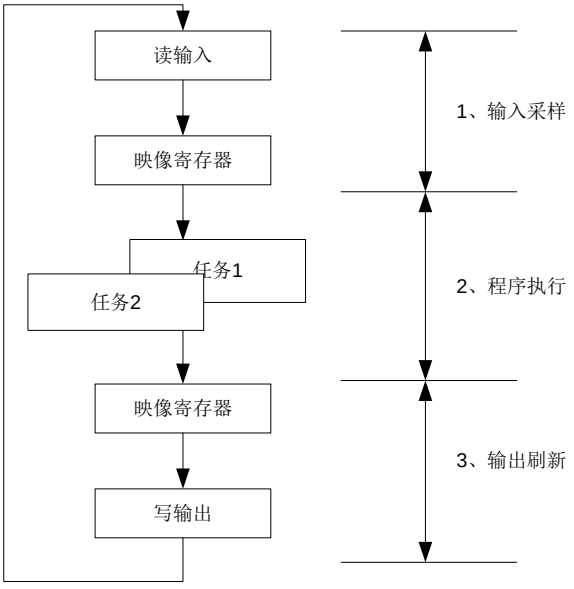


图 6-3 控制器执行流程

1）输入采样

每次扫描周期开始时，AX7x 可编程控制器检测输入设备（开关、按钮等）的状态，将状态写入输入映像寄存区内。在程序执行阶段，运行系统从输入映像区内读取数据进行程序解算。需要特别注意的是输入的刷新只发生在一个扫描开始阶段，在扫描过程中，即使输出状态改变，输入状态也不会发生变化。

2）执行程序

在扫描周期的执行程序阶段，AX7x 可编程控制器从输入映像区或输出映像区内读取状态和数据，并依照指令进行逻辑和算术运算，运算的结果保存在输出映像区相应的单元中。在这一阶段内，只有输入映像寄存器的内容保持不变，其他映像寄存器的内容会随着程序的执行而变化。

3）输出刷新

输出刷新阶段亦称为写输出阶段，AX7x 可编程控制器将输出映像区的状态和数据传送到输出点上，并通过一定的方式隔离和功率放大，驱动外部负载。AX7x 可编程控制器在一个扫描周期内除了完成上述三个阶段的任务外，还要完成内部诊断、通信、公共处理以及输入/输出服务等辅助任务。

AX7x 可编程控制器重复执行上述 1）至 3）的过程，每重复一次的时间就是一个工作周期（或扫描周期）。由 AX7x 可编程控制器的扫描方式可得知，AX7x 可编程控制器为了迅速响应输入输出数据的变化，完成控制任务扫描时间较短，控制器 的工作周期一般都控制在 ms 数量级，因此需要开发稳定、可靠、响应快的实时系统供 AX7x 可编程控制器运行系统所用。

由于 AX7x 可编程控制器采用循环的工作方式，输入信号只会在每个周期的开始阶段进行刷新，输出在每个工作周期的结束阶段进行集中输出，所以必然会产生输出信号相对输入信号的滞后现象。从 AX7x 可编程控制器的输入端有一个信号输入发生到变化到控制器 的输出端对该输入信号的变化做出反应需要一段时间。滞后延时时间是设计 AX7x 可编程控制器控制系统时应了解的一个重要参数。通常，滞后延时时间的长短和以下因素有关：

✧ 输入电路的滤波时间，它由硬件 RC 滤波电路的时间常数决定，通过改变时间常数可调整输入延迟时间。如表 6-1 为 AX-EM-1600D 数字量输入模块技术参数，其中“端口滤波时间”表示该输入模块的滤波时间。

表 6-1 AX-EM-1600D 数字量输入模块规格参数

项目	规格
输入通道	16
输入连接方式	18 点接线端子
输入电压等级	24V(最高可达 30V)
输入电流（典型）	4.7mA
ON 电压	>15VDC
OFF 电压	<5VDC
端口滤波时间	10ms
输入阻抗	5.4kΩ
输入信号形式	电压直流输入
隔离方式	光耦隔离
输入动态显示	输入有效时，指示灯亮

✧ 输出电路的滞后时间，与输出电路的方式有关系，继电器输出方式的滞后时间一般为 10ms 左右，晶体管输出方式滞后时间小于 1ms。

✧ 控制器循环扫描的工作方式。

✧ 用户程序中语句的安排。

为了能够让读者更好的理解这整个过程，如下通过一个简单的梯形图程序例子来反应其输入输出及滞后现象的工作原理。程序逻辑如图 6-4 所示。

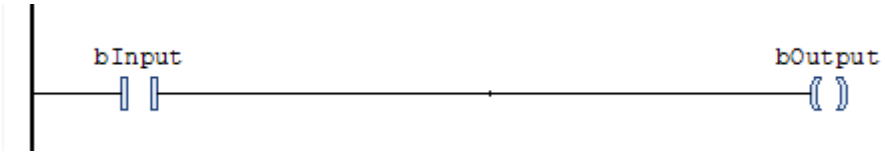


图 6-4 AX7x 可编程控制器程序

bInput 与外部的输入按钮有硬件映射关系，当按钮被按下时 bInput 为 ON ；bOutput 与外部继电器的线圈也有硬件映射关系，当 bOutput 为 ON 时，继电器的线圈也会得电。在 AX7x 可编程控制器的内部，其处理的关系如图 6.6 所示，当输入按钮按下时， bInput 不会马上被置为 ON，因为输入采样只有在一个工作周期的开始阶段才能被程序执行，由于该按钮信

号已经过了采样阶段，通常会在下一个周期开始阶段才被执行。在图 6.6 的程序中将 bInput 的状态赋值给 bOutput，由于在程序运行期间存在一定的程序计算，所以需要一定的程序处理时间 bOutput 才会被置为 ON。由于输出刷新是发生在程序处理的最后阶段，故在该周期的最后阶段 bOutput 通过输出刷新功能将其数值传递至实际硬件，最终线圈才能得电。下图是比较理想状态，最终的输出只有一个周期的延迟。

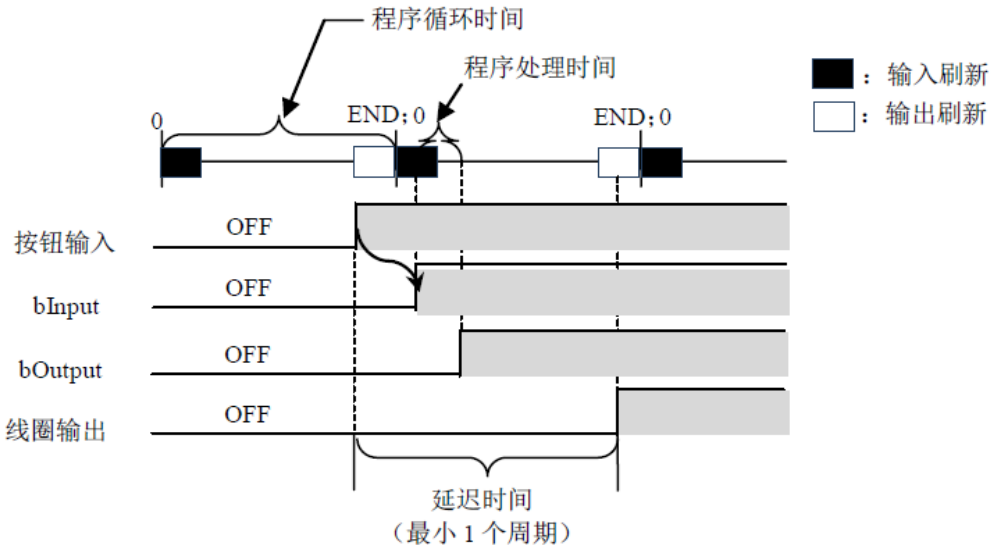


图 6-5 输出最快的情况

上图为比较理想的情况，那么也要考虑到比较糟的情况，当一个周期的输入采样刚刚结束的时候，此时外部输入按钮为 ON，由于需要在下一个周期开始时输入信号才能被载入至输入映像区，而实际输出则要等到第二个周期结束时才能被载入输出映像区，故整个过程如图 6-6 所示，在这种情况输出的延时接近于 2 个周期，这种情况为最迟的输出情况。

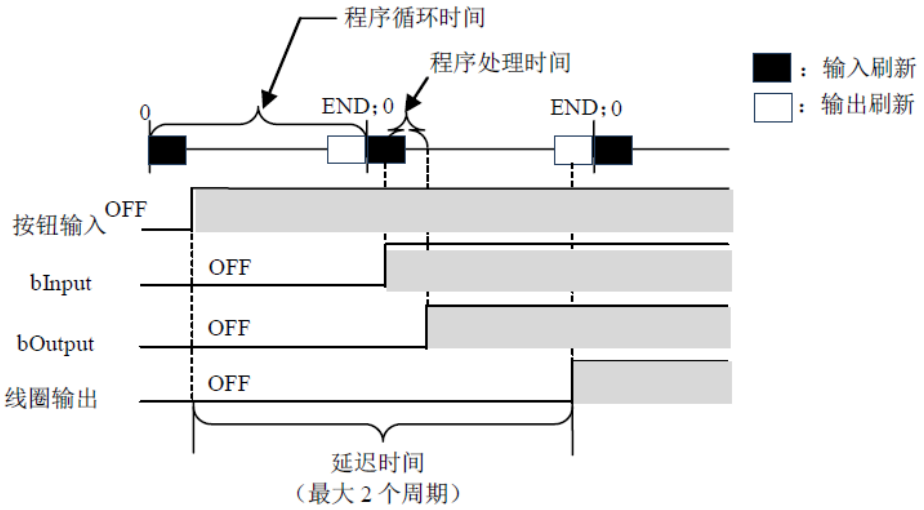


图 6-6 输出最迟的情况

6.4 任务的执行类型

在任务配置树的最顶端有条目“任务配置”。其中的内容是当前定义的任务，每个通过任务名代表。特定任务的 POUs 调用没有显示在任务配置树中。针对每个独立的任务可以对其进行执行的类型编辑及配置。包括固定周期循环、事件触发、自由运行和状态触发 4 种类型。详见图 6-7 所示。

图 6-7 任务执行的类型

1）固定周期循环-Cyclic

根据程序中所使用的指令执行与否，程序的处理时间会有所不同，所以实际执行时间在每个扫描周期都发生不同的变化，执行时间有长有短。通过使用固定周期循环方式，能保持一定的循环时间反复执行程序。即使程序的执行时间发生变化，也可以保持一定的刷新间隔时间。在这里，也推荐大家优先选择固定周期循环任务启动方式。例如，假设将程序对应的任务设定为固定周期循环方式，间隔时间设定为 10ms 时，实际程序执行的时序图如图 6-8 所示。

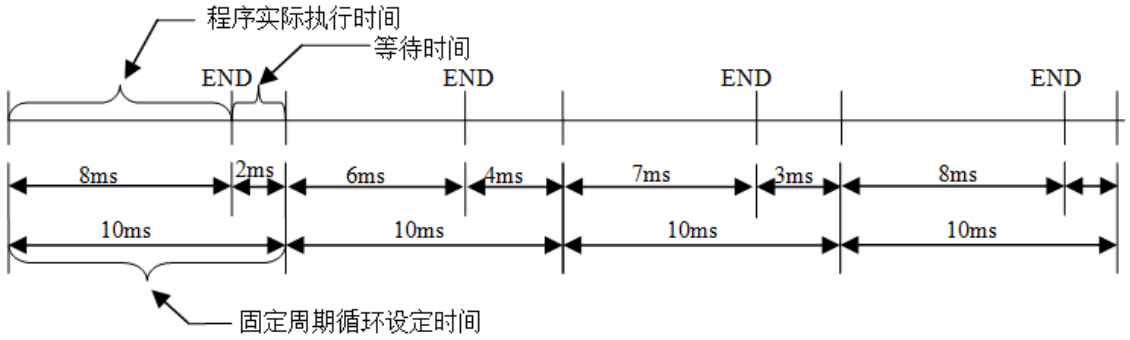


图 6-8 固定周期循环执行顺序

如果程序实际执行时间在规定的固定周期循环设定时间内执行完，则空余时间用作等待。如应用中还有优先级较低的任务未被执行，则剩下的等待时间用来执行相对低优先级的任务。任务的优先级在后文会有详细的说明。

2）自由运行-Freewheeling

程序一开始运行任务就被处理，一个运行周期结束后任务将在下一个循环中被自动重新启动。该执行方式不受程序扫描周期的影响。即确保每次执行完程序的最后一条指令后才进入下一个循环周期。否则不会结束该程序周期。图 6-9 为自由运行执行顺序的时序图。

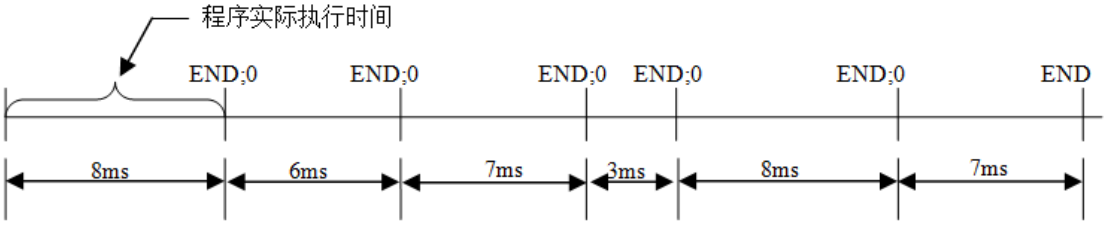


图 6-9 自由运行执行顺序

由于自由运行执行方式因为没有固定的任务时间，所以每次执行的时间可能都不一样。故不能保证程序的实时性，在实际的应用中选用此方式的场合较少。

3）事件触发-Event

如果事件区域的变量得到一个上升沿，任务开始。

4）状态触发-Status

如果事件区域的变量为 TRUE，任务开始。状态触发方式与事件触发功能类似，区别在于状态触发的触发变量只要为 TRUE 程序就执行，为 FALSE 则不执行。而事件触发只采集触发变量的上升沿有效信号。图 6-10 中针对事件触发和状态触发分别进行了比较，绿色实线为两种触发方式选择的布尔变量状态，表 6-2 为比较的结果。

图 6-10 任务输入触发信号

在采样点 1~4（紫色）不同类型的任务展示了不同的反应。这个具体的事件为 TRUE 完成了状态驱动任务的条件，然而一个事件驱动任务需要事件从 FALSE 变为 TRUE。如果任务计划的采样频率过低，事件的上升沿可能检测不到。

表 6-2 事件触发与状态触发执行结果比较

执行点	1	2	3	4
事件触发（Event）	不执行	执行	执行	执行
状态触发（Status）	不执行	执行	不执行	不执行

6.5 任务优先级

1）任务优先级设置

可以对任务的优先级进行设置，一共可以设 32 个级别（0～31 之间的一个数字，0 为最高优先级，31 为最低优先级）。当一个程序在执行时，优先级高的任务优先于优先级任务低的任务，高优先级任务 0 能中断同一资源中较低优先级的程序执行，使较低优先级程序执行被放缓。

注意：在任务优先级等级分配时，请勿分配具有相同优先级的任务。如果还存在其他任务视图先于具有相同优先级的任务，则结果可能不确定且不预知。

如果任务的类型为“循环”，则按照“间隔”中的时间循环执行，具体设置如下图 6-11 所示。

图 6-11 固定周期循环配置图

例：假设有 3 个不同的任务，分别对应三种不同的优先等级，具体分配如下。

：任务 1 具有优先级 0 和 循环时间 10 ms。

：任务 2 具有优先级 1 和 循环时间 30 ms。

：任务 3 具有优先级 2 和 循环时间 40 ms。

在控制器内部，各任务的时序关系如图 6-12 所示，具体说明如下：

0~10ms：先执行任务 1（优先级最高），如在本周期内已将程序执行完，剩余时间执行任务 2 程序。但是如果此时任务 2 没有被完全执行完，但时间已经到了第 10ms 时，由于任务 1 是每 10ms 执行一次的且优先级更高，此时将会打断任务 2 的执行。

10~20ms：先将任务 1 的程序执行完毕，如有剩余时间，再执行上个周期未完成的任务 2。

20~30ms：由于任务 2 是每 30ms 执行一次的，在 10～20ms 之间任务 2 已经全部执行完毕，此时不需要再执行任务 2，只需将优先级最高的任务 1 执行一次即可。

30~40ms：与之前类似。

40~50ms：此时出现了任务 3，任务 3 的优先级更低，所以只有在确保任务 2 彻底执行完后，才能执行任务 3。

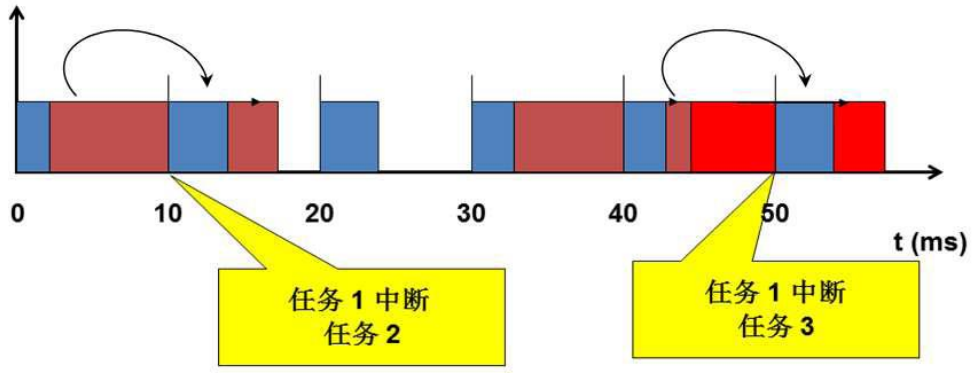


图 6-12 任务优先级中断执行顺序

2）AX7x 任务的优先级安排

AX7x 控制器的上位机软件创建新的标准工程时在任务配置中默认创建 MainTask，其优先级默认为 0。新创建的任务其优先级默认的也为 0，，但是为确保对运动控制等重要任务优先执行，在一些需要高性能运动控制（MC）的应用中，可让控制器的性能得到合理利用。任务优先级顺序设置推荐使用如下顺序（若仅有一个任务，此任务优先级可随意设置）：

表 6-3 任务优先级安排

任务类型	推荐优先级
PlcCfg 模块	31
ModbusTCP	15~30
ModbusRTU	15~30
高速 IO	1~15
模拟量输入输出	1~15
温度模块	1~15
EtherCAT	0

优先级设定值越小，优先级越高；高优先级的 POU 可以打断低优先级 POU 的执行，如图 6-13，其中 ECT 代表 EtherCAT。

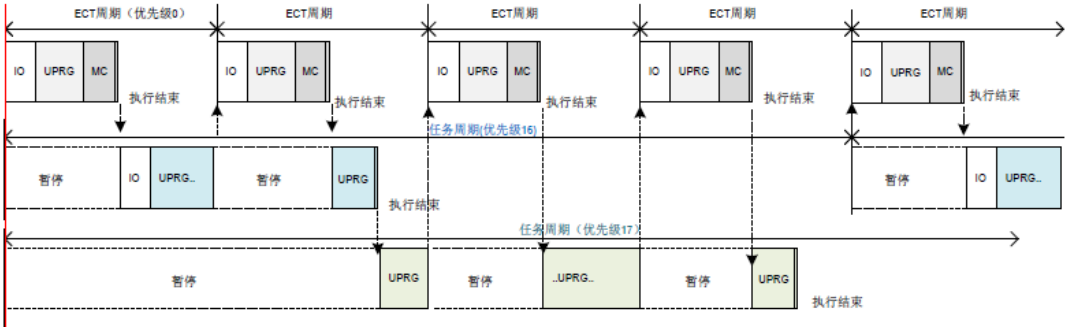


图 6-13 POU 执行顺序

从图 6-13 可知：

控制器执行任务时，有一个用户观察不到的时间对准点，如上图左侧，在这个时间点开始，按最高优先级->次高优先级->最低优先级的顺序开始执行。

低优先的任务在执行时，可能被高优先级的任务打断，待高优先级任务执行完成后，返回被打断的任务，继续执行该低优先级任务。

EtherCAT 任务为最高优先级任务，按 EtherCAT 周期进入该任务，完整执行一遍该任务内的所有 POU 后，才返回较低优先级任务。

3）任务配置中的执行周期设定的要求

AX7x 系统上位机软件采用多任务方式来执行用户程序的“任务”，而每个“任务”分配了不同的执行周期，有些全局变量可能要在不同的 POU 之间被访问和修改， 于是需要对全局变量进行交互同步，也是在任务的“时间对准点”进行的，在设置循环类型任务的周期时，不同类型的循环任务循环时间呈整数倍数的关系。

比如，设 EtherCAT 任务周期为 4ms、8ms，而设普通循环任务的周期为 400ms，更低优先级的循环任务周期为 100ms 或 200ms 等等。EtherCAT 周期不要设置为 5ms、7ms、9ms 等，容易造成非 2 的整数倍的关系。

4）子设备总线循环任务（Bus Cycle Options）配置注意事项

在控制器设备“PLC 设置 > 总线周期 > 总线周期任务”选项中，总线周期任务：选项列表提供了当前有效工程中任务配置里已定义的任务（例如"MainTask"，"EtherCAT Master"等）。选择其中一个任务作为当前工程的总线周期，或者选择选项<未标明的>，该选项意味着将应用最短的任务周期时间，也就是最快的执行周期。您可以切换到另一种设置，但是请务必注意下列事项：

注意：在修改“<unspecified>”设置之前，您已经清楚其影响。“<unspecified>”是指，将应用由设备描述定义的缺省行为。所以请检查这方面的相关描述：缺省方式下，该任务可能被定义为具有最短周期时间，但也可能具有最长周期时间。

因此在使用扩展模块以及 EtherCAT 模块时，为提高系统运行稳定性（尤其在使用 EtherCAT_Master_SoftMotion 模块时），在“EtherCAT I/O 映射->总线循环选项”中选择各模块对应的任务，参考例程如图 6-14。

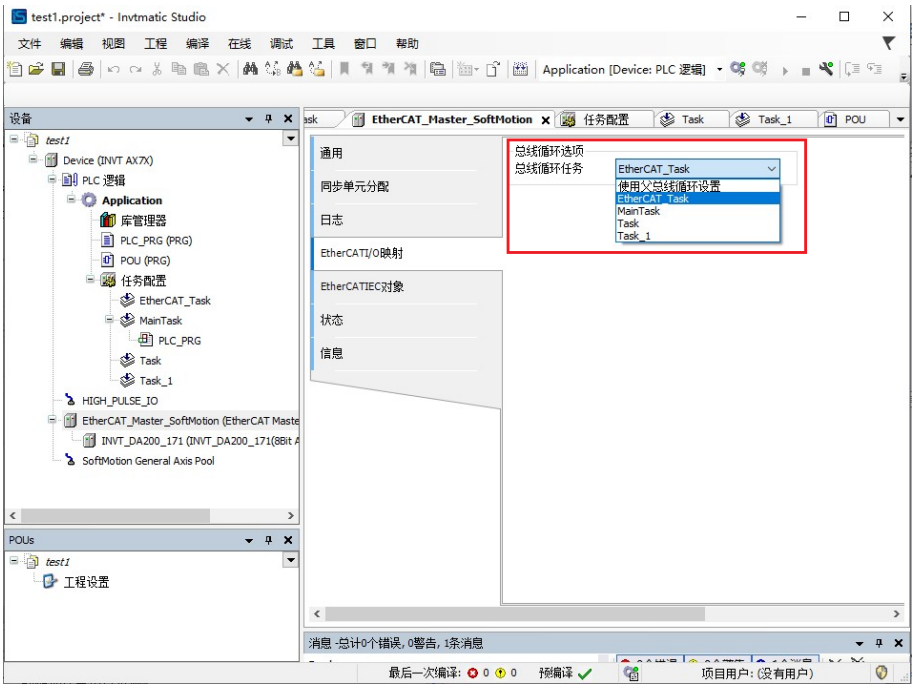


图 6-14 EtherCAT 总线循环任务设置示例

6.6 多子程序的运行

在实际的工程项目中，通常可以将程序按控制流程或者按照设备的对象分割成很多子程序，据此，设计人员将可以按各处理单元分别进行编程。如下图 6-15，以控制流程将主程序拆分为多个不同流程的子程序，拆分的目的主要是使主程序条理更清晰，并且方便今后的调试。

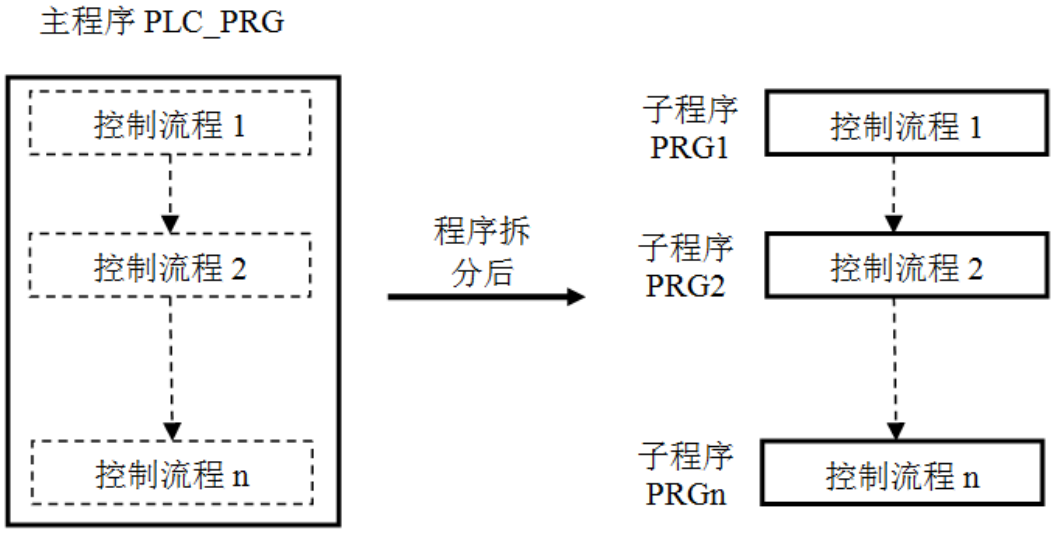


图 6-15 按流程拆分多子程序

图 6-15 中右半部份是按流程进行分类的各子程序 PRG1，PRG2..PRGn，图的左半部分为主程序 PLC_PRG，在主程序中可以分别调用 PRG1..PRGn 子程序。多子程序运行的方式有两种，第一种在任务配置中添加子程序。第二种方法是在主程序中调用子程序，也是比较常用及灵活的一种方式，如下，分别对两种方式的进行详细说明。

1）任务配置中添加子程序

用户可以通过在任务配置页面中添加子程序实现多程序的运行。按子程序的执行顺序依次点击“增加调用”进行添加子程序。如图 6-16 所示，添加后，对应的任务即按用户所指定的从上至下的顺序循环执行，也可以通过“上移”和“下移”功能再对顺序进行手动编辑。

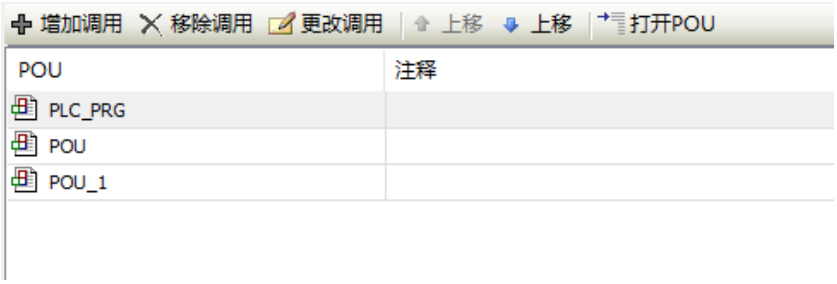


图 6-16 任务中添加子程序

2）主程序 **PLC_PRG** 中调用子程序

PLC_PRG 被系统默认为主程序，从某种意义上可以理解为汽车的电瓶，在生产汽车时，将各个零件进行组装，相当于子程序的编写；当汽车组装完成时，就要检查汽车是否可用，如果想启动汽车，就必须通过电瓶来启动汽车的各个部件，如发动机、车灯等，电瓶就相当于启动汽车的入口点。通过这样调用程序使操作性更强及使程序更灵活，能够在程序中加入判断语句等，且能实现嵌套。

PLC_PRG 是一个特殊的 POU，其默认的运行方式为“惯性滑行”。系统默认每个控制周期调用该 POU，用户不需要对其进行额外的任务配置，在任务配置中也可看该 POU 相应的配置。用户可以通过它来实现对其他子程序的调用，更可以在调用时添加必要的条件选择，或实现子程序嵌套，使程序调用更为灵活。如果要实现图 6-17 中的调用关系，可以在主程序 PLC_PRG 中写入如下代码。

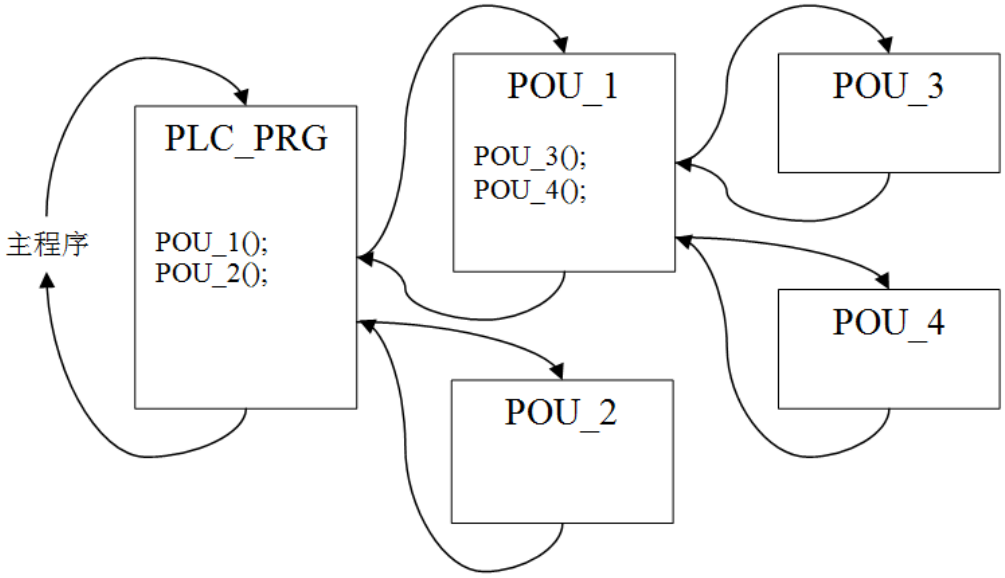


图 6-17 POU 调用顺序

如图 6-17 中所示，主程序为PLC_PRG，该主程序使用的是结构化文本编程语言，其中的程序的内容为POU_1()；POU_2()；

上述程序的主要功能是分别调用执行了 POU_1 和 POU_2 子程序。而 POU_1 中又分别调用了 POU_3 和 POU_4，实际 AX7x 可编程控制器内部实际按如下顺序执行程序：

a) AX7x 可编程控制器先执行子程序 POU_1。

b) 由于 POU_1 中依次调用了 POU_3 和 POU_4，故先执行 POU_3。

c) 执行 POU_4，POU_1 执行完成。

d) 最后执行 POU_2，完成一个完整任务周期。

重复上述步骤 a）至 d）即为 AX7x 系列可编程控制器内部的执行顺序。

第7章 EtherCAT 总线运动控制

7.1 EtherCAT 运行原理

7.1.1 协议介绍

EtherCAT 技术突破了其他以太网解决方案固有的局限性：一方面，无需像其它方案那样接收以太网数据包，将其解码，之后再将过程数据复制到各个设备。EtherCAT 从站设备在报文经过其节点时读取带有相应寻址信息的数据；同样，输入数据也是在报文经过时插入至报文中。整个过程中，报文只有几纳秒的时间延迟。

由主站发出的帧被传输并经过所有从站，直到网段（或分支）的最后一个从站。当最后一个设备检测到其开放端口时，便将帧返回给主站。另一方面，由于发送和接收的以太网帧压缩了大量的设备数据，所以可用数据率可达 90%以上。100 Mb/s TX 的全双工特性完全得以利用，因此，有效数据率可以达到 > 100 Mb/s （> 2 x 100 Mb/s 的 90%）。

EtherCAT 主站采用标准的以太网介质存取控制器（MAC），而无需额外的通讯处理器。因此，任何集成了以太网接口的设备控制器都可以实现 EtherCAT 主站，而与操作系统或应用环境无关。EtherCAT 从站采用 EtherCAT 从站控制器（EtherCAT Slave Controller，ESC）来高速动态地（onthe-fly）处理数据。网络的性能并不取决于从站使用的微处理器性能，因为所有的通讯都是在 ESC 硬件中完成的。过程数据接口（Process Data Interface，PDI）为从站应用层提供了一个双口随机存储器（Dual-Port-RAM,DPRAM）来实现数据交换。

精确同步在广泛要求同时动作的分布过程中显得尤为重要，如几个伺服轴在执行同时联动任务时。分布时钟的精确校准是同步的最有效解决方案。在通讯系统中，和完全同步通讯相比，分步式校准时钟在某种程度上具备错误延迟的容错性。

7.1.2 工作计数器 WKC

每个 EtherCAT 报尾拥有一个 16 位的工作计数器 WKC。WKC 是用于记录对 EtherCAT 从站设备读写次数的工作计数器，EtherCAT 从站控制器在硬件中计算 WKC，主站接收到返回数据后检查子报文中的 WKC，如果不等于预期值，则表示该子报文没有被正确的处理。子报文经过某一个从站时，如果是单独的读或写操作，WKC 加 1。如果是读写操作，读成功时 WKC 加 1，写成功时 WKC 加 2，全部完成时 WKC 加 3。WKC 为各个从站处理结果的累加。关于 WKC 增量的描述如表 7-1 所示。

表 7-1 WKC 增量

指令	数据类型	增量
读指令	读取失败	无变化
	成功读取	+1
写指令	写入失败	无变化
	成功写入	+1
读/写指令	不成功	无变化
	成功读取	+1
	成功写入	+2
	成功读写	+3

7.1.3 寻址方式

EtherCAT 通信由主站发送 EtherCAT 数据帧读写从站设备的内部存储区来实现，EtherCAT 报文使用多种寻址方式来操作 ESC 内部存储区实现多种通信服务。EtherCAT 的寻址方式如图 7.1 所示。一个 EtherCAT 网段相当于一个以太网设备，主站首先使用以太网数据帧头的 MAC 地址寻址到网段，然后使用 EtherCAT 子报文头中的 32 个位地址寻址到段内设备。段内寻址可以使用两种方式：设备寻址和逻辑寻址。设备寻址针对某一个从站进行读写操作。逻辑寻址面向过程数据，可以实现多播，同一个子报文可以读写多个从站设备。

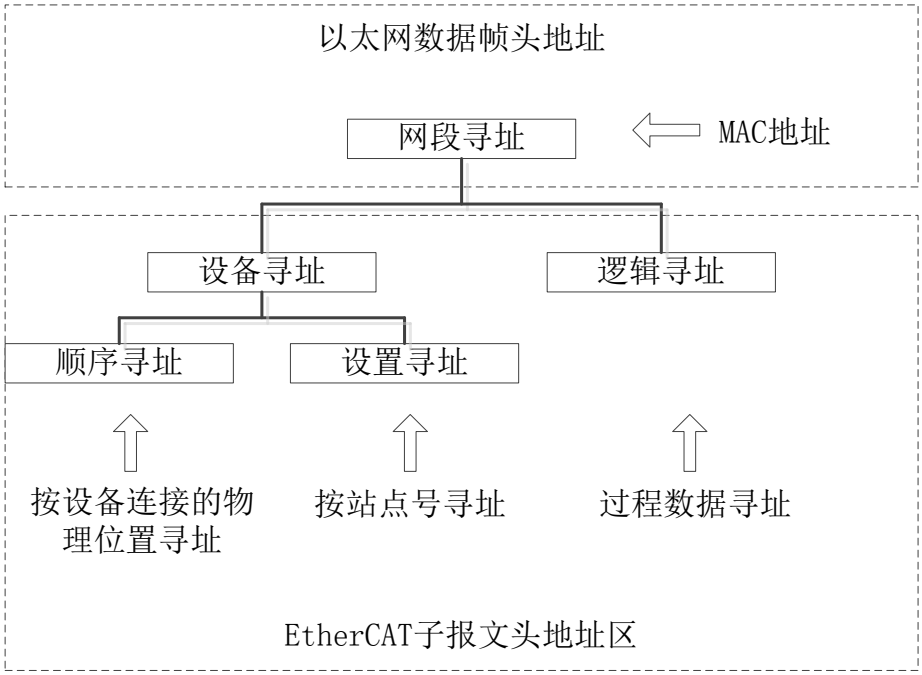


图 7-1 EtherCAT 网络寻址模式

7.1.3.1 网段寻址

根据 EtherCAT 主站及其网段的连接方式不同，可以使用如下两种方式寻址到网段：

✧ 直连模式

一个 EtherCAT 网段直接连接到主站设备的标准以太网端口，如图 7-2 所示。此时，主站使用广播 MAC 地址，EtherCAT 数据帧如图 7-3 所示。

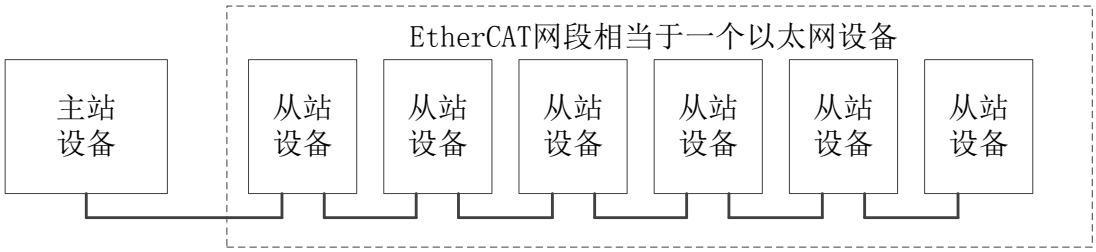


图 7-2 直连模式中的 EtherCAT 网段

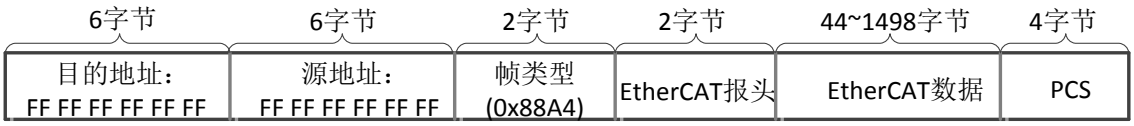


图 7-3 直连模式中的 EtherCAT 网段寻址地址内容

✧ 开放模式

EtherCAT 网段连接到一个标准以太网交换机上，如图 7-4 所示。此时，一个网段需要一个 MAC 地址，主站发送的 EtherCAT 数据帧中的地址是它所控制的网段的 MAC 地址，如图 7-5 所示。EtherCAT 网段内的第一个从站设备有 ISO/IEC 8802.3 的 MAC 地址，这个地址表示了整个网段，这个从站称为段地址从站，它能够交换以太网内的目的地址区和源地址区。如果 EtherCAT 数据帧通过 UDP 传送，这个设备也会交换源和目的的 IP 地址、以及源和目的的 UDP 端口号，使响应的数据帧完全满足 UDP/IP 协议标准。

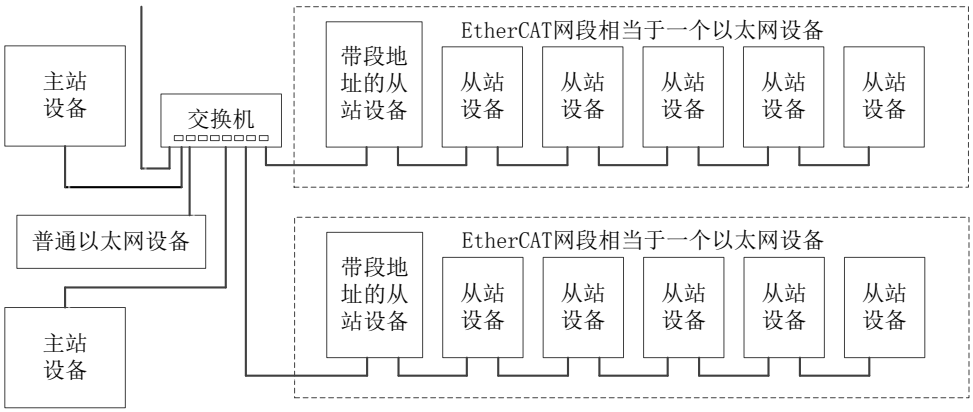


图 7-4 开放模式中的 EtherCAT 网段

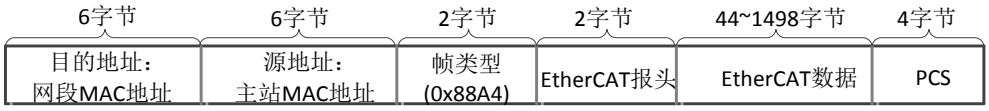


图 7-5 开放模式中的 EtherCAT 网段寻址地址内容

7.1.3.2 设备寻址

在设备寻址时，EtherCAT 子报文头内的 32 位地址分为 16 位从站设备地址和 16 位从站设备内部物理存储空间地址，如图 7-6 所示。16 位从站设备地址可以寻址 65535 个从站设备，每个设备内最多可以有 64 个本地地址空间。

设备寻址时，每个报文只寻址唯一的一个从站设备，但它有两种不同的设备寻址机制。

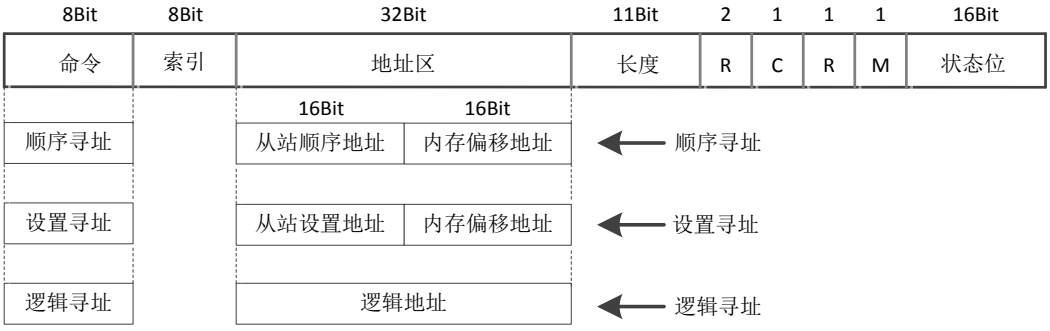


图 7-6 EtherCAT 的设备寻址结构

✧ 顺序寻址

顺序寻址时，从站的地址由其在网段内的连接位置确定，用一个负数来表示每个从站在网段内由接线顺序决定的位置。顺序寻址子报文在经过每个从站设备时，其顺序地址加 1；从站在接收报文时，顺序地址为 0 的报文就是寻址到自己的报文。由于这种机制在报文经过时更新设备地址，所以又被称为“自动增量寻址”。

图 7-7 中，网段中有 3 个从站设备，其顺序寻址的地址为 0、-1、-2 以此类推。主站使用顺序寻址访问从站时子报文的地址变化如图 7.8 所示。主站发出 3 个子报文分别寻址 3 个从站，其中的地址分别是 0、-1 和-2，如图数据帧为 1。数据帧达到从站①时，从站①检查到子报文 1 中的地址为 0，从而得知子报文 1 就是寻址到自己的报文。数据帧经过从站①后，所有的顺序地址都增加 1，称为 1、0 和-1，如图 7.8 中的数据帧 2。到达从站②时，从站②发现子报文 2 中的顺序地址为 0，即为自己的报文。同理，后续的从站都按此方法来寻址。如图 7.7. 在实际工程应用中，顺序寻址主要用于启动阶段，主站配置站点地址给各个从站。此后，可以使用与物理位置无关的站点地址来寻址从站。使用顺序寻址机制能自动为从站设定地址。如图 7-8。

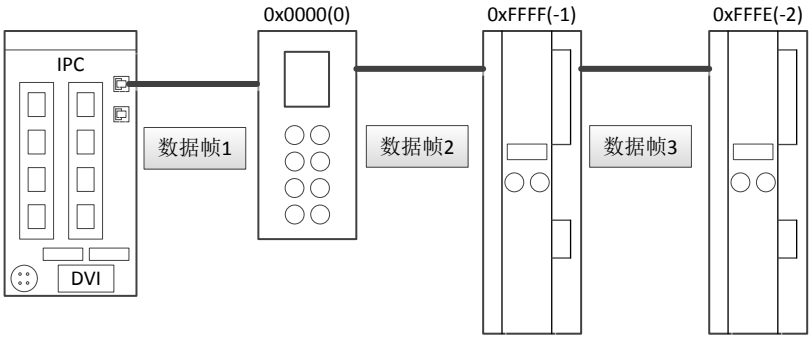


图 7-7 顺序寻址的从站地址

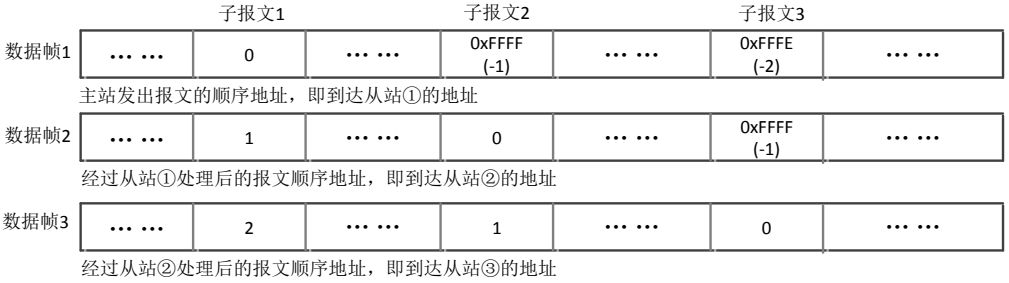


图 7-8 顺序寻址时子报文地址的变化

✧ 设置寻址

设置寻址时，从站的地址与其在网段内的连续顺序无关。如图 7-9 所示，地址可以有主站在数据链路启动阶段配置给从站，也可以由从站在上电初始化阶段的配置数据装载，然后由主站在链路启动阶段使用顺序寻址方式读取各个从站的设置地址。其报文结构如图 7-10 所示。

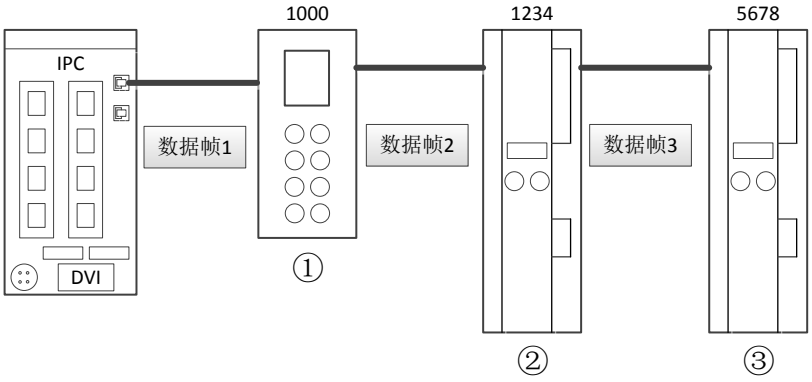


图 7-9 设置寻址时的从站地址

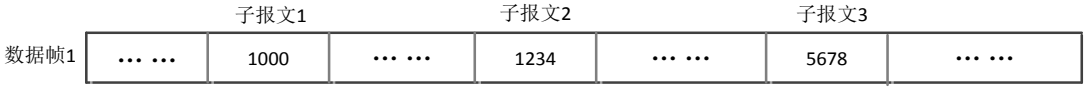


图 7-10 设置寻址时的报文结构

✧ 逻辑寻址

逻辑寻址时，从站地址并不是单独定义的，而是使用寻址段内 4GB 逻辑地址空间中的一段区域。报文内的 32 位地址区作为整体数据逻辑地址完成设备的逻辑寻址。逻辑寻址方式由现场总线内存管理单元（FMMU，Fieldbus Memory Management Unit）实现，FMMU 功能位于每一个 ESC 内部，将从站本地物理存储地址映射到网段内逻辑地址，其原理图如图 7-11 所

示。

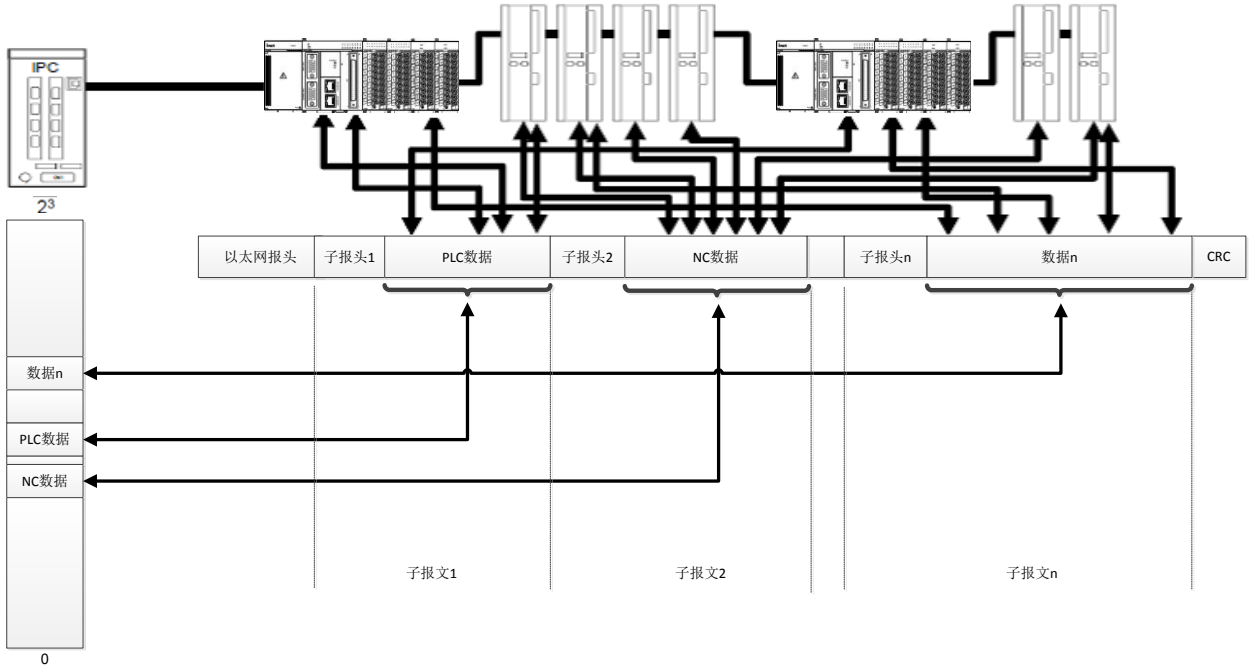


图 7-11 现场总线内存管理单元（FMMU）运行原理

从站设备收到一个数据逻辑寻址的 EtherCAT 子报文时，检查是否有 FMMU 单元地址匹配。如果有，他将输入类型数据插入到 EtherCAT 子报文数据区的对应位置，以及从 EtherCAT 子报文数据区的对应位置抽取输出类型数据。

7.1.4 分布时钟

7.1.4.1 分布时钟概念

精确同步对于同时动作的分布式过程而言尤为重要。例如，几个伺服轴同时执行协调运动时，便是如此。分布时钟机制能够使所有的从站都同步于一个参考时钟。主站连接的第一个具有分布时钟功能的从站作为参考时钟，以参考时钟来同步其他设备和主站的从时钟。为了实现精确的时钟同步控制，必须测量和计算数据传输延时和本地时钟偏移，并补偿本地时钟的漂移。同步时钟所涉及到如下 6 个概念。

✧ 系统时间

系统时间是分布时钟使用的系统计时。系统时间从 2001 年 1 月 1 日零点开始，使用 64 位二进制变量表示，单位为纳秒（ns），最大可以计时 500 年。也可以使用 32 位二进制变量表示，32 位时间值最大可以表示 4.2s，通常用于通信和时间戳。

✧ 参考时钟和从时钟

EtherCAT 协议规定主站连接的第一个具有分布时钟功能的从站作为参考时钟，其他从站的时钟称为从时钟。参考时钟被用于同步其他从站设备的从时钟和主站时钟。参考时钟提供 EtherCAT 系统时间。

✧ 主站时钟

EtherCAT 主站也具有计时功能，称为主站时钟。主站时钟可以在分布时钟系统中作为从站时钟被同步。在初始化阶段，主站可以按照系统时间的格式发送主站时钟给参考时钟从站，使分布时钟使用系统时间计时。

✧ 本地时钟、其初始偏移量和时钟漂移

每一个 DC 从站都有本地时钟，本地时钟独立运行，使用本地时钟信号计时。系统起动时，各从站的本地时钟和参考时钟之间有一定的差值，称为时钟初始偏移量。在运行过程中，由于参考时钟和 DC 从站时钟使用各自的时钟源等原因，它们的计时周期存在一定的漂移，这将导致时钟运行不同步，本地时钟产生漂移。因此，必须对时钟初始偏移和时钟漂移进行补偿。

✧ 本地系统时间

每个 DC 从站的本地时钟经过补偿和同步之后都产生一个本地系统时间，分布时钟同步机制就是使各个从站的本地系统时间保持一致。参考时钟也是相应从站的本地系统时钟。

✧ 传输延时

数据帧在从站之间传输时会产生一定的延迟。其中包括设备内部和物理连接延迟。所以在同步从站时钟时，应该考虑参考时钟与多个从站时钟之间的传输延时。

7.1.4.2 时钟同步过程

时钟同步有如下三个步骤组成：

a) 传输延时测量

分布时钟初始化时主站会给所有方向的从站初始化传输延时，并计算得到从时钟与参考时钟之间的偏差值，将其写入从站时钟站。

b) 参考时钟偏移补偿（系统时间）

每个从站的本地时钟会与系统时间进行比较，然后将不同的比较结果分别写入不同的从站内，这样所有的从站都会得到绝对的系统时间。

c) 参考时钟漂移补偿

时钟漂移补偿及本地时间是用于定期补偿本地时钟的误差及微调。如下的图形说明了补偿计算的两个应用案例，图 7-12 为系统时间小于从站本地时钟的案例。图 7-13 为大于本地时间的案例。

✧ 系统时间<本地时间

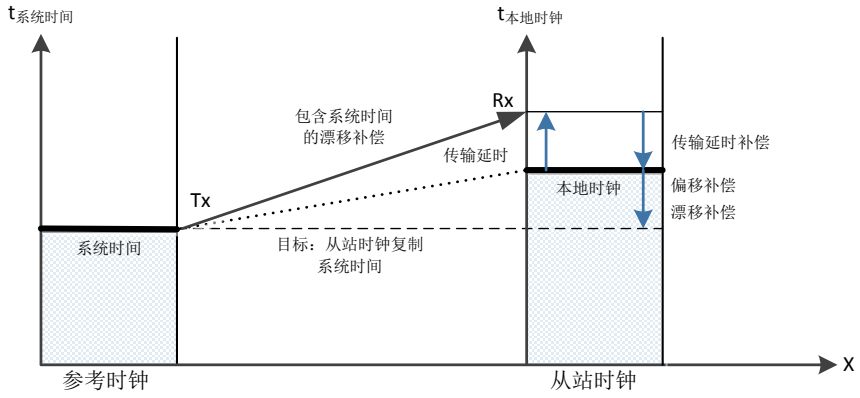


图 7-12 时钟同步过程：系统时间<本地时间

✧ 系统时间 > 本地时间

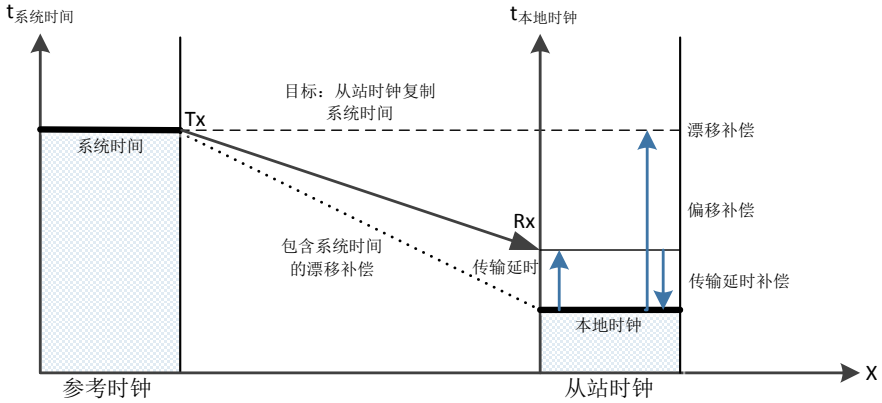


图 7-13 时钟同步过程：系统时间>本地时间

采用 EtherCAT，数据交换就完全基于纯硬件机制。由于通讯采用了逻辑环结构（借助于全双工快速以太网的物理层），主站时钟可以简单、精确地确定各个从站时钟传播的延迟偏移，反之亦然。分布时钟均基于该值进行调整，这意味着可以在网络范围内使用非常精确的、小于 1 微秒的、确定性的同步误差时间基。其结构图如图 7-14 所示。

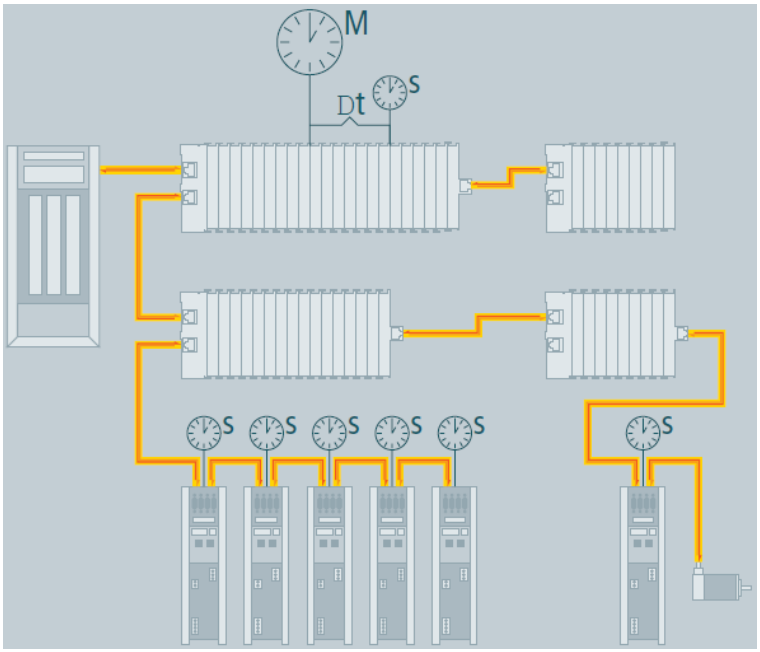


图 7-14 同步时钟原理

比如两设备之间相差 300 个节点，线缆的长度为 120 米，使用示波器抓取其通信信号，其结果如图 7-15 所示。

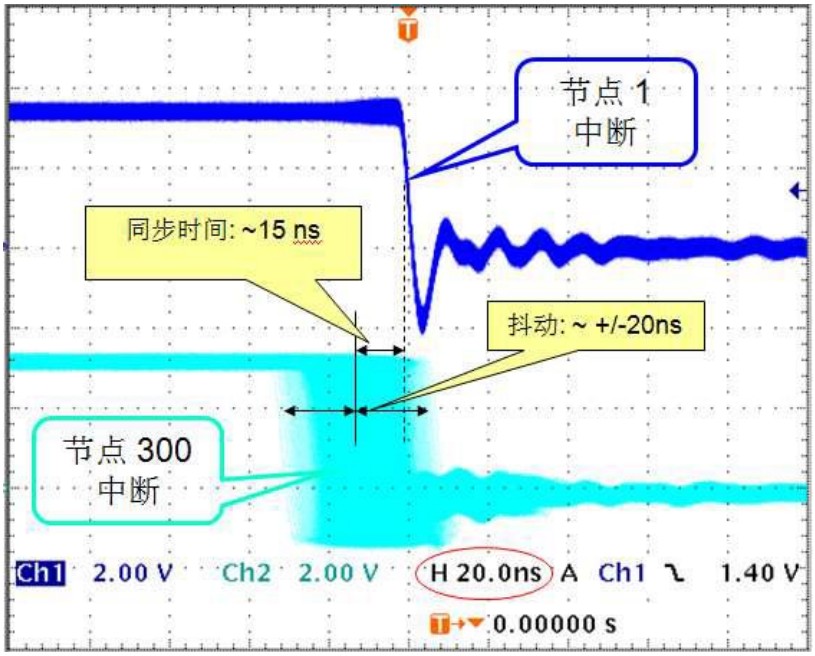


图 7-15 同步时钟性能测试

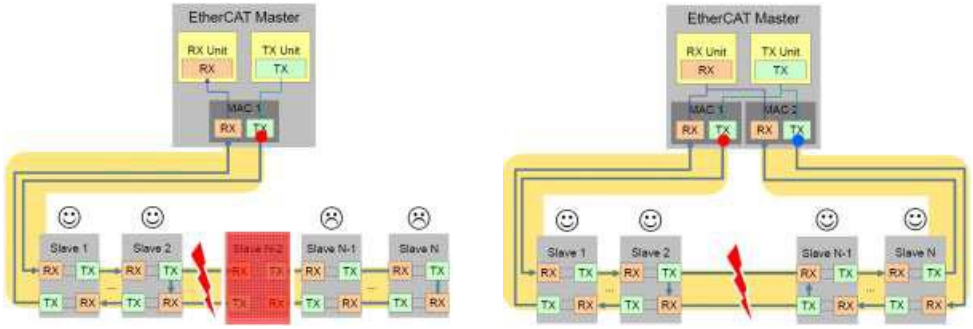
该功能对于运动控制是非常重要的，它通过连续检测到的位置值计算出速度，当采样时间非常短时，即使是位置测量出现一个很小的瞬时抖动，也会导致速度计算出现较大的阶跃变化。在 EtherCAT 中，引入时间戳数据类型作为一个逻辑延伸，可以为测量值附加高分辨率的系统时间，而以太网所提供的巨大带宽使这成为可能。

7.1.5 EtherCAT 线缆冗余

EtherCAT 可选用电缆冗余满足快速增长的系统可靠性需求，它可以保证无需关闭网络即可进行设备更换。增加冗余特性耗费不高，仅需在主站设备端增加一个标准的以太网端口（无需专用网卡或接口）和根电缆，这将线型拓扑结构转变为环型拓扑结构。当设备或电缆发生故障时，也仅需一个周期即可完成切换。因此，即使是针对运动控制要求的应用，电缆出现故障

时也不会有任何问题。

EtherCAT 使用热备份功能支持主站冗余。一旦出现中断、设备故障等问题，EtherCAT 从站控制器可以立即自动返回以太网帧，所以不会导致整个网络关闭。例如，标准 EtherCAT 拓扑结构如图 7-16 的 a）所示，如果在该拓扑结构中 Slave2 与 SlaveN-2 之间出现了网络中断现象，如图中的红色部分，则 Slave N-2 后的所有从站通讯也会相应中断。这也是标准拓扑结构的缺点。



a) 标准 EtherCAT 拓扑结构　　b) EtherCAT 冗余拓扑结构

图 7-16 EtherCAT 冗余

图 7-16 的 b）为 EtherCAT 冗余模式的拓扑结构，主站只需要有两个标准网口即可实现该拓扑结构，使用这两个网口将所有从站构成一条环路，即使在使用过程中网络出现中断，如图 7-16 中红色的部分断开，主站马上会检测到错误，自动将通讯分为两路，所有的从站还能继续通讯，以保障系统的稳定运行。

7.2 EtherCAT 通信模式

在实际自动化控制系统中，应用程序之间通常有两种数据交换形式：时间关键和时间非关键。时间关键表示特定的动作必须在确定的时间窗口内完成。如果不能再要求的时间窗口内完成通讯，则有可能引起控制失效。时间关键的数据通常周期性的发送，称为周期性过程数据通信。非时间关键数据可以非周期性发送，在 EtherCAT 中采用非周期性邮箱（Mailbox）数据通信。

7.2.1 周期性过程数据通信

主站可以使用逻辑读、写或读写命令同时控制多个从站。在周期性数据通信模式下，主站和从站有多种同步运行模式。

1）从站设备同步模式

✧　自由运行

在自由运行模式下，本地控制周期由一个本地定时器中断产生。周期时间可以由主站设定，这是从站的可选功能。自由运行模式的本地周期如图 7-17 所示。其中 T1 为本地微处理器从 EtherCAT 从站控制器复制数据并计算输出数据的时间；T2 为输出硬件延时；T3 为输入锁存偏移时间。这些参数反映了从站的时间响应性能。

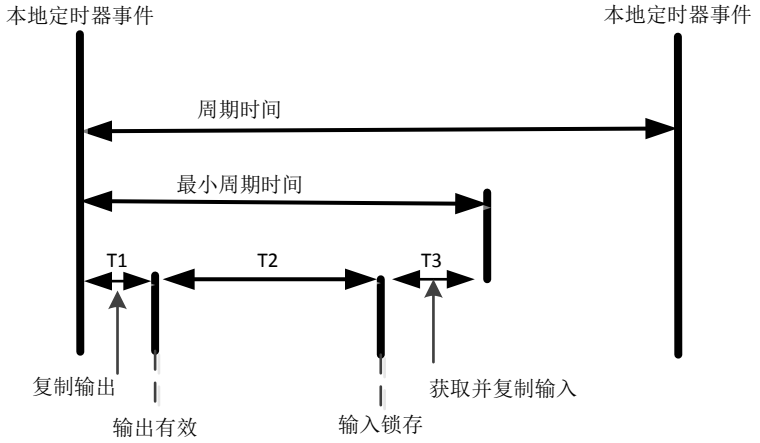


图 7-17 自由运行模式的本地周期

✧ 同步于数据输入或输出事件

本地周期在发生数据输入或输出事件的时候触发，如图 7-18 所示。主站可以将过程数据帧的发送周期写给从站，从站可以检查是否支持这个周期时间或对周期时间进行本地优化。从站可以选择支持这个功能。通常同步与数据输出事件，如果从站只有输入数据，则数据同步于输入事件。

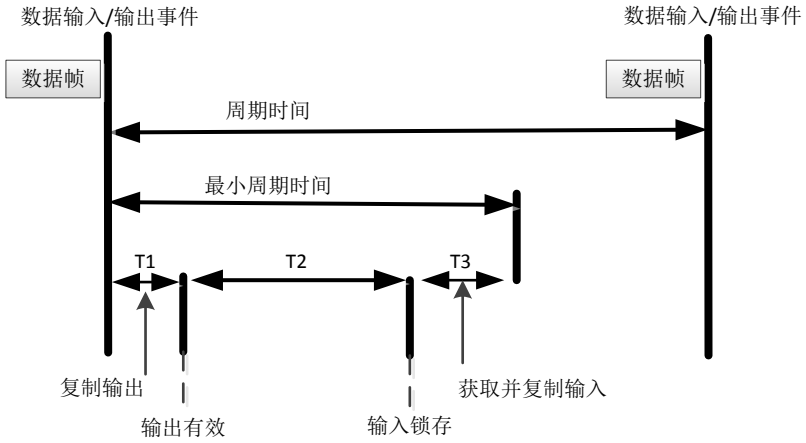


图 7-18 同步于数据输入/输出事件的本地周期

✧ 同步于分布式时钟同步事件

本地周期由 SYNC 事件触发，如图 7-19 所示。主站必须在 SYNC 事件之前完成数据帧的发送，为此要求主站时钟也要同步于参考时钟。

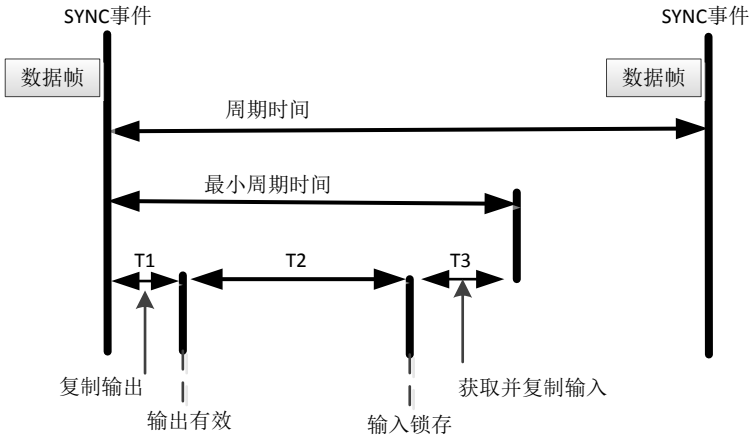


图 7-19 同步于 SYNC 事件的本地周期

为了进一步优化从站同步性能，主站应该在数据收发事件发生时从接收到的过程数据帧复制输出信息。然后等待 SYNC 信号到达后继续本地操作，如图 7-20 所示数据帧必须比 SYNC 信号提前 T1 时间到达，从站在 SYNC 事件之前已经完成数据交换和控制计算，接收 SYNC 信号后可以马上执行输出操作，从而进一步提高同步性能。

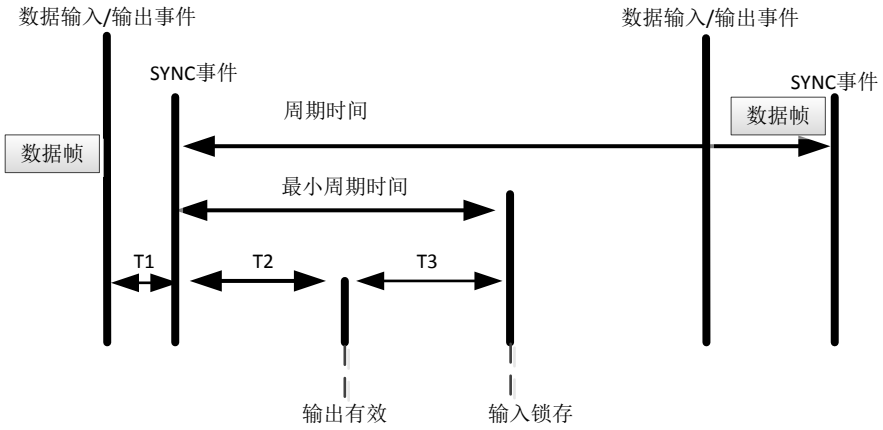


图 7-20 优化的同步于 SYNC 事件的本地周期

2）主站设备同步模式

主站有以下两种同步模式：

✧ 周期性模式

在周期性模式下，主站周期性的发送过程数据帧。主站周期通常由一个本地定时器控制。从站可以运行在自由运行模式或同步于接收数据事件模式。对于运行在同步模式的从站，主站应该检查相应的过程数据帧的周期时间，保证大于从站支持的最小周期时间。

主站可以以不同的周期时间发送多种周期性的过程数据帧，以便获得最优化的带宽。例如，使用比较短的周期发送运动控制数据，比较长的周期用来发送 I/O 数据。

✧ DC 模式

在 DC 模式下，主站运行与周期性模式类似，只是主站本地周期应该和参考时钟同步。主站本地定时器应该根据发布参考时钟的 ARMW 报文进行调整。在运行过程中，用于动态补偿时钟漂移的 ARMW 报文返回主站后，主站时钟可以根据读回的参考时钟时间进行调整，使之大致同步与参考时钟时间。

DC 模式下，所有支持 DC 的从站都应该同步与 DC 系统时间。主站也应该使其他通讯周期同步于 DC 参考时钟时间。图 7-21 表示本地周期与 DC 参考时钟同步的工作原理。

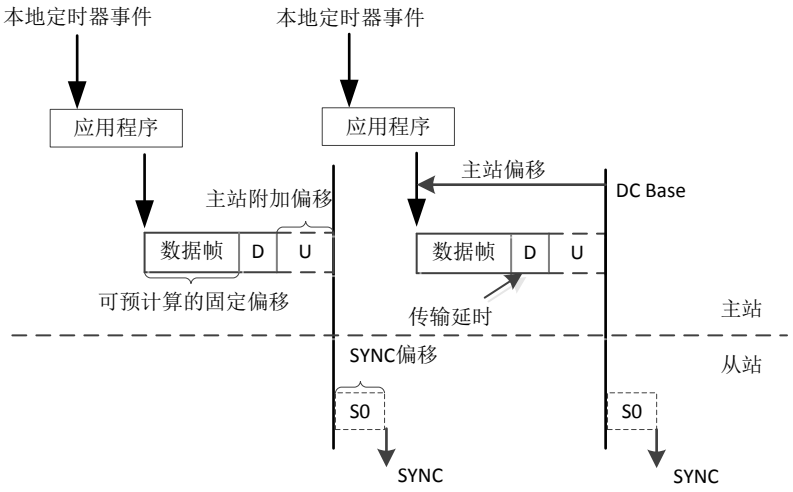


图 7-21 主站 DC 模式

主站本地运行由一个本地定时器启动。本地定时器应该比 DC 参考时钟定时存在一个提前量，提前量为以下时间之和。

✧ 控制程序执行时间。

✧ 数据帧传输时间。

✧ 数据帧传输延时 D。

✧ 附加偏移 U（与各从站延时时间的抖动和控制程序执行时间的抖动值有关，用于主站周期的调整）。

7.2.2 2. 非周期性邮箱数据通信

EtherCAT 协议中非周期性数据通信称为邮箱数据通信，它可以双向进行—主站到从站和从站到主站。它支持全双工、两个方向独立通信和多用户协议。从站到从站的通信由主站作为路由器来管理。邮箱通信数据头中包括一个地址域，使主站可以重寄邮箱数据。邮箱数据通信是由实现参数交换的标准方式，如果需要配置周期性过程数据通信或需要其他非周期性服务时需要使用邮箱数据通信。

邮箱数据报文结构如图 7-22 所示。通常邮箱通信值对应一个从站，所以报文中使用设备寻址模式。其数据头中各数据元素的解释如表 7-2 所列。

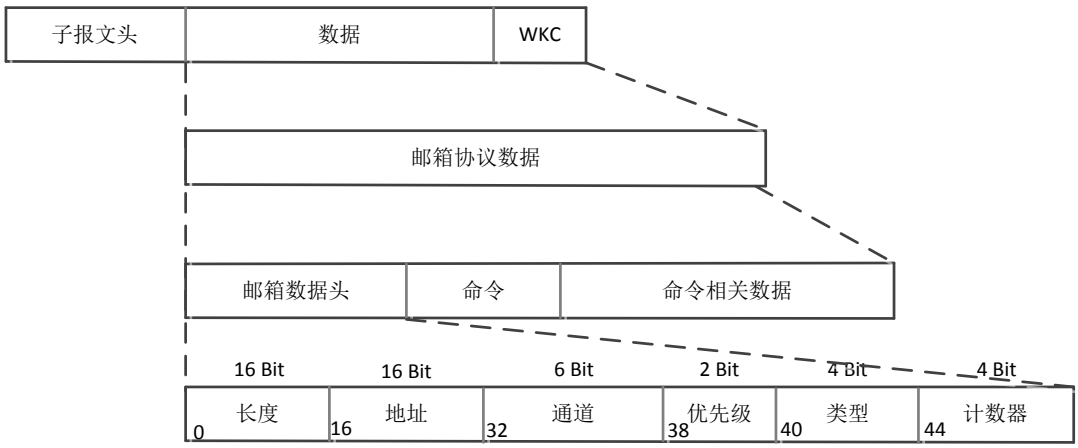


图 7-22 邮箱数据单元结构

表 7-2 邮箱数据头

数据元素	位数	描述
长度	16 位	跟随的邮箱服务数据长度
地址	16 位	主站到从站通信时，为数据源从站地址 从站到从站通信时，为数据目的从站地址
通道	6 位	保留
优先级	2 位	保留
类型	4 位	邮箱类型，即后续的协议类型： 0：邮箱通信出错； 2：EoE（Ethernet over EtherCAT）； 3：CoE（CANopen over EtherCAT）； 4：FoE（File Access over EtherCAT）； 5：SoE（Sercos over EtherCAT）； 15：VoE（Vendor Specific Profile over EtherCAT）
计数器 Ctr	4 位	用于重复检测的顺序编号，每个新的邮箱服务将加 1（为了兼容老版本而只使用 1～7）

✧ 主站到从站通信--写邮箱命令

主站发送写数据区命令将邮箱数据发送从站。主站需要检查从站邮箱命令应答报文中工作计数器 WKC。如果工作计数器为 1，表示写命令成功。反之，如果工作计数器没有增加，通常因为从站没有读完上一个命令，或在限定的时间内没有响应，主站必须重发写邮箱数据命令。

✧ 从站到主站通信—读邮箱命令

从站有数据要发送给主站，必须先将数据写入输入邮箱缓存区，然后由主站来读取。主站发现从站 ESC 输入邮箱数据区有效数据等待发送时，会尽快发送适当的读命令来读取从站数据。主站有两种方式来测定从站是否已经将邮箱数据填入输入数据区。一种是使用 FMMU 周期性的读取某一个标志位。使用逻辑寻址可以读取多个从站的标志位，但其缺点是每个从站都需要一个 FMMU 单元。另一个方法是将简单的轮训 ESC 输入到邮箱的输入区。读命令的工作计数器增加 1 表示从站已经将新数据填入到了输入数据区。

7.3 EtherCAT 状态机

EtherCAT 状态机（ESM，EtherCAT State Machine）负责协调主站和从站应用程序在初始化和运行时的状态关系。

EtherCAT 设备必须支持四种状态，另外还有一个可选的状态。

✧ Init：初始化，简写为 I。

✧ Pre- Operational：预运行，简写为 P。

✧ Safe-Operational：安全运行，简写为 S。

✧ Operational：运行，简写为 O。

✧ Boot-Strap：引导状态（可选），简写为 B。

以上各状态之间的转换关系如图 7-23 所示。从初始化状态向运行状态转化时，必须按照“初始化 > 预运行 > 安全运行 > 运行”的顺序转换，只有从运行状态返回时可以越级转化，其他状态均不可以越级转化。引导状态为可选状态，只允许与初始化状态之间相互转化。所有的状态改变都由主站发起，主站向从站发送状态控制命令请求新的状态，从站响应此命令，执行所请求的状态转换，并将结果写入从站状态指示变量。如果请求的状态转换失败，从站将给出错误标志。表 7-3 状态转换的总结。

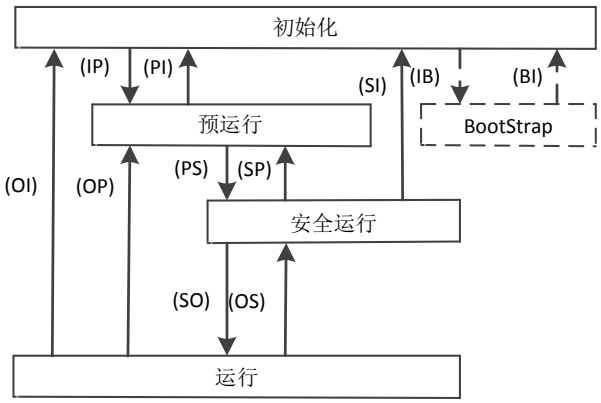


图 7-23 EtherCAT 状态转化关系

✧ Init：初始化

初始化状态定义了主站与从站在应用层的初始通信关系。此时，主站与从站应用层不可以直接通信，主站使用初始化状态来初始化 ESC 的一些配置寄存器。如果主站支持邮箱通信，则配置邮箱通信参数。

✧ Pre- Operational：预运行

在预运行状态下，邮箱通信被激活。主站与从站可以使用邮箱通信来交换与应用程序相关的初始化操作和参数。在这个状态下不允许过程数据通信。

✧ Safe-Operational：安全运行

在安全运行状态下，从站应用程序读入输入数据，但是不产生输出信号。设备无输出，处于“安全状态”。此时，仍然可以使用邮箱通信。

✧ Operational：运行

在运行状态下，从站应用程序读入数据，主站应用程序发出输出数据，从站设备产生输出信号。此时，仍然可以使用邮箱通信。

✧ Boot-Strap：引导状态

引导状态的功能是下载设备固件程序。主站可以使用 FoE 协议的邮箱通信下载一个新的固件程序给从站。

表 7-3 EtherCAT 状态机其转化过程总结过程

状态和状态转化	描述
初始化	应用层没有通信，主站只能读写 ESC 寄存器
初始化向预运行转化 Init to Pre-OP（IP）	主站配置从站站点地址寄存器； 如果支持邮箱通信，则配置邮箱通道参数； 如果支持分布式时钟，则配置 DC 相关寄存器； 主站写状态控制寄存器，以请求“Pre-Op”状态
预运行	应用层邮箱数据通信
Pre-Op to Safe-Op（PS）	主站使用邮箱初始化过程数据映射； 主站配置过程数据通信使用的 SM 通道； 主站配置 FMMU； 主站写状态控制寄存器，以请求“Safe-Op”状态
Safe-Operational	主站发送有效的输出数据； 主站写状态控制寄存器，以请求“Op”状态
Operational	输入和输出全部有效； 仍然可以使用邮箱通信

7.4 EtherCAT 伺服驱动器控制应用协议

IEC 61800 标准系列是一个可调速电子功率驱动系统通用规范。其中 IEC 61800-7 定义了控制系统和功率驱动系统之间的通信接口标准、包括网络通信技术和应用行规，如图 7-24 所示。EtherCAT 作为网络通信技术，支持了 CANopen 协议中的行规 CiA 402 和 SERCOS 协议的应用层，分别称为 CoE 和 SoE。

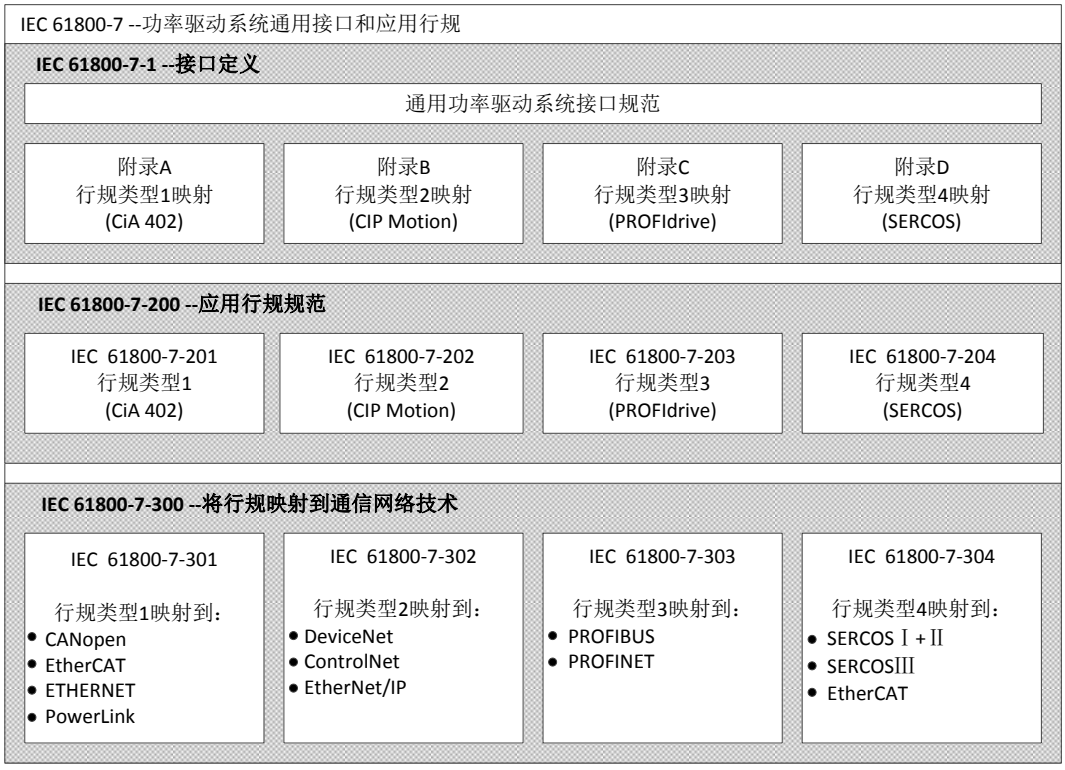


图 7-24 IEC 61800-7 体系结构

7.4.1 基于 EtherCAT 的 CAN 应用协议（CoE）

CANopen 设备和应用行规广泛用于多种设备类别和应用，如 I/O 组件、驱动、编码器、比例阀、液压控制器，以及用于塑料或纺织行业的应用行规等。EtherCAT 可以提供与 CANopen 机制相同的通讯机制，包括对象字典、PDO（过程数据对象）、SDO（服务数据对象），甚至相似的网络管理。因此，在已经实施了 CANopen 的设备中，仅需稍加变动即可轻松实现 EtherCAT，绝大部分的 CANopen 固件都得以重复利用。并且，可以选择性地扩展对象，以便利用 EtherCAT 所提供的巨大带宽资源。

EtherCAT 协议在应用层支持 CANopen 协议，并作了相应的补充，其主要功能有：

- ✧ 使用邮箱通信访问 CANopen 对象字典和对象，实现网络初始化；
- ✧ 使用 CANopen 应用对象和可选的时间驱动 PDO 消息，实现网络管理；
- ✧ 使用对象字典映射过程数据，周期性传输指令数据和状态数据。

图 7-25 为 CoE 设备结构图，其通讯方式主要包括周期性过程数据通信及非周期数据通信。下文会分别介绍两者在实际应用中的区别。

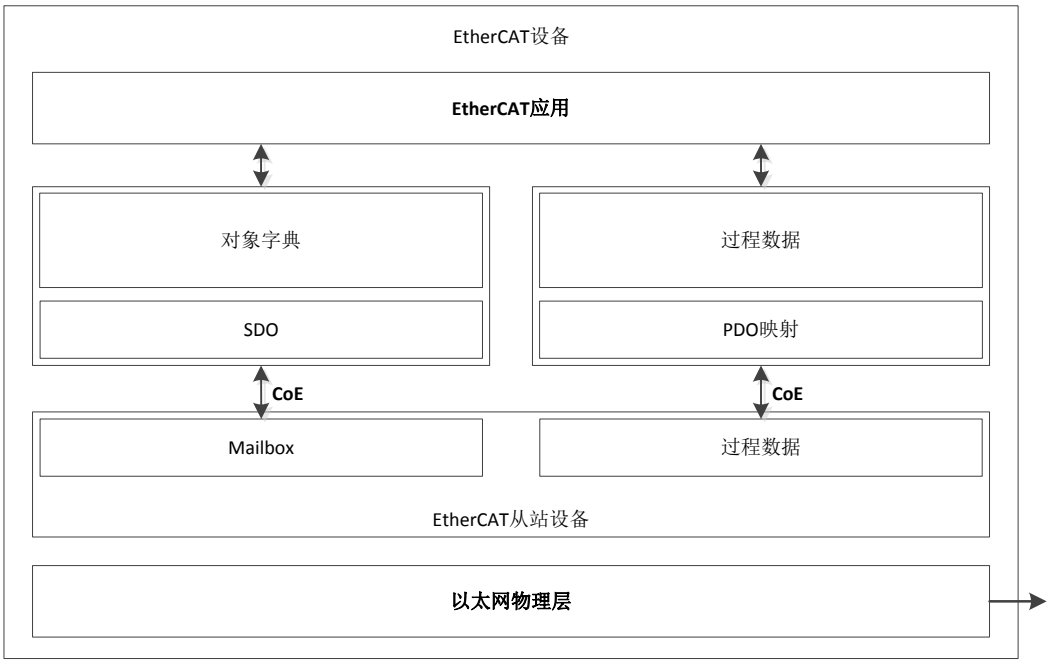


图 7-25 CoE 设备结构图

7.4.1.1 CoE 对象字典

CoE 协议完全遵从 CANopen 协议，其对象字典的定义也相同，如表 7-4 所示。表 7-5 列举了 CoE 的通信数据对象，其中针对 EtherCAT 通信扩展了相关通信对象 0x1C00~0x1C4F，用于设置存储同步管理器的类型、通信参数和 PDO 数据分配。

表 7-4 CoE 对象字典定义

索引号范围	描述
0x0000 - 0x0FFF	数据类型描述
0x1000 - 0x1FFF	通信对象包括： 设备类型、标识符、PDO 映射、与 CANopen 兼容 CANopen 专用数据对象，在 EtherCAT 中保留 EtherCAT 扩展数据对象
0x2000 - 0x5FFF	制造商定义对象
0x6000 - 0x9FFF	行规定义数据对象
0xA000 - 0xFFFF	保留

表 7-5 CoE 通信数据对象

索引	描述
0x1000	设备类型
0x1001	错误寄存器
0x1008	设备商设备名称
0x1009	制造商硬件版本
0x100A	制造商软件版本
0x1018	设备标识符
0x1600 - 0x17FF	RxPDO 映射
0x1A00 - 0x1BFF	TxPDO 映射
0x1C00	同步管理器通讯类型
0x0x1C10 - 0x1C2F	过程数据通信同步管理器 PDO 分配
0x0x1C30 - 0x1C4F	同步管理参数

7.4.1.2 CoE 周期性过程数据通信（PDO）

周期性数据通信中，过程数据可以包含多个 PDO 映射数据对象。CoE 协议使用的数据对象 0x1C10～0x1C2F 定义相应的 PDO 映射通道。表 7-6 为 EtherCAT 协议中对该通讯数据的具体结构。

表 7-6 CoE 通信数据对象

索引	对象类型	描述	类型
0x1C10	数组	SM0 PDO 分配	无符号整型 16 位
0x1C11	数组	SM1 PDO 分配	无符号整型 16 位
0x1C12	数组	SM2 PDO 分配	无符号整型 16 位
0x1C13	数组	SM3 PDO 分配	无符号整型 16 位
……	……	……	…… …… …… ……
0x1C2F	数组	SM31 PDO 分配	无符号整型 16 位

以下针对 SM2 PDO（0x1C12）进行分配举例，表 7-7 列出了其取值实例。如 PDO0 中映射了两个数据，第一个通讯变量为控制字，对应映射的索引及子索引地址为 0x6040：00；第二个通讯变量目标位置值，对应映射的索引及子索引地址为 0x607A：00。

表 7-7 SM2 通道 PDO 分配对象数据 0x1C12 举例

0X1C12 子索引	数值	PDO 数据对象映射			
		子索引	数值	字节数	描述
0	3			1	PDO 映射对象数目
1	PDO0 0x1600	0	2	1	数据映射数据对象数目
		1	0x6040：00	2	控制字
		2	0x607A：00	4	目标位置
1	PDO1 0x1601	0	2	1	数据映射数据对象数目
		1	0x6071：00	2	目标转矩
		2	0x6087：00	4	目标斜坡
1	PDO2 0x1602	0	2	1	数据映射数据对象数目
		1	0x6073：00	2	最大电流
		2	0x6075：00	4	马达额定电流

PDO 映射有以下几种方式：

（1）简单设备不需要映射协议：

✧ 使用简单的过程数据

✧ 在从站的 EEPROM 中读取。

（2）可读取的 PDO 映射：

✧ 固定过程数据映射

✧ 使用 SDO 通信读取

（3）可选择的 PDO 映射

✧ 多组固定的 PDO 通过对象 0x1C1X 选择

✧ 通过 SDO 通信读取

（4）可变的 PDO 映射

✧ 通过 CoE 通信配置

7.4.1.3 CoE 非周期性过程数据通信（SDO）

EtherCAT 主站通过读写邮箱数据 SM 通道实现非周期性数据通信。CoE 协议邮箱数据结构如图 7-26 所示。

图 7-26 CoE 数据头

针对图 7-26 中的编号部分在表 7-8 中有详细的解释。

表 7-8 CoE 命令定义

CoE 命令字段编号	描述
编号	PDO 发送时的编号
类型	消息类型： 0：保留 1：紧急事件信息 2：SDO 请求 3：SDO 响应 4：TxPDO 5：RxPDO 6：远程 TxPDO 发送请求 7：远程 RxPDO 发送请求 8：SDO 信息 9-15：保留

✧ SDO 服务

CoE 通信服务类型 2 和 3 为 SDO 通信服务，SDO 数据结构如图 7-27 所示。

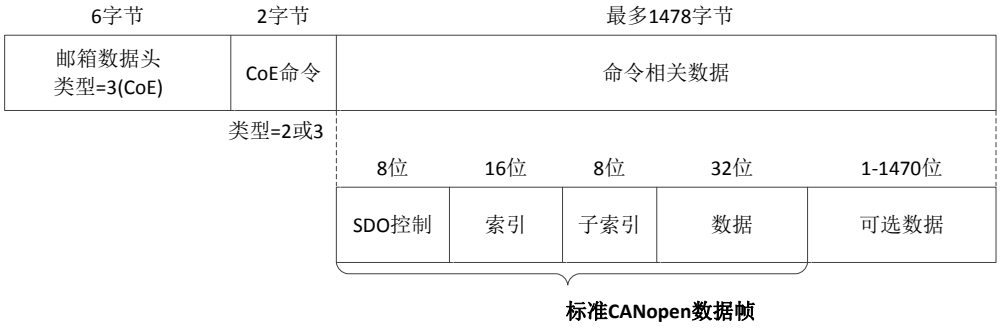


图 7-27 SDO 数据帧格式

SDO 按传输方式通常有如下三种类型，表 7-9 为 SDO 数据帧具体的内容。其结构图如图 7-28 所示：

快速传输服务：与标准的 CANopen 协议相同，只使用 8 个字节，最多传输 4 个字节有效数据。

常规传输服务：使用超过 8 个字节，可以传输超过 4 个字节的有效数据，最大可传输的有效数据取决于邮箱 SM 所管理的存储区容量。

分段传输服务：对于超过邮箱容量的情况，使用分段的方式进行传输。

表 7-9 CoE 数据帧内容

SDO 控制	标准 CANopen SDO 服务
索引	设备对象索引
子索引	子索引
数据	SDO 中的数据
数据（可选）	有四个字节的可选数据可被加截至数据帧中

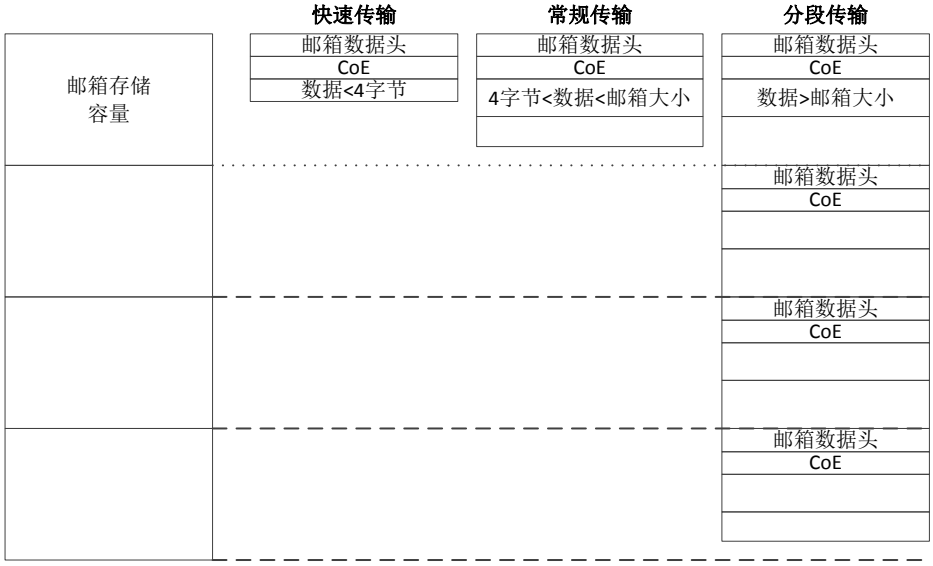


图 7-28 SDO 传输类型

如果要传输的数据大于 4 个字节，则使用常规传输服务；在常规传输时用快速传输时的 4 个数据字节表示要传输的数据的完整大小，用扩展数据部分传输有效数据，有效数据的最大容量为邮箱容量减去 16。

7.4.2 IEC 61800-7-204 的伺服驱动行规（SERCOS）

SERCOS 被公认为用于高性能实时系统的通讯接口，尤其适用于运动控制的应用场合。用于伺服驱动和通讯技术的 SERCOS 行规属于 IEC61800-7-204 标准的范畴。该伺服驱动行规到 EtherCAT 的映射(SoE)在 304 部分定义。用于访问位于驱动中的全部参数以及功能的服务通道基于 EtherCAT 邮箱。在此，关注焦点还是 EtherCAT 与现有协议的兼容性（访问 IDN 的数值、属性、名称、单位等），以及与数据长度限制相关的扩展性。过程数据，即格式为 AT 和 MDT 的 SERCOS 数据，都使用 EtherCAT 设备协议机制进行传送，其映射与 SERCOS 映射相似。并且，EtherCAT 从站的状态机也可以非常容易地映射为 SERCOS 协议状态。

7.4.2.1 SoE 状态机

SERCOS 协议的通信阶段与 EtherCAT 状态机的比较图如图 7-29 所示，其特点有以下几个方面：

（1） SERCOS 协议通信阶段 0 和 1 被 EtherCAT 初始化状态覆盖；

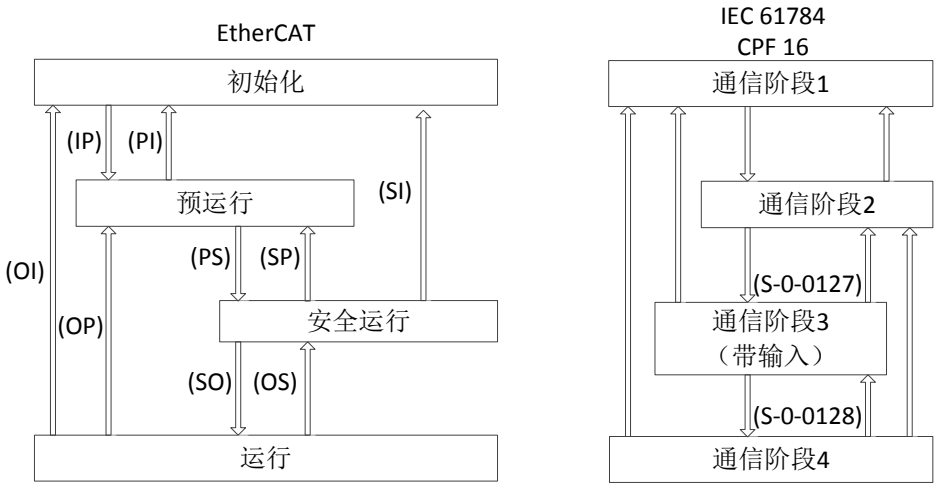
（2） 通信阶段 2 对应于与运行状态，允许使用邮箱通信实现服务通道，操作 IDN 参数；

（3） 通信阶段 3 对应于安全运行状态，开始传输周期性数据，只有输入数据有效，输出数据被忽略，同时可以实现时钟同步；

（4） 通信阶段 4 对应于运行阶段，所有的输入和输出都有效；

（5） 不使用 SERCOS 协议的阶段切换过程命令 S-0-0127（通信阶段 3 切换检查）和 S-0-0128（通信阶段 4 切换检查），分别由 PS 和 SO 状态转化取代；

（6） SERCOS 协议只允许高级通信阶段向下切换到通信阶段 0，而 EtherCAT 允许任意的状态向下切换（如图 7-29 的 a）所示）。例如从运行状态切换到安全运行状态，或从安全运行状态切换到预运行状态。SoE 也应该支持这种切换图 7-29 的 b）中所示，如果从站不支持，则应该 EtherCAT AL 状态寄存器中设置错误位。



a) EtherCAT 状态机　　b) SERCOS 状态机

图 7-29 SoE 状态机

7.4.2.2 IDN 继承

SoE 协议继承 SERCOS 协议的 DIN 参数定义。每个 IDN 参数都有一个唯一的 16 位标识号 IDN，对应一个唯一的数据块，保存参数的全部信息。数据块由 7 个元素组成，如表 7-10 所列。IDN 参数分为标准数据和产品数据两部分，每部分又分为 8 个参数组，使用不同的 IDN 表示，如

表 7-11 所列。

表 7-10 IDN 数据块结构

编号	名称
元素 1	IDN
元素 2	名称
元素 3	属性
元素 4	单位
元素 5	最小允许值
元素 6	最大允许值
元素 7	数据值

表 7-11 IDN 编号定义

位	15	14-12	11-0
含义	分类	参数组	参数编号
取值	0：标准数据 S 1：产品数据 P	0-7:8 个参数组	0000-4095

在使用 EtherCAT 作为通信网络时，取消了一些 SERCOS 协议中用于通信接口控制的 IDN，如表 7-12 所列。此外，还对一些 IDN 的定义做了些修改，如

表 7-13 所列。

表 7-12 删除的 IDN

IDN 描述	IDN 描述
S-0-0003	最小 AT 发送的开始时间
S-0-0004	发送到接收状态切换时间
S-0-0005	最小反馈采样提前时间
S-0-0009	主站数据报文中的开始地址
S-0-0010	主站数据报文长度
S-0-0088	接收 MDT 后准备好接收 MST 所需要的恢复时间
S-0-0090	命令处理时间
S-0-0127	通信阶段 3 切换检查
S-0-0128	通信阶段 4 切换检查

表 7-13 修改的 IDN

IDN	原描述	新描述
S-0-0006	AT 发送的开始时间	在从站内部于同步信号之后应用程序向 ESC 存储区写入 AT 数据的时间偏移
S-0-0014	通信接口状态	映射从站 DL 状态和 AL 状态码
S-0-0028	MST 错误技术	映射从站 RX 错误计数器到丢失计数器
S-0-0089	MDT 发送开始时间	在从站内部于同步信号之后从 ESC 存储区得到新的 MDT 数据的时间偏移

7.4.2.3 SoE 周期性过程数据

输出过程数据（MDT 数据内容）和输入过程数据（AT 数据内容）由 S-0-0015、S-0-0016 和 S-0-0024 配置。过程数据不包括服务通道数据，只有周期性过程数据。输出过程数据包括伺服控制字和指令数据，输入过程包括状态字和反馈数据。

S-0-0015 设定了周期性过程数据的类型，如表 7-14 所列，参数 S-0-0016 和参数 S-0-0024 如表 7-15 所列。主站在“预运行”阶段通过邮箱通信写这三个参数，以配置周期性过程数据的内容。

表 7-14 参数 S-0-0015 定义

S-0-0015	指令数据	反馈数据
0：标准类型 0	无	无反馈数据
1：标准类型 1	扭矩指令 S-0-0080（2 字节）	无反馈数据
2：标准类型 2	速度指令 S-0-0036（4 字节）	速度反馈 S-0-0053（4 字节）
3：标准类型 3	速度指令 S-0-0036（4 字节）	位置反馈 S-0-0051（4 字节） 速度反馈 S-0-0053（4 字节）
4：标准类型 4	位置指令 S-0-0047（4 字节）	
5：标准类型 5	位置指令 S-0-0047（4 字节） 速度指令 S-0-0036（4 字节）	位置反馈 S-0-0051（4 字节） 或速度反馈 S-0-0053（4 字节）+ 位置反馈 S-0-0051（4 字节）
6：标准类型 6	速度指令 S-0-0036（4 字节）	无反馈数据
7：自定义	S-0-0024 配置	S-0-0016 配置

表 7-15 参数 S-0-0016 和参数 S-0-0024 定义

数据字	S-0-0024 定义	S-0-0016 定义
0	输出数据最大长度（Word）	输入数据最大长度（Word）
1	输出数据实际长度（Word）	输入数据实际长度（Word）
2	指令数据映射的第一个 IDN	反馈数据映射的第一个 IDN
3	指令数据映射的第二个 IDN	反馈数据映射的第二个 IDN
…	…	… … …

7.4.2.4 SoE 非周期性服务通道

EtherCAT SoE 服务通道 SSC（SoE Service Channel）由 EtherCAT 邮箱通信功能完成，它用于非周期性数据交换，如读写 IDN 及其元素。SoE 数据头格式如图 7-30 所示。

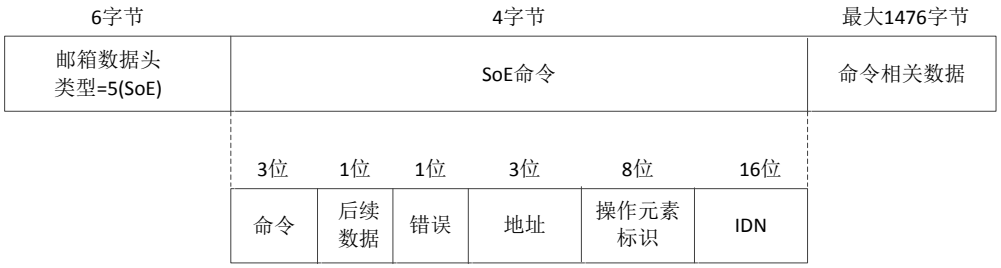


图 7-30 SoE 数据头格式

表 7-16 SoE 数据命令描述

数据区	描述
命令	即指令类型： 0x01：读请求 0x02：读响应 0x03：写请求 0x04：写响应 0x05：通报

数据区	描述
	0x06：从站信息 0x07：保留
后续数据	后续数据信号： 0x00：无后续数据帧 0x01：未完成传输，有后续数据帧
错误	错误信号： 0x00：无错误 0x01：发生错误，数据区有 2 个字节的错误码
地址	从站设备的具体地址
操作元素标识	单个元素操作时为元素选择，按位定义，每一个位对应一个元素； 寻址结构体时为元素的数目
IDN	参数的 IDN 编号，或分段操作时的剩余片段

常用的 SSC 操作包括 SSC 读操作、SSC 写操作和过程命令。

- ✧ SSC 读操作：SSC 读操作由主站发起，写 SSC 请求到从站。从站接收到读操作请求后，用所请求的 IDN 编号和数据值作为回答。主站可以同时读多个元素，从站应该同时回答多个元素，如果从站只支持单个元素操作，应该以所请求的第一个元素作为响应。
- ✧ SSC 写操作：该操作用于主站下载数据到从站，从站应该以写操作的结果回答。分段操作由一个或多个分段写操作及一个 SSC 写响应服务组成。
- ✧ SSC 过程命令：过程命令是一种特殊的非周期数据，每一个过程命令都有唯一标识的 IDN 和规定的数据元素，用于启动伺服装置的某些特定功能或过程。执行这些功能或过程通常需要一段时间，过程命令只是触发其开始，随后它所占用的服务通道立即变为可用，用以传输其他非周期数据或过程命令，而不用等到被触发的功能或过程执行完毕。

第8章 应用编程

8.1 单轴控制

8.1.1 单轴控制编程说明

AX7x 系列控制器与伺服轴（如 DA200）配合的运动控制是基于 EtherCAT 总线网络来实现的，每个 EtherCAT 总线周期会进行一次计算、发布一次控制命令，从而实现对伺服的控制，不同与以往的脉冲控制方式，EtherCAT 总线完全通过软件来实现，应用时需要注意以下几点：

- ✧ MC 相关的 POU 应配置在 EtherCAT 任务下执行，多数 MC 功能块放在低优先级 Main 任务的 POU 中无法正常运行。
- ✧ PDO 配置表需要配置相关的数据对象，否则会由于通信数据对象配置缺省而导致伺服无法运行，也不会有出错报警，排查原因难度加大。
- ✧ 控制器可以通过配置 SDO 的方式对伺服进行参数设定。
- ✧ MC 功能块实例只能用于唯一的伺服轴控制，如果用于多个会导致控制出错。
- ✧ 必须有 MC 功能块来监控运行中的伺服轴，避免程序逻辑跳转无 MC 功能块监控而引发的报错，此类错误通常不易排查。
- ✧ 注意调试的安全处理，信号配置与实际应用相符。若伺服系统采用增量式编码器，正常运行之前需要回零，对于在有限范围内运动（如丝杆），应设置极限与安全保护信号。

8.1.2 单轴控制常用的 MC 功能块

MC 功能块（FB）也称为 MC 指令，实际上，用户程序中使用的是 MC 功能块的对象实例，伺服轴通过 MC 对象实例来进行控制，例如：

MC_Power1：MC_Power；// 声明实例 MC_Power1

MC_Power1（Axis=Axis1，）

单轴的控制，一般用于定位的控制，即伺服电机拖动外部机构运动到指定的位置；有时也需要伺服以指定的速度或力矩运行等，在单轴控制中，常用到如下的 MC 功能块：

表 8-1 单轴控制常用 MC 功能块

控制操作	需要使用的 MC 指令	说明
伺服使能	MC_Power	运行该指令，使伺服轴使能，才能进行后续的运行控制
绝对定位	MC_MoveAbsolute	命令伺服运行到指定的坐标点
相对定位	MC_MoveRelative	以当前位置为参考，运行指定距离
伺服点动运行	MC_Jog	伺服电机的点动运行，常用于低速试车，用于检验设备或调整伺服电机位置
相对叠加定位	MC_MoveAdditive	在伺服当前运行指令的基础上，再相对运行指定距离
速度控制	MC_MoveVelocity	命令伺服以指定的速度运行
伺服暂停	MC_Halt	命令伺服暂停运行，若 MC_Movexxx 再次触发，伺服可以再运行
紧急停机	MC_Stop	命令伺服紧急停机，只有 stop 命令复位后，触发 MC_Movexxx，伺服才可以再运行
告警复位	MC_Reset	当伺服出现告警停机后，运行该指令进行复位
伺服原点回归	MC_Home	命令伺服开始原点回归操作，应用系统的原点信号、两侧极限信号等都接在伺服的 DI 端口

控制操作	需要使用的 MC 指令	说明
控制器原点回归	MC_Homing	控制系统开始原点回归操作，应用系统的原点信号、两侧极限信号等都接在控制器的 DI 端口

8.2 凸轮同步控制

电子凸轮(英文简称 ECAM)是利用构造的凸轮曲线来模拟机械凸轮，以达到机械凸轮系统相同的凸轮轴与主轴之间相对运动的软件系统。电子凸轮可以应用在诸如汽车制造、冶金、机械加工、纺织、印刷、食品包装等各个领域。电子凸轮曲线是以主轴脉冲（主动轴）输入为 X，伺服电机（凸轮轴）对应输出为 Y=F(X)的一个函数曲线。

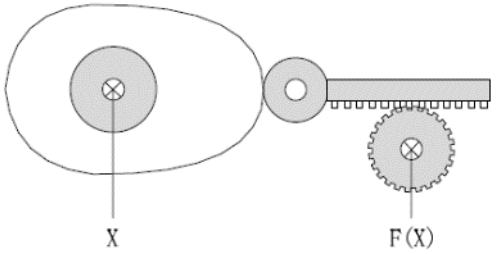


图 8-1 电子凸轮示意图

AX 系列可编程控制器电子凸轮功能有如下特点：

✧ 凸轮曲线易于绘制：可通过凸轮表、凸轮曲线或数组描述凸轮且支持多个凸轮表选择和运行中动态切换。

✧ 凸轮曲线易于修正：支持对运行中的凸轮表进行动态修改。

✧ 支持一主多从：一个主轴可有多个从轴与之对应。

✧ 凸轮挺杆：允许有多个凸轮挺杆、多个设置区间。

✧ 凸轮离合器：用户程序可使之进入与退出凸轮运行。

✧ 特有功能：支持虚拟主轴、相位偏移和输出叠加。

注意：所谓“在线修改 CAM 曲线”，是指用户编写的程序在执行过程中，根据控制特性的需要，对 CAM 曲线的关键点坐标，进行的修改。修改的内容，一般是修改关键点坐标，但也可以修改关键点的个数、修改主轴的距离范围等。

AX 系列可编程控制器电子凸轮有三个控制要素：

（1）主轴：同步控制的参考轴。

（2）从轴：按照非线性特性跟随主轴运动的伺服轴。

（3）凸轮表：描述主轴-从轴相对位置与范围、周期性等的数据表或凸轮曲线。

常用的电子凸轮相关功能块见下表：

表 8-2 常用电子凸轮功能块

MC 指令	说明
MC_CamTableSelect	运行该指令，关联主轴、从轴凸轮表三者关系
MC_CamIn	让从轴进入凸轮运行
MC_CamOut	让从轴退出凸轮运行
MC_Phasing	主轴相位修改

8.2.1 凸轮表的周期模式

（1）单周期模式（Periodic:=0）：凸轮表周期运行完毕后，从轴脱离凸轮运行状态，如图 8-2 所示；

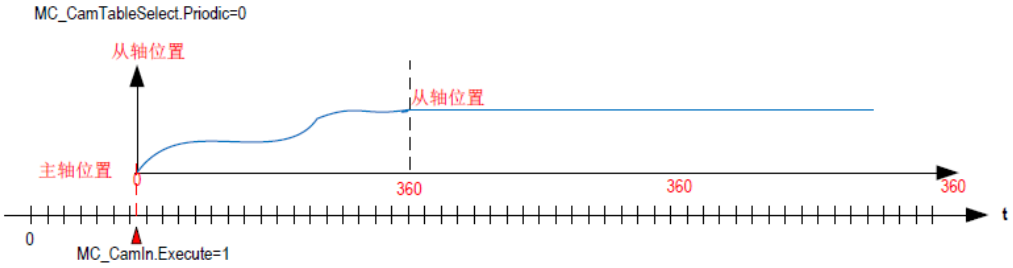


图 8-2 单周期模式

（2）周期模式（Periodic:=1）：凸轮表周期运行完毕后，从轴又开始下一凸轮周期的运行，直到用户程序命令其退出凸轮运行状态，如图 8-3 所示：

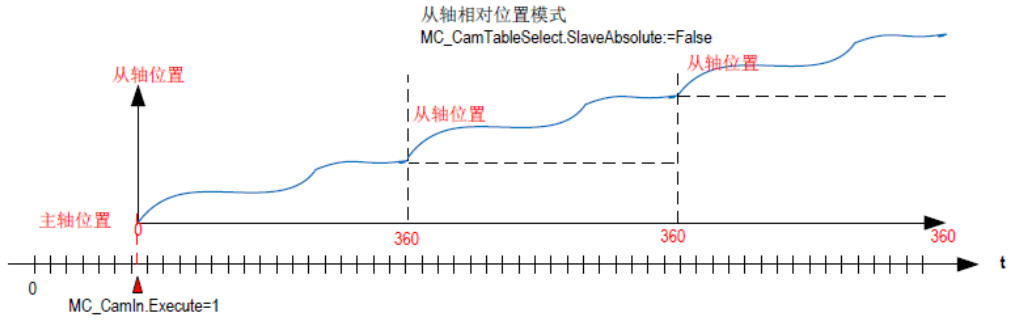


图 8-3 周期模式

8.2.2 凸轮表的输入方法

（1） 新建一个凸轮表时，系统会自动生成一个最简单的凸轮曲线，用户在此基础上进行编辑，形成自己的 CAM 曲线表。

（2） 用户可以增减凸轮曲线的关键点个数或修改关键点的坐标。

（3） 凸轮曲线两个关键点之间的线型可设置为直线或五次多项式，并且系统会对每条曲线作最佳优化，以尽量减小速度和加速度的突变。

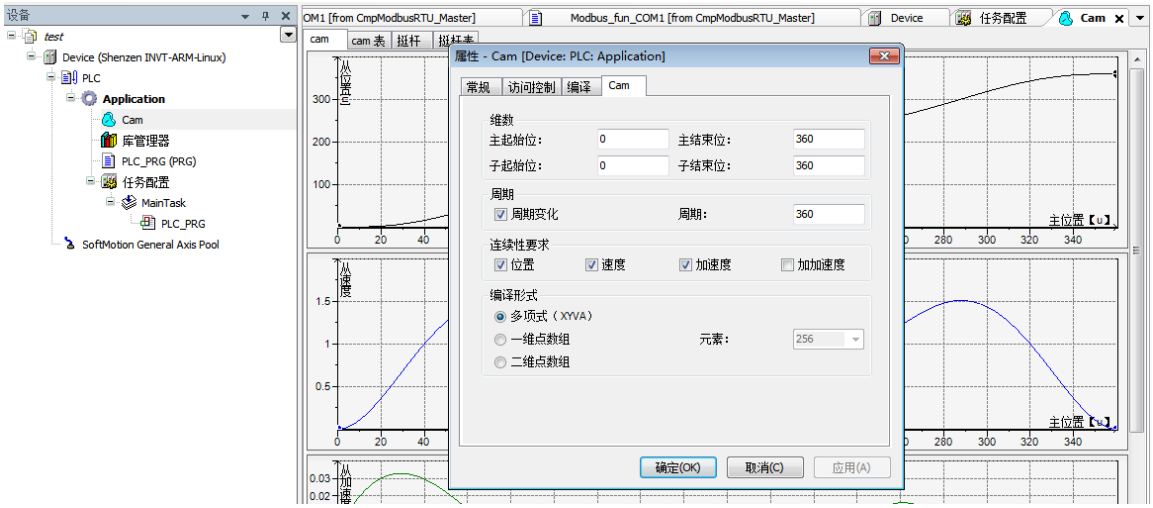


图 8-4 凸轮曲线

8.2.3 凸轮表的数据结构

在 Invtmatic Studio 中，对每一个 CAM 表，都有描述该 CAM 表的数据结构，描述该 CAM 表的特征数据。下图为“CAM0”凸轮表的描述数据结构，请注意其结构各变量名称。

cam | cam 表 | 挺杆 | 挺杆表

	X	Y	V	A	J	段类型	最小(...	最大(...	最大(|...	最大(|...
	0	0	0	0	0					
						Poly5	0	120	1.5120...	0.0328...
	120	120	1	0	0					
						Poly5	120	240	1	0
	240	240	1	0	0					
						Poly5	240	360	1.512	0.0328...
	360	360	0	0	0					

图 8-5 凸轮表数据结构

Invtmatic Studio 内部有一个数据结构来描述 CAM 凸轮表的特征，我们也可以手动编写一个 CAM 表，或者我们可以通过数据结构的访问操作，对 CAM 特性数据进行修改。

注意：我们在声明 CAM0 凸轮表时，系统自动默认声明了全局变量类型的 CAM0 数据结构，同时声明了 CAM0_A[i] 数组。例如在用户程序中，修改 CAM0 凸轮表关键点个数或坐标:

```
CAM0. nElements:=10; // 将关键点个数改为 10 个
CAM0. xEnd:=300; // 将主轴的结束点改为 300
// 例如在用户程序中，修改其中 2 个关键点的坐标：
CAM0_A[2].dx:=10;
CAM0_A[2].dy:=30;
CAM0_A[2].dv:=1;
CAM0_A[2].da:=0;
CAM0_A[3].dx:=30;
CAM0_A[3].dy:=50;
CAM0_A[3].dv:=1;
CAM0_A[3].da:=0;
```

8.2.4 凸轮表的引用与切换

CAM 凸轮表在控制器内部是用一个数组来保存，可以通过特定的 MC_CAM_REF 变量类型来指向，例如声明：

```
凸轮表 q: MC_CAM_REF;
```

可以给该变量赋值，也可认为是将其指向某具体的凸轮表：

```
凸轮表 q:= Cam0; // 指向所需的凸轮表
凸轮表 q: MC_CAM_REF; // 凸轮表指针；
TableID: uint; // 凸轮表选择命令，可由 HMI 设置；
Case TableID of
0: 凸轮表 q := 凸轮表 A;
1: 凸轮表 q := 凸轮表 B;
```

2: 凸轮表 q := 凸轮表 C;

End_case

MC_CamTableSelect_0(// 凸轮关系

Master:= 虚主轴

Slave:= 凸轮从轴

CamTable:= 凸轮表 q

Execute:= bSelect， // 上升沿触发凸轮表选择

Periodic:= TRUE，

MasterAbsolute:=FALSE，

SlaveAbsolute:= FALSE);

上面的例程，利用该 MC_CAM_REF 变量的赋值运算，就可以实现多个凸轮表的切换操作了。

附录A 功能模块指令

A.1 ModbusRTU 库指令

A.1.1 ModbusRTU 主站指令库变量定义及使用

A.1.1.1 变量定义

变量从属模块	变量名称	类型		功能	注释
ModbusRTU_Master_Init_COM1	Execute1	INPUT	BOOL	串口初始化功能激活	0：非激活 1：激活
	Baud1		DINT	波特率	例：115200
	Databits1		INT	数据位	例：8 位（无 7 位 ASCII）
	Stopbits1		INT	停止位	例：1 停止位、2 停止位
	Parity1		INT	读写标志	0：无校验 1：偶校验 2：奇校验
	Slave1		UINT	从站 ID	1-128
	Timeout1		DINT	超时时间	例：1000
	bDone1	OUTPUT	BOOL	完成标志	0：指令正在执行 1：执行指令完成
	Error1		BOOL	错误标志	0：无错误 1：有错误
	ErrorID1		INT	错误码	见 ModbusRTU 错误码表
ModbusRTU_Master_Fun_COM1	xExecute1	INPUT	BOOL	读写功能激活	0：非激活 1：激活
	Fun_Code1		INT	功能码	0x01、0x03、0x05、0x06、0x0f、0x10
	Addr1		UINT	地址	0x0000~0xFFFF
	DataCount1		UINT	数目	读：1~250 写：1~240
	DataPtr1		POINTER TO INT	数据指针	指向读写数据存放的地址
	Error1	OUTPUT	BOOL	错误标志	0：无错误 1：有错误
	ErrorID1		INT	错误码	见 ModbusRTU 错误码表

串口 2 做 ModbusRTU_Master 主站时变量数量相同，变量名称后的数字由“1”改变成“2”，例如“ModbusRTU_Master_Init_COM2”。

A.1.1.2 使用教程

1）ModbusRTU_Master 主站连接从站设置

从属模块	设置项	功能	示例
ModbusRTU_Master_Init_COM1	Execute1	从站使能变量	Enable:=TRUE
	Baud1	波特率	Baud1:=19200
	Databits1	数据位	Port:=8
	Stopbits	停止位	Unit:=1
	Parity1	校验位	Parity1:=2

从属模块	设置项	功能	示例
	Slave1	从站 ID	Slave1:=12
	Timeout1	超时时间	Delay Time:=1000

若定义将要连接的 ModbusRTU 从站时，参考上表 COM1 参数进行统一配置。参考示例（结构化文本 ST）如下：

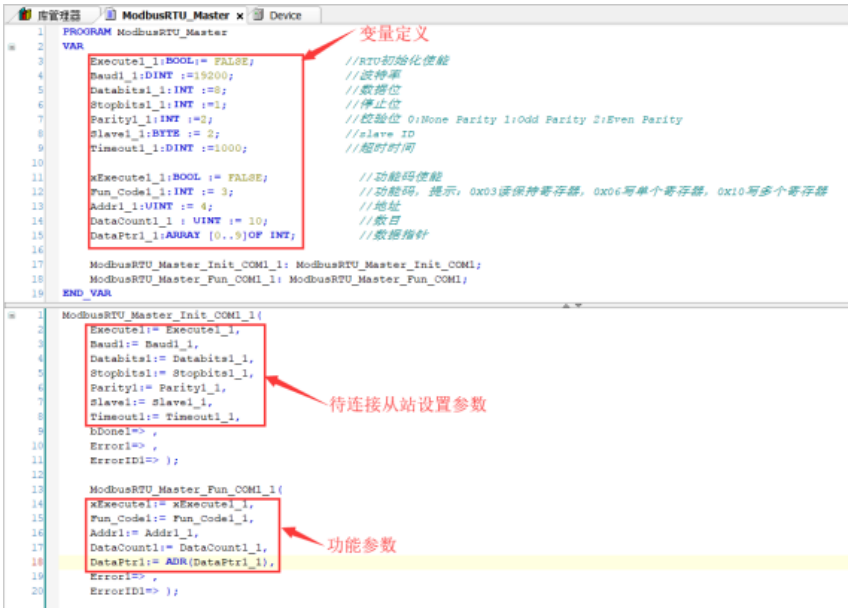


图 A-1 ModbusRTU 主站连接从站参数配置示例

2）完成 ModbuRTU_Master 主站连接从站相关参数配置后，对通信功能进行设置，设置参数如下表，设置参考示例如下图。

设置项	功能	示例
xExecute1	RTU 通信功能使能码	RW:=TRUE
Fun_Code1	功能码	Fun_Code1:=0x03
Addr1	读写寄存器开始地址	Addr:= 2001
DataCount1	读写寄存器数量	Conut:=12
DataPtr1	读写数据存放区域地址指针	ADR（DATE_RTU1）

```
ModbusRTU_Master_Init_COM1_1(
    Execute1:= Execute1_1,
    Baud1:= Baud1_1,
    Databits1:= Databits1_1,
    Stopbits1:= Stopbits1_1,
    Parity1:= Parity1_1,
    Slave1:= Slave1_1,
    Timeout1:= Timeout1_1,
    bDone1=> ,
    Error1=> ,
    ErrorID1=> );

    ModbusRTU_Master_Fun_COM1_1(
    xExecute1:= xExecute1_1,
    Fun_Code1:= Fun_Code1_1,
    Addr1:= Addr1_1,
    DataCount1:= DataCount1_1,
    DataPtr1:= ADR(DataPtr1_1),
    Error1=> ,
    ErrorID1=> );
```

图 A-2 ModbusRTU 主站和从站通信参数设置示例

A.1.2 ModbusRTU 从站库变量定义及使用

A.1.2.1 变量定义

变量从属模块	变量名称	类型		功能	注释
ModbusRTU_Slave1	Execute1	INPUT	BOOL	串口初始化功能激活	0：非激活 1：激活
	Baud1		DINT	波特率	例：115200
	Databits1		INT	数据位	例：8 位、7 位
	Stopbits1		INT	停止位	例：1 停止位、2 停止位
	Parity1		INT	读写标志	0：无校验 1：偶校验 2：奇校验
	Slave_Addr1		UINT	从站号	1~128
	Enable1		BOOL	读写功能激活	0：非激活 1：激活
	Done1	OUTPUT	BOOL	完成标志	0：未完成 1：完成
	ErrorID1		BYTE	错误码	见 ModbusRTU 错误码表

A.1.2.2 使用教程

1）配置串口参数，建立 ModbusRTU 主站与从站连接

从属模块	设置项	功能	示例
ModbusRTU_Slave1	Execute1	从站使能变量	Enable:=TRUE
	Baud1	波特率	Baud1:=19200
	Databits1	数据位	Port:=8
	Stopbits	停止位	Unit:=1
	Parity1	校验位	Parity1:=2
	Timeout1	超时时间	Delay Time:=1000
	Slave_Addr1	从站号	Slave1:=12

按照 ModbusRTU 主站的串口配置参数，参考上表中的参数对从站进行设置。（此处的 Slave_Addr1 应和主站的 Slave1 对应）

2）ModbusRTU 主站和 ModbusRTU 从站进行读写数据通信

使能 Execute1 使 ModbusRTU 从站处于激活状态，若主站功能码为 0x03 读保持寄存器，若主站功能码为 0x10 写多个寄存器，可在变量区定义相应的存储区域，其大小不应小于 ModbusTCP 主站将要写入数据的大小，相应的主站功能码为 0x0F（写多个线圈）或者其他功能码时，操作与上述过程相同。

A.2 ModbusTCP 库指令

A.2.1 ModbusTCP 主站指令库变量定义及使用

A.2.1.1 变量定义

变量名称	类型		功能	注释
Enable	INPUT	BOOL	ModbusTCP 功能激活	0：非激活 1：激活
IP		STRING	从站 IP 地址	例如：’192.168.1.13’
Port		DINT	从站端口号	例：502
Unit		INT	从站单元号	非负整数
DelayTime		INT	回复延时时间	非负整数
Fun_Enable		BOOL	功能码使能	0：非激活 1：激活
fun_code		BYTE	功能码	0x03：读多个寄存器模式 0X10：写多个寄存器模式
Addr		UINT	读写寄存器地址	例；2000、2001
Count		INT	读写寄存器数量	一次性读写寄存器数量最多 120 个
CoilSingleData		INT	写单个线圈	值为 0 或 1
BitPtr		POINTER TO BOOL	读写位数据指针	保存需要读写的位数据
DataPtr		POINTER TO INT	读写指针	存放读取到数据的位置信息或存放要写到寄存器的数据
Done	OUTPUT	BOOL	完成标志	0：指令正在执行 1：执行指令完成
Error		BOOL	错误标志	0：无错误 1：有错误
ErrorID		INT	错误码	见 ModbusTCP 错误码表

A.2.1.2 使用教程

1）ModbusTCP_Master 主站连接从站设置

在工程监控状态下设置将要连接的 ModbusTCP 从站的参数如下表。

设置项	功能	示例
Enable	从站使能变量	Enable := TRUE
IP 地址	主站连接 ModbusTCP 从站的 IP 地址	IP := ‘192.168.1.13’
Port	主站连接 ModbusTCP 从站的端口号	Port := ‘502’
Unit	主站连接 ModbusTCP 从站的单元号	Unit := 3
Delay Time	功能启动超时时间	Delay Time := 1000

主站访问单个从站时，需将上述变量分别赋值。参考示例（功能块图 FDB 创建主程序）如下：

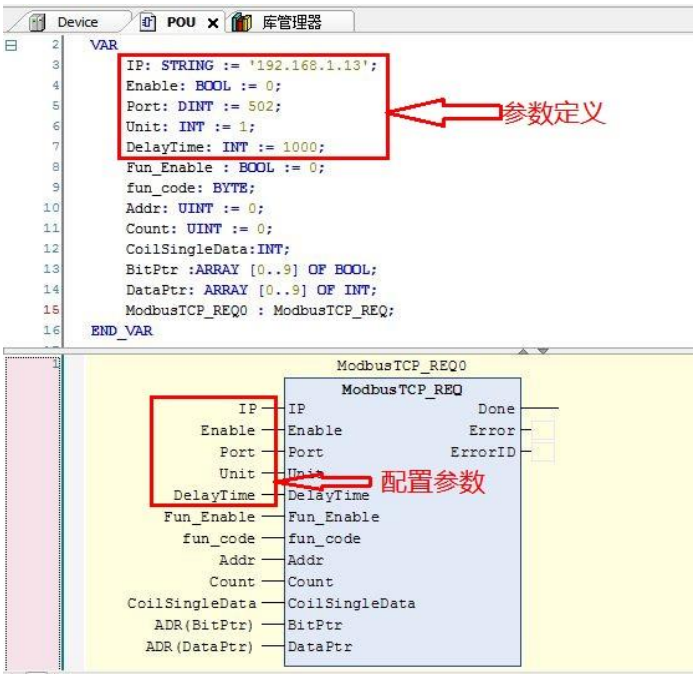


图 A-3 ModbusTCP 主站连接从站参数配置示例

上图中的功能块表示一个独立的 ModbusTCP 主站和从站连接，若添加新的 ModbusTCP 主站从站连接，可先创建一个新的功能块，按上图参数配置示例配置新的参数即可完成新的 ModbusTCP 主站从站连接。

2）完成 ModbusTCP 主站连接从站相关参数配置后，对通信参数进行设置，设置参数如下表，设置参考示例如下图。

设置项	功能	示例
Fun_Enable	功能码使能开关	Fun_Enable:=TRUE
fun_code	读写多个寄存器线圈功能	Fun_code:=3
Addr	读写寄存器开始地址	Addr:=2001
Count	读写寄存器数量	Conut:=12
DataPtr	读写数据存放区域地址指针	ADR（DATE_TCP）

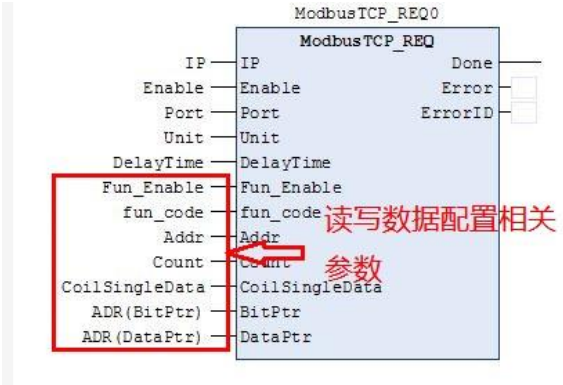


图 A-4 ModbusTCP_Master 主站和从站通信参数设置示例

上图中的每一个运算块表示一个 ModbusTCP 请求。图中定义了一个 ModbusTCP_Master 主站和从站的连接，第一个和第三个运算模块分别表示读不同从站的保持寄存器（0X03）操作，第二个和第四个运算模块表示写入不同从站寄存器内一定数目数据。

若想继续添加上述同一 ModbusTCP_Master 主站从站连接的不同通信要求，可先创建相同 ModbusTCP_Master 主站从站连接的运算块，再按图中示例更改通信参数即可实现新的通信请求。

A.2.2 ModbusTCP 从站指令库变量定义及使用

A.2.2.1 变量定义

变量名称	类型		功能	注释
Enable	INPUT	BOOL	ModbusTCP_Slave 功能激活	0：非激活 1：激活
Port		DINT	从站端口号	采用默认值 502
Unit		INT	从站单元号	从站单元号（1-247）
Done	OUTPUT	BOOL	完成标志	0：指令正在执行 1：指令执行完成
IP		STRING	从站的 IP 地址	本机的 IP 地址（此处不可更改）
Error		BOOL	错误标志	0：无错误 1：有错误
ErrorID		INT	错误码	见 ModbusTCP 错误码表

A.2.2.2 使用说明

1）ModbusTCP 主站从 ModbusTCP_Slave 从站读取数据

使能 Enable 使 ModbusTCP_Slave 从站处于激活状态，若主站功能码为 0x03 读保持寄存器，首先设置 InputSize 的大小，并创建 InputSize 大小的数组用于存放主站要读取的数据，其次将数组的地址赋值给 Inputs 指针。相应的主站功能码为 0x01（读线圈）时，操作与上述过程相同。

2）ModbusTCP 主站将数据写入 ModbusTCP_Slave 从站数据

使能 Enable 使 ModbusTCP_Slave 从站处于激活状态，若主站功能码为 0x10 写多个寄存器，可在变量区定义相应的存储区域，其大小不应小于 ModbusTCP 主站将要写入数据的大小，相应的主站功能码为 0x0F（写多个线圈）或者其他功能码时，操作与上述过程相同。

A.3 CmpHSIO_C 库说明

CmpHSIO_C 库包含了计数，锁存，预设值，脉宽测量，定时采样，计数值比较等多个功能块，通过调用这些功能块来完成计数所需的应用。

A.3.1 Counter_HP

通过此功能块可以实现单脉冲，正交，计时，方向+脉冲计数。

计数功能块在其它模块需要用到计数器时，先调用这个模块对对应的计数器进行设置，参数“更新频率的任务周期数”是为了更新频率的时间内至少能读到 1 个脉冲变化，否则频率显示为 0。通道数 Channel 的范围为 0 到 7。由于高速计数输入会出现干扰，需要设置滤波参数 Filt_Set，在设备描述文件中设置，建议设置值为 2us。

表 A-1 Counter

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	true 使能开始计数，false 不计数
Channel	BYTE	IN	通道数[0,7]
CounterParameter	Counter_Parameter	IN	计数器参数，看 CounterParameter 参数说明
Value	DINT	OUT	当前计数值
Frequency	DWORD	OUT	计数频率（Hz）
Velocity	DWORD	OUT	计数速度（r/min）
Direction	BOOL	OUT	True 为负方向，false 为正方向
Break	BOOL	OUT	true 断线，false 无断线

Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

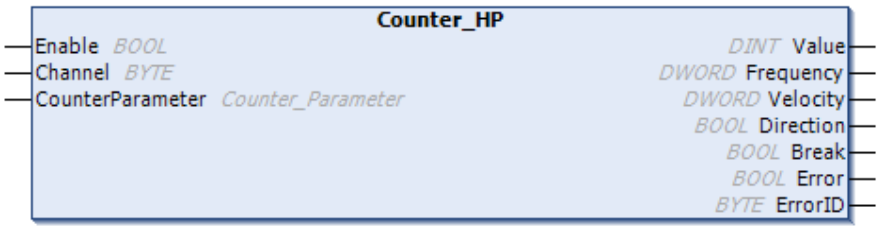
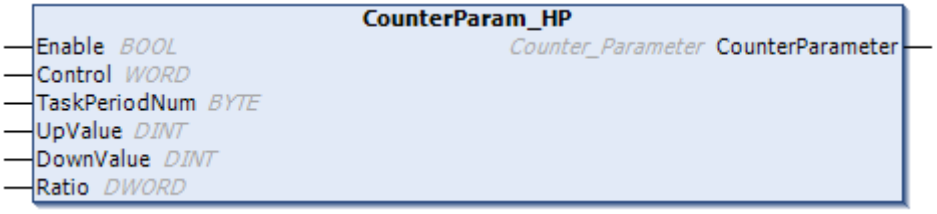


图 A-5 Counter

CounterParameter 参数说明

STRUCT **Counter_Parameter**

Name	Type	Inherited from	Address	Initial	Comment
Control	WORD			1	控制信号设置
TaskPeriodNum	BYTE			1	更新频率的任务周期数
UpValue	DINT			20000	上限值设置
DownValue	DINT			-100	下限值设置
Ratio	DWORD			10000	分辨率



Control：设置可查看如下 Control 设置说明。

TaskPeriodNum：设置多少个任务周期更新一次脉冲频率。

UpValue：计数器的上限值，设置计数为线性计数时最大为这个值，不再往上计数。

DownValue：计数器的下限值，设置计数为线性计数时最小为这个值，不再往下计数。

Ratio：计数器的分辨率，表示一圈的计数值，用于频率计算。

控制字的 bit 位与功能的对应关系如下表：

	控制字功能	功能值说明
0	计数(计时)使能	0：无效 1：有效
1~2	倍频模式	0：正交倍频 1 倍频 1：正交倍频 4 倍频
3	计数(计时)清零	0：无效 1：有效
4~6	计时单位	0：1us 1：10us 2：100us 3：1ms
7	单脉冲和计时方向	0：单脉冲和计时方向，正方向 1：单脉冲和计时方向，负方向

	控制字功能	功能值说明
8	计数模式	0：环形计数 1：线性计数
9~11	预设值和计数值锁存控制	1：软件触发写入 2：外部触发写入，外部触发源 CnT 3：比较一致时触发写入 4：锁存功能，外部触发源 CnT
12~15	保留	保留

A.3.1.1 单脉冲计数

输入端口配置成计数功能，计数模式配置为单脉冲计数，每个计数通道有 CxA、CxB 两个信号，A 为脉冲输入，B 为低电平。x 为通道数，0 =< x <= 7，目前计数器最多只支持 8 个通道。

功能配置

A：计数模式配置

计数器模式功能配置

//计数 0 和 1 计数模式配置 单脉冲设置值为 0，byte 的低 4 位设置计数器 0，高 4 位设置计数器 1;

xmodea:=16#00;

//计数 2 和 3 计数模式配置 单脉冲设置值为 0，byte 的低 4 位设置计数器 2，高 4 位配置计数器 3;

xmodeb := 16#00;

配置计数器模式变量映射

Application.xmodea		XMode_SetA	%QB16	BYTE
Application.xmodeb		XMode_SetB	%QB17	BYTE

B：输入端口功能配置，设置为计数功能

in0:=in1:=1;//计数器 0 输入端口设置为计数功能

输入端口变量映射

Application.in0		In0_Configure	%QB0	BYTE
Application.in1		In1_Configure	%QB1	BYTE

C：信号滤波参数配置

filt_set:= 8;//单位 0.25us，相当于 2us，对于不同干扰可以调整这个值

滤波参数变量映射

Application.filt_set		Filt_Set	%QB20	BYTE

D：控制参数配置

根据功能块设置控制参数

控制字设置，以下为按位（bit）操作

//计数使能

```
Control.0:=1;

//倍频设置 1 为 4 倍频只有正交计数有效，0 为 1 倍频，
Control.1:=0;
Control.2:=0;

//计数清零 1 清零 0 不清零
Control.3:=0;

//计数方向 0 正方向 1 负方向
Control.7:=0;

//计数方式 1 线性计数，0 循环计数
Control.8:=0;

//计时单位选择。0 为 1us， 1 为 10us， 2 为 100us， 3 为 1ms，非计时模式这个参数无效
Control.4:=0;
Control.5:=0;
Control.6:=0;
```

预设值控制：

1 软件触发写入 ；

2 外部触发写入，外部触发源可以选择 X8，X9，XA，XB 中任何一个（改为：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

3 比较一致时触发写入。

计数（计时）值锁存控制：

4 计数值锁存使能，外部触发源可以选择 X8，X9，XA，XB 中任何一个（改为：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

```
Control.9:=0;
Control.10:=0;
Control.11:=0;

counterparam[0].Control:= Control;//控制字
counterparam[0].TaskPeriodNum:=1;// 多少个任务周期更新一次脉冲频率
counterparam[0].UpValue:=10000000;
counterparam[0].DownValue:=-1000;
```

counterparam[0].Ratio:=10000;

示例程序代码

```
Counter0(
    Enable:= TRUE,
    Channel:= 0, //选择计数器 0，不同计数器修改这个值范围【0,7】
    CounterParameter:=counterparam[0],
    Value=> value0, //输出计数
    Frequency=> fre0, //输出计数频率值
    Velocity=> vel0, //输出计数速度值
    Direction=> ,
    Break=> ,
    Error=> ,
    ErrorID=> );
```

时序说明

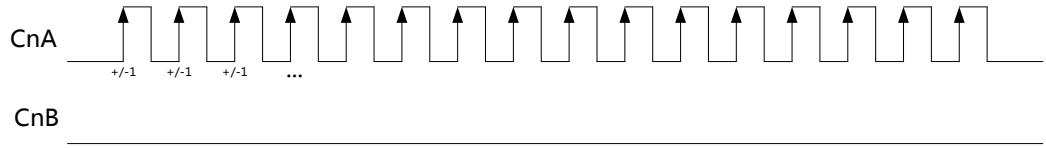


图 A-6 单脉冲输入示意图

说明：

单脉冲计数需要根据配置的计数方向来确定是累加还是累减。当方向是正向时，每次来一个脉冲计数器加一，反之减一。n 表示计数通道，0 =< n <= 7。

单脉冲常见于生产线物件的计数，感应器每次检测到一个物件则输出一个高电平脉冲。

A.3.1.2 正交计数

正交信号常见于正交编码器输出信号，包含有 A、B、Z 三个信号，其中 A、B 是相位相差 90°的脉冲信号，Z 是原点信号，每圈产生一个脉冲。Z 信号一般用来清除计数器、补偿、原点定位。在计数中，大多数使用情况并没有用到 Z 信号。

输入端口配置成计数功能，计数模式配置为正交计数，16 个输入端口都可以选择作为正交计数，目前计数器最多只支持 8 个通道。

功能配置

A：计数模式配置

计数器模式功能配置

//计数 0 和 1 计数模式配置 正交计数设置值为 1，byte 的低 4 位设置计数器 0，高 4 位设置计数器 1;

xmodea:=16#11;

//计数 2 和 3 计数模式配置 正交计数设置值为 1，byte 的低 4 位设置计数器 2，高 4 位配置计数器 3;

xmodeb := 16#11;

配置计数器模式变量映射

Application.xmodea		XMode_SetA	%QB16	BYTE
Application.xmodeb		XMode_SetB	%QB17	BYTE

B：输入端口功能配置，设置为计数功能

in0:=in1:=1;//计数器 0 输入端口设置为计数功能

输入端口变量映射

Application.in0		In0_Configure	%QB0	BYTE
Application.in1		In1_Configure	%QB1	BYTE

C：信号滤波参数配置

filt_set:= 8;//单位 0.25us，相当于 2us，对于不同干扰可以调整这个值

滤波参数变量映射

Application.filt_set		Filt_Set	%QB20	BYTE

D：控制参数配置

控制字设置，以下为按位（bit）操作

//计数使能

Control.0:=1;

//倍频设置 1 为 4 倍频只有正交计数有效，0 为 1 倍频

Control.1:=0;

Control.2:=0;

//计数清零 1 清零 0 不清零

Control.3:=0;

//计数方向 0 正方向 1 负方向

Control.7:=0;

//计数方式 1 线性计数，0 循环计数

Control.8:=0;

//计时单位选择。0 为 1us， 1 为 10us， 2 为 100us， 3 为 1ms，非计时模式这个参数无效

Control.4:=0;

Control.5:=0;

Control.6:=0;

预设值控制：

1 软件触发写入 ；

2 外部触发写入，外部触发源可以选择 X8，X9，XA，XB 中任何一个（即：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

3 比较一致时触发写入。

计数（计时）值锁存控制：

4 计数值锁存使能，外部触发源可以选择 X8，X9，XA，XB 中任何一个（即：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

```
Control.9:=0;
Control.10:=0;
Control.11:=0;
```

计数功能块设置控制参数

```
counterparam[0].Control:= Control;//控制字
counterparam[0].TaskPeriodNum:=1;// 多少个任务周期更新一次脉冲频率
counterparam[0].UpValue:=10000000;
counterparam[0].DownValue:=-1000;
counterparam[0].Ratio:=10000;
```

示例程序代码

```
Counter0(
    Enable:= TRUE,
    Channel:= 0, //选择计数器 0，不同计数器修改这个值范围【0,7】
    CounterParameter:=counterparam[0],
    Value=> value0, //输出计数
    Frequency=> fre0, //输出计数频率值
    Velocity=> vel0, //输出计数速度值
    Direction=> ,
    Break=> ,
    Error=> ,
    ErrorID=> );
```

时序说明

（1）正向

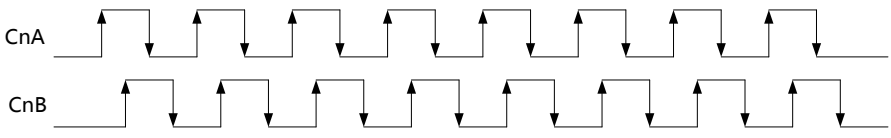


图 A-7 正交脉冲正向输入示意图

（2）反向

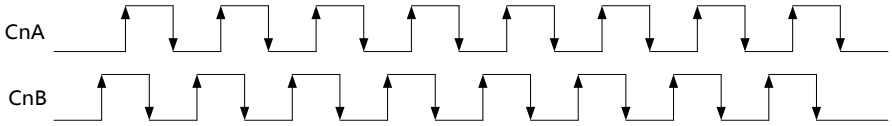


图 A-8 正交脉冲反向输入示意图

说明：

正交脉冲计数需要根据编码器转动方向来确定是累加还是累减。当方向是正向时（A 相超前于 B 相 90°），根据倍频方式来进行累加，如果一倍频则 CnA 完整一个周期加一，如果四倍频则 CnA 和 CnB 每个信号沿都加一。当方向是反向时（B 相超前于 A 相 90°）计数器进行累减，如果一倍频则 CnA 完整一个周期减一，如果四倍频则 CnA 和 CnB 每个信号沿都减一。n 表示计数通道，0 =< n <= 7。

A.3.1.3 计时计数

输入端口可以不配置，计数模式配置为计时计数，按设置的时间单位进行计数，目前计数器最多只支持 8 个通道。

计时计数，实际上就是实现了一个功能简单的时钟功能。可以预设起始计时点、时间单位、计时时长（通过设置比较值），达到计时时长时可输出比较相等信号。计时完成后还可以重置各参数并重新定时。计时计数需要根据配置的计数方向来确定是累加还是累减。当方向是正向时，每隔一个周期计数器加一，反之减一。

功能配置

A：计数模式配置

计数器模式功能配置

//计数 0 和 1 计数模式配置 计时计数设置值为 2，byte 的低 4 位设置计数器 0，高 4 位设置计数器 1;

xmodea:=16#22;

//计数 2 和 3 计数模式配置计时计数设置值为 2，byte 的低 4 位设置计数器 2，高 4 位配置计数器 3;

xmodeb := 16#22;

配置计数器模式变量映射

Application.xmodea		XMode_SetA	%QB16	BYTE
Application.xmodeb		XMode_SetB	%QB17	BYTE

B：输入端口功能配置，设置为计数功能（可以不配置没有影响）

in0:=in1:=1;//计数器 0 输入端口设置为计数功能

输入端口变量映射

Application.in0		In0_Configure	%QB0	BYTE
Application.in1		In1_Configure	%QB1	BYTE

C：信号滤波参数配置（可以不配置没有影响）

filt_set:= 8;//单位 0.25us，相当于 2us，对于不同干扰可以调整这个值

滤波参数变量映射

Application.filt_set		Filt_Set	%QB20	BYTE

D：控制参数配置

控制字设置，以下为按位（bit）操作

//计数使能

Control.0:=1;

//倍频设置 1 为 4 倍频只有正交计数有效，0 为 1 倍频

Control.1:=0;

Control.2:=0;

//计数清零 1 清零 0 不清零

Control.3:=0;

//计数方向 0 正方向 1 负方向

Control.7:=0;

//计数方式 1 线性计数，0 循环计数

Control.8:=0;

//计时单位选择。0 为 1us， 1 为 10us， 2 为 100us， 3 为 1ms，计时计数按这个设置单位计数

Control.4:=0;

Control.5:=0;

Control.6:=0;

预设值控制：

1 软件触发写入；

2 外部触发写入，外部触发源可以选择 X8，X9，XA，XB 中任何一个；

3 比较一致时触发写入。

计数（计时）值锁存控制：

4 计数值锁存使能，外部触发源可以选择 X8，X9，XA，XB 中任何一个；

Control.9:=0;

Control.10:=0;

Control.11:=0;

计数功能块设置控制参数

counterparam[0].Control:= Control;//控制字

counterparam[0].TaskPeriodNum:=1;// 多少个任务周期更新一次脉冲频率

counterparam[0].UpValue:=10000000;

counterparam[0].DownValue:=-1000;

counterparam[0].Ratio:=10000;

示例程序代码

```
Counter0(
    Enable:= TRUE,
    Channel:= 0, //选择计数器 0，不同计数器修改这个值范围【0,7】
    CounterParameter:=counterparam[0],
    Value=> value0, //输出计数
    Frequency=> fre0, //输出计数频率值
    Velocity=> vel0, //输出计数速度值
    Direction=> ,
    Break=> ,
    Error=> ,
    ErrorID=> );
```

时序说明

说明：

所有计数通道都可以进行计时计数。

A.3.1.4 脉冲+方向计数

脉冲+方向信号包含 CxA、CxB，CxA 接脉冲信号，CxB 接方向信号。方向信号高电平表示正向，低电平表示反向。x 是通道数，0 =< x <= 7。

输入端口配置成计数功能，计数模式配置为脉冲+方向计数，16 个输入端口都可以选择作为方向加脉冲计数，目前计数器最多只支持 8 个通道。

功能配置

A：计数模式配置

计数器模式功能配置

//计数 0 和 1 计数模式配置脉冲+方向计数设置值为 3，byte 的低 4 位设置计数器 0，高 4 位设置计数 1;

```
xmodea:=16#33;
```

//计数 2 和 3 计数模式配置脉冲+方向计数设置值为 3，byte 的低 4 位设置计数器 2，高 4 位配置计数 3;

xmodeb := 16#33;

配置计数器模式变量映射

Application.xmodea		XMode_SetA	%QB16	BYTE
Application.xmodeb		XMode_SetB	%QB17	BYTE

B：输入端口功能配置，设置为计数功能

in0:=in1:=1;//计数器 0 输入端口设置为计数功能

输入端口变量映射

Application.in0		In0_Configure	%QB0	BYTE
Application.in1		In1_Configure	%QB1	BYTE

C：信号滤波参数配置

filt_set:= 8;//单位 0.25us，相当于 2us，对于不同干扰可以调整这个值

滤波参数变量映射

Application.filt_set		Filt_Set	%QB20	BYTE

D：控制参数配置

控制字设置，以下为按位（bit）操作

//计数使能

Control.0:=1;

//倍频设置 1 为 4 倍频只有正交计数有效，0 为 1 倍频

Control.1:=0;

Control.2:=0;

//计数清零 1 清零 0 不清零

Control.3:=0;

//计数方向 0 正方向 1 负方向

Control.7:=0;

//计数方式 1 线性计数，0 循环计数

Control.8:=0;

//计时单位选择。0 为 1us， 1 为 10us， 2 为 100us， 3 为 1ms，非计时模式这个参数无效

Control.4:=0;

Control.5:=0;

Control.6:=0;

预设值控制：

1 软件触发写入 ；

2 外部触发写入，外部触发源可以选择 X8，X9，XA，XB 中任何一个（改为：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

3 比较一致时触发写入。

计数（计时）值锁存控制：

4 计数值锁存使能，外部触发源可以选择 X8，X9，XA，XB 中任何一个（改为：CnT，n 为计数通道，0 =< n <= 3，每个触发源对应一个计数通道。）；

Control.9:=0;

Control.10:=0;

Control.11:=0;

计数功能块设置控制参数

counterparam[0].Control:= Control;//控制字

counterparam[0].TaskPeriodNum:=1;// 多少个任务周期更新一次脉冲频率

counterparam[0].UpValue:=10000000;

counterparam[0].DownValue:=-1000;

counterparam[0].Ratio:=10000;

示例程序代码

```
Counter0(
    Enable:= TRUE,
    Channel:= 0, //选择计数器 0，不同计数器修改这个值范围【0,7】
    CounterParameter:=counterparam[0],
    Value=> value0, //输出计数
    Frequency=> fre0, //输出计数频率值
    Velocity=> vel0, //输出计数速度值
    Direction=> ,
    Break=> ,
    Error=> ,
    ErrorID=> );
```

时序说明

（1）正向

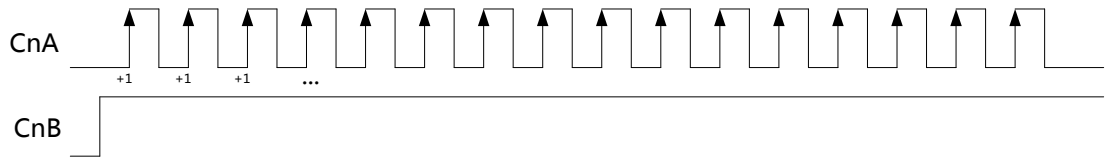


图 A-9 脉冲+方向正向输入示意图

（2）反向

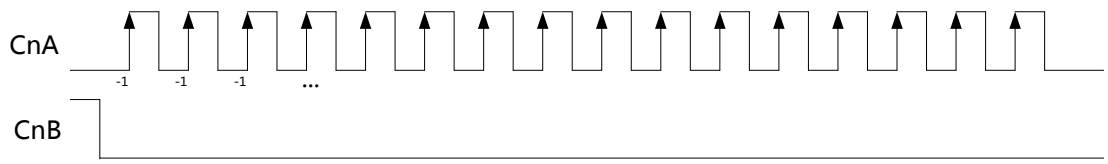


图 A-10 脉冲+方向反向输入示意图

说明：

脉冲+方向计数需要根据方向信号来确定是累加还是累减。当方向是正向时，每次来一个脉冲计数器加一，反之减一。n 表示计数通道，0 =< n <= 7。

A.3.2 LatchValue_HP

锁存值读取，调用这个模块前要调用 Counter_HP 模块对使用的计数器进行参数设置。本模块通过选择 CxT 选择触发信号，有信号时（上升沿触发锁存）锁存对应的值，只有信号 X8，X9，XA，XB 有触发功能。在计数器中需要设置计数值锁存控制等参数，说明 Done 在锁存值为 0 时 done 不会置为 true。

表 A-2 Latch_Value

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Channel	BYTE	IN	通道数[0,3]
Value	DINT	OUT	锁存值
Done	BOOL	OUT	执行完成标志
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-11 Latch_Value

A.3.2.1 功能配置

A：配置 Counter_HP 功能块

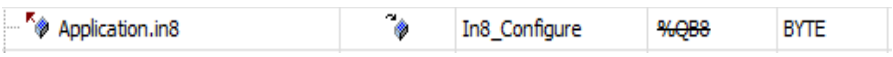
详见 Counter_HP 功能块说明。

针对锁存功能的特别配置

1：配置输入端口为触发锁存功能，

例程：配置 X8 为触发锁存端口

in8:=2;



2：控制参数配置为锁存使能

例程：配置锁存使能

Control.9:=0;

Control.10:=0;

Control.11:=1;

B：中断配置（如果需要中断功能）

详见探针中断说明

C：配置 LatchValue_HP 功能块

功能块 LatchValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致

例程：选用计数器 0，锁存值存放在 latch0

```
latchValue0(
    Enable:= TRUE,
    Channel:= 0,
    Value=> latch0,
    Done=> ,
    Error=> ,
    ErrorID=> );
```

A.3.2.2 时序说明

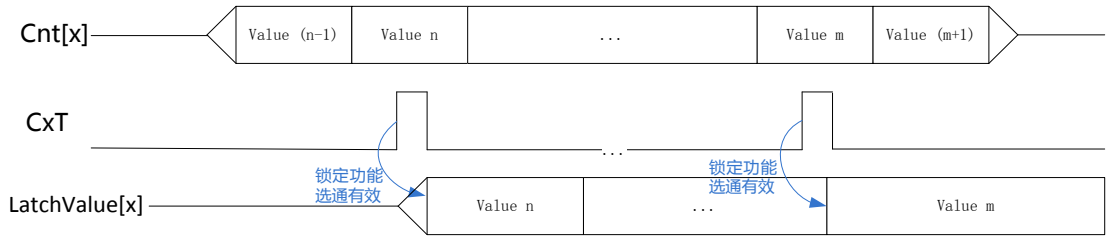


图 A-12 锁存功能示意图

说明：

x 是计数通道，0 =< x <= 3，Cnt[x]是第 x 计数通道的计数值，CxT 是第 x 通道的锁存信号，LatchValue[x]是第 x 通道的锁存值。当 CxT 锁存触发信号到来时（锁存功能必须配置正确），此时 Cnt[x]计数值将被锁存至 LatchValue[x]，上位机可根据需要读取 LatchValue[x]值。LatchValue[x]是 32bit 有符号数，最高位是符号位。

A.3.3 PresetValue_HP

计数器预设值写入，有三种方式：软件写入、外部触发写入、计数值比较相等写入。在调用这个模块前要调用 Counter_HP 模块对使用的计数器进行参数设置。只有输入计数器 0，1，2，3 四个通道有参数预设功能，在计数器中需要设置预设值控制等参数。说明：Done 表示预设值已写入 FPGA，要根据设置参数在计数器中启作用，在预设值为 0 时 done 不会置为 true。

表 A-3 Preset_Value

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Channel	BYTE	IN	写入通道数[0,3]
Value	DINT	IN	预设值(起始值)
Done	BOOL	OUT	完成标志，1 为完成
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-13 Preset_Value

A.3.3.1 功能配置

有三种预设值的方式，在实际使用过程中可以根据需要选用一种方式。

软件触发写入

这个方式下功能块 *PresetValue_HP* 使能写入预设值。所谓的软件写入，就是由上位机 ARM 进行写入。

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明。针对预设值功能的特别设置，

控制参数配置为预设值软件触发写入。

Control.9:=1;

Control.10:=0;

Control.11:=0;

B：配置 PresetValue_HP 功能块

功能块 PresetValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致。

例程：选用计数器 0，预设值为 10000

```
Set_Value0(
    Enable:= bPreSetFlag,
    Channel:= 0,
    Value:= 10000,
    Done=> ,
    Error=> ,
    ErrorID=> );
```

外部触发写入

这个方式下功能块 *PresetValue_HP* 使能，在有外部触发信号 CxT 时写入预设值。CxT 上升沿有效。

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明。

针对预设值功能的特别设置如下，

1：配置输入端口为触发锁存功能，

例程：配置 X8 为触发锁存端口

in8:=2;

2：控制参数配置为预设值为外部触发写入。

Control.9:=0;

Control.10:=1;

Control.11:=0;

B：配置 PresetValue_HP 功能块

功能块 PresetValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致。

例程：选用计数器 0，预设值为 10000，在端口触发时写入预设值。

```
Set_Value0(
 Enable:= bPreSetFlag,
 Channel:= 0,
 Value:= 10000,
 Done=> ,
 Error=> ,
 ErrorID=> );
```

比较一致触发写入

这个方式下功能块 *PresetValue_HP* 使能，在功能块 CompareSingleValue_HP 比较一致时写入预设值。

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明，针对比较一致触发功能的特别设置如下。

控制参数配置为预设值为比较一致触发写入。

```
 Control.9:=1;
    Control.10:=1;
    Control.11:=0;
```

B：配置 CompareSingleValue_HP 功能块

详见 CompareSingleValue_HP 功能块说明。

C：配置 PresetValue_HP 功能块

功能块 PresetValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致。

例程：选用计数器 0，预设值为 10000，在 CompareSingleValue_HP 比较一致时写入预设值。

```
Set_Value0(
 Enable:= bPreSetFlag,
 Channel:= 0,
 Value:= 10000,
 Done=> ,
 Error=> ,
 ErrorID=> );
```

A.3.3.2 时序说明

（1）软件触发

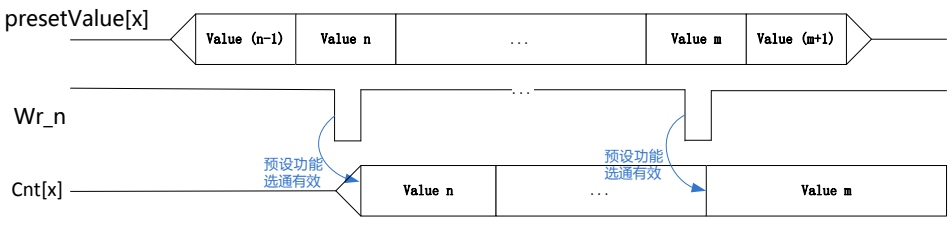


图 A-14 软件触发预设功能示意图

说明：

x 表示计数通道，0 =< x <= 3，presetValue[x]是第 x 计数通道的预设值，Wr_n 是上位机写入信号，低电平有效，Cnt[x]是 x 通道计数器计数值。当 Wr_n 低电平到来时，presetValue[x]的值被预设进 Cnt[x]。

（2）外部触发

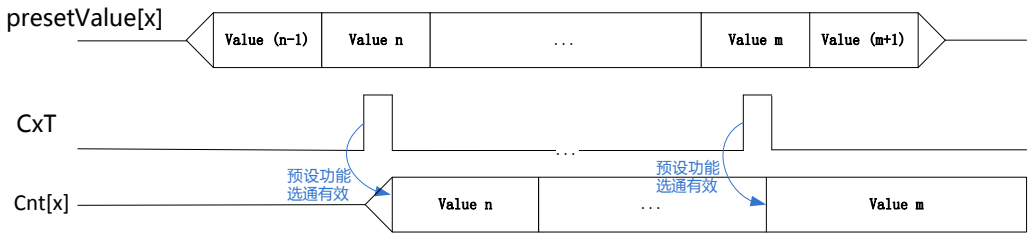


图 A-15 外部触发预设功能示意图

说明：

x 表示计数通道，0 =< x <= 3。presetValue[x]是第 x 计数通道的预设值，实际就是 CompareSingleValue_HP 模块下发的 Value 值。CxT 是第 x 通道外部预设触发信号，上升沿有效，Cnt[x]是 x 通道计数器计数值。当 CxT 上升沿到来时，presetValue[x]的值被预设进 Cnt[x]。

（3）计数相等触发

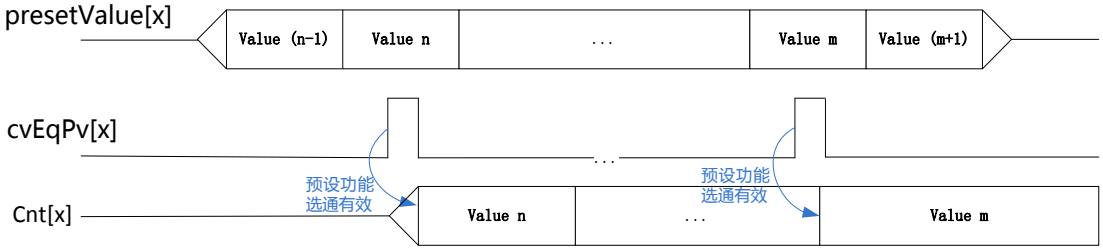


图 A-16 单值比较相等触发预设功能示意图

说明：

x 表示计数通道，0 =< x <= 3，presetValue[x]是第 x 计数通道的预设值，cvEqPv[x]是第 x 通道单值比较相等信号，高电平有效，Cnt[x]是 x 通道计数器计数值。当 cvEqPv[x]高电平到来时，presetValue[x]的值被预设进 Cnt[x]。presetValue[x]是 32bit 有符号数，最高位是符号位。

A.3.4 PulsewidthMeasure_HP

脉宽测量信号 PWCx，本模块只有输入信号 X8，X9，XA，XB 信号配置相应功能有效。通道数采用低 4 位使能通道，0 位表示 1 通道，1 位表示 2 通道，2 位表示 3 通道，3 位表示 4 通道。例：Channel：=2#00001010；表示通道 2、4 使能。

表 A-4 Pulsewidth_Measure

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Channel	BYTE	IN	通道数（低 4 位有效）
Mode	BYTE	IN	脉宽测量模式：1 为高电平 0 为低电平
Value0	DINT	OUT	通道 0 脉宽测量值（0.01us）
Value1	DINT	OUT	通道 1 脉宽测量值（0.01us）
Value2	DINT	OUT	通道 2 脉宽测量值（0.01us）

Value3	DINT	OUT	通道 3 脉宽测量值（0.01us）
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

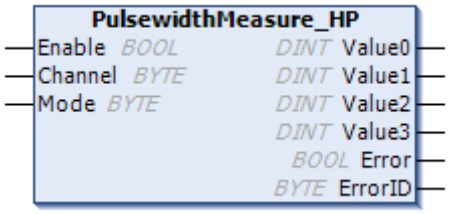


图 A-17 Pulsewidth_Measure

A.3.4.1 功能配置

这个功能块只要对输入端口配置为脉宽测量功能 PWC 就可以调用功能块 *PulsewidthMeasure_HP 进行脉宽测量*。

例程 1：对 X9 进行脉宽测量，ch1_Value 为高电平脉宽测量值

输入端口配置

in9:=4;

Application.in9		In9_Configure	%QB9	BYTE

功能块程序，

```
PWM0(
    Enable:= TRUE,
    Channel:= 2#00000010,
    Mode:= 2#00000010, //高电平测量有效
    Value0=> ,
    Value1=> ch1_Value,
    Value2=>,
    Value3=>,
    Error=> ,
    ErrorID=> );
```

例程 2：对 X8，X9，XA ，XB 进行脉宽测量，ch0_Value, ch1_Value, ch2_Value,ch3_Value 分别为 4 个端口的高电平脉宽测量值。

输入端口配置

in8:= in9:=4;

inA:= inB:=4;

Application.in8		In8_Configure	%QB8	BYTE
Application.in9		In9_Configure	%QB9	BYTE
Application.inA		InA_Configure	%QB10	BYTE
Application.inB		InB_Configure	%QB11	BYTE

功能块程序

```
PWM0(
    Enable:= TRUE,
    Channel:= 2#00001111, //4 个通道
    Mode:= 2#00001111, //低 4 位表示 4 个通道
    Value0=>ch0_Value ,
    Value1=> ch1_Value,
    Value2=> ch2_Value,
    Value3=>ch3_Value ,
    Error=> ,
    ErrorID=> );
```

A.3.4.2 时序说明

（1）正脉冲脉宽检测

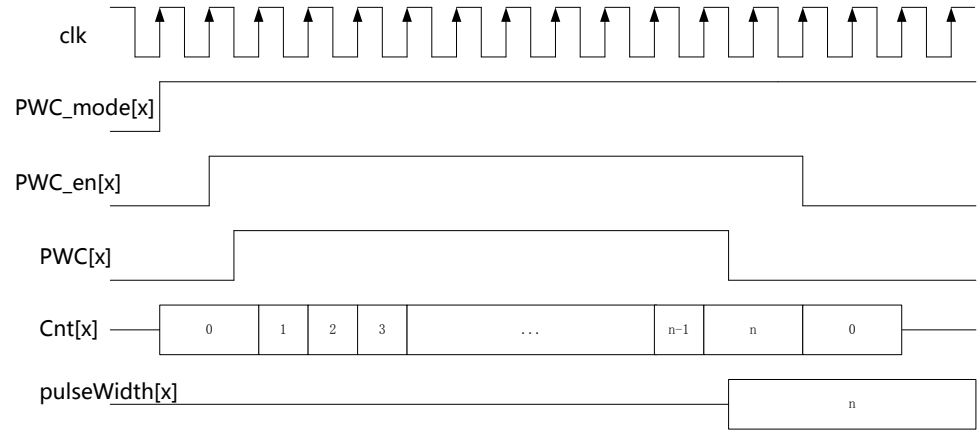


图 A-18 高电平正脉冲脉宽检测示意图

（2）负脉冲脉宽检测

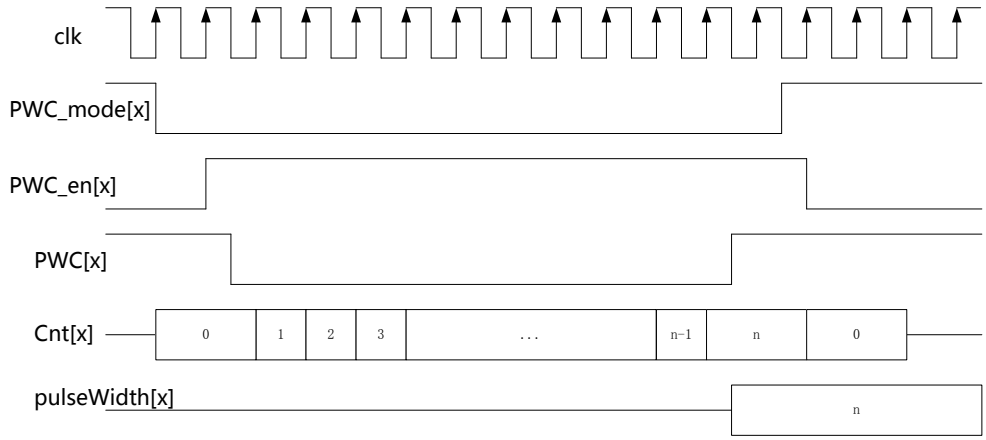


图 A-19 高电平负脉冲脉宽检测示意图

正负脉宽检测说明：

x 是计数通道，0 =< x <= 3。PWC_mode[x]是第 x 通道的检测模式，高电平表示检测正脉冲，低电平表示负脉冲。PWC_en[x]是第 x 通道的使能，高有效。PWC[x]是第 x 通道脉冲输入信号。Cnt[x]是第 x 通道脉宽检测计数器。PulseWidth[x]是第 x 通道脉冲宽度值，单位是 0.01us，是无符号数。

A.3.5 SetCompareInterruptParam_HP

这个功能块用来设置比较中断源的选择。

表 A-5 SetCompareInterruptParam

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
MoreOrSingle_Sel	BYTE	IN	多值比较中断选择
MoreValueCount_Sel	BOOL	IN	多值比较计数通道选择
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-20 SetCompareInterruptParam

MoreOrSingle_Sel 参数说明：

计数器中断状态输出选择，每一 bit 位控制一个中断通道。比较中断总共有 8 个。MoreOrSingle_Sel 各 Bit 位值对应的比较中断说明如下：

MoreOrSingle_Sel 位	对应中断	位值为 1	位值为 0
0	比较中断 0	多值比较计数器第 0 个比较点的中断	计数器 0 单值比较中断
1	比较中断 1	多值比较计数器第 1 个比较点的中断	计数器 1 单值比较中断
2	比较中断 2	多值比较计数器第 2 个比较点的中断	计数器 2 单值比较中断
3	比较中断 3	多值比较计数器第 3 个比较点的中断	计数器 3 单值比较中断
4	比较中断 4	多值比较计数器第 4 个比较点的中断	计数器 4 单值比较中断
5	比较中断 5	多值比较计数器第 5 个比较点的中断	计数器 5 单值比较中断
6	比较中断 6	多值比较计数器第 6 个比较点的中断	计数器 6 单值比较中断
7	比较中断 7	多值比较计数器第 7 个比较点的中断	计数器 7 单值比较中断

MoreValueCount_Sel 参数说明：

用于多值比较中断的计数通道选择。MoreValueCount_Sel 值含义说明如下：

MoreValueCount_Sel 值	选择的计数通道
0	计数器 0
1	计数器 1
2	计数器 2
3	计数器 3

A.3.5.1 功能配置

使用这个功能块需要调用 CompareMoreValue_HP 功能块配合使用，详细看 CompareMoreValue_HP 说明。

例程：选择计数器 3 作为多值比较中断，并在比较中断 0 和比较中断 1 产生中断。

```
interrupt_sel:=16#11;//选择多值比较中断
count_sel:=16#3;//选择多值比较中断计数器
SetCompareInterruptParam(
Enable:= enableparam,
MoreOrSingle_Sel:= interrupt_sel,
MoreValueCount_Sel:= count_sel,
Error=> ,
ErrorID=> );
```

A.3.6 TimingSampling_HP

定时采样，就是在某个给定的时间范围内，计算出所采集到的脉冲数量，可以是输入通道支持的各种脉冲信号，包括单脉冲、CW/CCW、计时、脉冲+方向。调用这个模块前要调用 Counter_HP 模块对使用的计数器进行参数设置。修改采样时间前请使 enable 为 false，不然可能采样可能会出现异常。

表 A-6 Timing_Sampling

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Channel	BYTE	IN	通道数[0,7]
SampleEnable	BOOL	IN	采样使能
Timeset	DWORD	IN	采样时间设置（us）
Value	DINT	OUT	采样值
Done	BOOL	OUT	执行标志 1 为完成
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-21 Timing_Sampling

A.3.6.1 功能配置

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明。针对定时采样功能块不需要另外设置特别参数。

B：配置 TimingSampling_HP 功能块

功能块 *TimingSampling_HP* 的 Channel 设置值与 Counter_HP 的 Channel 值一致。

例程：选用计数器 1，采样时间为 20000us，采样脉冲值输出到 sampleValue1。

```
Sampling1(
```

```
Enable:= TRUE,
Channel:= 1,
SampleEnable:=TRUE,
Timeset:= 20000, //us
Value=> sampleValue1,
Done=> ,
Error=> ,
ErrorID=> );
```

A.3.6.2 时序说明

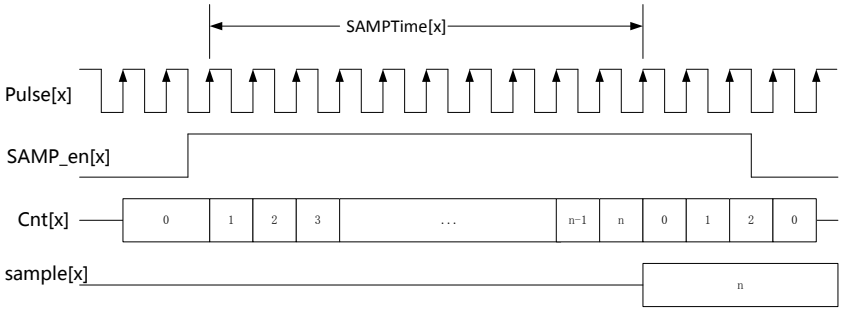


图 A-22 采样示意图

说明：

x 表示第 x 通道，0 =< x <= 3。Pulse[x]表示第 x 通道的输入脉冲信号，可以是输入通道支持的各种脉冲信号，包括单脉冲、CW/CCW、计时、脉冲+方向。SAMP_en[x]表示第 x 通道使能，高有效。SAMPTime[x]表示第 x 通道采样时间。Sample[x]表示第 x 通道采样得到的脉冲数，是无符号数。

A.3.7 CompareSingleValue_HP

单值比较输出，调用这个模块前要调用 Counter_HP 模块对使用的计数器进行参数设置。使能上升沿更新参数有效，低电平模块无效。OutChanne 的值范围为 0 到 7。

表 A-7 compare_singlevalue

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Start_Cmp	BOOL	IN	开始比较
Channel	BYTE	IN	计数通道[0,7]
OutChannel	DINT	IN	选择输出通道[0,7]
CmpValue	DINT	IN	设定比较值
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

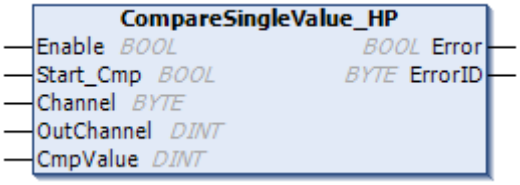


图 A-23 compare_singlevalue

A.3.7.1 功能配置

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明，针对单值比较功能块没有特别配置。

B：中断配置

详见比较中断说明

C：配置 CompareSingleValue_HP 功能块

功能块 CompareSingleValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致

例程：选用计数器 3，设置比较值为 10000，输出通道为 0。

```
Cmp3(
 Enable:= TRUE,
 Start_Cmp:= bStart,
 Channel:= 3, //计数器
 OutChannel:= 0, //输出通道
 CmpValue:= 10000, //比较值
 Error=> ,
 ErrorID=> );
```

A.3.7.2 时序说明

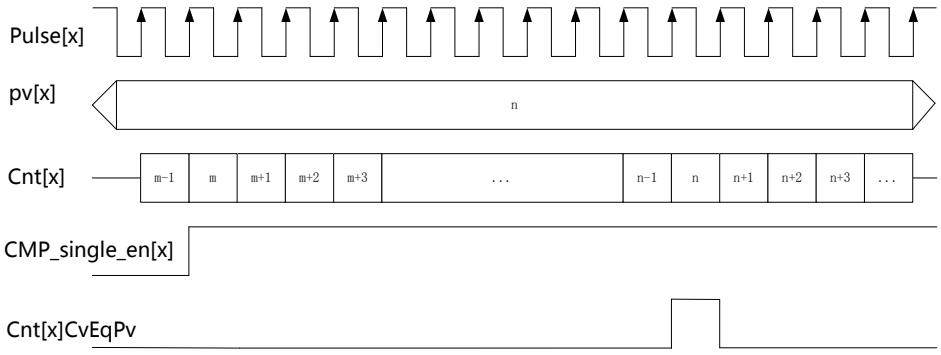


图 A-24 单值比较中断时序示意图

单值比较说明：

x 表示计数通道，0 =< x <= 7。Pulse[x]表示第 x 通道输入的脉冲，可以是输入通道支持的各种脉冲信号，包括单脉冲、CW/CCW、计时、脉冲+方向。Pv[x]是第 x 通道的比较值。Cnt[x]是第 x 通道计数器计数值。CMP_single_en[x]是第 x 通道单值比较使能。Cnt[x]CvEqPv 是通道 x 的单值比较输出，高电平有效，高电平表示计数值与 pv 相等。

此处给出累加计数的情形，累减计数也一样的道理，计数值与 pv 值相等则输出 Cnt[x]CvEqPv 有效。

A.3.8 CompareMoreValue_HP

多值比较输出，调用这个模块前要调用 Counter_HP 模块对使用的计数器进行参数设置。比较值一定顺序增加或顺序减小，对应计数器设置为正方向或反方向。比较最大为 8 个数。使能上升沿更新参数有效，低电平模块无效。

表 A-8 compare_morevalue

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Start_Cmp	BOOL	IN	开始比较

参数名称	参数类型	输入输出类型	参数作用
Channel	BYTE	IN	计数通道[0,7]
CmpValue_Num	BYTE	IN	比较值个数[1,100]
CmpValue	POINTER TO DINT	IN	设定比较值
CmpEqual_Num	BYTE	OUT	比较相等的个数[1,100]
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-25 compare_morevalue

A.3.8.1 功能配置

A：配置 Counter_HP 功能块

详见 Counter_HP 功能块说明，针对多值比较功能块没有特别配置。

B：中断配置（如果需要多值比较中断）

详见比较中断说明

C：配置 *SetCompareInterruptParam_HP*（如果需要多值比较中断）

详见 SetCompareInterruptParam_HP 功能块说明

C：配置 CompareSingleValue_HP 功能块

功能块 CompareMoreValue_HP 的 Channel 设置值与 Counter_HP 的 Channel 值一致。

例程：选用计数器 0，设置比较值为从 1000 到 8000，总共 8 个比较值，多值比较中断输出通道为 0 和 1。

```
    FOR comp_num:=0 TO 7 BY 1 DO
      cmpvalue[comp_num]:=1000+1000*comp_num;
END_FOR


interrupt_sel:=16#3;
count_sel:=16#0;
SetCompareInterruptParam(
Enable:= enableparam,
MoreOrSingle_Sel:= interrupt_sel,
MoreValueCount_Sel:= count_sel,
Error=> ,
ErrorID=> );


compValue_num:=8;
pcmpvalue:=ADR(cmpvalue[0]);//取比较值数组地址
```

```
cmpmore0(
    Enable:= benabele,
    Start_Cmp:= bcmpmore,
    Channel:= 0,
    CmpValue_Num:=compValue_num , //总共多少个比较值
    CmpValue:= pcmpvalue, //比较值存放地址输入指针
    CmpEqual_Num=> CmpEqual_Num0, //当前第几个比较值相等
    Error=> ,
    ErrorID=> );
```

A.3.8.2 时序说明

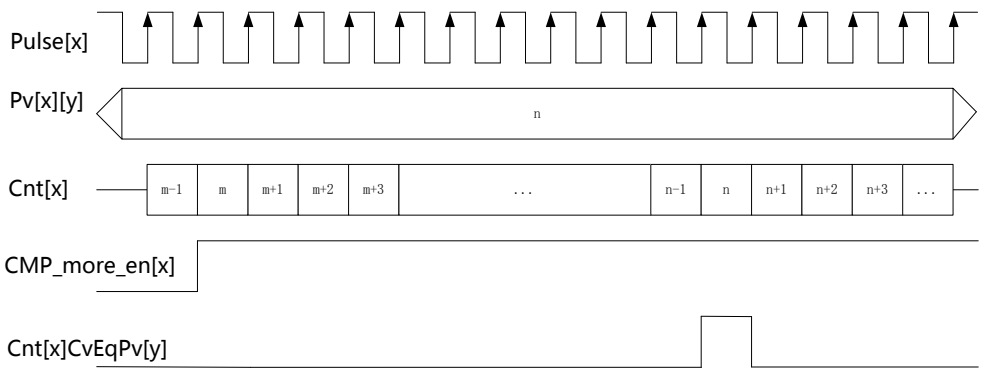


图 A-26 多值比较中断时序示意图

多值比较说明：

x 表示计数通道，0 =< x <= 3。y 表示所选计数通道的第几个比较输出值，0 =< y <= 7。Pulse[x]表示所选的第 x 计数通道的输入脉冲，可以是输入通道支持的各种脉冲信号，包括单脉冲、CW/CCW、计时、脉冲+方向。Pv[x][y]是第 x 计数通道的第 y 个比较值。Cnt[x]是第 x 通道计数器计数值。CMP_more_en 是多值比较使能。Cnt[x]CvEqPv[y]是通道 x 的第 y 个值比较输出，高电平有效，高电平表示计数值与 pv 相等。

此处给出累加计数的情形，累减计数也一样的道理，计数值与 pv 值相等则输出 Cnt[x]CvEqPv[y]有效。

A.3.9 GetVersion_HP

表 A-9 get_version

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
Version	STRING	OUT	版本

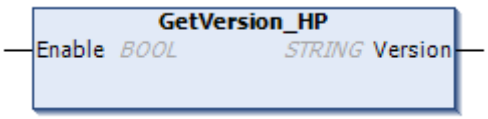


图 A-27 get_version

A.3.10 Zphase_Clearpulse_HP

计数通道 Z 信号清 0 功能，当高速计数器检测到计数通道的 Z 信号时，对计数器值清 0，实际使用时需要对输入信号配置为 Z 信号功能，输入端口 X4，X5，X6，X7 有 Z 信号功能。Enable 使能上升沿更新轴使能有效，低电平模块无效。

如果清零和补偿功能同时打开，清零优先级高执行清零功能。

表 A-10 Zphase_Clearpulse

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
bEnableAxis0	BOOL	IN	使能通道 0 Z 相清脉冲
bEnableAxis1	BOOL	IN	使能通道 1 Z 相清脉冲
bEnableAxis2	BOOL	IN	使能通道 2 Z 相清脉冲
bEnableAxis3	BOOL	IN	使能通道 3 Z 相清脉冲
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-28 Zphase_Clearpulse

A.3.10.1 功能配置

A：配置 Counter_HP 功能块

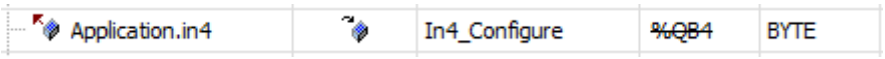
详见 Counter_HP 功能块说明。

针对计数通道 Z 信号清 0 的特别配置

1：配置输入端口为 Z 信号功能，

例程：配置 X4 为 Z 信号功能

in4:=2;



B：配置 Zphase_Clearpulse_HP 功能块

例程：选用计数通道 0 和计数通道 1 具有 Z 相清零功能

```
Zphase_Clearpulse_FB(
  Enable:= TRUE,
  bEnableAxis0:= TRUE ,
  bEnableAxis1:= TRUE ,
  bEnableAxis2:= ,
  bEnableAxis3:= ,
  Error=> ,
  ErrorID=> );
```

A.3.10.2 时序说明

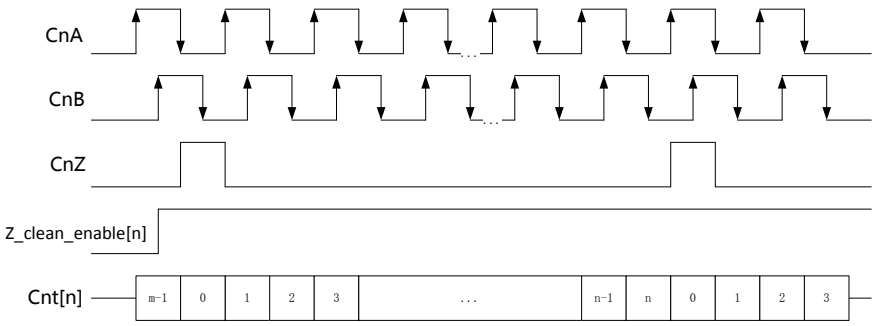


图 A-29 清零功能时序示意图

说明：

n 表示第 n 个通道，0 =< n <= 3。Z_clean_enable[n]表示第 n 通道 Z 清零使能，高有效。Cnt[n]表示第 n 通道计数器计数值。此处给出正向计数方式作为例子说明，反向计数也一样的道理，Z 到来清零然后反向计数。

A.3.11 Zphase_Compensate_HP

计数通道 Z 信号补偿功能，当高速计数器检测到计数通道的 Z 信号时，对计数器值按计数器分辨率参数 Ratio 进行补偿，实际使用时需要对输入信号配置为 Z 信号功能，输入端口 X4，X5，X6，X7 有 Z 信号功能。Enable 使能上升沿更新轴使能有效，低电平模块无效。如果清零和补偿功能同时打开，清零优先级高执行清零功能。上电后补偿功能有效需要计数值最少有一次变化输入，否则不会补偿。

表 A-11 Zphase_Compensate

参数名称	参数类型	输入输出类型	参数作用
Enable	BOOL	IN	使能
bEnableAxis0	BOOL	IN	使能通道 0 Z 相脉冲补偿
bEnableAxis1	BOOL	IN	使能通道 1 Z 相脉冲补偿
bEnableAxis2	BOOL	IN	使能通道 2 Z 相脉冲补偿
bEnableAxis3	BOOL	IN	使能通道 3 Z 相脉冲补偿
Error	BOOL	OUT	出错标志
ErrorID	BYTE	OUT	错误码

图 A-30 Zphase_Compensate

A.3.11.1 功能配置

A：配置 Counter_HP 功能块

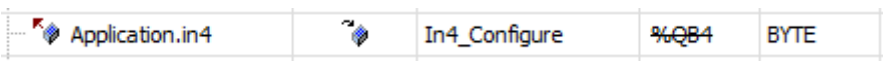
详见 Counter_HP 功能块说明。

针对计数通道 Z 信号补偿的特别配置

1：配置输入端口为 Z 信号功能，

例程：配置 X4 为 Z 信号功能

in4:=2;



B：配置 Zphase_Compensate_HP 功能块

例程：选用计数通道 0 和计数通道 1 具有 Z 相补偿功能

```
Zphase_Compensate_FB(
Enable:= TRUE,
bEnableAxis0:= TRUE,
bEnableAxis1:=TRUE ,
bEnableAxis2:= ,
bEnableAxis3:= ,
Error=> ,
ErrorID=> );
```

A.3.11.2 时序说明

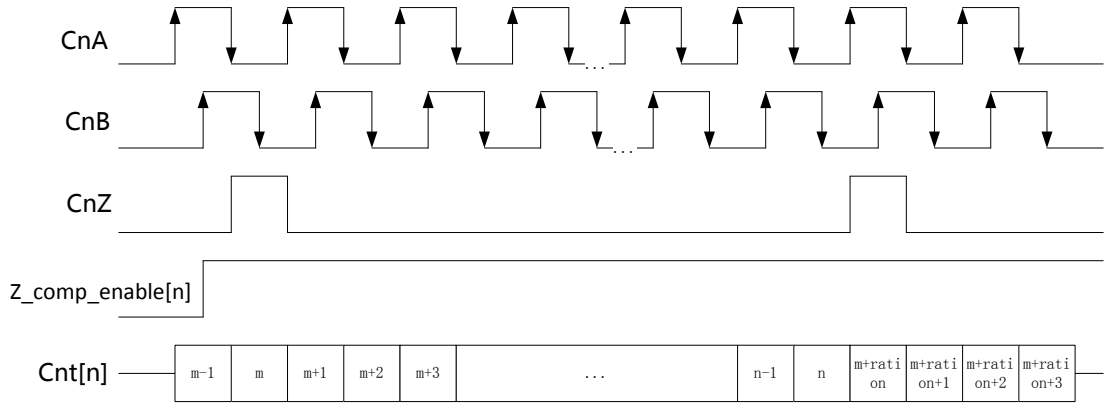


图 A-31 补偿功能时序示意图

说明：

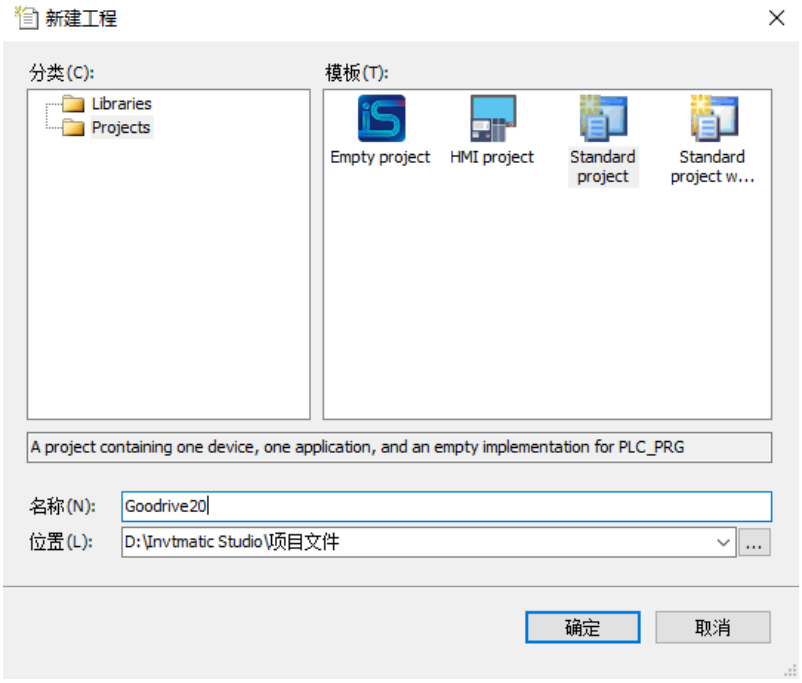
n 表示第 n 个通道，0 =< n <= 3。Z_comp_enable[n]表示第 n 通道 Z 补偿使能，高有效。Cnt[n]表示第 n 通道计数器计数值。此处给出正向计数补偿作为参考例，反向也是一样的道理，Z 到来进行反向补偿（减去 ration）然后反向计数。

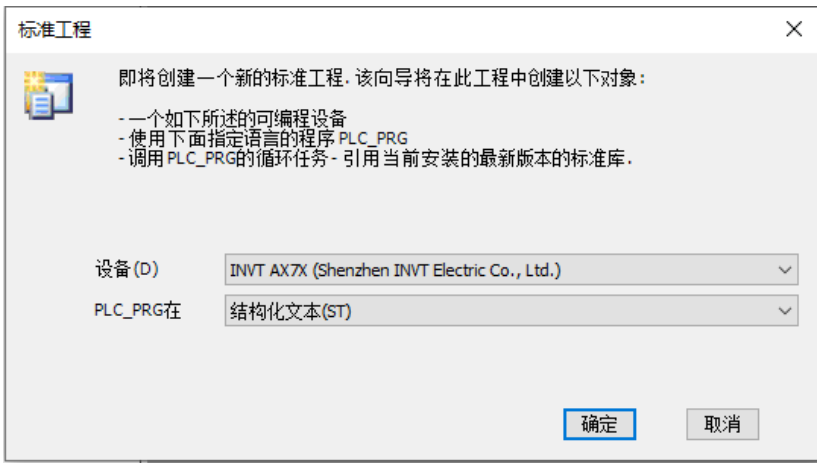
附录B 工程实例

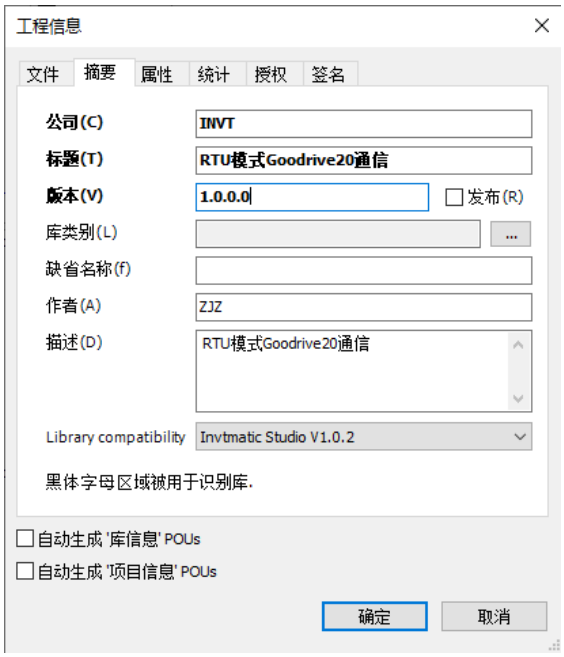
B.1 控制器与 Goodrive20 系列变频器配置实例

现在我们将 AX 系列控制器设为主机，将一台 Goodrive20 系列变频器设为从机，控制器使用 Modbus /RTU 通讯协议，其物理层为两线制 RS485，通过 COM2 口与变频器通信。我们编写一个小程序，使用上位机对 Goodrive20 变频器的功能参数进行读写。

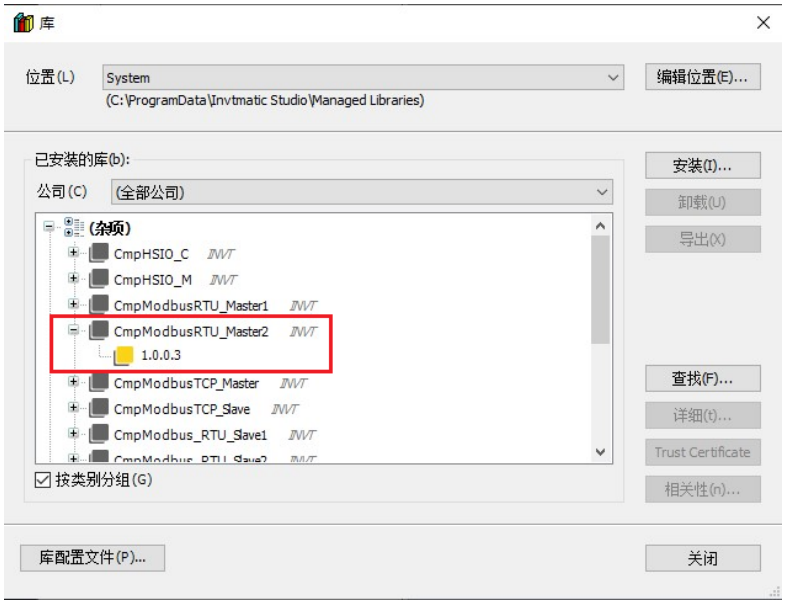
1、 新建一个工程，选择菜单“文件 > 新建工程”，新建一个标准工程，设备为 INVT AX7x，编程语言为结构化文本（ST），根据实际需要，编辑工程信息，如下图所示。



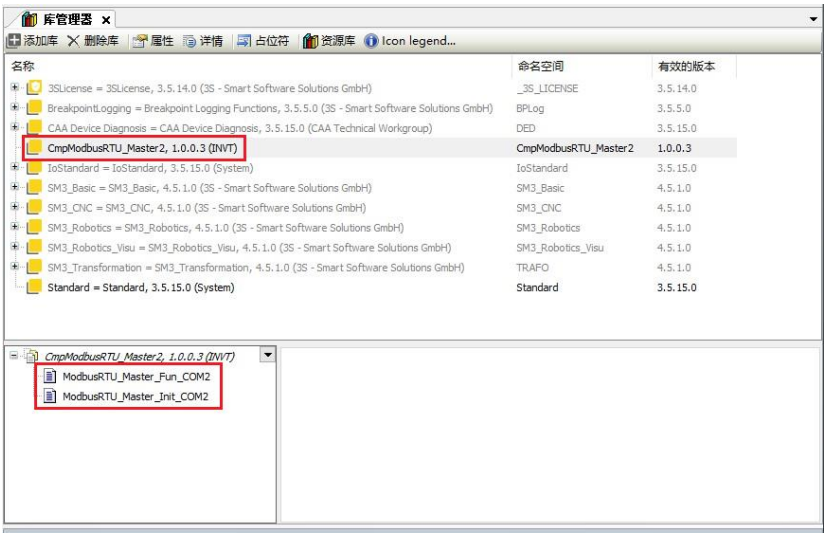




2、 选择菜单“工具 > 库”，安装库文件 CmpModbusRTU_Master2_1.0.0.3.library，如下图所示：



3、 选择“库管理器 > 添加库”，将安装好的库添加到应用，如下图所示：



4、 双击 PLC_PRG，在声明编辑器上输入以下代码。

```
PROGRAM PLC_PRG
VAR
    ModbusRTU_Master_Fun_COM2: ModbusRTU_Master_Fun_COM2;
    ModbusRTU_Master_Init_COM2: ModbusRTU_Master_Init_COM2;
    DatePtr2:ARRAY[0..0]OF INT;
    input_registers_Ptr2:ARRAY[0..9]OF INT;
    CoilDataPtr2:ARRAY[0..9]OF BOOL;
    input_bits_Ptr2:ARRAY[0..9]OF BOOL;
    CoilSingleData2:INT;
    Fun_Code2:INT;
    Addr2:UINT;
    DataCount2 : UINT：=1;
END_VAR
```

在主体代码编辑器里输入以下代码：

```
ModbusRTU_Master_Init_COM2(
        Execute2:= 1,
        Baud2:= 19200,
        Databits2:= 8,
        Stopbits2:=1 ,
        Parity2:=2 ,
        Timeout2:= 1000,
        bDone2=> ,
        Error2=> ,
        ErrorID2=> );
ModbusRTU_Master_Fun_COM2(
        xExecute2:= 1,
        Fun_Code2:= Fun_Code2,
        Addr2:= Addr2,
        Slave2:= 1,
        DataCount2:= DataCount2,
        CoilDataPtr2:=ADR(CoilDataPtr2) ,
        CoilSingleData2:= CoilSingleData2,
        input_bits_Ptr2:= ADR(input_bits_Ptr2),
    input_registers_Ptr2:=ADR(input_registers_Ptr2) ,
```

DataPtr2:=ADR(DatePtr2),

Done2=> ,

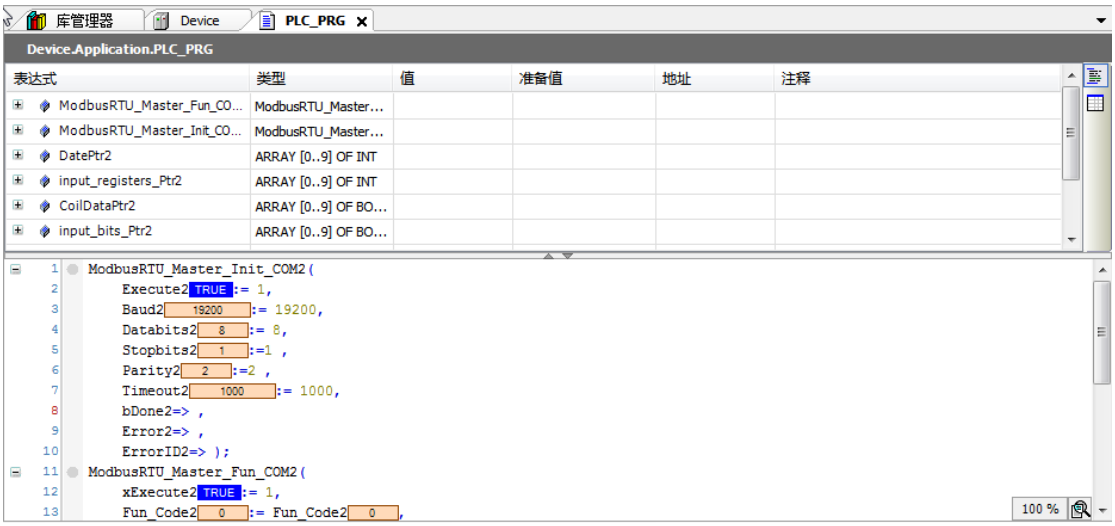
Error2=> ,

ErrorID2=>);

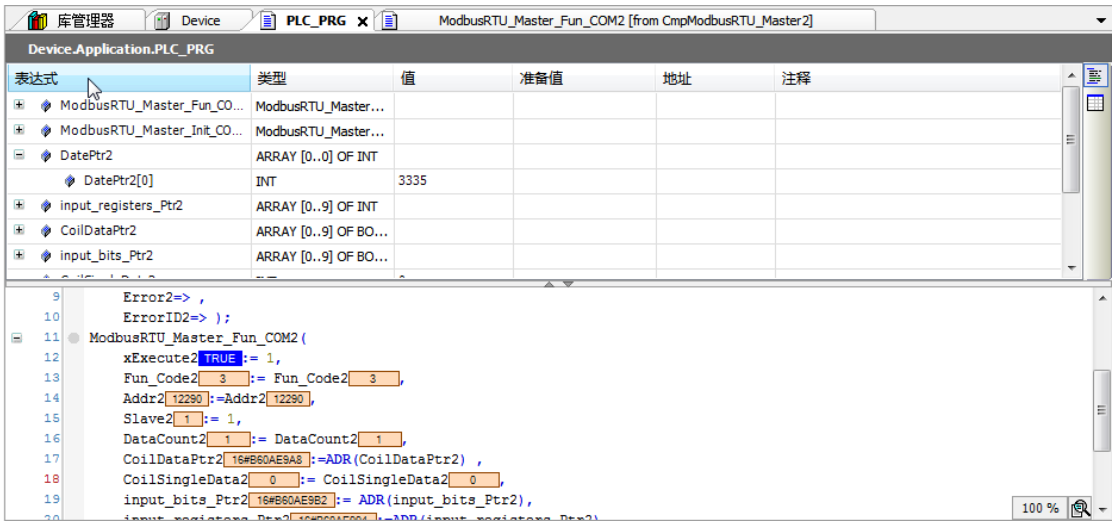
现在我们对程序做必要的说明。程序调用库 CmpModbusRTU_Master2 的两个功能块，ModbusRTU_Master_Init_COM2 和 ModbusRTU_Master_Fun_COM2。其中 ModbusRTU_Master_Init_COM2 用于初始化 RTU Master2，这里将波特率设定为 19200，数据位为 8，停止位为 1，校验位为偶检验，超时时间为 1000ms；ModbusRTU_Master_Fun_COM2 是功能模块的使能和具体应用，变量 Fun_Code2 是标准 Modbus 功能码，Addr2 是变频器 Goodrive20 功能的地址，关于 MODBUS 其他功能的地址说明可以参考 INVTGoodrive20 系列变频器的产品说明书，Slave2 是变频器从机地址，这里设定为 1。

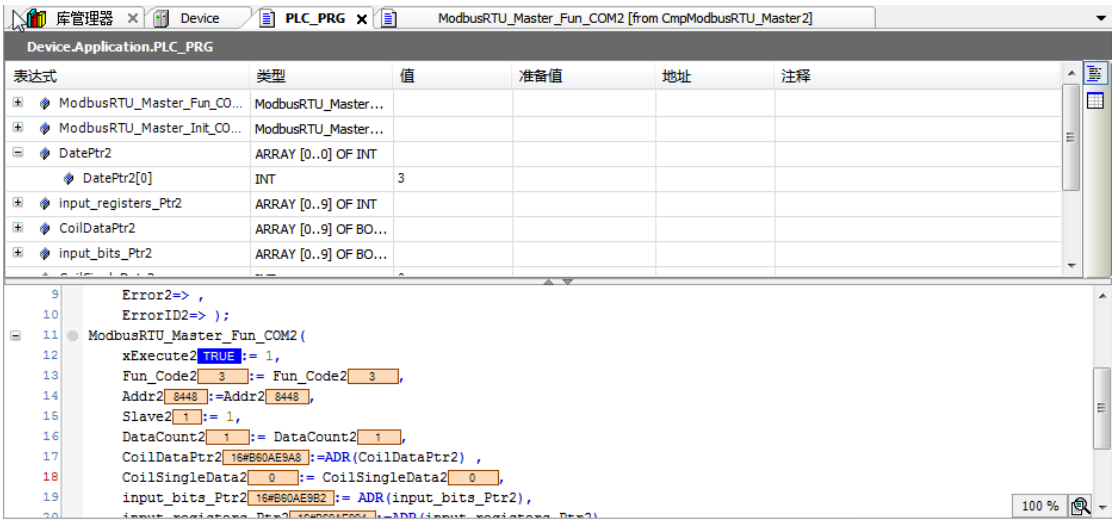
将变频器通过两线制 RS485 和控制器相连接后，启动变频器。通过变频器键盘设置功能码 P00.01 为 2，使其运行命令可由上位机通过通讯方式进行控制；设置 P00.06 为 8，即选择 MODBUS 通讯方式；设置 P14 组的串行通讯参数，使其与上位机的初始化设定参数包括波特率、数据位、校验位、从机地址、超时时间等一致。

点击工具栏的 按钮编译代码。编译没有错误后，点击工具栏的 按钮登录控制器。检查控制器数码管无报错，变频器 Goodrive20 与控制器顺利连接，通讯正常。上位机界面如下图所示。

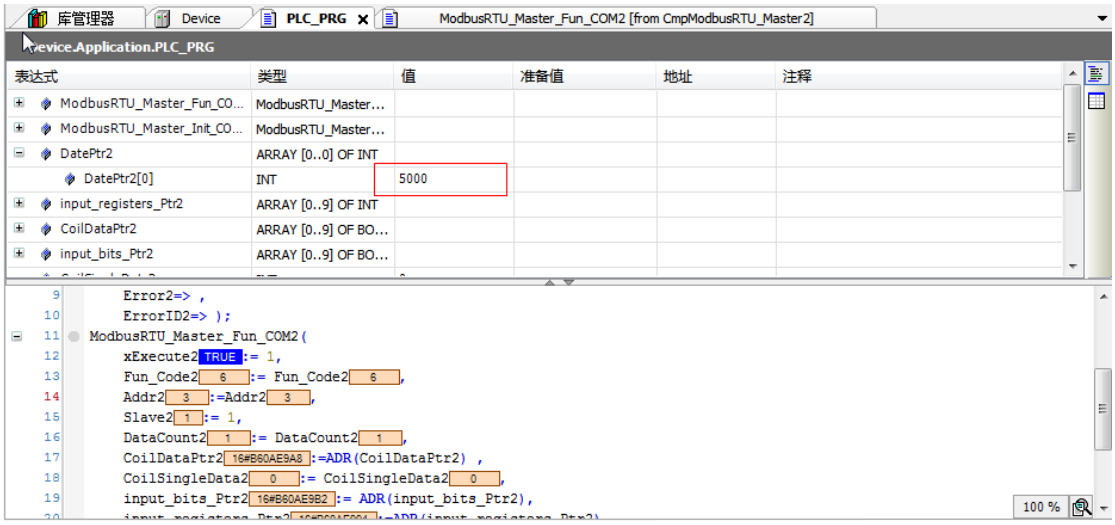


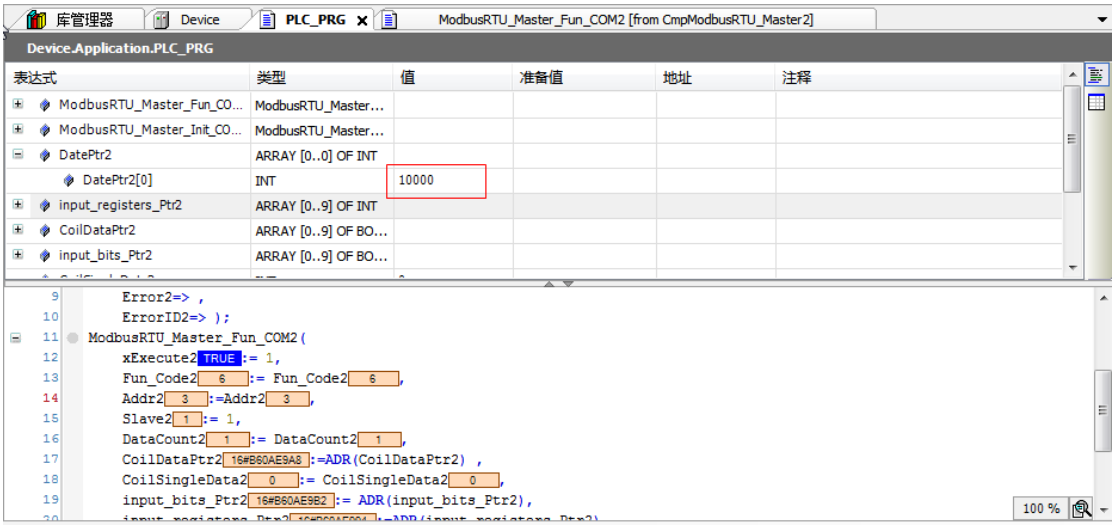
现在我们对读操作进行举例。在登录状态对变量写值，Fun_Code 写入值 3，表示 03H 功能码 Read Holding Registers ，Addr 写入值 16#3002，表示从 3002H 开始读取 1 个地址，可以在数组 DataPtr2 即 3002H 地址，读到值 3335，参考变频器产品手册，表示母线电压为 333.5V。同样地，Fun_Code 写入值 3，表示 03H 功能码 Read Holding Registers ，Addr 写入值 16#2100，可以在数组 DataPtr2 即 2100H 地址，读到值 3，参考变频器产品手册，表示变频器停机中，如图所示。





现在我们对写操作进行举例。在登录状态对变量写值，Fun_Code 写入值 6，表示 06H 功能码 Write Single Register ，Addr 写入值 16#0003，表示往地址 0003H 开始写入 1 个数值，参考变频器产品手册，0003H 是变频器的最大输出频率地址，其值默认为 50.00HZ。在未对该地址写值前，我们可以在上位机看到地址 0003H 的值为 5000，这个值是由 50.00Hz 乘上比例值 100 得到的，现在我们将变频器的最大输出频率设为 100Hz，则需在 0003H 写入值 100Hz*100，即 10000，这样以后，在变频器上使用键盘查看 P00.03 的值，可看到该值从 50.00 变成了 100.00，说明控制器对变频器写入成功。如图所示。

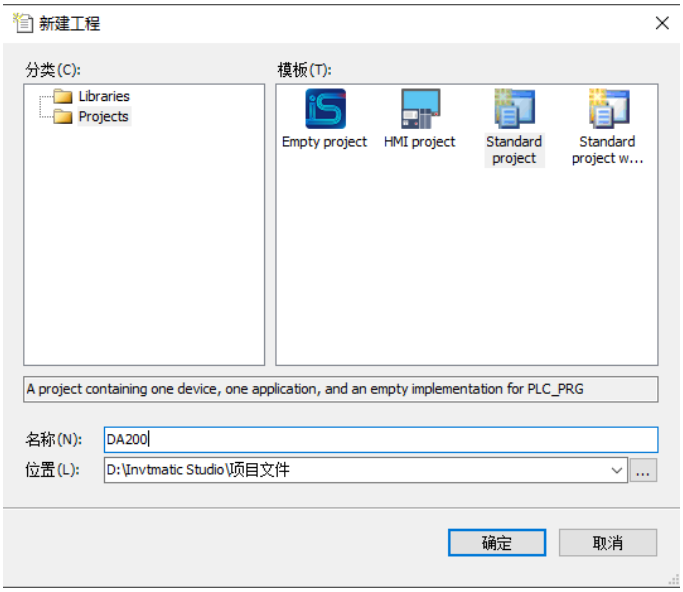


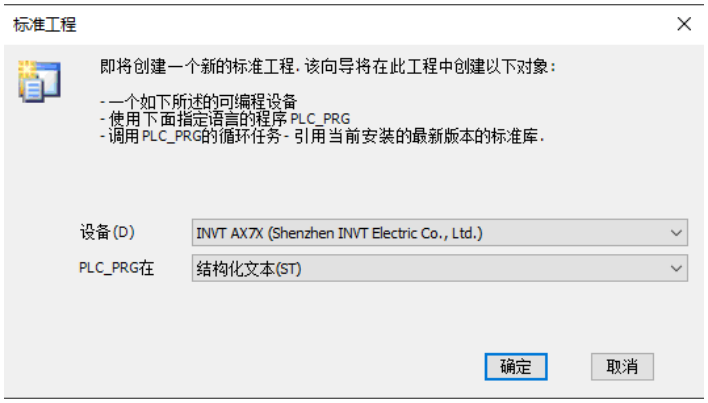


B.2 控制器与 DA200 系列伺服驱动器配置实例

现在我们编写一个小程序，控制 4 台 DA200 系列的伺服驱动器，驱动 4 台电机轴做匀速正反转运动。

1、 新建一个工程，选择菜单“文件 > 新建工程”，新建一个标准工程，设备为 INVT AX7x，编程语言为结构化文本（ST），根据实际需要，编辑工程信息，如下图所示。

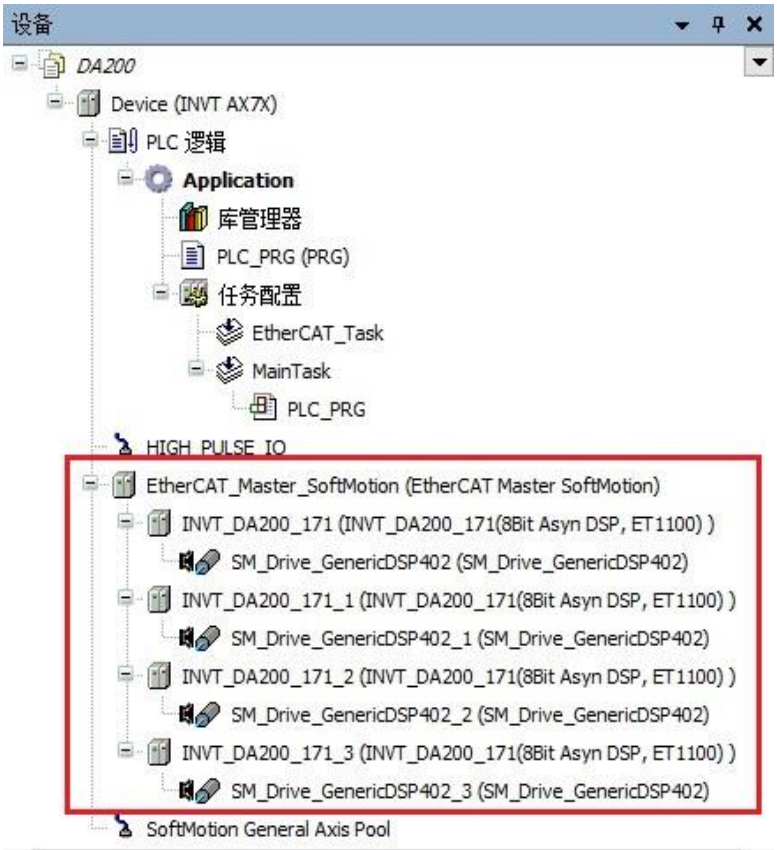




2、 在设备栏选中“Device”，右键选择“添加设备”，添加 EtherCAT 主站设备，这里选择“EtherCAT Master SoftMotion”，版本 3.5.15.0，如下图所示。

3、 在设备栏选中“EtherCAT Master SoftMotion”设备，右键选择“添加设备”，添加 4 台伺服驱动器，这里选择“INVT_DA200_171”，如下图所示。

4、 依次在设备栏选中“INVT_DA200_171”设备，右键选择“添加 SoftMotion 的 CiA402 轴”，完成 4 台伺服电机的添加，如下图所示。



5、 双击 PLC_PRG，在声明编辑器上输入以下代码。

```
PROGRAM PLC_PRG
VAR
    iStatus: INT;
    MC_Power_0: MC_Power;
    MC_Power_1: MC_Power;
    MC_Power_2: MC_Power;
    MC_Power_3: MC_Power;
    MC_MoveAbsolute_0: MC_MoveAbsolute;
    MC_MoveAbsolute_1: MC_MoveAbsolute;
    MC_MoveAbsolute_2: MC_MoveAbsolute;
    MC_MoveAbsolute_3: MC_MoveAbsolute;
END_VAR
```

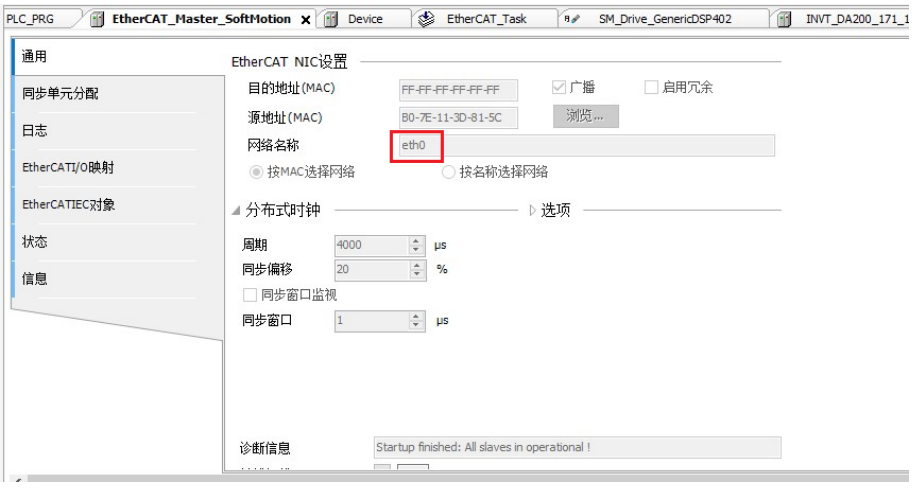
6、 在主体代码编辑器里输入以下代码。

```
CASE iStatus OF
0:
MC_Power_0(Axis:= SM_Drive_GenericDSP402, Enable:= TRUE, bRegulatorOn:= TRUE, bDriveStart:=TRUE , );
MC_Power_1(Axis:= SM_Drive_GenericDSP402_1, Enable:= TRUE, bRegulatorOn:= TRUE, bDriveStart:=TRUE , );
MC_Power_2(Axis:= SM_Drive_GenericDSP402_2, Enable:= TRUE, bRegulatorOn:= TRUE, bDriveStart:=TRUE , );
MC_Power_3(Axis:= SM_Drive_GenericDSP402_3, Enable:= TRUE, bRegulatorOn:= TRUE, bDriveStart:=TRUE , );
IF MC_Power_0.Status AND MC_Power_1.Status AND MC_Power_2.Status AND MC_Power_3.Status  THEN
        iStatus:=iStatus+1;
END_IF
1:
MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402 , Execute:= TRUE, Position:=50 , Velocity:=3 , Acceleration:= 2,
Deceleration:= 100,);
MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_1, Execute:= TRUE, Position:=50 , Velocity:=3 , Acceleration:= 2,
Deceleration:=100,);
MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_2, Execute:= TRUE, Position:=50 , Velocity:=3 , Acceleration:= 2,
Deceleration:=100,);
MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_3, Execute:= TRUE, Position:=50 , Velocity:=3 , Acceleration:= 2,
Deceleration:=100,);
IF MC_MoveAbsolute_0.Done AND MC_MoveAbsolute_1.Done AND MC_MoveAbsolute_2.Done AND
MC_MoveAbsolute_3.Done  THEN
        MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402 , Execute:= FALSE,);
    MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_1 , Execute:= FALSE,);
        MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_2 , Execute:= FALSE,);
```

```
            MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_3 , Execute:= FALSE,);
            iStatus:=iStatus+1;
END_IF
2:
MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402 , Execute:= TRUE, Position:=0 , Velocity:=3, Acceleration:= 2, Deceleration:= 100,);
MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_1, Execute:= TRUE, Position:=0 , Velocity:=3 , Acceleration:= 2, Deceleration:=100,);
MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_2, Execute:= TRUE, Position:=0 , Velocity:=3, Acceleration:= 2, Deceleration:=100,);
MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_3, Execute:= TRUE, Position:=0 , Velocity:=3 , Acceleration:= 2, Deceleration:=100,);
IF MC_MoveAbsolute_0.Done AND MC_MoveAbsolute_1.Done AND MC_MoveAbsolute_2.Done AND MC_MoveAbsolute_3.Done THEN
            MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402 , Execute:= FALSE,);
            MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_1 , Execute:= FALSE,);
            MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_2 , Execute:= FALSE,);
            MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_3 , Execute:= FALSE,);
iStatus:=1;
END_IF
END_CASE
```

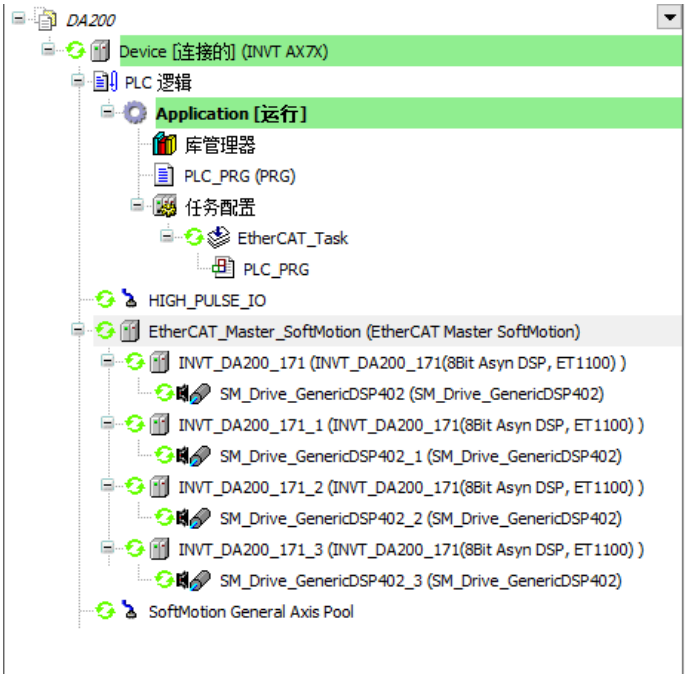
程序主体采用状态机的形式，通过判断 iStatus 的值来决定执行哪部分的代码。程序启动时，iStatus 值为 0，程序初始化 MC_Power 功能块，使能相应的电机轴，如果使能成功，则 iStatus 值为 1，进入下一个状态。iStatus 值为 1 时，执行 MC_MoveAbsolute 功能块，电机以指定的速度转动到指定的位置。若电机正常运动到指定的位置，则 iStatus 值加 1，进入下一个状态。iStatus 值为 2 时，继续执行另一个方向的 MC_MoveAbsolute 功能块，电机继续以该功能块指定的速度转到到指定的位置。若电机正常运动到指定的位置，则 iStatus 值置为 1。如此反复，实现电机的正反转运动。

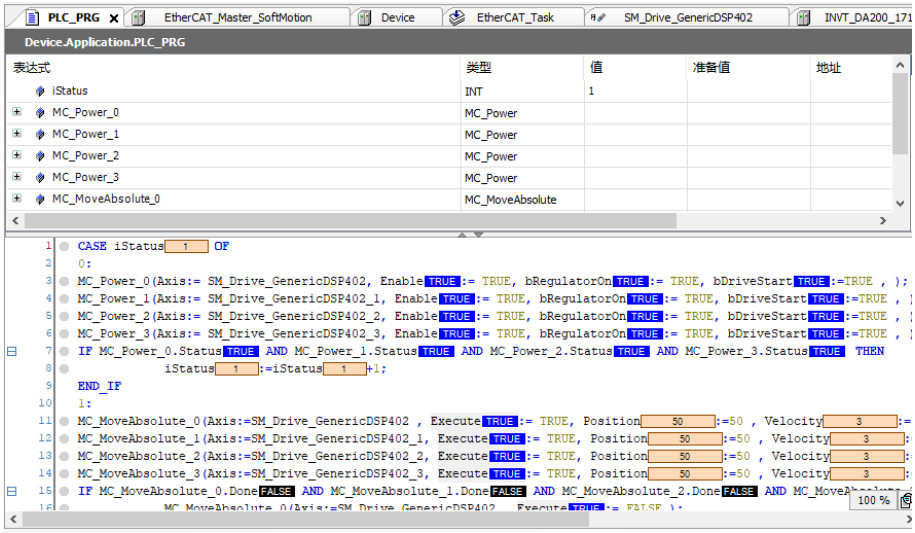
双击设备栏的 EtherCAT 主站设备“EtherCAT Master SoftMotion”，点击“浏览”选择相应的 EtherCAT 通信网口，这里选择“eth0”。根据需要选择分布式时钟，这里选择循环时间为 4000us。如下图所示。



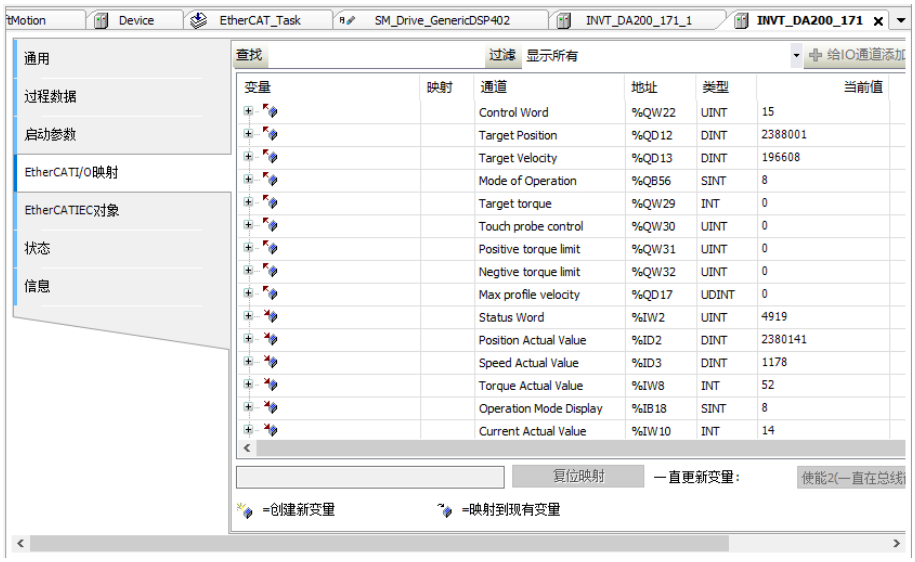
点击工具栏的 按钮编译代码。编译没有错误后，点击工具栏的 按钮登录控制器。伺服正常启动，电机顺利运行，

上位机界面如图所示。

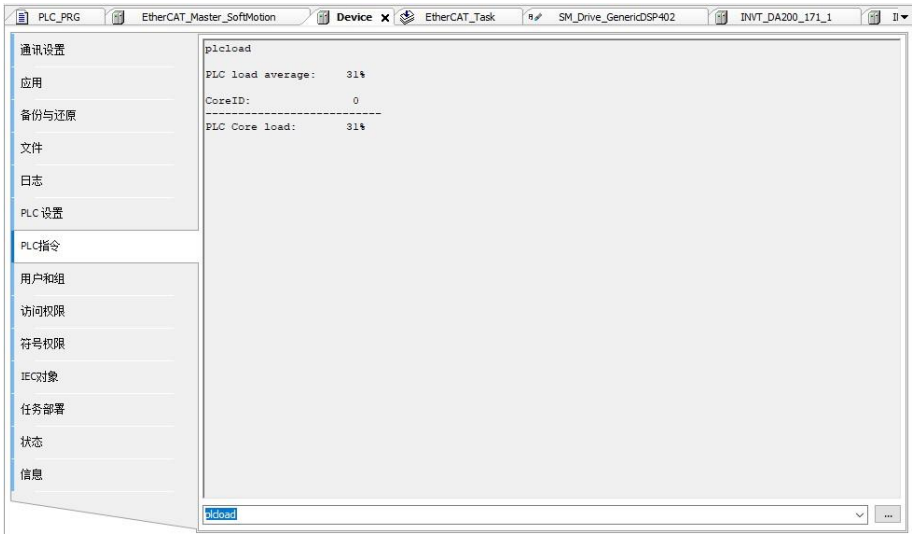




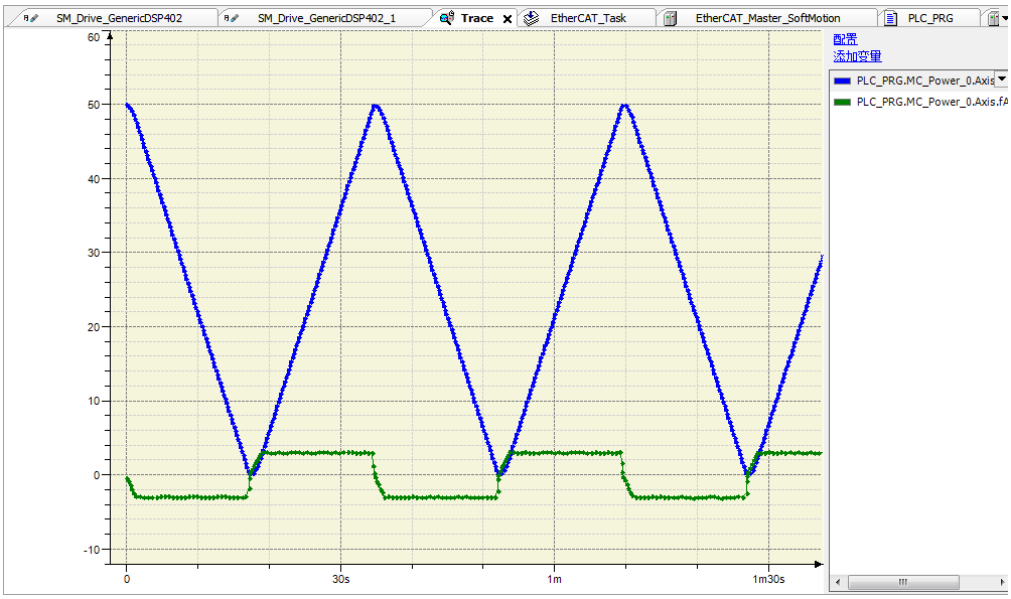
双击设备栏“INVT_DA200_171”，在 I/O 映射界面可查看或设置当前电机的运行参数。如下图所示。



在登录状态，选择 Device 的“控制器指令”，在右下角找到 ... 按钮，选择“plcload”，执行后可以查看当前控制器的 CPU 负载率，如下图所示：



为了更直观地观察电机轴的运行情况，跟踪轴的实际位置，我们新建一个 trace。鼠标右击“Application”，“添加对象”选择“跟踪”，设置任务属性为“EtherCAT_Task”，在 Trace 里添加 PLC_PRG.MC_Power_0.Axis.fActPosition 和 PLC_PRG.MC_Power_0.Axis.fActVelocity 变量，适当调整坐标的显示属性。鼠标右击图形，选择“下载跟踪”，可以跟踪电机的实际位置和实际速度，如图所示。



服务热线：400-700-9997　网址：www.invt.com.cn

深圳市英威腾电气股份有限公司（产地代码：01）
地址:深圳市光明区马田街道松白路英威腾光明科技大厦

苏州英威腾电力电子有限公司（产地代码：06）
地址：苏州高新区科技城昆仑山路1号

工业自动化：■HMI ■PLC ■变频器 ■伺服系统
■电梯智能控制系统 ■轨道交通牵引系统
能 源 电 力：■UPS ■数据中心基础设施 ■光伏逆变器 ■SVG
■新能源汽车动力总成系统 ■新能源汽车充电系统 ■新能源汽车电机

66001-00649

202010 (V1.1)