

AX Series Programmable Controller Software Manual



SHENZHEN INVT ELECTRIC CO., LTD.

No.	Change description	Version	Release date
1	First release	V1.0	September 2020
2	 Modified the variables PLC and ErrorID to setError and getError in Table 4-1. Updated the remarks of inTime and inDate in Table 4-1 and Table 4-2. Updated the variable definition table in A.1.1.1, A.1.2.1, and A.2.1.1. 	V1.1	June 2021
3	 Added a Table 3-2 "Example of bit, byte, word, and double word correspondence of AX series controllers" and updated the data in Table 3-1 and Table 3-3. Updated the section 5.2, changing the "error code" to "fault code". Added the description of PC communication configuration for Windows10 when the hardware is connected with Mini USB cable in section 2.3 	V1.2	November 2021
4	 Added Figure 2-28, Figure 2-29, and updated related descriptions in section 2.6.2. Added section 4.2.1.1 "P-type model port configuration description" and section 4.2.1.2 "N-type model port configuration description". Added Figure 4-7, Figure 4-8, and Figure 4-9, and updated related descriptions in sections 4.4 and 4.5. Deleted section 4.5.3 "Temperature module" and Appendix A "Function module command". Added section 4.7 "Distributed I/O module", section A.3 "Controller and DA200 Series Servo CANopen Configuration Example" and Appendix B "SMC_ERROR description". 	V1.3	August 2022

Preface

Thank you for using the AX series programmable controller (programmable controller for short).

This manual contains the information required to use the AX series programmable controllers. Please read this manual carefully before using the product. Then you can fully understand the functions, performance, and system build-up, which helps to give full play to the advanced performance.

Target audience

Personnel with electrical professional knowledge (such as qualified electrical engineers or personnel with equivalent knowledge)

Applicable product

AX70 programmable controller

- AX71 programmable controller
- AX72 programmable controller
- AX series programmable controller backplane expansion modules
- AX series programmable controller bus expansion modules

Online support

You can also obtain product documentation and technical support from INVT website:

http://www.invt.com

If the product is ultimately used for military affairs or weapon manufacture, abide by the export control regulations in the *Foreign Trade Law of the People's Republic of China* and complete related formalities.

The manual is subject to change without prior notice.

Contents

Pretace	i
Target audience	i
Applicable product	i
Online support	i
Contents	ii
1 Product Introduction	1
1.1 AX series programmable controller	1
1.1.1 Overview	1
1.1.2 Product configuration and module description	1
1.1.3 System application process	2
1.2 Programming platform	2
1.2.1 Invtmatic Studio	2
1.2.2 Software programming interface	2
1.3 PLCopen specification	3
2 Getting Started	4
2.1 Software installation and uninstallation	4
2.1.1 Software obtaining	4
2.1.2 Software installation requirements	4
2.1.3 Preparing	4
2.1.4 Installing the software	4
2.1.5 Uninstalling the software	8
2.2 AX series programmable controller connection	8
2.3 PC communication configuration	8
2.4 Project creation	15
2.4.1 Starting the programming environment	15
2.4.2 Creating new project	16
2.5 Typical steps of project writing	18
2.6 Examples of program writing and debugging	18
2.6.1 Adding devices	18
2.6.2 Writing a function to handle POU	20
2.6.3 Setting motor parameters	22
2.6.4 Writing motor positive and reverse	23
2.6.5 Compiling user program	24
2.6.6 Running monitor program	25
3 Network Configuration	26
3.1 ModbusTCP	26
3.1.1 ModbusTCP_Master	26
3.1.2 ModbusTCP_Slave	26
3.2 ModbusRTU	27
3.2.1 ModbusRTU_Master	27
3.2.2 ModbusRTU_Slave	27
3.3 EtherCAT master node	27
3.4 CANopen	29
3.4.1 CANopen master node configuration	30
3.4.2 Parameter configuration of CANopen master	31
4 Module Configuration	33
4.1 CPU module	33

4.2 High speed I/O module	
4.2.1 Creating high speed I/O module project	
4.2.2 Input port function description	
4.2.3 Output port function description	
4.2.4 High-speed I/O mapping table	
4.2.5 Interrupt instruction	
4.3 Digital input/output module	
4.3.1 Creating a project for digital input/output module	63
4.3.2 Variable definition and use	63
4 4 Analog input/output module	64
4.4.1 Creating a project for analog input/output module	64
4 4 2 Variable definition and use	65
4 5 Temperature module	66 66
4.5.1 Creating a project for temperature module	66
4.5.2 Variable definition and use	66
4.6 Communication module	67
4.6.1 Digital input module	68
4.6.2 Digital nutbut module	68
4.6.3 Analog input module	00 69
4.0.5 Analog niput module	
4.6.5 Tomporature module	
4.0.5 Temperature module	
4.7 Distributed I/O module	
4.7.1 Creating a project for distributed i/O module	
4.6 Fhority Setting priority	
4.6.1 Setting priority	
4.6.2 Configuring sub-device bus cycle options	
5 Device Diagnosis	80
5. I Fault Indicator	
5.1.1 System and bus fault indicator	
5.1.2 High-speed input/output indicator	
5.2 Digital tube fault code	
6 Controller Program Structure and Execution	
6.1 Program structure	
6.3 Program execution	
6.4 Task execution type	
6.5 Task priority	
6.6 Operation of multiple subprograms	
7 EtherCAI Bus Motion Control	
7.1 EtnerCAT operation principle	
7.1.2 Work counter WKC	
7.1.3 Addressing mode	
7.1.4 DISTRIBUTED CIOCKS	
7.1.5 EtherCAI cable redundancy	
7.2 EtherCAI communication mode	
7.2.1 Periodic process data communication	
7.2.2 Non-periodic mailbox data communication	
7.3 EtherCAT state machine	
7.4 EtherCAT servo drive controller application protocol	110
7.4.1 EtherCAT-based CAN application protocol (CoE)	

7.4.2 Servo drive profile according to IEC 61800-7-204 (SERCOS)	115
8 Application Programming	120
8.1 Single axis control	120
8.1.1 Single axis control programming description	120
8.1.2 MC function blocks commonly used for single-axis control	120
8.2 Cam synchronization control	121
8.2.1 Periodic mode of the cam table	122
8.2.2 Input method of cam table	123
8.2.3 Data structure of cam table	123
8.2.4 CAM table reference and switch	124
Appendix A Project Instance	125
A.1 Controller and Goodrive20 Series VFD Configuration Example	125
A.2 Controller and DA200 Series Servo Drive Configuration Example	131
A.3 Controller and DA200 Series Servo CANopen Configuration Example	137
Appendix B SMC_ERROR description	141

1 Product Introduction

1.1 AX series programmable controller

1.1.1 Overview

The AX series programmable controller is a high-performance programmable controller designed with a modular structure to provide users with intelligent automation solutions. It adopts IEC61131-3 programming language system and supports six standard programming languages: IL, LD, FBD, ST, SFC, and CFC. High-level motion control functions such as electronic cams, electronic gears, synchronous control, and positioning can be realized through EtherCAT bus. Supporting 200kHz high-speed I/O, the programmable controller can realize motion control functions such as linear interpolation and circular interpolation.

The programmable controller is rack-mounted. Each rack can embrace 16 functional extension modules, including digital input/output modules, analog input/output modules, temperature modules and communication modules. Remote I/O extension can be carried on via EtherCAT fieldbus.

In addition, programmable controller supports various communication interfaces such as EtherCAT, CANopen, RS485 and Ethernet to meet the diverse application requirements of users.

1.1.2 Product configuration and module description

The AX series programmable controller CPU supports the following modules: power supply module, digital input module, digital output module, analog output module, temperature module and communication module. Take AX70-C-1608P as an example, the diagram of system combination is as follows.



Figure 1-1 System integration

1.1.3 System application process



1.2 Programming platform

1.2.1 Invtmatic Studio

Invtmatic Studio is a programming platform developed by Shenzhen INVT Electric Co., Ltd. It fully supports the IEC61131-3 programming language system and six standard programming languages: IL, LD, FBD, SFC, ST, and CFC.

1.2.2 Software programming interface

The interface of Invtmatic Studio software after creating an application project is shown in Figure 1-2.



Figure 1-2 Invtmatic Studio software application engineering interface

1.3 PLCopen specification

Founded in 1992, PLCopen is a vendor- and product-independent worldwide association. One of the core activities of PLCopen is focused around IEC 61131-3, the only global standard for industrial control programming. A standard programming interface allows people with different backgrounds and skills to create different elements of a program during different stages of the software lifecycle: specification, design, implementation, testing, installation and maintenance. Yet all pieces adhere to a common structure and work together harmoniously. The standard includes the definition of six programming languages: Continuous Function Chart (CFC), Sequential Function Chart (SFC), Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD) and Structured Text (ST). Via decomposition into logical elements, modularization, and modern software techniques, each program is structured, increasing its re-usability. For programmers, the programming technology based on IEC61131-3 can be widely used in the entire industrial control field.

Invtmatic Studio programming platform used in AX series programmable controller fully supports the PLCopen specification and allows users to reference many standard function libraries. The high-level language programming approach makes it easy for controller manufacturers and users to develop their own proprietary function blocks and instruction libraries and to borrow existing similar control programs to form industry-specific "process packages", which can significantly improve user programming efficiency.

2 Getting Started

2.1 Software installation and uninstallation

2.1.1 Software obtaining

INVT AX series programmable controller user programming software contains Invtmatic Studio platform, installation files and related reference materials. You can get them by the following ways:

- 1. Visit INVT website (www.invt.com) and go to **Support** > **Download** > **Software** to download the software installation package for free.
- 2. Obtain software installation CDs from all levels of INVISTA distributors.

2.1.2 Software installation requirements

You can install the software on a computer or desk:

- 3. Installed with Windows 7/ Windows 8/ Windows 10 operation system
- 4. CPU clock speed: 2GHz or higher
- 5. Memory: 2GB or higher
- 6. Available hardware space: 5GB or higher

2.1.3 Preparing

If it is the first time to install Invtmatic Studio, check whether your computer meets the software installation requirements. If yes, you can install it directly.

If you want to install the latest version of Invtmatic Studio, check the version information about the installed software by choosing **Help** > **About**. If it is not the latest version, you can upgrade the software using the online upgrade method.



Figure 2-1 Version information

2.1.4 Installing the software

- 1. Locate the installation file storage path, and double-click **Invtmatic Studio Setup 64 Vx.x.x.exe**. (take V1.0.2 as an example)
- 2. The installation starts. See the following figure.



Figure 2-2 Installation preparation

3. When the dialog box shown in the following figure appears, click **Next**.



Figure 2-3 Installation wizard

4. Then the license agreement dialog box appears. Select I accept the terms in the license agreement, and then click Next.



Figure 2-4 License agreement

5. Set the software installation path, and click **Next**.

😸 Invtmati	c Studio V1.0.2 - InstallShield Wizard	J
Destinati Click Nex	on Folder at to install to this folder, or click Change to install to a different folder.	
Ø	Install Invtmatic Studio V1.0.2 to: C:\Program Files\Invtmatic Studio\ Change	
InstallShield -	< Back Next > Cancel	-

Figure 2-5 Installation path

6. The installation component selection interface appears. Select an installation option. If you have no special requirement, keep the default selection, and click **Next**.

🚽 Invtmatic Stud	dio V1.0.2 - InstallShield Wizard	X
Setup Type Choose the set	up type that best suits your needs.	55
Please select a	setup type.	
Complete	All program features will be installed. (Requires the most disk space.)	
Custom	Choose which program features you want installed and where they will be installed. Recommended for advanced users.	
InstallShield	< Back Next > C	ancel

Figure 2-6 Installation type

7. When the following interface appears, click **Install**.

岃 Invtmatic Studio V1.0.2 - InstallShield Wizard
Ready to Install the Program The wizard is ready to begin installation.
Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield

Figure 2-7 Start installation

8. An installation progress bar appears. Click **Finish** when the installation is completed.

🛃 Invtmati	c Studio V1.0.2 - InstallShield Wizard 📃 🔲 🔀			
Installing The prog	Installing Invtmatic Studio V1.0.2 The program features you selected are being installed.			
1 2	Please wait while the InstallShield Wizard installs Invtmatic Studio V1.0.2. This may take several minutes. Status:			
InstallShield –	< Back Next > Cancel			

Figure 2-8 Installation progress

🛃 Invtmati	c Studio V1.0.2 - InstallShield Wizard			
Installing Invtmatic Studio V1.0.2 The program features you selected are being installed.				
Þ	Please wait while the InstallShield Wizard installs Invtmatic Studio V1.0.2. This may take several minutes.			
	Status:			
InstallShield -				
	< Back Next > Cancel			

Figure 2-9 Installation complete

2.1.5 Uninstalling the software

Uninstall Invtmatic Studio by using the standard software uninstallation method of a Windows system. The procedure is as follows:

Step 1 Shut down Invtmatic Studio running programs, including the backend running program.

Step 2 Enter the control panel, find and right-click Invtmatic Studio, and click Uninstall.

Step 3 Wait until the software is uninstalled.

2.2 AX series programmable controller connection

There are two types of hardware connections between an upper computer and programmable controller:

- 1. Using Mini USB cable
- 2. Using LAN network cable





2.3 PC communication configuration

 If the hardware is connected with a LAN network cable, ensure that the IP address of the PC and the IP address of the controller are in the same network segment. The factory default IP address of the AX series programmable controller is 192.168.1.10, so the IP address of the PC should be set to 192.168.1.xxx. (xxx means any integer value in the range of 1 - 254 except the end address of the controller IP).

Getting Started

AX series programmable controller software manual

Local Area Connection 2 Properties	Internet Protocol Version 4 (TCP/IPv4) Properties
Networking Sharing	General
Connect using:	کرج You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator
Thtel(H) Ethemet Connection (4) 1219-LM	for the appropriate IP settings.
Configure	Obtain an IP address automatically
I his connection uses the following items:	Ose the following IP address:
QoS Packet Scheduler	IP address: 192 . 168 . 1 . 20
Ele and Printer Sharing for Microsoft Networks	Subnet mask: 255 . 255 . 0
Internet Protocol Version 4 (TCP/IPv4)	Default gateway:
Link-Layer Topology Discovery Mapper I/O Driver Link-Layer Topology Discovery Responder	Obtain DNS server address automatically
	O Use the following DNS server addresses:
Install Uninstall Properties	Preferred DNS server:
Description Transmission Control Protocol/Internet Protocol. The default	Alternate DNS server:
wide area network protocol that provides communication across diverse interconnected networks.	Validate settings upon exit
OK Cancel	OK Cancel

Figure 2-11 PC communication configuration for LAN network cable connection

• If the hardware is connected with Mini USB cables, configure the PC as follows.

When the PC runs Windows7:

- ♦ Install USB drive
 - 1) In **Computer Management** window, select **Device Manager**, right click the RNDIS/Ethernet Gadget device and select **Update driver**.

Network adapters Aventail VPN Adapter Aventail VPN Adapter Network adapter Network Intel(R) Dual Band Wire Intel(R) Ethernet Conne VMware Virtual Etherne VMware Virtual Etherne Other devices Other devices	ess-AC 8265 tion (4) 1219-LM Adapter for VMnet1 Adapter for VMnet8	н	
 Ports (COM & LPT) Processors Sensor I/O devices Sound, video and game c System devices Universal Serial Bus contr 	Update Driver Software Disable Uninstall Scan for hardware changes Properties		
Launches the Update Driver Software Wizard for the selected device.			

Figure 2-12 RNDIS/Ethernet Gadget

2) Select Browse my computer for driver software > Let me pick from a list of device drivers on my computer > Network adapter > Microsoft Corporation > Remote NDIS Compatible Device, and then click Next.

	0 Hadata Driver Coffman, DNDIC/Ethanast Codest	×	
	Select Network Adapter Click the Network Adapter that matches your hardware, then click OK. If you have an		
	Manufacturer		
	Microsoft Microsoft Corporation Motorola, Inc. NEC		
This driver is digitally signed. Have Disk			
	Next	ancel	

Figure 2-13 Select driver software

3) After the installation, start the controller and connect it to the PC with a Mini USB cable. The USB driver is displayed in the computer device manager.



Figure 2-14 Install the driver

♦ Configure USB IP address

1) Go to Control Panel > Network and Internet, right click Local Area Connection of RNDIS and select Properties. In the Properties window, select Internet Protocol Version 4 (TCP/IPv4).

Getting Started

• 49 Search Network		Local Area Connection 4 Properties Networking Sharing Connect using:
 i) I b) L b) L c) L <lic) l<="" li=""> <lic) l<="" li=""> <lic) l<="" li=""> <li< td=""><td>Image: Status Diagnose Bridge Connections Create Shortcut Delete Rename Image: Properties</td><td>RNDIS/Ethemet Gadget #2 Configure This connection uses the following items: VMware Bridge Protocol QoS Packet Scheduler QoS Packet Scheduler Internet Protocol Version 4 (TCT VIPv4) Internet Protocol Version 4 (TCT VIPv4) Link-Layer Topology Discovery Responder Install Uninstall Properties Description Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks. OK Cancel</td></li<></lic)></lic)></lic)>	Image: Status Diagnose Bridge Connections Create Shortcut Delete Rename Image: Properties	RNDIS/Ethemet Gadget #2 Configure This connection uses the following items: VMware Bridge Protocol QoS Packet Scheduler QoS Packet Scheduler Internet Protocol Version 4 (TCT VIPv4) Internet Protocol Version 4 (TCT VIPv4) Link-Layer Topology Discovery Responder Install Uninstall Properties Description Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks. OK Cancel

Figure 2-15 Select local area connection of RNDIS

2) Configure the IP address on network segment 192.168.2.xxx, in which xxx is within 1-254 (except 10). Click **OK** to complete the IP address configuration.

Internet Protocol Version 4 (TCP/IPv4)	Properties		
General			
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.			
Obtain an IP address automatical	ly		
Ouse the following IP address:			
IP address:	192 . 168 . 2 . 100		
Subnet mask:	255 . 255 . 255 . 0		
Default gateway:	· · ·		
Obtain DNS server address autor	natically		
Ose the following DNS server add	resses:		
Preferred DNS server:			
Alternate DNS server:	· · ·		
Validate settings upon exit	Advanced		
	OK Cancel		

Figure 2-16 IP address configuration

When the PC runs Windows10:

♦ Install the driver

kindle_rndis.inf_amd64 is the USB driver file.

1) Right-click the file "5-runasadmin_register-CA-cer.cmd" and select Run as administrator.



2) Press any key.

fxact match:
Blement 2:
Serial Number: 3c914b8e900338844fafb20aadb325c8
Issuer: CN=MobileRead-CodeSigning-CA, O=MobileRead Forums, OU=Marco77
NotBefore: 3/20/2016 7:36 PM
NotAfter: 1/1/2040 7:59 AM
Subject: CN=MobileRead-CodeSigning-CA, O=MobileRead Forums, OU=Marco77
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): 58e285d47509e810dccfa865a8f6d99b8a297fa5
Certificate "MobileRead-CodeSigning-CA" already in store.
CertUtil: -addstore command completed successfully.
,
:\Users\Administrator\Desktop\rndis.inf_amd64\rndis.inf_amd64\pause
Press any key to continue

3) Connect the computer and the PLC with a USB cable, open **Device Manager**, and right-click the USB serial device under the **Ports** node.



- 4) Click Browse my computer for drivers and select the driver folder.
 - ← 🚦 Update Drivers USB 串行设备 (COM8)

How do you want to search for drivers?



5) Wait for the installation process completed.

Update Drivers - Kindle USB RNDIS Device (USBNetwork enabled)

Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:



Kindle USB RNDIS Device (USBNetwork enabled)

The USB RNDIS item has been added to the Network Adapters node in Device Manager.



♦ Configure USB network port

1) Right-click the Network menu and select Properties.

_	Disconnect network and en
> 👝 Local Disk (E:)	Delete
> 👝 Local Disk (F:)	Properties
> 💣 Network	

2) Click Change adapter settings.

← → × ↑ ½ → Control P	anel > All Control Panel Items > Network	and Sharing Center
Control Panel Home	View your basic network infor	mation and set u
Change adapter settings	View your active networks	
Change advanced sharing settings	invt.cn Private network	Ai Ci
Media streaming options		

3) Right-click the Unidentified network with "USB RNDIS" in its name, and select Properties.

以太网 2 未识别的网络		_
Kindle USB RNDIS Device (USBNet	. ᠹ Disable	1
	Status	
	Diagnose	
	Bridge Connections	
\mathbf{X}	Create Shortcut	
\mathbf{X}	💎 Delete	
	🗣 Rename	
	Properties	

4) Select Internet Protocol Version 4 (TCP/IPv4) and click Configure....

Networking	Sharing	
Connect us	ing:	
Ţ Kind	e USB RNDIS Device (USBNetwork enabled)	
	Configur	e
This conne	ction uses the following items:	
🗹 🥮 Qa	S数据包计划程序	~
🗹 🔔 Int	emet 协议版本 4 (TCP/IPv4)	
🗆 🔔 Mi	crosott 网络适配器多路传送器协议	
🛛 🗹 🔔 Mi	crosoft LLDP 协议驱动程序	
🛛 🗹 🔔 İnt	ernet 协议版本 6 (TCP/IPv6)	
🗹 💶 链	路层拓扑发现响应程序	
🗹 💶 链	路层拓扑发现映射器 I/O 驱动程序	~
<		>

5) Set the IP address manually. The IP address must be in the network segment 192.168.2.xxx, in which xxx is within 1-254 (except 10).

Internet 协议版本 4 (TCP/IPv4) Properties			
General			
You can get IP settings assigned autor this capability. Otherwise, you need to for the appropriate IP settings.	matically if your network supports ask your network administrator		
Obtain an IP address automatical	lly		
Use the following IP address:			
IP address:	192.168.2.10		
Subnet mask:	255.255.255.0		
Default gateway:	192 . 168 . 2 . 1		
Obtain DNS server address autor	natically		
• Use the following DNS server add	lresses:		
Preferred DNS server:			
Alternate DNS server:			

2.4 Project creation

2.4.1 Starting the programming environment

1. Take Invtmatic Studio V1.0.2 as an example. Double-click the software icon of Invtmatic Studio. The programming environment is as follows:



Figure 2-17 Invtmatic Studio homepage

2. In the tool bar, select **Tool** > **Device repository** to add a device profile.



Figure 2-18 Add device profile

3. In the Device repository pop-up window, click Install.

cation	System Repository			~	Edit Locations
	(C:\ProgramData\Invtmatic	tudio\Devices)		
stalled d	evice descriptions				
tring for	a fulltext search	Vendor:	<all vendors=""></all>	~	Install
Name	Vendor	Version	Description		Uninstall
* 100 M + 100 Fi + 200 Fi + 200 Fi H + 100 Fi P + 200 Fi P S	iscellaneous ieldbuses MI devices LCs oftMotion drives				
					Detelle

Figure 2-19 Install device

4. From the **Install device profile** window, select the device profile to be installed from a local folder and then click **Open**.

Install Device Descrip	ption			×
← → × ↑ 📙 «	AX70 > AX70_APP_V2.0.5.1 > AX7X	ٽ ~	搜索"AX7X"	Q
组织 ▼ 新建文件夹				
△ WPS网盘	^ 名称	修改日期	类型	大小
🔜 此电脑	Shenzen INVT-AX7X-CPU_1.2.0.4.dev	2020/4/2 16:19	XML 文档	106
■ 视频				
◆ N载 ♪ 音乐				
🔜 桌面 🏪 系统 (C:)				
程序安装 (D:)	v <			>
<u></u> χί	牛名(N): Shenzen INVT-AX7X-CPU_1.2.0.4.devdes	sc.xml ~	Sercos XML device 打开(O)	descripti ~ 取消

Figure 2-20 Install device profile

Note: All device profiles provided by INVT can be added by following the steps above.

2.4.2 Creating new project

 Click the project creation icon at the upper left corner or choose File > New Project, or directly click New Project in the window to quickly create a project. Select the project category, template, save path and file name, as shown in the following figure.

管 New Pro	oject				×
Categories	ojects	Templates	HMI project	Standard project	Standard project w
A project of Name Location	Untitled2 D:\Invtmatic Studio\Projec	plication, and an o	empty implemen	tation for PLC_	PRG
				OK	Cancel

Figure 2-21 New project

2. Click **OK**. On the standard project setting interface that appears, select the device type and programming language. See the following figure.

Standard Pr	roject	
61	You are abou objects within	t to create a new standard project. This wizard will create the following n this project:
	- A program F - A cyclic task - A reference	LC_PRG in the language specified below : which calls PLC_PRG to the newest version of the Standard library currently installed.
	Device	INVT AX7X (Shenzhen INVT Electric Co., Ltd.)
	PLC_PRG in	Structured Text (ST)
		Continuous Function Chart (CFC) Continuous Function Chart (CFC) - page-oriented Function Block Diagram (FBD) Ladder Logic Diagram (LD) Sequential Function Chart (SFC) Structured Text (ST)

Figure 2-22 Standard project setting page

3. On the configuration and programming interface, double-click **PLC_PRG(PRG)** to write programs. See the following figure.

e Edit View Project Build Onlin	Debug Tools Window Help	
🛎 🖬 🚳 🗠 🗠 👗 🖻 🛝 🗙 🚧 🎕	🛓 🏠 則 🧐 🦄 職 100 + 了 世目 Application [Device: PLC Logic] 🔹 🧐 🕠	■ ≪ 印 印 印 印 谷 々 第 干 や
ces 🗸 🗘	PLC_PRG X	
GI Chatted I	1 PROGRAM PLC_PRG	
BU Device (INVI ACX)	3 END_VAR	
Andratian		
Application		
Task Configuration		100 % [
B SS MainTask	1	
(B) PLC_PRG		
A HIGH PULSE TO		
SoftMotion General Axis Pool		
		100
	< []	
	Messages - Total 0 error(s), 0 warning(s), 0 message(s)	*
	- O error(s) 🙂 0 warning(s) 🖲 0	message(s) 🗙 💥
	Description	Project Object Position

Figure 2-23 Invtmatic Studio configuration and programming page

2.5 Typical steps of project writing

From the above example, writing a user program with MC motion control functions generally requires the following steps.

Step 1 Application system hardware configuration

Configure network according to the main controller, expansion module, network type, servo slave node and other hardware used.

Step 2 User program writing

According to the control function to be implemented, write motion control with one POU (such as POU1), and write common logic control with a POU (such as POU2).

Step 3 Servo driver parameter configuration

Configure the objects of SDO and PDO according to the servo name in the hardware configuration and the operation mode of the servo. Ensure that the communication objects required between the MC function block of the user program and the servo are filled in the configuration table.

Step 4 Servo motor parameter configuration

Correctly fill in the resolution of the servo motor encoder, the transmission ratio of the mechanical structure, the characteristics of the axis movement range and other parameters, so that the displacement command of the control object corresponds accurately to the actual displacement.

Step 5 Task arrangement

Based on the real-time requirements of control, execute the motion control function POU1 in the EtherCAT task and set the cycle to 4ms, the priority to 0; execute the common logic control POU2 in common tasks and set the cycle to 20ms, the priority to 16.

Step 6 Online debugging

Connect the AX series programmable controller to PC via LAN network correctly. Power on the programmable controller, download and debug the user program, and eliminate user program bugs (if possible, you can connect the servo drive system to the programmable controller and then debug. If the servo system is not available, you can set the servo as a virtual axis; if the programmable controller is not available, you can simulate and debug the user program on the PC to eliminate possible errors in the user program).

2.6 Examples of program writing and debugging

Here is an example of a basic servo control program to give you a first glimpse of the programming process before you go through the principle of the programming system and the method of compiling the motion control program.

Write a simple program that allows the AX series CPU programmable controller to implement the following functions:

The servo motor repeats rotating forward 50 revolutions, and then reversing 50 revolutions.

The programming method and steps of the routine are as follows:

Step 1 Add the corresponding equipment: EtherCAT master node, servo drive, motor shaft.

Step 2 Handle the motion control of the servo in the high real-time EtherCAT task cycle.

Step 3 Set relevant parameters.

Step 4 Write program.

2.6.1 Adding devices

1. Add an EtherCAT SoftMotion master node and an EtherCAT network bus.



Figure 2-24 Add EtherCAT master node

2. Add a servo device.

E LUIL VIEW FI	roject Ruild Opling Debug	k Window Help	
	X III III X IIII 25 M 25 III I	1 2 [Charles of Charles and Charles C	े न । स्ट
	So was seen and the linear cost and complete	w w w w w w w w w w w w w w w w w w w	11 1 4
es	- a x		
Untibled2			
Device (INVT AX7)	0		
🗏 🔛 PLC Logic			
Application	ion		
- 🛍 Library	y Manager		
	RG (PRG)		
= 🔯 Task (👗 Cut		
	Па Сору		
Æ	Paste		
- & HIGH PULSE	× Delete		
EtherCAT_Ma	Refactoring		
SoftMotion Ge	T Store		
	Properties		
	Add Object		
	add Folder		
	Add Device		
	Insert Device		
1	Scan For Devices		
ji Untibled2	Disable Device		
Project Settings	Update Device		
	🕤 Edit Object		
	Edit Object with	0 error(s), 0 warnino(s), 1 message(s)	v 3
	Edit IO mapping	- O 0 error(s) O warning(s) O 1 message(s) X X	
	Edit IO mapping Import mappings from CSV	O error(s) O warning(s) I message(s) X	

AX series programmable controller software manual

5 Untitled2.project	• - Invtmatic Studio —	
File Edit View	Project Build Online Debug Tools Window Help	T
🖻 📽 🖬 🐻 🗠	Add Device	× - 1 =
Devices	Name [INT_DA200_252 Action @ Append device () Elusert device () Elup device	
PLC Logic	String for a fulltext search Vendor <all vendors=""></all>	7
= ○ Appl - ∰ L - ∰ - = = = = = = = = = = = = =	Name Vendor A	
- & SoftMotio	Construction of the second secon	
C POUs Chttled2 Project Settin	The intervent of the types of the transmission of the type of type of the type of	
	Append selected device as last child of EtherCAT_Haster_SoftHotion (You can select another target node in the navigator while this window is open.)	- a x
	Add Device Close	0

Figure 2-25 Add EtherCAT slave node

3. Add a servo axis.



Figure 2-26 Add a servo axis

2.6.2 Writing a function to handle POU

In Invtmatic Studio programming environment, there is an EtherCAT_Task task and a MainTask task for the default task configuration. The MainTask task contains a POU named PLC_PRG which is created at the same time as the new project is created. Create a POU for servo control under the EtherCAT_Task task.

AX series programmable controller software manual

🔄 Untitled1.project* - Invtmatic Studio	-		×
File Edit View Project Build Online	Debug Tools Window Help		₹
🎦 📽 📓 📾 🗠 a 🕹 🖻 📾 🗙 🖓 🎲 i	🎽 🏠 📕 🐄 🎢 🆓 🎼 🎬 - 😚 🖽 Application [Device: PLC Logic] - 🧐 🧐 🕠 💼 帐 [[크 오	1 da +a Q	5
Devices - 7 ×	PLC PRG X		•
🖃 🎒 Untitled1	1 PROGRAM PLC_PRG		1
B- M Device (INVT AX7X)	2 VAR		
PLC Logic	S BND VAR		
🖹 🔘 Application			
Library Manager			
PLC_PRG (PRG)		100 %	2
Task Configuration			
EtherCAT_Task			
- S Maniask			
HIGH PLUSE TO			
Fill EtherCAT Master SoftMotion (EtherCAT Mas			
fil INVT DA200 262 (DA200-N EtherCAT(C			
SoftMotion General Axis Pool			
	4	100 9	%
Messages - Total 0 error(s), 0 warning(s), 1 message	a)		
in the state of th	Last build: 😫 0 🕐 0 Precomple 🖌 🦛 Project user: (nobody) INS I n 1 Col 1	Ch 1	Ø .
			¥ 1.0

Figure 2-27 PLC_PRG programming page

1. Right-click the **Application** in the device tree, select **Add Object** > **POU** to add a POU for EtherCAT servo control.

🖸 Un	titled1.project* - Invtmat	tic Stu	idio				-		×
File	Edit View Project	Bi	uild Online Debug Tools W	ndow	v Help				
10 a		5 (6)	× (AA 15 AA 14 11 11 11 11 11 11	(B)	1 Application (Device: PLC	Logici -	or of the water water	d-1 +11	8
		n ets		1	Appression (bencer re-	cogicj	-9 -9 -9 -9 -9 -9 -9 -9 -9		V 1
Devices									
B-Cal /	Untitled 1		1 PROGRAM	PLC P	RG				1
8-1	Device (INVT AX7X)		2 VAR						
	- I PLC Logic		3 END_VAR						
	Application	ч	A.	7					
	👘 Library Mana	ð	Cut						
	PLC_PRG (PR	40	Сору						
	🖻 🧱 Task Configu		Paste					100.96	
	StherCA	\cap	Delete	F	A 7				<u></u>
	= Se Manias		Refactoring +						
		Π,	Store		Alarm Configuration	1			
	E-fil EtherCAT Master So	G	Properties	0	Application	1			
	INVT_DA200_26	ħ	Add Object	2	Axis Group				
	SoftMotion General A	6	Add Folder	8	Cam table	1			
		ñ	Edit Object	8	CNC program				
			Edit Object with		CNC settings				
		CS.	Login	5	Data Sources Manager				
		~?	Delete exclination from device	*	DUT				
		_	Delete application from device	. 1	External File				
		-19	Visual Element Repository	2	Global Variable List				
		8	License Manager		Image Pool	1			
			Program manual	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	Interface				
			Hardware manual	2	Network Variable List (Receiver)				
			Software manual	7	Persistent Variables			100)%
Mar	reages - Total 0 error(s) 0 :		INVT Website	æ	POU				
E Mes	ssages - rotar 0 error(s), 0 (Advice feedback	-	POU for implicit checks	08	Draiast usan (aabadu)		0
						103	Projeccuser: (hobody)		V

Figure 2-28 Add a POU

2. Double-click EtherCAT_Task in the device tree and click Add Call in the configuration interface to select EtherCAT POU.

File	Edit View Project Build	i Online Debu < 444 🏠 🐴 🍇	ug Tools W 베 왜 해 작	findow 📾 🖞	Help I TÎÎÊ	oplication (Devi	ce: PLC Logi	c] • 😋 🕫	> = %	Ç≡ 9∃	± <u>1</u> +3 3	3
Devices = 0 L = 6 - 6 1	Antikel // Denke (INIT AX2)) The Clock)	A X A Confi Confi Price T Confi Price T Confi Price Confi Price Confi Price Confi Price Confi Price Confi Price Confi Confi Price Confi Confi	PC_PRG [guration porty (0.1]): [porty (0.1]): [porty (0.1]): [porty (0.1]): [porty (0.1]]: [po	1 move Cal	Interval (e.g. t#	EtherCAT_Task	p & Move	Down 141	Open POU		ms	

Figure 2-29 Call a POU for EtherCAT task

2.6.3 Setting motor parameters

For precise control of the movement position, the programmable controller must accurately calculate the position of the servo motor. Based on the operating characteristics and stroke characteristics of the application system, select the **Axis type and limit**. Therefore, the programmable controller can calculate the feedback information of the motor encoder to obtain the accurate position, and then avoid errors caused by the accumulated overflow of the encoder pulse number.



Figure 2-30 Motor parameter settings

For the reciprocating mechanism of the lead screw type, **Finite** is preferred as the lead screw stroke is limited and we should know its absolute position within the stroke range.

For a single-direction shaft, **Modulo** is preferred as the linear mode may cause position counting overflow, resulting in position calculation errors.

The encoder parameters of the motor (such as resolution) and the mechanical deceleration ratio of the application system may be different. They need to be set based on the actual situation during programming, as shown in the following figure.



Figure 2-31 Motor encoder parameter settings

The DA200 servo matching motor has two typical resolutions. The resolution of normal incremental encoders is 20bit, that is, 1048576 pulses per revolution; and the resolution of absolute encoders is 23bit, i.e. 8388608 pulses per revolution. In

AX series programmable controller software manual

actual operation, the programmable controller sends the required number of pulses to the servo drive by EtherCAT communication to control the servo operation. Therefore, the encoder resolution needs to be accurately set according to the actual situation, as shown in the figure above. Take a 20bit encoder without a reducer as an example. When the servo is commanded to run 1 unit, the servo will select 1 revolution (axis moves 360°). If the field unit in application (circled in red in the figure above) is set to 360, the servo will select 1/360 circle (axis moves 1°) when the servo is commanded to run 1 unit, and so on. After setting the corresponding parameters (commonly known as electronic gear ratio) according to the actual mechanical structure, you can input the distance command according to the physical unit of the application system movement distance, making the control parameters intuitive and easy to understand.

Please note that only integer numbers can be entered in the fields circled in red in the figure above. Because the ratio of the parameters in the corresponding rows on the left and right sides is effective, you can enter appropriate integer values in the corresponding rows on the left and right sides. For example, to enable the drive lead with screw rod 6.8mm (that is, the screw rod rotates 1 circle and the screw slide block moves 6.8mm) to move after the servo motor passes through a mechanical deceleration mechanism with a ratio of 4:1, please set as shown in the following figure.

Scaling Invert direction		
16#20000	increments <=> motor turns	1
4	motor turns <=> gear output turns	1
10	gear output turns <=> units in application	68

Figure 2-32 Setting example

The dimension of the parameters circled in red can be used as the dimension of the distance in the MC control command later. The settings of the servo driver and motor described above must be set and verified in the corresponding items of the servo axis, otherwise the motor will not operate as expected.

2.6.4 Writing motor positive and reverse

For the motion control of the servo axis, the default synchronization period is 4ms. Users can choose according to the actual need, as shown in the following figure.

S Untitled2.project* - Invtmatic Studio			- 🗆 ×
File Edit View Project Build Online I	Debug Tools Window Help		τ
🎦 😅 🖬 🕼 🗠 여 🐇 🖻 🛝 🗙 🖓	🍇 📗 別 別 別 間 🛅 🖆	🛗 Application [Device: PLC Logic] 🔹 😋 🕠 🕟	■ ♥[대 맨 앱 앱 왕 [ㅎ [悪] 문 [장]
Devices 👻 🕂 🗙	PLC_PRG M SM_Drive_	GenericDSP402	•
Untitled2 Device (INVT AX7X)	General	Autoconfig Master/Slaves	EtherCAT
	Sync Unit Assignment	EtherCAT NIC Setting	
- Diplication	Log	Destination address (MAC) FF-FF-FF-FF-FF	Broadcast Enable redundancy
PLC_PRG (PRG) Section Section	EtherCAT I/O Mapping	Source address (MAC) 00-00-00-00-00 I Network Name	Browse
EtherCAT_Task	EtherCAT IEC Objects	Select network by MAC Select network by m	ame
	Status	Distributed Clock Distributed Clock	ions
EtherCAT_Master_SoftMotion (EtherCAT Master		Sync offset 20 . %	
SM_Drive_GenericDSP402 (SM_Drive_		Sync window monitoring	
		Sync window 1 📮 µs	
< >			
POUs - 4 X			
Untitled2 Project Settings			
	<		>
	Messages - Total 0 error(s), 0 warning(s),	1 message(s)	→ ∓ ×
	Devices	- O error(s) 😗 0 warning(s) 🜒	1 message(s) 🗙 💥
	Description	Project Object	Position
		Last build: 🔕 0 😗 0 🛛 Precompile 🗸	😭 Project user: (nobody) 🔇 📑

Figure 2-33 Servo axis motion control cycle setting

The program in the above figure is written in ST language. The relevant code is as follows:



Figure 2-34 ST codes

2.6.5 Compiling user program

If there is a writing error, the error type and reason will be listed in Figure 2-30. Double-click the error description, and the cursor will jump to the corresponding program editing window to facilitate revision. After the revision, compile again until all compilation problems are eliminated.

Substituted2.project* - Invtmatic Studio File Edit View Project Build Online Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute Substitute <th>Debug Tools Window Help 19 1일 - 11 개 개 개 대 대급 1월~ 1일</th> <th>🎽 Application [Device: PLC Logic] 👻 🧐</th> <th>- □ × ₹) = ¥[ा ा ा ं 1 % % % %</th>	Debug Tools Window Help 19 1일 - 11 개 개 개 대 대급 1월~ 1일	🎽 Application [Device: PLC Logic] 👻 🧐	- □ × ₹) = ¥[ा ा ा ं 1 % % % %
Devices • 2 2 2 • Device (NOT AX70) • Device (NOT	PC_PRG * SM_Drive General Sync Unit Assignment Log EtherCAT I/O Mapping EtherCAT IEC Objects Status Information	GenerLOSP402	tion x CEtherCAT.
POUS • 3 X · ③ UnsteidZ · ③ Project Settings	< Messages - Total 0 error(s), 0 warning(s) Devices Description), 1 message(s) Project Ob	s) 0 1 message(s) X X performance Position

Figure 2-35 Program compilation

Click **Device** > **Communication Settings** > **Scan Network** to select the device and click **Wink**, then the SF, BF, CAN and ERR indicators of the connected device will wink three times. After the device is confirmed, download the user program into the AX series CPU module.

EtherCAT.project* - Invtmatic Studio			– 🗆 X
<u>File Edit View Project Build Online Debug</u>	<u>T</u> ools <u>W</u> indow <u>H</u> elp		τ
🖹 📽 📓 I 🖉 I 이 이 강 🖻 🛍 🗙 I 🖊 책 🐴 🌿 I 📗	। ११ भे भे 🔓 🛅 - 🗗 🖽	Application [Device: PLC Logic] 🝷 🧐 🔅 🔋 📲 🔏 🕻 🕮 🕾	·□ = 栗 ☆ 祭 I = E ⁵
B-CAT			
=-1 Device (INVT AX7X)	Communication Settings	Scan Network Gateway + Device +	
PLC Logic	Applications	Select Device	×
Application Application	Backup and Restore	Select the network path to the controller:	
PLC_PRG (PRG)	The second se	Gateway-1	节点名: AX70-C- Scan Network
🖹 🙀 Task Configuration	riies		1608P_V208 Wink
B S MainTask	Log	Connect the scanned device	Ce 节点地址:
E PLC_PRG	PLC Settings	through the network cable	e
- & HIGH_PULSE_IO	PLC Shell		快程动: UDP
Enerchi Mastel Solubitori (Enerchi Mastel Solubitori INVT_DA200_262 (DA200-N EtherCAT(CoE) Drive)	Users and Groups		B 16 m
SM_Drive_GenericDSP402 (SM_Drive_GenericDSP	SP4		1631 0003
SoftMotion General Axis Pool	Access Rights		目标版本:
	Symbol Rights		3.5.15.0
	IEC Objects		目标供应商: Shenzhen INVT
	Task Deployment		Electric Co.,Ltd 🗸
	Chature		
			<u>OK</u> <u>Cancel</u>
٢	> Information		v
2 Devices POUs	<		>
Messages - Total 0 error(s), 0 warning(s), 1 message(s)		Last build: O 0 D Press	mole (CE Desistuases (schodu))
🗎 📽 📕 🕌 (이 이 👗 ங 🛍 🗙) 👬 4월 4월 48	실 비 왜 왜 젖 톱 ‱~	G 🛗 Application [Device: PLC Logic] 🛛 🥵 🦉 🔸	■ ♥ 〔= ?= 4= 3 + 悪 = ∛
Devices 🗸 🕈 🗙 🖊	🖹 PLC_PRG 🦯 🗑 EtherCAT_	Master_SoftMotion 🗙 📆 Device 🗙	-
Unitiled I	Seperal	Automolie Master/Clause	
Device (INVT AX7X)		Autocoming master/slaves	Ether CAT
Application	Sync Unit Assignment	EtherCAT NIC Setting	
Library Manager	.og	Destination address (MAC) FF-FF-FF-FF-FF	Broadcast Enable redundancy
EtherCAT_pou (PRG)	TherCAT I/O Manping	Source address (MAC) 00-00-00-00-00 Br	rowse
PLC_PRG (PRG)		Network Name eth1	
EtherCAT_Task	therCAT IEC Objects	Select network by MAC Select network by na	me
🗏 🏶 MainTask 🛛 S	Status	Distributed Clock Option	ons
PLC_PRG	information	Ovcle time 4000	
EtherCAT Master SoftMotion (EtherCAT Mas		Sync offset 20 \$ %	
- 🕤 INVT_DA200_262 (DA200-N EtherCAT(C		Sync window monitoring	
SM_Drive_GenericDSP402 (SM_Driv		Sync window 1 🌲 µs	
SoftMotion General Axis Pool			
< > `			,
Massages - Total () error(s) () warning(s) 1 mossage(s)			
Messages - Total 0 error(s), 0 warning(s), 1 message(s)			

Figure 2-36 User program download

2.6.6 Running monitor program

After logging in to the device through the button marked in a red square in Figure 2-36, the program is running if you can observe the actual operation of the servo or check the position value of the servo axis of the host computer. At this point, the required servo jogging and the 2-cycle running triggering functions has been implemented, which shows the programming process is complete.

3 Network Configuration

The network configuration of AX series programmable controllers mainly includes: ModbusTCP, ModbusRTU, EtherCAT, and CANopen.

3.1 ModbusTCP

3.1.1 ModbusTCP_Master

The number of variables that ModbusTCP can access is defined as follows:

- Read coil (0x01), number of coils 1–2000 (0x7D0)
- Read discrete coils (0x02), number of coils 1–2000 (0x7D0)
- Read holding register (0x03), number of registers 1–125 (0x7D)
- Read input register (0x04), number of registers 1–125 (0x7D)
- Write a single coil (0x05)
- Write a single register (0x06)
- Write multiple coils (0x0F), number of coils 1–1968 (0x7B0)
- Write multiple register (0x10), number of register 1–120 (0x78)

ModbusTCP_Master is an important component of the ModbusTCP_Master function module. Before using the master node, the corresponding library files must be added as follows:

- Create an application project for the ModbusTCP_Master.
- Add the library file "CmpModbusTCP_Master_x.x.x.x.library" required by this module.

3.1.2 ModbusTCP_Slave

- Create an application project for the ModbusTCP_Slave.
- Add the library file "ModbusTCP_Slave_x.x.x.x.library" required by this module.

The ModbusTCP_Slave defines the storage area that can be accessed from outside. The detailed area is as follows:

Table 3-1 ModbusTCP_Slave function codes

Function code of TCP master node	Address name	Range	Offset
01	%QX	0.0-511.7	N/A
05	%QX	0.0-511.7	N/A
02	%IX	0.0-511.7	N/A
04	%IW	0-511	N/A
03/06	%MW	0-8192	5000
03/06	%QW	0-511	N/A
01	%MX	0.0-7565.7	5000
05	%MX	0.0-7565.7	5000

Table 3-2 Example of bit, byte, word, and double word correspondence of AX series controllers

%_X	195.7 – 195.0	194.7 – 194.0	193.7 – 193.0	192.7 – 192.0		
%_B	195 (8 most significant bits)	194 (8 leaset significant bits)	193 (8 most significant bits)	192 (8 leaset significant bits)		
%_W	97 (16 most sig	gnificant bits)	96 (16 leaset s	significant bits)		
%_D	48					

3.2 ModbusRTU

AX series programmable controllers support two Modbus serial communications, COM1_RS485 and COM2_RS485, both of which support the standard ModbusRTU protocol, and can be independently configured as a master or slave, supporting 2400, 4800, 9600, 19200, 38400, 57600, 115200, etc. 7 baud rates.

The number of variables that ModbusRTU can access is defined as follows:

- Read coil (0x01), number of coils 1–2000
- Read discrete coils (0x02), number of coils 1–2000 (0x7D0)
- Read holding register (0x03), number of registers 1–125 (0x7D)
- Read input register (0x04), number of registers 1–125 (0x7D)
- Write a single coil (0x05)
- Write a single register (0x06)
- Write multiple coils (0x0F), number of coils 1–1968 (0x7B0)
- Write multiple register (0x10), number of register 1–120 (0x78)

3.2.1 ModbusRTU_Master

Create an application project for the ModbusRTU_Master. There are two serial ports in AX series programmable controllers. To add ModbusRTU_Master module, the corresponding library files "ModbusRTU_Master1_x.x.x.k.library" and "ModbusRTU_Master 2_x.x.x.k.library" are needed (ModbusRTU_Master1_x.x.x.k.library for the hardware COM1_RS485 port and ModbusRTU_Master2_x.x.x.k.library for the hardware COM2_RS485 port).

3.2.2 ModbusRTU_Slave

Create an application project for the ModbusRTU_Slave. There are two serial ports in AX series programmable controllers. To add ModbusRTU_Slave module, the corresponding library files "ModbusRTU_Slave1_x.x.x.k.library" and "ModbusRTU_Slave2_x.x.x.k.library" are needed (ModbusRTU_Slave1_x.x.x.k.library for the hardware COM1_RS485 port and ModbusRTU_Slave2_x.x.x.k.library for the hardware COM2_RS485 port).

The ModbusRTU_Slave defines the storage area that can be accessed from outside. The detailed area is as follows:

Function code of RTU master node	Address name	Range	Offset
01	%QX	0.0-511.7	N/A
05	%QX	0.0-511.7	N/A
02	%IX	0.0-511.7	N/A
04	%IW	0-511	N/A
03/06	%MW	0-8192	5000
03/06	%QW	0-511	N/A
01	%MX	0.0-7565.7	5000
05	%MX	0.0-7565.7	5000

Table 3-3 ModbusRTU_Slave function code

3.3 EtherCAT master node

For the parameter configuration of the EtherCAT master node, please refer to the relevant instruction in Invtmatic Studio help documents. Here is an example of the connection between an EtherCAT master and a DA200 servo drive slave for reference.

1. Creating the DA200 servo application project

Add the library file "INVT_DA200_xxx.devdesc.xml" required for this module. Take INVT_DA200_262 as an example.

Note:

- The highest priority 0 is recommended for the creation of EtherCAT Master SoftMotion projects.
- It is recommended that the synchronization period and the task period be set consistently at 4ms or more.
- Create EtherCAT Master SoftMotion through a separate task. Separate the EtherCAT Master SoftMotion tasks from I/O, analog input/output, Modbus communication and other tasks.

2. Select the motion controller device profile in the device tree, right-click it and add the EtherCAT Master SoftMotion as shown in the following figure.

Add Device							
me EtherCAT_Master_SoftMotion							
Action							
Append device O Insert device O Plug	device O	Jpdate device					
tring for a fulltext search	Vendor	<all vendors=""></all>			~		
Name	Vende	or	Version	Description	^		
🕴 🄟 Miscellaneous							
Fieldbuses							
E CAN CANbus							
EtherCAT							
B- a Master							
til EtherCAT Master	35 - Sm	art Software Solutions GmbH	3.5.15.0	EtherCAT Master			
EtherCAT Master SoftMotion	3S - Sm	art Software Solutions GmbH	3.5.15.0	EtherCAT Master SoftMotion			
t - BB Ethernet Adapter							
Group by category Display all versions	(for experts (only) 🗌 Display outdated ve	ersions				
Name: EtherCAT Master SoftMotion				^			
Vendor: 3S - Smart Software Solutions G	SmbH						
Categories: Master							
Version: 3.5.15.0							
Order Number:				~			
pend selected device as last child of							
evice							
You can select another target node in th	e navigator v	while this window is open.)					
			-	Add Davies	a 1		

Figure 3-1 Add the EtherCAT motion control master

3. Select EtherCAT_Master_SoftMotion in the device tree, right-click it and add INVT DA200 servo drive as shown in the following figure.

Add Device				>
ame INVT_DA200_262]
String for a fulltaxt search	Vendor			~
Name			Vendor	^
Eieldhusen				
EtherCAT				
But Effercat				
But Slave				
Delta Electronics, Inc Ser	vo Drives			
ifm electronic - ifm electroni	c EtherCAT De	vices		
INVT				
INVT INDUSTRIAL				
Servo Drives				
1 DA200-N EtherCAT	(CoE) Drive		INVT INDUSTRIAL	
🖲 🔄 Panasonic Corporation, App	oliances Compa	any - AC Servo Driver		
🛞 🔁 Darker Hannifin - Darker Cer	nuo Drivio 1M		>	Ť
Group by category Display all version:	s (for experts o	only) 🗌 Display outdated versions		-
Name: DA200-N EtherCAT(CoE) Drive			^	ī
Vendor: INVT INDUSTRIAL				
Categories: Slave			\$	
Version: Revision=16#000000AB				
Order Number: INVT_DA200_262				
Description: EtherCAT Slave imported	from Slave XM	L: INVT_DA200_EtherCAT_V262_200313.xml Device	2:	
DA200-N EtherCAT(CoE) Drive			¥	
Append selected device as last child of EtherCAT_Master_SoftMotion				
• (You can select another target node in th	ne navigator v	while this window is open.)		
		2	Add Device Close	

Figure 3-2 Add the DA200 servo drive

4. Select the INVT_DA200_262 in the device tree, right-click it and add the motor axis (select **SoftMotion's CiA 402 axis**). Add the call program as shown in the following figure.



Figure 3-3 DA200 servo drive application example

3.4 CANopen

CANopen is a high-level communication protocol that is based on the CAN (Controller Area Network) protocol, including communication profile and device profile.

The communication model defines four types of messages (communication objects).

• Management message

Layer management, network management and ID assignment services: such as initialization, configuration and network management (including: node protection).

The services and protocols conform to the LMT, NMT and DBT services sections of the CAL. These services are based on the master-slave communication mode, which means there can only be one LMT, NMT or DBT master node and one or more slave nodes in a CAN network.

• Service Data Object (SDO)

By using indexes and sub-indexes (in the first few bytes of a CAN message), the SDO enables clients to access items (objects) in the device (server) object dictionary.

SDO is implemented through a multi-domain CMS object in CAL that allows the transfer of data of any length. The data will be split into several messages when it exceeds 4 bytes.

The protocol confirms the service type: generating an answer for each message (two IDs are required for an SDO). SDO request and answer messages always contain 8 bytes (meaningless data lengths are indicated in the first byte which carries the protocol information). SDO communication has many protocols.

• Process Data Object (PDO)

PDO is used to transfer real-time data from a creator to one or more recipient s. Data transfer is limited to 1 to 8 bytes (for example, one PDO can transfer up to 64 digital I/O values, or 4 16-bit AD values).

PDO communication has no protocol defined. PDO data content is defined only by its CAN ID, assuming that the creator and recipient s know the data content of the PDO.

Each PDO is described by two objects in the object dictionary:

1. PDO communication parameters: determine which COB-ID will be used by the PDO, transmission type, prohibition time, and timer period.

2. PDO mapping parameter: a list of objects in the object dictionary that are mapped to the PDO, including their data

lengths (in bits). The creator and recipients must know this mapping to interpret PDO content.

PDO message content is predefined (or configured at network startup).

Mapping application objects to the PDO is described in the device object dictionary. If the device (creator and recipients) supports variable PDO mappings, the PDO mapping parameters can be configured using SDO messages.

PDO can be delivered in the following modes:

1. Synchronization (by receiving SYNC objects)

Aperiodic: The transmission is pre-triggered by a remote frame or by an object-specific event defined in the device profile.

Periodic: The transmission is triggered after every 1 to 240 SYNC messages.

2. Asynchronization

The transmission is triggered by a remote frame or by an object-specific event defined in the device profile.

• Predefined messages or special function objects:

SYNC

Time Stamp

Emergency

Node guarding

3.4.1 CANopen master node configuration

3.4.1.1 Master node usage process

• Install the CANopen slave devices.

The associated CANopen slave device profile must first be installed into the system. The device profile can be a *.Devdesc.xml file or an EDS (Electronic Data Sheet) file for the manufacturer.

• Add CANbus to the device tree.

The base node of CANopen (the uppermost entry in the CANbus configuration tree) must be a CANbus object. A CANbus can be inserted underneath the device node of the AX series programmable controller. The device tree structure after adding a CANbus is shown in the following diagram.





3.4.1.2 Adding CANopen management device

Under the CANbus, add a **CANopen Management** device, which can be used as a CANopen master. The device tree structure after adding the device is shown in the following diagram.


Figure 3-5 Device tree structure with a CANopen master

3.4.1.3 Adding CANopen slave node

Take our DA200 CANopen slave as an example. Add the DA200 slave device under CANopen Manager after adding the EDS file of this slave, as shown in the following diagram.



Figure 3-6 Device tree structure with a CANopen slave

The software configuration of the CANopen master is complete.

3.4.2 Parameter configuration of CANopen master

Configure Network and Baud Rate of the CANbus first.

Network: the number of CAN networks connected via the CANbus, range: 0-100.

Baud Rate: the baud rate used for transmission on the bus, the following baud rates can be set: 10kbits/s, 20kbits/s, 50kbits/s, 100kbits/s, 125kbits/s, 250kbits/s, 500kbits/s, 800kbits/s, and 1000kbits/s.

PLC_PRG	us X		
General	General		
Log	Network	0	CAN
CANbus IEC Objects	Baudrate (kbit/s)	250 ~	
Status			
Information			

Figure 3-7 Parameter configuration of CANbus

CANopen Management is a node under the CANbus node that supports CANbus configuration through internal functions.

It is generally used as the CANbus master. The configuration page is shown in the following figure.

General	General							
Log	Node ID 127 🗘 Check	and Fix Configuration						
CANopen I/O Mapping	🖌 Autostart CANopen Manager 🛛 Polling	of optional slaves						
CANopen IEC Objects	Start Slaves NMT error behaviour Restart Slave 🗸							
NMT start all (if possible)								
⊿ Guarding								
Information	Enable heartbeat producing							
	Node ID 127							
	Producer time (ms) 200 🖨							
	✓ SYNC							
	Enable SYNC producing	Enable TIME producing						
	COB ID (Hex) 16# 80 *	COB ID (Hex) 16# 100						
	Cycle period (µs) 1000	Producer time (ms) 1000						
	Window length (µs) 1200 🔹]						
	Enable SYNC consuming							

Figure 3-8 Parameter configuration of CANopen master

Node ID: Provides an array pair module that CANopen Manager can correspond to one-to-one, with ID values of 1-127 (must be a decimal integer).

Guarding: Heartbeat mode is a traditional protection mechanism that can be handled by the master station and the slave station modules, different form node protection. Normally the master is configured to send a heartbeat to the slave.

Enable heartbeat producing: If this option is enabled, the master will send heartbeats continuously according to an internally defined heartbeat time. If a new slave heartbeat function is added, their heartbeat actions will be automatically activated and configured, i.e. the node ID is automatically set in the management configuration and the heartbeat interval is automatically multiplied by a factor of 1 and 2. If this option is disabled, the node protection (with a life time factor of 10 and a protection time of 100ms) is activated in the slave.

Node ID: Unique identifier of heartbeat generation (1–127) on the bus.

Producer time (ms): Defines the internal heartbeat time in milliseconds.

4 Module Configuration

4.1 CPU module

Please follow the steps to configure the AX series programmable controller real time and IP address.

1. Create a controller Cfg project.

Add the library file CmpPlcCfg_x.x.x.x.library required for this module to create a standard project.

2. Define and use variables.

Variable	Туре		Function	Remarks		
setEnable		BOOL	Time setting function	0: Disabled 1: Enabled		
getEnable		BOOL	Time reading function	0: Disabled 1: Enabled		
inTime		ARRAY OF UINT	Time to be entered in format: hour minute second	E.g. 14 48 56		
inDate	INPUT	ARRAY OF UINT	Date to be entered in format: year month day	E.g. 2018 12 26		
r_Enable		BOOL	IP reading function	0: Disabled 1: Enabled		
w_Enable	BOOL IP setting function		0: Disabled 1: Enabled			
new_IP		STRING	Set a new IP	E.g. 192.168.1.16		
new_netmask		STRING	Set a new subnet mask	E.g. 255. 255. 255.0		
setDone		BOOL	Completion mark of time setting	 The execution of commands is in progress. The execution of commands is completed. 		
getDone		BOOL	Completion mark of time obtaining	 The execution of commands is in progress. The execution of commands is completed. 		
setError		INT	Configuration error sign	See Controller Cfg error code table		
getError	OUTPUT	INT	Get error sign	See Controller Cfg error code table		
outTime		ARRAY OF UINT	Read the native hour, minute and second information.	E.g. 14 48 56		
outDate		ARRAY OF UINT	Read the native year, month and day information.	E.g. 2018 12 26		
Done		BOOL	Completion mark	0: The execution of commands is in progress. 1: The execution of commands is completed.		
read_IP		STRING	IP read	E.g. 192.168.1.16		
read_netmask		STRING	Subnet mask read	E.g. 255. 255. 255.0		

Table 4-1 Variable definition

Variable	Function	Remarks
setEnable	Time setting function	0: Disabled 1: Enabled
getEnable	Time reading function	0: Disabled 1: Enabled
inDate	Date to be entered in format: year month day	E.g. 2018 12 16
inTime	Time to be entered in format: hour minute second	E.g. 14 48 56

Table 4-2 Local time configuration

According to the time array in format inTime and inDate, where inTime[0] is hour, inTime[1] is minute, inTime[2] is second, inDate[0] is year, inDate[1] is month, inDate[2] is day, enter the time (all inputs are required). After the settings, enable setEnable to set the above time to current time of the AX series programmable controller CPU.

Enable getEnable to get the real time of the AX series programmable controller CPU, which is displayed in outTime and outDate arrays.

Variable	Function	Remarks				
r_Enable	IP reading function	0: Disabled 1: Enabled				
w_Enable	IP setting function	0: Disabled 1: Enabled				
new_IP	Set a new IP	E.g. 192.168.1.16				
new_netmask	Set a new subnet mask	E.g. 255. 255. 255.0				

Table 4-3 Local IP configuration

Enter the IP and subnet mask in the required format, and then enable w_Enable to set the above IP and subnet mask to the current IP and subnet mask of AX series programmable controller EtherNET network port after entering the setup.

Note: The USB virtual network port is independent of the EtherNET network port, and the IP and subnet mask modified by CmpPlcCfg_x.x.x.x.library is still the IP and subnet mask of the EtherNET network port when the device is connected with a USB. After the IP or subnet mask modification, it will take some time for the AX series CPU to connect to Invtmatic Studio on the PC.

Enable r_Enable to get the IP address and subnet mask of the controller EtherNET network port, which are displayed in the read_IP and read_netmask strings respectively.

4.2 High speed I/O module

4.2.1 Creating high speed I/O module project

Create the high speed I/O module application and add the corresponding library file. Then complete the corresponding variable configuration in HIGH_PULSE_IO device.

HSIO stands for High Speed Input and Output. HSIO can be used for high speed counting and high speed pulse output with three interrupt functions that can be configured as needed. HSIO contains the device profile Shenzen INVT-AX7X-CPU_x.x.x.devdesc, the high speed counting function block library CmpHSIO_C.library and the motion

control function block library CmpHSIO_M.library or CmpIMC_P.library.

The HSIO device profile is used to configure various functions of the high-speed IO, including input/output port function, counter, high-speed pulse output, filter parameters, and interruption.

The high-speed counting function block library CmpHSIO_C.library contains several function blocks, such as counter setting, count value reading, latching, preset value, pulse width measurement, timing sampling, and count value comparison. These function blocks can be called to complete the application needed for counting.

The motion control function block library CmpHSIO_M.library is described in detail via dedicated instructions.

AX7 \Box -C-1608P, hereinafter referred to as the P-type model, and AX7 \Box -C-1608N, hereinafter referred to as the N model. The software of the P model and the N model are the same but the hardware ports are different.

4.2.1.1 P-type model port configuration description

At present, AX7 -C-1608P programmable controller integrates 16-channel high-speed pulse input (The first 6 channels support 24V single-ended input or differential input, and the last 10 channels support 24V single-ended input) and 8-channel pulse output which supports pulse+direction mode, FWD/REV pulse mode and quadrature pulse mode, and each port can be configured with different functions. The configuration table is shown as follows.

Input port	Common input function (default)	Counting function	Trigger latching and Z-signal function	Positive and negative limit zero function	Pulse width measure -ment function	Output port	Common input function (default)	High speed pulse output function	Compare Output Function
	Function value is 0	Function value is 1	Function value is 2	Function value is 3	Function value is 4		Function value is 0	Function value is 1	Function value is 2
X0 (In0)	Common input	COA		CH0N		Y0 (Out0)	Common output	CH0CW/PULS0	CMP0
X1 (In1)	Common input	COB		CH1N		Y1 (Out1)	Common output	CH0CCW/SIGN0	CMP1
X2 (In2)	Common input	C1A		CH2N		Y2 (Out2)	Common output	CH1CW/PULS1	CMP2
X3 (In3)	Common input	C1B		CH3N		Y3 (Out3)	Common output	CH1CCW/SIGN1	CMP3
X4 (In4)	Common input	C4A	C0Z	CH0P		Y4 (Out4)	Common output	CH2CW/PULS2	CMP4
X5 (In5)	Common input	C4B	C1Z	CH1P		Y5 (Out5)	Common output	CH2CCW/SIGN2	CMP5
X6 (In6)	Common input	C5A	C2Z	CH2P		Y6 (Out6)	Common output	CH3CW/PULS3	CMP6
X7 (In7)	Common input	C5B	C3Z	СНЗР		Y7 (Out7)	Common output	CH3CCW/SIGN3	CMP7
X8 (In8)	Common input	C2A	С0Т		PWC0				
X9 (In9)	Common input	C2B	C1T		PWC1				
XA (InA)	Common input	C3A	C2T		PWC2				
XB (InB)	Common input	C3B	СЗТ		PWC3				
XC (InC)	Common input	C6A		CH0Z					

Input port	Common input function (default)	Counting function	Trigger latching and Z-signal function	Positive and negative limit zero function	Pulse width measure -ment function	Output port	Common input function (default)	High speed pulse output function	Compare Output Function
	Function value is 0	Function value is 1	Function value is 2	Function value is 3	Function value is 4		Function value is 0	Function value is 1	Function value is 2
XD (InD)	Common input	C6B		CH1Z					
XE (InE)	Common input	C7A		CH2Z					
XF (InF)	Common input	C7B		CH3Z					

Note:

- X0-XF is the input port and Y0-Y7 is the output port.
- Common input and common output mean a common I/O signal, usually a switching signal.
- CxA, CxB, and CxZ are signals of encoder A, B, and Z respectively.
- CxT refers to the trigger and latch function channel and supports 4 channels, C0T–C3T.
- CHxP and CHxN refer to positive and negative limit signals, with N being the negative direction and P being the positive direction. CHxZ refers to the zero signal.
- PWCx means pulse width check signal.
- CHxCW is a clockwise signal and CHxCCW is a counterclockwise signal.
- PULSx means pulse.
- SIGNx means the direction of the pulse.
- CMPx means the output comparison.

4.2.1.2 N-type model port configuration description

At present, AX7 -C-1608N programmable controller integrates 16-channel high-speed pulse input (The first 4 channels support differential input, and the last 12 channels support 24V single-ended input) and 8-channel high-speed pulse output which supports pulse+direction mode, FWD/REV pulse mode and quadrature pulse mode, and each port can be configured with different functions. The configuration table is shown as follows.

Input port	Common input function (default)	Counting function	Trigger latching and Z-signal function	Positive and negative limit zero function	Pulse width measure -ment function	Output port	Common input function (default)	High speed pulse output function	Compare Output Function
	Function value is 0	Function value is 1	Function value is 2	Function value is 3	Function value is 4		Function value is 0	Function value is 1	Function value is 2
A0 (In0)	Common input	COA		CHON		Y0 (Out0)	Common output	CH0CW/PULS0	CMP0
B0 (In1)	Common input	COB		CH1N		Y1 (Out1)	Common output	CH0CCW/SIGN0	CMP1
A1 (In2)	Common input	C1A		CH2N		Y2 (Out2)	Common output	CH1CW/PULS1	CMP2

Input port	Common input function (default)	Counting function	Trigger latching and Z-signal function	Positive and negative limit zero function	Pulse width measure -ment function	Output port	Common input function (default)	High speed pulse output function	Compare Output Function
	Function value is 0	Function value is 1	Function value is 2	Function value is 3	Function value is 4		Function value is 0	Function value is 1	Function value is 2
B1 (In3)	Common input	C1B		CH3N		Y3 (Out3)	Common output	CH1CCW/SIGN1	CMP3
X4 (In4)	Common input	C4A	C0Z	CH0P		Y4 (Out4)	Common output	CH2CW/PULS2	CMP4
X5 (In5)	Common input	C4B	C1Z	CH1P		Y5 (Out5)	Common output	CH2CCW/SIGN2	CMP5
X6 (In6)	Common input	C5A	C2Z	CH2P		Y6 (Out6)	Common output	CH3CW/PULS3	CMP6
X7 (ln7)	Common input	C5B	C3Z	CH3P		Y7 (Out7)	Common output	CH3CCW/SIGN3	CMP7
X8 (In8)	Common input	C2A	СОТ		PWC0				
X9 (In9)	Common input	C2B	C1T		PWC1				
X10 (InA)	Common input	C3A	C2T		PWC2				
X11 (InB)	Common input	C3B	СЗТ		PWC3				
X12 (InC)	Common input	C6A		CH0Z					
X13 (InD)	Common input	C6B		CH1Z					
X14 (InE)	Common input	C7A		CH2Z					
X15 (InF)	Common input	C7B		CH3Z					

Note:

- A0/B0/A1/B1/X4-X15 is the input port and Y0-Y7 is the output port.
- Common input and common output mean a common I/O signal, usually a switching signal.
- A0/B0/A1/B1 is recommended not to be a normal input port. In special cases, if it is used as a normal input port, a 2K resistor needs to be connected in series in the circuit, otherwise the point will be burned out.
- CxA, CxB, and CxZ are signals of encoder A, B, and Z respectively.
- CxT refers to the trigger and latch function channel and supports 4 channels, C0T–C3T.
- CHxP and CHxN refer to positive and negative limit signals, with N being the negative direction and P being the positive direction. CHxZ refers to the zero signal.
- PWCx means pulse width check signal.
- CHxCW is a clockwise signal and CHxCCW is a counterclockwise signal.
- PULSx means pulse.
- SIGNx means the direction of the pulse.

• CMPx means the output comparison.

4.2.2 Input port function description

The input port can be set to five functions, which are: common input function, counting function, triggering latch and Z-signal function, positive and negative limit zero function, and pulse width measurement function. Here is the mapping table of configuration input function corresponding to Inx_Configure parameters, where x ranges from 0 to F.

HIGH_PULSE_IO Parameters	Find Filter Show all								
HIGH PULSE TO I/O Mapping	Variable	Mappi	Channel	Address	Туре	Unit	Descri		
	Application.in0	~⊘	In0_Configure	%QB0	BYTE				
Status	Application.in1	~¢	In1_Configure	%QB1	BYTE				
	Application.in2	~⊘	In2_Configure	%QB2	BYTE				
Information	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				

For P-type model, the In0_Configure–InF_Configure port function configuration parameters correspond to ports X0–XF in turn.

For N-type model, the In0_Configure–InF_Configure port function configuration parameters correspond to ports A0/B0/A1/B1/X4–X15 in turn.

HIGH_PULSE_IO Parameters	Find		•				
HIGH PULISE TO I/O Mapping	Variable	Mappi	Channel	Address	Туре	Unit	Descri
India_rococ_roll/olimpping	Application.in0	~⊘	In0_Configure	%QB0	BYTE		
Status	Application.in1	~∕	In1_Configure	%QB1	BYTE		
-	Application.in2	~	In2_Configure	%QB2	BYTE		
Information	🍢 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		
	- Note 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 Common input function

If the function value is 0, the signal port is configured to be used as a common input port.

Common input: P-type model										
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring			
	X0	Common input	40	39	Common input	X1	<u> </u>			
24VDC - +			38	37			±⊥24VDC			
L	COM	Input	36	35	Input	СОМ	-			

Wiring of common input ports

Common input: P-type model										
External wiring	Port	Function	CN5 teri	minal No.	Function	Port	External wiring			
		common			common					
		port			port					
	Vo	Common	0.4	00	Common	Vo				
24VDC	Χ2	input	input 34	33	input	X3	<u> </u>			
			32	31			+ 24VDC			
		Input			Input					
	COM	common	30	29	common	COM				
		port			port					
_/	V4	Common	20	27	Common	VE				
	Λ4	input	20	21	input	72				
			26	25						
-		Input			Input					
	COM	common	24	23	common	СОМ				
		port			port					
24VDC							-1 + +			
		Input			Input					
	SS1	common	22	21	common	SS2				
		port			port		$\neg \neg$			

Common input: N-type model										
External wiring	Port	Function	CN5 terminal No.		Function	Port	External wiring			
	X4	Common input	X4	X5	Common input	X5	└─└└└└└└└└ ┾ _┿ ╪╴			

Configuration of common input ports

Define the variables to configure the ports and map them to the high speed pulse mapping table.

Configuration routine:

1: Configure X4 port of P-type model and X4 port of N-type model as common input ports.

in4:=0;

Application.in4	~∕	In4_Configure	%QB4	BYTE
-----------------	----	---------------	------	------

2: Configure X5 port of P-type model and X5 port of N-type model as common input ports.

in5:=0;



4.2.2.2 Counting function

If the function value is 1, the signal port is configured as a counter function and all 16 input ports can be used as counter inputs.

Counting function module can count and calculate the input pulse, and detect the position, speed and frequency. The maximum frequency of input pulse is 200kHz.

Wiring of counting function ports

Counting function (Single-end source): P-type model									
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring		
<u></u>	C0A	Phase A pulse input	40	39	Phase B pulse input	C0B	<u></u>		
<u>+</u>			38	37			±_		
-	СОМ	Input common port	36	35	Input common port	СОМ	-		
<u>7.</u>	C1A	Phase A pulse input	34	33	Phase B pulse input	C1B	<u></u>		
<u> </u>			32	31			±_		
	СОМ	Input common port	30	29	Input common port	СОМ			
<u>7.</u>	C4A	Phase A pulse input	28	27	Phase B pulse input	C4B	<u></u>		
<u> </u>			26	25			±_		
	СОМ	Input common port	24	23	Input common port	СОМ	-		
Т _л							<u></u> 		
<u></u>							<u></u>		
<u></u>	SS1	port	22	21	port	SS2	<u></u>		
<u></u>							<u></u>		
							<u></u>]		

Counting function (Single-end source): N-type model										
External wiring	Port	Function	CN5 terminal No.		Function	Port	External wiring			
External wiring	Port C4A	Function Phase A pulse input	X4	ninal No. X5	Function Phase B pulse input	Port X5	External wiring			
+ 24VDC							+ - 24VDC			

Counting function (Single-end sink): P-type model									
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring		
	СОМ	Input common port	40	39	Input common port	СОМ			
_ <u>+</u> + T-			38	37			<u>+ </u> -T		
	C0A	Phase A pulse input	36	35	Phase B pulse input	C0B	<u></u>		
	СОМ	Input common port	34	33	Input common port	СОМ			
			32	31			<u>+ </u> T		
L. <u>.</u>	C1A	Phase A pulse input	30	29	Phase B pulse input	C1B	<u></u>]		
	СОМ	Input common port	28	27	Input common port	СОМ			
			26	25			<u>+ </u> Ţ		
L <u>-T-</u>	C4A	Phase A pulse input	24	23	Phase B pulse input	C4B	<u></u>]		
······································	SS1	Input common port	22	21	Input common port	SS2			

Counting function (Single-end sink): N-type model										
External wir	ring	Port	Function	CN5 tern	ninal No.	Function	Port	External wiring		
ء	<u>л</u> .							<u></u>		
-	<u></u>	C4A	Phase A pulse input	X4 X5		Phase B pulse input	C4B	<u></u>		
-	Л							<u> </u>		
-	<u>л</u>				Х5			<u></u>		
-	<u></u>							<u></u>		
 24VDC +	<u>π</u>							<u>-</u> - <u> </u> + 24VDC		

Counting function (differential signal): P-type model									
External wiring	Port	Function	CN5 terminal No.		Function	Port	External wiring		
			40	39					
	C0A+	Phase A differential +	38	37	Phase B differential +	C0B+	4		
	C0A-	Phase A differential -	36	35	Phase B differential -	C0B-			
			34	33					
	C1A+	Phase A differential +	32	31	Phase B differential +	C1B+	4		
	C1A-	Phase A differential -	30	29	Phase B differential -	C1B-			
			28	27					
	C4A+	Phase A differential +	26	25	Phase B differential +	C4B+	A		
	C4A-	Phase A differential -	24	23	Phase B differential -	C4B-			

Counting function (differential signal): N-type model											
External wiring	Port	Function	CN5 tern	ninal No.	Function	Port	External wiring				
	C4A	Phase A pulse input	X4	X5	Phase B pulse input	C4B	X				

Configuration of counting ports

Function value configuration:

Define the variables to configure the ports with data type BYTE, and map them to the high-speed pulse mapping table.

Configuration routine:

1) Configure X0 port of P-type model and A0 port of N-type model as counting ports.

in0:=1;

AX series programmable controller software ma	Module Configuration				
··· 🍞 Application.in0	- •	In0_Configure	%QB0	BYTE	

2) Configure X1 port of P-type model and B0 port of N-type model as counting ports.

: A	. 4	
Int	· = 1	
		,

🍫 Application.in1	~⊘	In1_Configure	%QB1	BYTE
-------------------	----	---------------	------	------

Configure other ports by analogy.

4.2.2.3 Trigger, latch and Z-signal function

If the function value is 2, the signal port is configured as trigger, latch and Z-signal functions.

The trigger function can preset count value for the counter and the rising edge of the trigger signal is valid. The preset value will be written to the counter once the signal is valid. Normally there are three ways to write the preset value of the counter: software writing, external trigger writing, and consistent comparison trigger writing. This product uses external trigger writing.

The latch function can lock the counter value instantly for the upper computer to read.

The trigger and latch functions support 4 channels, C0T–C3T (P-type model mapping ports X8, X9, XA, XB and N-type model mapping ports X8, X9, X10, X11).

Z-signal function is used for Z clearing and Z compensation functions and Z-signal encoders generate one pulse per revolution.

Z-signal function supports 4 channels, C0T–C3T (mapping ports X4, X5, X6, X7)

Wiring of trigger, latch and Z-signal ports

Input function 3:	: (CnT wiri	ng refers to c	ommon i	input; Cr	Z wiring refer	s to coun	ting pulse input)		
P-type model									
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring		
<u></u>	C0Z	Phase Z single-ended input	28	27	Phase Z single-ended input	C1Z	<u></u>		
T-	C0Z+	Phase Z differential input	26	25	Phase Z differential input	C1Z+			
	СОМ	Input common port	24	23	Input common port	СОМ]		
#I-	SS1	Input common port	22	21	Input common port	SS2	<u></u>		
┋╴╶╤┚┖	C2Z	Z signal input	20	19	Z signal input	C3Z			
	С0Т	Probe signal input	18	17	Probe signal input	C1T			
	C2T	Probe signal input	16	15	Probe signal input	C3T			

	•	•		• •	•		•••••••••••••••••••••••••••••••••••••••
N-type model							
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring
п	C0Z	Z signal input	X4	X5	Z signal input	C1Z	
	C2Z	Z signal input	X6	X7	Z signal input	C3Z	
·	СОТ	Probe signal input	X8	X9	Probe signal input	C1T	
	C2T	Probe signal input	X10	X11	Probe signal input	СЗТ	
<u></u> <u>.</u> <u>.</u>	SS	Input common port	SS	SS	Input common port	SS	<u></u>

Input function 3: (CnT wiring refers to common input; CnZ wiring refers to counting pulse input)

Configuration of the trigger, latch and Z-signal ports

Function value configuration: Define the variables to configure the ports with data type BYTE, and map them to the high-speed pulse mapping table.

Configuration routine:

1) Configure X8 port of P-type model and X8 port of N-type model as trigger and latch ports.

in8:=2;

Application.in8	°∳	In8_Configure	%QB8	BYTE

2) Configure X4 port of P-type model and X4 port of N-type model as Z-signal ports

in4:=2;

Application.in4	~⊘	In4_Configure	%QB4	BYTE

4.2.2.4 Positive and negative limit zero function

If the function value is 3, the signal port is configured as positive and negative limit zero function.

CHxP/CHxN/CHxZ are positive limit, negative limit and zero signal functions on the x channel, where x ranges from 0 to 3. The positive limit serves to limit the positive direction, where motor movement needs to stop or reverse. The negative limit serves to limit the negative direction, where motor movement needs to stop or reverse.

Wiring of positive and negative limit zero ports

Input function 4: (CHnZ and CHnP wiring refers to common input; CHnZ wiring refers to counting pulse input) P-type model									
External wiring	Port	Function	CN5 te N	erminal Io.	Function	Port	External wiring		
	CH0N	Negative limit input	40	39	Negative limit input	CH1N	<u> </u>		
			38	37			<u>+</u>		
	СОМ	Input common port	36	35	Input common port	СОМ			
	CH2N	Negative limit input	34	33	Negative limit input	CH3N			
			32	31			-		

		СОМ	Input common port	30	29	Input common port	СОМ	
Г	/	CH0P	Positive limit input	28	27	Positive limit input	CH1P	<u> </u>
	-			26	25			+ -T
L		СОМ	Input common port	24	23	Input common port	СОМ	
		SS1	Input common port	22	21	Input common port	SS2	-1 +
│ [┿] ┿ └──┾╯──	/	CH2P	Positive limit input	20	19	Positive limit input	СНЗР	
	л							л
	Л	CH0Z	Home signal	14	13	Home signal	CH1Z]
		CH2Z	Home signal	12	11	Home signal	CH3Z	

Input function 4: (CHxZ and CHxP wiring refers to common input; CHxZ wiring refers to counting pulse input) N-type model

External w	viring	Port	Function	CN5 te N	erminal Io.	Function	Port	External wiring
24VDC 2		CH0N	Negative limit input	A0+	B0+	Negative limit input	CH1N	2KΩ <u>+</u> 24VDC
		СОМ	Input common port	A0-	B0-	Input common port	СОМ	
24VDC <u>+</u> +2 -	K12	CH2N	Negative limit input	A1+	B1+	Negative limit input	CH3N	24VDC
)	_/	СОМ	Input common port	A1-	B1-	Input common port	СОМ	
		CH0P	Positive limit input	X4	X5	Positive limit input	CH1P	
	л	CH2P	Positive limit input	X6	Х7	Positive limit input	СНЗР	л
				X8	X9			
	. . 			X10	X11			╶┨╴╧
T	+	CH0Z	Home signal	X12	X13	Home signal	CH1Z	<u> </u>
		CH2Z	Home signal	X14	X15	Home signal	CH3Z	
		SS	Input common port	SS	SS	Input common port	SS	

Configuration of positive and negative limit zero ports

Function value configuration:

Define the variables to configure the ports, and map them to the high-speed pulse mapping table.

Configuration routine:

1. Configure X3 port of P-type model and B1 port of N-type model as negative limit ports.

in3:=3;

Application.in3	~	In3_Configure	%QB3	BYTE
-----------------	----------	---------------	-----------------	------

2. Configure XC port of P-type model and X12 port of N-type model as zero ports.

in**C**:=3;

Application.inC	~ ⊘	InC_Configure	%QB12	BYTE
-----------------	------------	---------------	-------	------

4.2.2.5 Pulse width measurement function

If the function value is 4, the signal port is configured as a pulse width measurement function.

PWCx is a pulse width measurement input channel x, where x ranges from 0 to 3. The ports corresponding to P-type model are X8, X9, XA and XB, and the ports corresponding to N-type model are X8, X9, X10 and X11.

Wiring of pulse measurement ports

Input function 5: (PWCn wiring refers to counting pulse input) P-type model									
External w	iring	Port	Function	CN5 terminal No.		Function	Port	Externa	al wiring
		004	Input common	22	01	Input common	000		
		551	port	22	21	port	552		
<u> </u>								-1	±
⊺ - ⊤ ∓	•							+	
			Pulse			Pulse			
	Л	PWC0	measurement	18	17	measurement	PWC1	<u> </u>	
			signal			signal			
L.			Pulse			Pulse			
		PWC2	measurement	16	15	measurement	PWC3		
			signal			signal			

Input function 5: (PWCn wiring refers to counting pulse input) N-type model									
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring		
		Pulse			Pulse				
— п	PWC0	measurement	X8	X9	measurement	PWC1			
		signal			signal		·		
		Pulse			Pulse		<u></u>		
	PWC2	measurement	X10	X11	measurement	PWC3			
		signal			signal				
			X12	X13					
			X14	X15					
<u></u>		Input common	22	22	Input common	<u> </u>	<u>++</u> ;		
	35	port	35	- 55	port	55			

Configuration of pulse width measurement ports

Function value configuration:

Define the variables to configure the ports, and map them to the high-speed pulse mapping table.

Configuration routine:

1. Configure X8 port of P-type model and X8 port of N-type model as a pulse width measurement port in8:=4;

Application.in8	~₽	In8_Configure	%QB8	BYTE

2. Configure X9 port of P-type model and X9 port of N-type model as a pulse width measurement port.

in9:=4;

Application.in9	~	In9_Configure	%QB9	BYTE

4.2.3 Output port function description

The output port can be set for 3 functions: common output function, high-speed pulse output function and output comparison function.

4.2.3.1 Common output function

If the function value is 0, the signal port is configured to be used as a common output port. The following are the parameters of $Outx_Configure$ in the mapping table of the configuration output function, where the range of x is 0–7.

HIGH_PULSE_IO Parameters	Find		Filter Show all				•
HIGH PULISE TO I/O Mapping	Variable	Mappi	Channel	Address	Туре	Unit	Descri
	Application.xmodec	~⊘	XMode_SetC	%QB18	BYTE		
Status	Application.xmoded	~⊘	XMode_SetD	%QB19	BYTE		
	Application.filt_set	~∳	Filt_Set	%QB20	BYTE		
Information	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		

Wiring of common output ports

Common port: P-type model									
External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring		
	Y0	Common output	10	9	Common output	Y1			
	Y 2	Common output	8	7	Common output	Y3	Load		
	Y4	Common output	6	5	Common output	Y5	Load		
Load 24VDC Fuse	Y6	Common output	4	3	Common output	Y7	Load Fuse 24VDC		
└ュ╠╤══╾	СОМ	Output common	2	1	Output common port	СОМ	-□-‡ -┘		

Common port: N-type model										
External wiring	Port	Function	CN5 tern	ninal No.	Function	Port	External wiring			
Fuse Load Load Load Load Load	СОМ	Output common port	СОМ	СОМ	Output common port	СОМ	Fuse Load Load Load Load Load			

The output port of P-type model contains 8 output signals. Only single-ended outputs are supported, and the signal type is source type output. Y0, Y2, Y4 and Y6 share the common port COM1, and Y1, Y3, Y5 and Y7 share the common port

COM2.

The output port of N-type model contains 8 output signals. Only single-ended outputs are supported, and the signal type is sink type output. Y0–Y7 share the common port COM2.

Configuration of common output ports

Function value configuration:

Define the variables to configure the ports and map them to the high speed pulse mapping table.

Configuration routine:

1. Configure Y0 as a common output port.

out0:=0;

Application.out0	~ >	Out0_Configure	%QB21	BYTE

2. Configure Y1 as a common output port.

out1:=0;

Application.out1	°)	Out1_Configure	%QB22	BYTE
-				

4.2.3.2 High speed pulse output function

If the function value is 1, the signal port is configured as a high-speed pulse output function, and all 8 output ports can be configured for high-speed pulse output.

The high-speed pulse output support pulse + direction, FWD/REV pulse, and quadrature pulse modes.

Wiring of high-speed pulse output ports

Output pulse: P-type model									
External wiring	Port	Function	CN5 te N	rminal o.	Function	Port	External wiring		
Encoder 0 Pulse + Pulse -	Plus0	Pulse output	10	9	Direction output	Sign0	Direction+ Encoder 0 Direction-		
Encoder 1 Pulse+ Pulse-	Plus1	Pulse output	8	7	Direction output	Sign1	Direction+ Direction- Encoder 1		
Encoder 2 Pulse+ Pulse-	Plus2	Pulse output	6	5	Direction output	Sign2	Direction+Encoder 2		
Encoder 3 Pulse+ Pulse-	Plus3	Pulse output	4	3	Direction output	Sign3	Direction+Encoder 3		
	СОМ	Output common port	2	1	Output common port	СОМ			

Output pulse: N-type model									
External wiring	Port	Function	CN5 te	erminal o.	Function	Port	External wiring		
Г	СОМ	Output common port	СОМ	СОМ	Output common port	СОМ			
Encoder 0 Pulse+ Pulse-	Plus0	Pulse output	Y0	Y1	Direction output	Sign0	Direction+ Direction-Encoder 0		
Encoder 1 Pulse+ Pulse-	Plus1	Pulse output	Y2	Y3	Direction output	Sign1	Direction+ Encoder 1 Direction-		
Encoder 2 Pulse+	Plus2	Pulse output	Y4	Y5	Direction output	Sign2	Direction+ Encoder 2 Direction-		
Encoder 3 Pulse+ Pulse-	Plus3	Pulse output	Y6	Y7	Direction output	Sign3	Direction+Encoder 3		

Configuration of high-speed pulse output ports

Function value configuration:

Define the variables to configure the ports, and map them to the high-speed pulse mapping table.

Configuration routine:

1. Configure Y0 as a high-speed pulse output port.

out0:=1;

2. Configure Y1 as a high-speed pulse output port.

out1:=1;

Application.out1	°\$	Out1_Configure	%QB22	BYTE
-				

4.2.3.3 Output comparison function

If the function value is 2, the signal port is configured as an output comparison function with 8 channels.

The output comparison outputs the result of the counter single value comparison, and each counter channel has an output comparison function. If the counter value is equal to the set comparison value, it will output high, and if it is not equal, it will output low.

Wiring of output comparison ports

Comparison consistent output: P-type model											
External wiring	Port	Function	CN5 teri	minal No.	Function	Port	External wiring				
Land	Y0	Common output	10	9	Common output	Y1					
Load	Y2	Common output	8	7	Common output	Y3	Load				
Load	Y4	Common output	6	5	Common output	Y5	Load				
Load 24VDC	Y6	Common output	4	3	Common output	Y7	Load 24VDC				
│ └₋ª┠ _┿	СОМ	Output common	2	1	Output common	СОМ	┝╾╾╾╤┨╺╾╌┚				
		port			port						

Comparison consistent output: N-type model										
External wiring	Port	Function	CN5 tern	ninal No.	Function	Port	External wiring			
Fuse	СОМ	Output common port	СОМ	СОМ	Output common port	СОМ	Fuse			
	YO	Common output	Y0	Y1	Common output	Y1				
Load	Y2	Common output	Y2	Y3	Common output	Y3	Load			
Load	Y4	Common output	Y4	Y5	Common output	Y5	Load			
	Y6	Common output	Y6	Y7	Common output	Y7				

Configuration of output comparison ports

Function value configuration:

Define the variables to configure the ports, and map them to the high-speed pulse mapping table.

Configuration routine:

1. Configure Y0 as a comparison output port.

out0:=2;

Application.out0	°∳	Out0_Configure	%QB21	BYTE

2. Configure Y1 as a comparison output port.

out1:=2;

Application.out1	°)	Out1_Configure	%QB22	BYTE

4.2.4 High-speed I/O mapping table

The device profile Shenzen INVT-AX7X-CPU_x.x.x.devdes is a CPU device profile that contains description of the high speed counting function, which is used for functional configuration of the input and output ports as well as the use and configuration of the interrupt function. See the following table.

Serial No.	Variable	Input/out put type	Data type	Meaning
1	Gpi_Value	IN	Word	16-Channel general input feedback
2	Version_FPGA	IN	BYTE	FPGA version number. bit6–bit7: major version. bit3–bit5: minor version. bit0-bit2: revision number.
3	In0_Configure	IN	BYTE	Input terminal function configuration
4	In1_Configure	IN	BYTE	0: Standard input function
5	In2_Configure	IN	BYTE	1: Counting function
6	In3_Configure	IN	BYTE	2: Trigger, latch and zero-signal function
7	In4_Configure	IN	BYTE	3: Positive and negative limit zero
8	In5_Configure	IN	BYTE	function

Serial No.	Variable	Input/out put type	Data type	Meaning
9	In6_Configure	IN	BYTE	4: Pulse width measurement function
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	Counting function configuration for channel 0 (bit0-bit3), channel 1(bit4-bit7): 0: Single pulse 1: Quadrature encoder pulses (QEP) 2: Timing 3: SIGN+PULS
20	XMode_SetB	OUT	BYTE	Counting function configuration for channel 2 (bit0-bit3), channel 3(bit4-bit7) 0: Single pulse 1: Quadrature encoder pulses (QEP) 2: Timing 3: SIGN+PULS
21	XMode_SetC	OUT	BYTE	Counting function configuration for channel 4 (bit0-bit3), channel 5(bit4-bit7) 0: Single pulse 1: Quadrature encoder pulses (QEP) 2: Timing 3: SIGN+PULS
22	XMode_SetD	OUT	BYTE	Counting function configuration for channel 6 (bit0-bit3), channel 7(bit4-bit7) 0: Single pulse 1: Quadrature encoder pulses (QEP) 2: Timing 3: SIGN+PULS
23	Filt_Set	OUT	BYTE	Input signal filter parameter setting (unit: 0.25us)
24	Out0_Configure	OUT	BYTE	
25	Out1_Configure	OUT	BYTE	
26	Out2_Configure	OUT	BYTE	Output terminal function configuration
27	Out3_Configure	OUT	BYTE	U: Common output function
28	Out4_Configure	OUT	BYTE	2: Comparison output function
29	Out5_Configure	OUT	BYTE	3–255: Reserved
30	Out6_Configure	OUT	BYTE	
31	Out7_Configure	OUT	BYTE]

Serial No.	Variable	Input/out put type	Data type	Meaning
32	GPO_Set	OUT	BYTE	Common output signal setting bit0-bit7
33	Run_Enable	OUT	BYTE	 bit0: Output channel 0 (1: enabled, 0: disabled) bit1: Output channel 1 (1: enabled, 0: disabled) bit2: Output channel 2 (1: enabled, 0: disabled) bit3: Output channel 3 (1: enabled, 0: disabled) bit6-bit7: Reserved.
34	YMode_Set	OUT	BYTE	Reserved
35	Interrupt	OUT	BOOL	Global interrupt enable
36	Interrupt_Enable	OUT	DWORD	Interrupt enable bit0: Interrupt 0 enable bit1: Interrupt 1 enable bit19: Interrupt 19 enable
37	Interrupt_Mode	OUT	DWORD	Interrupt mode bit0-bit1: X0 interrupt mode bit2-bit3: X1 interrupt mode bit4-bit5: X2 interrupt mode bit6-bit7: X3 interrupt mode bit8-bit9: X4 interrupt mode bit10-bit11: X5 interrupt mode bit12-bit13: X6 interrupt mode bit14-bit15: X7 interrupt mode bit16-bit17: Probe 0 interrupt mode bit18-bit19: Probe 1 interrupt mode bit20- bit21: Probe 2 interrupt mode bit22-bit23: Probe 3 interrupt mode 0: rise edge 1: fall edge 2: Two edges

The operation interface of Invtmatic Studio is displayed as follows:

Devices	→ # X	👌 HIGH_PULSE_IO 🗙 🔮 Ta	sk_4	🐉 Task_s	5 📄 Zphas	e_Compensate	1	MainTask	😸 Task_	3 V Prop
hsio_demo2000 hovice (INVT AXTV)	-	HIGH_PULSE_IO Parameters	Find		Filter Show a	all			•	Y F
Device (INV AXX)		HIGH_PULSE_IO I/O Mapping	Variable #- * Application.Input_V	Mappi	Channel Gpi_Value	Address %IW0	Type WORD	Unit	Descri	Pro
hIGH_PULSE_IO		Status	Application.version	٩	Version_FPGA	%IB2	BYTE			
SoftMotion General Axis Pool			- V Application.in0	٩	In0_Configure	%Q80	BYTE			
		Information	Application.in1	*	In 1_Configure	%Q81	BYTE			
			- V Application.in2	*	In2_Configure	%Q82	BYTE			
			Application.in3	~ *	In3_Configure	%Q83	BYTE			
			- 🍫 Application.in4	~	In4_Configure	%Q 84	BYTE			
			Application.in5	~	In5_Configure	%Q85	BYTE			
			- 🍫 Application.in6	*	In6_Configure	%QB6	BYTE			
			- 🍫 Application.in7	٩	In7_Configure	%Q87	BYTE			
			Application.in8	٩	In8_Configure	%Q88	BYTE			
			Application.in9	~⊗	In9 Configure	%Q89	BYTE			
				Reset Ma	pping Always	updatevariabl	es Enabl	ed 2 (always	in bus cycle task	:)
			🍫 = Create new variable	°∳ =	Map to existing va	riable				
			Bus Cycle Options Bus cycle task Task		~					

4.2.4.1 General input value

The variable corresponding to the device profile is Gpi_Value with the data type of WORD. This parameter is used when the input signal is set to the common input function. The input signals corresponding to the bits of the variable Gpi_Value are shown in the following table.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XF	XE	XD	хс	ХВ	ХА	X9	X8	Х7	X6	X5	X4	X3	X2	X1	X0

If you need to read a common input signal, you can use either WORD mapping or bit mapping.

In WORD variable mapping mode, 16 input signal values can be read at the same time.

💷 🤎 Application.Input_Value	~⊘	Gpi_Value	%IW0	WORD
-----------------------------	----	-----------	-----------------	------

In Bit mapping mode, one variable can only read one signal value, and the variable type is BOOL.

🚍 ᡟ		Gpi_Value	%IW0	WORD
🍄 Application.Xn0_Bit	~∲	Bit0	%IX0.0	BOOL
🍗		Bit1	%IX0.1	BOOL
🍫		Bit2	%IX0.2	BOOL
🍗		Bit3	%IX0.3	BOOL

4.2.4.2 Version

The variable corresponding to the device profile is Version_FPGA with data type BYTE. It is used to read the FPGA version, where bit6–bit7: major version, bit3–bit5: minor version, bit0-bit2: revision number.

🗝 🤎 Application.version_fpga	~	Version_FPGA	%IB2	BYTE

4.2.4.3 Input terminal function configuration

Configure the function of the input port with data type BYTE. There are 16 input ports that can be configured for 5 functions. Including standard input function, counting function, triggering, latching, and Z-signal function, positive and negative limit zero function, and pulse width measurement function.

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	~ø	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 Counting mode configuration

There are 4 variables to configure the counting mode with the data type BYTE. Each variable can be configured for the counting mode of 2 channels. A total of 8 counter modes can be configured. See the following figure.

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

Use 4 bits to set the counter mode with the following values:

Bit	Counting mode
0	Single pulse
1	Quadrature encoder pulses
2	Timing counting
3	Pulse + direction

Configure the bits of XMode_SetA to set the mode of different counters.

7	6	5	4	3	2	1	0
	Cour	nter 1			Cour	nter 0	

Configure the bits of XMode_SetB to set the mode of different counters.

7	6	5	4	3	2	1	0
	Cour	nter 3			Cour	nter 2	

Configure the bits of XMode_SetC to set the mode of different counters.

7	6	5	4	3	2	1	0
	Cour	nter 5			Cour	nter 4	

Configure the bits of XMode_SetD to set the mode of different counters.

7	6	5	4	3	2	1	0
	Cour	nter 7			Cour	nter 6	

4.2.4.5 Filter parameters

The variable of the corresponding device profile is Filt_Set in 0.25µs, which sets the filter parameters of input and output signals, with the data type BYTE and the maximum filter width 64µs. Adjust this parameter to improve the anti-interfere of the signal.

If the signal interference is strong, set the parameter value larger. If the interference is weak, set it smaller. The filter parameters are usually set to 1/4-1/3 (no more than 1/2) of the reference width which is the smaller one of the high pulse and low pulse width. The upper limit is 64μ s. A parameter value that is too large will filter out the effective pulses, while a value that is too small may not filter out the clutter effectively.

Application.filt set	20	Filt Set	%OB20	BYTE
* hppicedonnine_bee	¥	· mc_bee	102020	0112

4.2.4.6 Output terminal function configuration

Configure the function of the output port with data type BYTE. There are 8 output ports that can be configured for 3 functions. For details, see the output port function description.

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 Common output value

Common means the common function output. The variable corresponding to the device profile is GPO_Set with the data type of BYTE. This parameter is used when the output signal is set to the standard output function. The output signals corresponding to the bits of the variable GPO_Set are shown in the following table.

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

If you need to set a common output signal, you can use either BYTE mapping or bit mapping.

In BYTE variable mapping mode, 8 output signal values can be set at the same time.

Application.OutPut_Byte	°∲	Gpo_Set	%Q829	BYTE

In Bit mapping mode, one variable can only set one signal value, and the variable type is BOOL.

i − *		Gpo_Set	%QB29	BYTE
Application.Yn0_Bit	~	Bit0	%QX29.0	BOOL
* ø		Bit1	%QX29.1	BOOL
* ø		Bit2	%QX29.2	BOOL

4.2.4.8 High-speed pulse output function

The variable corresponding to the device profile is Run_Enable with the data type of BYTE. This parameter is used for channel enable at high speed pulse output. The bits of the variable Run_Enable corresponds to the channel enable, 1 indicates enabled, 0 indicates disabled. The following table shows the correspondence between channels and bits.

7	6	5	4	3	2	1	0
	Rese	erved		Channel 3	Channel 2	Channel 1	Channel 0

4.2.4.9 Global interrupt enable

The variable corresponding to the device profile is Interrupt, which is the master switch that enables all interrupts, with the data type of BOOL. 1 indicates total interrupt enabled and 0 indicates disabled.

Serial No.	Variable	Input/output type	Data type	Meaning	
35	Interrupt	OUT	BOOL	Global interrupt enable	

4.2.4.10 Interrupt enable

The variable corresponding to the device profile is Interrupt_Enable with the data type of DWORD. HSIO supports 20 types of interrupts, including 8 external input interrupts, 8 count-comparison interrupts, and 4 probe interrupts, each of which can be enabled with the bit of Interrupt_Enable. The mapping is shown in the following table.

19 18	8	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Probe interrupt enable				Comparison interrupt enable					External interrupt enable										

Bit0-bit7 corresponds to external interrupt 0-7 respectively.

Bit8-bit15 corresponds to comparison interrupt 0-7 respectively.

Bit16-bit19 corresponds to probe interrupt 0-3 respectively.

4.2.4.11 Interrupt mode

The variable corresponding to the device profile is Interrupt_Mode with the data type of DWORD. Only external interrupts and probe interrupts require an interrupt mode. Each mode consists of 2 bits. The mapping of interrupt modes and bits is shown in the following table.

AX series programmable controller software manual

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Exte	rnal	Exte	ernal	Exte	ernal	Exte	ərnal	Exte	rnal	Exte	rnal	Exte	ernal	Ext	ernal
interru	upt 7	inter	rupt 6	inter	rupt 5	inter	rupt 4	interru	upt 3	interru	upt 2	inter	rupt 1	inter	rrupt 0

23	22	21	20	19	18	17	16	
Probe interrupt 3		Probe inte	errupt 2	Probe inte	errupt 1	Probe interrupt 0		

Use 2 bits to set the interrupt mode with the following values:

Motion mode configuration	Motion mode
0	Rising edge
1	Falling edge
2	Two edges

4.2.5 Interrupt instruction

The HSIO supports 20 types of interrupts, including 8 external input interrupts, 8 count-comparison interrupts and 4 probe interrupts. To use the interrupt function, configure the corresponding IO port function. Then, enable the global interrupt and the required interrupt bits. If an external input interrupt or probe interrupt is used, the interrupt mode must also be set.

4.2.5.1 External interrupt instruction

The corresponding input port numbers for P-type model external interrupts are X0–X7, and the corresponding input port numbers for N-type model external interrupts are A0/B0/A1/B1/X4–X7. Configure these ports as common input ports, set an interrupt mode to enable interrupts, and configure the interrupt task so that the operations can be performed in the interrupt task.

External interrupt configuration

Follow the steps to implement the interrupt function:

Step 1 Set the input terminal as standard input function

For details, see Input terminal function description.

Step 2 Set global interrupt

Set Interrupt to true. See Global interrupt enable in the device profile parameter description.

Serial No.	Variable	Input/output type	Data type	Meaning
35	Interrupt	OUT	BOOL	Global interrupt enable

Step 3 Set input port interrupt

Set the 8 input port bits of the Interrupt_Enable the device profile, with Gpix of input port x set to true. Set a bit to enable the interrupt function mapping to that bit.

🚔 🍢	Interrupt_Enable	%QD9	DWORD
**	Gpi0	%QX36.0	BOOL
* ø	Gpi 1	%QX36.1	BOOL
* @	Gpi2	%QX36.2	BOOL
* ø	Gpi3	%QX36.3	BOOL
* ø	Gpi4	%QX36.4	BOOL
* ø	Gpi5	%QX36.5	BOOL
* @	Gpi6	%QX36.6	BOOL
···· * ø	Gpi7	%QX36.7	BOOL

Step 4 Set interrupt mode

The interrupt mode setting consists of 2 bits, and different interrupts correspond to different bits. For details, see Interrupt mode in the device profile parameter description.

Step 5 Select interrupt task

In the Invtmatic Studio task, set the type to **External**, and select the event inxInterrupt of the input port X0–X7, where x ranges from 0 to 7.

畳 💧 MainTask 🗙 🐶 Y	। ।∎∎ FD	H MChome	1
Configuration			
Priority (031): 2			
Туре			
🎸 External 🗸 🗸	External event	in0Interrupt	
		in0Interrupt	
Watchdog		in 1Interrupt in 2Interrupt	
Enable		in3Interrupt in4Interrupt	
Time (e.g. t#200ms)		in5Interrupt in6Interrupt	
		in7Interrupt	

An external signal generates an interrupt based on the interrupt mode and calls the corresponding task execution.

External interrupt timing



Figure 4-1 External input interrupt timing

GPIx represents the xth external general input channel where 0 = < x <= 7, and Interrupt[] is the interrupt state output of GPIx. The high-level pulse output by Interrupt[] uses a dotted line to indicate that interrupts can be output only if the interrupt mode is valid and the interrupt enable is valid. The upper computer interrupt process and the interrupt_clean[] signal only appear after the output of the Interrupt[], so they are also presented as dotted lines. Interrupt_clean[] is the clear signal given by the upper computer in response to the Interrupt[], which clears the Interrupt[] to zero.

4.2.5.2 Probe interrupt instruction

The corresponding input port numbers for P-type model probe interrupts are X8–XB (i.e. CxT, 0 = < x <= 3), and the corresponding input port numbers for N-type model probe interrupts are X8–X11. The input port signal function should be configured as a latching function.

Probe interrupt wiring

External wiring	Port	Function	CN5 terr	ninal No.	Function	Port	External wiring
_# <u>1-</u>	SS1	Input common port	22	21	Input common port	SS2	<u> </u>
							ŦŦ
⊢∕	СОТ	Probe signal input	18	17	Probe signal input	C1T	
	C2T	Probe signal input	16	15	Probe signal input	СЗТ	

Probe interrupt configuration

Follow the steps to implement the interrupt function:

Step 1 Set the input terminal as latching function.

For details, see Input terminal function description.

Step 2 Set global interrupt.

Set Interrupt to true, see Global interrupt enable in the device profile parameter description.

Serial No.	Variable	Input/output type	Data type	Meaning
35	Interrupt	OUT	BOOL	Global interrupt enable

Step 3 Set input port interrupt.

Set the 4 input port bits of the Interrupt_Enable the device profile, with Trigx of input port x set to true. Set a bit to enable the interrupt function mapping to that bit.

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

Step 4 Set interrupt mode.

The interrupt mode setting consists of 2 bits, and different interrupts correspond to different bits. For details, see Interrupt mode in the device profile parameter description.

Step 5 Select interrupt task.

In the Invtmatic Studio task, set the type to **External**, and select the event prbxInterrupt of the input port X8–XB (X8–X11), where x ranges from 0 to 3. Read the probe latching value in the *LatchValue_HP function block* via the interrupt task flag.

PulseCounter 🛛 🕸 MainTask	🗙 🎯 GVL_Param
Configuration	
Priority (031):	
Type	
	- OT- to
External V External event:	Invinterrupt
	in0Interrupt
Wetch de e	in 1Interrupt
watchdog	in2Interrupt
Enable	in3Interrupt
	in Sinterrunt
Time (e.g. t#200ms):	in6Interrupt
	in7Interrupt
Sensitivity: 1	cmp0Interrupt
Sensitivity.	cmp 1Interrupt
	cmp2Interrupt
	cmp3Interrupt
	cmp4Interrupt
🕂 🕂 Add Call 🔀 Remove Call 📝 Change	cmp5Interrupt
	cmp6interrupt
POU	orb0Interrupt
- •	prb1Interrupt
PulseCounter	prb2Interrupt
	prb3Interrupt

An external signal generates an interrupt based on the interrupt mode and calls the corresponding task execution.

Probe interrupt timing



Figure 4-2 Probe input interrupt timing

CxT represents the xth probe input channel where 0=<x<=3, and Interrupt[] is the interrupt state output of CxT. The high-level pulse output by Interrupt[] uses a dotted line to indicate that interrupts can be output only if the interrupt mode is valid and the interrupt enable is valid. The upper computer interrupt process and the interrupt_clean[] signal only appear after the output of the Interrupt[], so they are also presented as dotted lines. Interrupt_clean[] is the clear signal given by the upper computer in response to the Interrupt[], which clears the Interrupt[] to zero.

4.2.5.3 Comparison interrupt instruction

Comparison interrupt includes single-value comparison interrupt and multi-value comparison interrupt. Single-value comparison interrupt is generated by calling the function block CompareSingleValue_HP, and multi-value comparison interrupt is generated by calling CompareMoreValue_HP. The following steps describe the generation of single-value interrupt and multi-value interrupt respectively.

Comparison interrupt configuration

- Single-value comparison interrupt:
 - 1: Set the input terminal as counting function.
 - For details, see Input terminal function description.
 - 2: Set global interrupt.

Set Interrupt to true, see Global interrupt enable in the device profile parameter description.

Serial No.	Variable	Input/output type	Data type	Meaning
35	Interrupt	OUT	BOOL	Global interrupt enable

3: Set input port interrupt.

Set the 8 input port bits of the Interrupt_Enable the device profile, with Compx of input port x set to true. Set a bit to enable the interrupt function mapping to that bit.

Variable	Mapping	Channel	Address	Туре
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: Set the comparison interrupt output.

If comparison interrupt output is not needed, skip this step.

Select the port to be output, set the corresponding port in the device profile as the comparison output function, and select any one of the following 8 channels through the single-value comparison function block CompareSingleValue_HP parameter OutChannel. The OutChanne value ranges from 0 to 7. One output channel OutChannel value can only correspond to one CMP channel.

Output terminal	Standard output function	High-speed pulse output function	Comparison output function
XO	General		CMDO
YU	Common 0	CHOCW/POLSO	CIMPU
Y1	Common 1	CH0CCW/SIGN0	CMP1
Y2	Common 2	CH1CW/PULS1	CMP2
Y3	Common 3	CH1CCW/SIGN1	CMP3
Y4	Common 4	CH2CW/PULS2	CMP4
Y5	Common 5	CH2CCW/SIGN2	CMP5
Y6	Common 6	CH3CW/PULS3	CMP6
Y7	Common 7	CH3CCW/SIGN3	CMP7

5: Select interrupt task.

In the Invtmatic Studio task, set the type to External, and select cmpxInterrupt, where x ranges from 0 to 7.

PulseCounter	🔹 🖄 MainTask	🗙 🧭 GVL_Param
onfiguration		
Priority (031):		
Туре		
External	 External event: 	cmp0Interrupt
		in0Interrupt
		in 1Interrupt
Watchdog		in2Interrupt
Enable		in3Interrupt
		in4Interrupt
Time (e.g. t#200ms):		in6Interrunt
inite (eight: 200110)		in7Interrupt
Consitivity	1	cmp0Interrupt
Sensitivity:	1	cmp1Interrupt
		cmp2Interrupt
		cmp3Interrupt
		cmp4Interrupt
🕂 Add Call 🔀 Ren	nove Call 🛛 🗹 Change	cmp5Interrupt
		cmp6interrupt
DOLL		chip/interrupt

If the comparison value is equal, an interrupt is generated and the corresponding task execution is called. The channel x corresponds to the cmpxInterrupt comparison interrupt task and cannot be modified at will.

6: Call function block to generate interrupt

Single-value comparison calls the function block CompareSingleValue_HP to generate an interrupt. Setting the comparison value to be the same as the count value can also generate an interrupt output.

• Multi-value comparison interrupt:

1: Set the input terminal as counting function

For details, see Input terminal function description.

2: Set global interrupt

Set Interrupt to true, see Global interrupt enable in the device profile parameter description.

Serial No.	Variable	Input/output type	Data type	Meaning
35	Interrupt	OUT	BOOL	Global interrupt enable

3: Set input port interrupt

Set the 8 port bits of the Interrupt_Enable the device profile, with Compx of port x set to true. Since a multi-value comparison function block can be used to generate multiple interrupts, the first value is the enable bit of Cmp0 interrupt, the second value is the enable bit of Cmp1 interrupt, and so on, and the eighth value is the enable bit of Cmp7 interrupt. It cannot be modified arbitrarily.

Variable	Mapping	Channel	Address	Туре
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: Select interrupt task

In the Invtmatic Studio task, set the type to External, and select cmpxInterrupt, where x ranges from 0 to 7.

PulseCounter	🌑 🍪 MainTask	🗙 🧭 GVL_Param
Lonnguration		
Priority (031): 1		
-		
Type		
External	 External event: 	cmp0Interrupt
		in0Interrupt
		in 1Interrupt
Watchdog		in2Interrupt
—		in3Interrupt
Enable		in4Interrupt
- () ()		in5Interrupt
Time (e.g. t#200ms):		in6Interrupt
		in/Interrupt
Sensitivity:	1	cmp0Interrupt
		cmp2Interrupt
		cmp2Interrupt
		cmp4Interrupt
- Add Call X Page	ove Cell 🗖 Change	cmp5Interrupt
	iove call 🖉 change	cmp6Interrupt
		cmp7Interrupt

The multi-value comparison function block has multiple comparison values, each of which corresponds to an interrupt enabled bit of Compx. It shares a one-to-one mapping with the interrupt task cmpxInterrupt where x ranges from 0 to 7 and cannot be modified at will.

5: Call function block to generate interrupt

Multi-value comparison calls the function block CompareMoreValue_HP to generate an interrupt. Setting the comparison value to be the same as the count value will generate an interrupt output. For now, only eight comparison values are supported for multi-value comparisons to generate interrupts, that is, the first eight values of a multi-value comparison can generate interrupts.

Comparison interrupt timing

• Single-value comparison interrupt



Figure 4-3 Single-value comparison interrupt timing

Cnt[x]CvEqPv represents the single-value comparison signal of the xth counting channel, in which 0 = < x <= 7. A high pulse indicates that cv and pv are equal. Interrupt[] is the interrupt state output corresponding to Cnt[x]CvEqPv. The high-level pulse output by Interrupt[] uses a dotted line to indicate that interrupts can be output if the interrupt enable is valid. The upper computer interrupt process and the interrupt_clean[] signal only appear after the output of the Interrupt[], so they are also presented as dotted lines. Interrupt_clean[] is the clear signal given by the upper computer in response to the Interrupt[], which clears the Interrupt[] to zero.

• Multi-value comparison interrupt



Figure 4-4 Multi-value comparison interrupt timing

Cnt[x]CvEqPv[y] represents the yth comparison value signal of the xth counting channel, in which 0=<x<=3 and 0=<y<=7. A high pulse indicates that cv and pv are equal. Interrupt[] is the interrupt state output corresponding to Cnt[x]CvEqPv[y]. The high-level pulse output by Interrupt[] uses a dotted line to indicate that interrupts can be output if the interrupt enable is valid. The upper computer interrupt process and the interrupt_clean[] signal only appear after the output of the Interrupt[], so they are also presented as dotted lines. Interrupt_clean[] is the clear signal given by the upper computer in response to the Interrupt[], which clears the Interrupt[] to zero.

In the single-value comparison interrupt, each counting channel has only one interrupt signal output, and all counting channels (0-7) can output single-value comparison interrupt signals. In the multi-value comparison interrupts, only counting channels 0-3 can output multi-value interrupts, and each counter can output 8 (0-7) interrupt signals. When a multi-value counting channel is selected, its yth comparison value corresponds to the interrupt signal one by one. Only one counting channel is valid at a time for the multi-value comparison interrupt.

4.3 Digital input/output module

4.3.1 Creating a project for digital input/output module

Create a digital I/O application. Add the device profile AX_EM_1600D_x.x.x.x.devdesc.xml, AX_EM_0016DP_x.x.x.x.devdesc.xml, and AX_EM_0016DP_x.x.x.x.devdesc.xml required by the module.

PCI-Bus IEC Objects	Find	F	ilter Show all				-
Internal Parameters	Variable	Mapping	Channel	Address	Туре	Unit	Descriptio
	📮 · ᡟ		IB1	%IB48	BYTE		
Internal I/O Mapping	* >		Bit0	%IX48.0	BOOL		
	🍫		Bit1	%IX48.1	BOOL		
Status	🍫		Bit2	%IX48.2	BOOL		
nformation	🍫		Bit3	%IX48.3	BOOL		
			Bit4	%IX48.4	BOOL		
	🍫		Bit5	%IX48.5	BOOL		
			Bit6	%IX48.6	BOOL		
	*		Bit7	%IX48.7	BOOL		
	😑 ᡟ		IB2	%IB49	BYTE		
	🍫		Bit0	%IX49.0	BOOL		
	*		Bit1	%IX49.1	BOOL		
	🍫		Bit2	%IX49.2	BOOL		
	* >		Bit3	%IX49.3	BOOL		
	🍬		Bit4	%IX49.4	BOOL		
	*		Bit5	%IX49.5	BOOL		
	🍫		Bit6	%IX49.6	BOOL		
	* ø		Bit7	%IX49.7	BOOL		
	L		Version FPGA	%IB50	BYTE		

4.3.2 Variable definition and use

Figure 4-5 Variable mapping of input module

The IB1/IB2 input point status can be obtained by BYTE or BOOL type.

PCI-Bus IEC Objects	Find		Filter Show all				•
Internal Parameters	Variable	Mapping	Channel	Address	Туре	Unit	Description
	📮 🍫 📃		QB1	%QB88	BYTE		
Internal I/O Mapping	* ø		Bit0	%QX88.0	BOOL		
	* ø		Bit1	%QX88.1	BOOL		
Status	*>		Bit2	%QX88.2	BOOL		
Information	- *		Bit3	%QX88.3	BOOL		
	* @		Bit4	%QX88.4	BOOL		
	* ø		Bit5	%QX88.5	BOOL		
	*		Bit6	%QX88.6	BOOL		
	* @		Bit7	%QX88.7	BOOL		
	🚍 🍫		QB2	%QB89	BYTE		
	* @		Bit0	%QX89.0	BOOL		
	···· **		Bit1	%QX89.1	BOOL		
	* ø		Bit2	%QX89.2	BOOL		
	*>		Bit3	%QX89.3	BOOL		
	🍫		Bit4	%QX89.4	BOOL		
	*>		Bit5	%QX89.5	BOOL		
	* @		Bit6	%QX89.6	BOOL		
			Bit7	%QX89.7	BOOL		
AX_EM_0016DN X	1		Version_FPGA	%IB51	BYTE		
AX_EM_0016DN X	Find		Version_FPGA	%IB51	BYTE		•
AX_EM_0016DN X	Find Variable	Mapping	Version_FPGA Filter Show all Channel	%IB51	Туре	Unit	▼ Description
AX_EM_0016DN X	Find Variable E- *p	Mapping	Filter Show all Channel QB1	%IB51 Address %QB90	Туре ВУТЕ	Unit	• Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping	Find Variable	Mapping	Filter Channel QB1 Bit0	%IB51 Address %QB90 %QX90.0	Type BYTE BOOL	Unit	• Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping	Find Variable	Mapping	Filter Show all Channel QB1 Bit0 Bit1	%IB51 Address %QB90 %QX90.0 %QX90.1	BYTE Type BYTE BOOL BOOL	Unit	• Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status	Find Variable Frid	Mapping	Version_FPGA Filter Show all Channel QB1 Bit0 Bit1 Bit2	%IB51 Address %QB90 %QX90.0 %QX90.1 %QX90.1	BYTE Type BYTE BOOL BOOL BOOL	Unit	Descriptio
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status	Find Variable P. ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	Mapping	Version_FPGA Filter Show all Channel QB1 Bit0 Bit1 Bit2 Bit3	%IB51 Address %QB90 %QX90.0 %QX90.1 %QX90.2 %QX90.3	BYTE Type BYTE BOOL BOOL BOOL BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable P ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	Mapping	Version_FPGA Filter Show all Channel QB1 Bit0 Bit2 Bit2 Bit3 Bit4	%IB51 Address %QB90 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.4	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable □ - ⁷ % □ - ⁷ %	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5	%IB51 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.4 %QX90.4	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable ⇒ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	Mapping	Version_FPGA Filter Show all Channel Q81 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6	%IB51 Address %QB90 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.3 %QX90.5 %QX90.5	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable □ - ^ 0 □ ^ 0 □ - ^ 0 □ -	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit6 Bit7	%(IBS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.4 %QX90.5 %QX90.5 %QX90.5	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable □ - *• □ - *•	Mapping	Version_FPGA Filter Show all Channel QB1 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 QB2	%(BS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.4 %QX90.5 %QX90.7 %QX90.7 %QX90.7	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable ⇒ * • - * •	Mapping	Version_FPGA Filter Show all Channel QB1 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 QB2 Bit0	%(IBS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.4 %QX90.5 %QX90.6 %QX90.7 %QX90.7 %QX91.0	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable P ~ 0 C ~	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Q82 Bit0 Bit1	%(IBS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.4 %QX90.5 %QX90.6 %QX90.6 %QX90.7 %QX91.0 %QX91.1	Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BYTE BOOL BOOL BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable P ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Q82 Bit0 Bit1 Bit0 Bit1 Bit2	%(BS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.4 %QX90.5 %QX90.6 %QX90.6 %QX90.7 %QX91.0 %QX91.1 %QX91.2	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable Find Find Find Find Find Find Find Find	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Q82 Bit0 Bit1 Bit2 Bit2 Bit1 Bit2 Bit3	%4B51 Address %42890 %42X90.0 %42X90.1 %42X90.2 %42X90.3 %42X90.4 %42X90.5 %42X90.6 %42X90.6 %42X91.0 %42X91.1 %42X91.1	Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable 	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Q82 Bit0 Bit1 Bit2 Bit0 Bit1 Bit2 Bit1 Bit2 Bit3 Bit4	%4B51 Address %42890 %4290.0 %4290.1 %4290.2 %4290.4 %4290.4 %4290.4 %4290.4 %4290.4 %4290.4 %4291.0 %4291.1 %4291.1 %4291.2 %4291.4	BYTE Type BYTE BOOL BOOL BOOL BOOL BOOL BOOL BOOL BOO	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal Parameters Internal I/O Mapping Status Information	Find Variable	Mapping	Version_FPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Q82 Bit6 Bit7 Q82 Bit0 Bit1 Bit2 Bit3 Bit4 Bit3 Bit4 Bit3 Bit4 Bit5 Bit4 Bit5 Bit4 Bit5 Bit4 Bit5 Bit4 Bit5 Bit4 Bit5 Bit5 Bit4 Bit5 Bit5 Bit5 Bit6 Bit7 Bit7 Bit7 Bit7 Bit8 Bit8 Bit8 Bit8 Bit8 Bit8 Bit8 Bit8	%IBS1 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.5 %QX90.6 %QX90.6 %QX91.0 %QX91.1 %QX91.1 %QX91.1 %QX91.2 %QX91.4 %QX91.3	BYTE Type BYTE BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal I/O Mapping Status Information	Find Variable □ - *• □ - *•	Mapping	Version_FPGA Filter Show all Channel Q81 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit5 Bit6 Bit7 Q82 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit5 Bit5 Bit5 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit6 Bit7 Q82 Bit0 Bit1 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit6 Bit5 Bit6 Bit7 Q82 Bit6 Bit7 Bit7 Bit7 Bit6 Bit7 Bit7 Bit7 Bit7 Bit7 Bit6 Bit7	%dB51 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.4 %QX90.4 %QX90.5 %QX90.6 %QX90.7 %QX91.0 %QX91.1 %QX91.2 %QX91.3 %QX91.4 %QX91.4 %QX91.6	BYTE Type BYTE BOOL	Unit	Description
AX_EM_0016DN X PCI-Bus IEC Objects Internal I/O Mapping Status Information	Find Variable Find Variable Find Variable Find Variable Find Variable Find Find Variable Find	Mapping	Version_PPGA Filter Show all Channel Q81 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit7 Q82 Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5 Bit5	%4B51 Address %Q890 %QX90.0 %QX90.1 %QX90.2 %QX90.3 %QX90.4 %QX90.5 %QX90.5 %QX90.5 %QX91.0 %QX91.0 %QX91.1 %QX91.1 %QX91.1 %QX91.4 %QX91.5 %QX91.7	ВУТЕ Туре ВУТЕ ВООL		Description

Figure 4-6 Variable mapping of output module

The QB1/QB2 output point status can be controlled by BYTE or BOOL type.

4.4 Analog input/output module

4.4.1 Creating a project for analog input/output module

Create an analog I/O application project. Add the device profile **AX_EM_4AD_x.x.x.devdesc.xml** and **AX_EM_4DA_x.x.x.devdesc.xml** required by the module.

4.4.2 Variable definition and use

PCI-Bus IEC Objects	Find		Filter Show all				-
Internal Parameters	Variable	Mapping	Channel CH0	Address %0W59	Type UINT	Unit	Description
Internal I/O Mapping			CH1	%QW60	UINT		
	😟 🍢		CH2	%QW61	UINT		
Status	±		CH3	%QW62	UINT		
Information	🍫		FP	%QW63	UINT		
	*>		FP0	%QB128	USINT		
	🍫		FP1	%QB129	USINT		
	*		FP2	%QB130	USINT		
	*>		FP3	%QB131	USINT		
	¥ø		IN0	%IW28	INT		
	🍫		IN1	%IW29	INT		
	* >		IN2	%IW30	INT		
	🍗		IN3	%IW31	INT		
	* >		Version_FPGA	%IW32	INT		
	i 🗤		Version_MCU	%IW33	INT		

Figure 4-7 Variable mapping of analog input module

Variable	Input/output type	Data type	Meaning
CH0~CH3	IN	UINT	Control word of sampling channel 0–3. See the module user manual for specific meaning.
FP	IN	UINT	AD sampling chip filter selection. See the module user manual for specific meaning.
FP0~FP3	IN	USINT	Filter parameter configuration of sampling channel 0–3. See the module user manual for specific meaning.
IN0~IN3	OUT	INT	Analog code value of sampling channel 0–3
Version_FPGA	OUT	INT	FPGA version number, converted to octal number
Version_MCU	OUT	INT	MCU version number, converted to octal number

PCI-Bus IEC Objects	Find		Filter Show all				•
Internal Parameters	Variable	Mapping	Channel	Address	Туре	Unit	Description
	I		Configuration_CH0	%QW47	INT		
Internal I/O Mapping	····· **		Data_CH0	%QW48	INT		
	*		Data_Default0	%QW49	INT		
Status	۰۰۰ ۲۵		Configuration_CH1	%QW50	INT		
Information	*		Data_CH1	%QW51	INT		
	*		Data_Default1	%QW52	INT		
	🗎 🍢		Configuration_CH2	%QW53	INT		
			Data_CH2	%QW54	INT		
	* ø		Data_Default2	%QW55	INT		
	🗄 🍢		Configuration_CH3	%QW56	INT		
	* @		Data_CH3	%QW57	INT		
	···· **		Data_Default3	%QW58	INT		
	¥ø		Version_FPGA	%IB54	BYTE		
	- ×		Version MCU	%IB55	BYTE		

Figure 4-8 Variable mapping of analog output module

Variable	Input/output type	Data type	Meaning
Configuration_CH0~3	IN	INT	Control word of output channel 0–3. See the module user manual for specific meaning.
Data_CH0~3	IN	INT	Output analog code value of output channel 0-3
Data_Default0~3	IN	INT	Output analog preset code value of output

Variable	Input/output type	Data type	Meaning				
			channel 0–3				
Version_FPGA	OUT	BYTE	FPGA version number, converted to octal number				
Version_MCU	OUT	BYTE	MCU version number, converted to octal number				

4.5 Temperature module

4.5.1 Creating a project for temperature module

Create a temperature module application. Add the device profile **AX_EM_4PTC_.x.x.x.devdesc.xml** required by the module.

4.5.2 Variable definition and use

CI-Bus IEC Objects	Find	Filter	r Show all			-	🕂 Add FB for IO C	hannel
ternal Parameters	Variable	Mapping	Channel	Address	Туре	Unit	Description	
ternal I/O Mapping			Breakun0	%IB72	BYTE			
			Overrun0	%IB73	BYTE			
atus			Temperature 1	%ID19	REAL			
			Breakun 1	%IB80	BYTE			
ormation			Overrun1	%IB81	BYTE			
			Temperature2	%ID21	REAL			
	×		Breakup2	%IB88	BYTE			
			Overrun2	%IB89	BYTE			
			Temperature3	%ID23	REAL			
	*		Breakup3	%IB96	BYTE			
			Overrun3	%IB97	BYTE			
			Version_FPGA	%IB98	BYTE			
			Version MCU	%IB99	BYTE			
	🍫		In_CJC	%ID25	REAL			
			Out_CJC	%ID26	REAL			
	50		Basic_Set_0	%QB132	BYTE			
	N		Sampling_Period_0	%QB133	BYTE			
	- **		Sensor_Type_0	%QB134	BYTE			
	* *		Filtering_Time_0	%QB135	BYTE			
	- 50		Upper_Value_0	%QW68	INT			
	50		Lower_Value_0	%QW69	INT			
	* >		Basic_Set_1	%QB140	BYTE			
			Sampling_Period_1	%QB141	BYTE			
	🍫		Sensor_Type_1	%QB142	BYTE			
	🍫		Filtering_Time_1	%QB143	BYTE			
	🍫		Upper_Value_1	%QW72	INT			
	* >		Lower_Value_1	%QW73	INT			
	* *		Basic_Set_2	%QB148	BYTE			
	* ø		Sampling_Period_2	%QB149	BYTE			
	* >		Sensor_Type_2	%QB150	BYTE			
	* *		Filtering_Time_2	%QB151	BYTE			
	* >		Upper_Value_2	%QW76	INT			
	* >		Lower_Value_2	%QW77	INT			
	* >		Basic_Set_3	%QB156	BYTE			
			Sampling_Period_3	%QB157	BYTE			
	···· 🍫		Sensor_Type_3	%QB158	BYTE			
	` >		Filtering_Time_3	%QB159	BYTE			
			Upper Value 3	%OW/80	INT			

Figure 4-9 Variable mapping of temperature module

Variable	Input/output type	Data type	Meaning
Basic_Set_0~3	IN	BYTE	Control word of temperature sampling channel 0–3. See the module user manual for specific meaning.
Sampling_period_0~3	IN	BYTE	Sampling period of temperature sampling channel 0– 3 (reserved)
Sensor_type_0~3	IN	BYTE	Sensor type of temperature sampling channel 0–3
Filtering_time_0~3	IN	BYTE	Sampling filter configuration of temperature sampling channel 0–3 (reserved)
Upper_value_0~3	IN	INT	Temperature upper limit of temperature sampling channel 0–3
Variable	Input/output type	Data type	Meaning
-----------------	----------------------	---	--
Lower_value_0~3	IN	INT	Temperature lower limit of temperature sampling channel 0–3
Temperature0~3	OUT	REAL	Temperature sampling value of temperature sampling channel 0–3
Breakup0~3	OUT	BYTE	Disconnection flag of temperature sampling channel 0-3 (reserved)
Overrun0~3	OUT	BYTE	Over-limit flag of temperature sampling channel 0–3
In_CJC	OUT	REAL	Internal cold junction temperature
Out_CJC	OUT	REAL External cold junction temperature	
Version_FPGA	OUT	BYTE	FPGA version number, converted to octal number
Version_MCU	OUT	BYTE	MCU version number, converted to octal number

4.6 Communication module

The EtherCAT communication module is used as an EtherCAT slave. Before using the module, add the device profile **INVT_ECAT_SLAVE_Vx.xx.xml**. For detailed instructions, refer to the case of adding DA200 servo drive to the EtherCAT master node.

1. Create a new project in the Invtmatic Studio upper computer, Right click **Device** to add a device, and add an EtherCAT Master SoftMotion module, as shown in the following figure:



Figure 4-10 Add an EtherCAT Master SoftMotion module

2. Right click the EtherCAT Master SoftMotion module to add a device, and add the EtherCAT Slave Module (AX-EM-RCM-ET), as shown in the following figure:

File Edit View Project Build C	Add Device			×
	Name AX_EM_ECM_ET			
levices	Action Append device Insert device	O Plug device O Update device		
Unotied2	String for a fulltext search	Vendor <all vendors=""></all>		~
Borkes (NOT AX7) BY RCLope Cope Appleation Borker (Nonager Borker (Nonager Borker (Nonager Borker (Nonager) Borker (Nonager) Borker (Nonager) Borker (Nonager) Borker (Nonager) Borker (Nonager) Softwaten General Ass Pool	Name	c Serva Drives Ketoron: Ether CAT Devices AE larer Module on, Applances Company - AC Serva Driver darer from the Mat	Vendor per	*
: Ous :- @r Project Settings	Name:EtherCAT Size holds Vendor: INVT Vendor: INVT Vendor: INVT Vendor: Size Vension: Revision + 8540000:10 Order: EtherCAT Size in Description: EtherCAT Size TherCAT_size, Softholion Original selected device as last child EtherCAT_size, Softholion Original select another target no	0 ported from Silve XM1_ECAT_SLANE_FOR 6 of de in the nevigator while this window is open.	_CCCEDIS_VILOR.vml Device: EtherCAT Slave Module	×.

Figure 4-11 Add a EtherCAT remote expansion module

The following section explains how to use the EtherCAT remote expansion module to extend our existing IO.

4.6.1 Digital input module

EtherCAT remote extension module (AX-EM-Rcm-ET) is used to extend the digital input module (AX-EM-1600D) through the backplane. The instructions are as follows:

1. Right click **AX-EM-ECM-ET** in the device panel to add the digital input module (AX_EM_1600D). Control 16 channels through two sets of variables InByte0 and InByte1 in the Module/IO mapping tab, as shown in the following figure.



Figure 4-12 Variable mapping of digital input module

2. After compiling, log in to download the project and run it.

4.6.2 Digital output module

EtherCAT remote extension module (AX-EM-Rcm-ET) is used to extend the digital output module (AX-EM-0016DP/ AX-EM-0016DN) through the backplane. The instructions are as follows:

1. Right click **AX-EM-ECM-ET** in the device panel to add the digital output module (AX_EM_0016DP). Control 16 channels through two sets of variables OutByte0 and OutByte1 in the Module/IO mapping tab, as shown in the following figure.

Ratup Parameters Ratup Parame	🔄 Untitled2		et- 4		films of				
Marketion	- m Device (INVT AX7X)	Startup Parameters	Find		Filter Sho	wall			•
Opplication Outly web %644 USRT USRT USRT USRT </td <td>🖙 🔝 PLC Logic</td> <td>Module I/O Mapping</td> <td>Variable</td> <td>Mapping</td> <td>Channel</td> <td>Address</td> <td>Туре</td> <td>Unit</td> <td>Description</td>	🖙 🔝 PLC Logic	Module I/O Mapping	Variable	Mapping	Channel	Address	Туре	Unit	Description
Module EC Objects BR0 %QK4.0 BOOL If Tak Configuration BR1 %QK4.1 BOOL If Direct AT Task BR1 %QK4.4 BOOL If Direct AT Tasker Softwison (Ether CAT Tasker Softwison BR1 %QK4.4 BOOL If AT LEW, DOI ETC Ether CAT Tasker Softwison BR1 %QK4.4.5 BOOL Direct Tasker Softwison If AT LEW, DOI ETC Ether CAT Tasker Softwison BR1 %QK4.5.6 BOOL Direct Tasker Softwison If AT LEW, DOI ETC Ether CAT Tasker Softwison BR1 %QK4.5.5 BOOL Direct Tasker Softwison If AT LEW, DOI ETC Ether CAT Tasker Softwison BR1 %QK4.5.5 BOOL Direc Tasker Softwison If	Application				OutByte0	%QB44	USINT		OutByte0
Implementation Implementation Implementation Implementa	- 💼 Library Manager	Module IEC Objects	- **		Bit0	%QX44.0	BOOL		
Image: Softward Contract Softward Carl Master Softward Carl M	PLC_PRG (PRG)		- **		Bit1	%QX44.1	BOOL		
 	Task Configuration	Information	- **		Bit2	%QX44.2	BOOL		
Image: Section of the sector of the	- 😂 EtherCAT_Task		- **		Bit3	%QX44.3	BOOL		
Image: Part of the control of the	🖹 🍪 MainTask		- **		Bit4	%QX44.4	BOOL		
Inter_USE_ID Bits \$VQ44.6 BOOL Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool Image: SoftWoton General Axis Pool <t< td=""><td>PLC_PRG</td><td></td><td>- **</td><td></td><td>Bit5</td><td>%QX44.5</td><td>BOOL</td><td></td><td></td></t<>	PLC_PRG		- **		Bit5	%QX44.5	BOOL		
Image: SoftWoton General Axis Pool	- & HIGH_PULSE_IO		- **		Bit6	%QX44.6	BOOL		
AX_EM_EON_ET (Exter CAT Save Module) AX_EM_GOUGE (Digital Output 16 Bits) AX_EM_GOUGE (Digital Output	EtherCAT_Master_SoftMotion (EtherCAT Master Soft	*	- *		Bit7	%QX44.7	BOOL		
Image: SoftWattion General Axis Pod 900. 900. SoftWattion General Axis Pod 911. 542(45.1) 800. -1 9 811. 542(45.1) 800. -1 9 813. 542(45.2) 800. -1 9 813. 542(45.2) 800. -1 9 814. 542(45.2) 800. -1 9 814. 542(45.4) 800. -1 9 815. 542(45.6) 800. -1 9 815.5 542(45.6) 800. -1 9 815.5 542(45.6) 800. -1 9 815.5 542(45.6) 800. -1 9 817.5 542(45.6) 800. -1 9 817.5 542(45.7) 800. -1 9 817.7 542(45.7) 800. -1 9 817.7 542(45.7) 800. -1 9 817.7 542(45.7) <t< td=""><td>AX_EM_ECM_ET (EtherCAT Slave Module)</td><td></td><td>.≘- **</td><td></td><td>OutByte1</td><td>%QB45</td><td>USINT</td><td></td><td>OutByte 1</td></t<>	AX_EM_ECM_ET (EtherCAT Slave Module)		.≘- **		OutByte1	%QB45	USINT		OutByte 1
SofMotion General Axis Pool 90 881 19QH45.1 800. -10 882 19QH45.2 800. 1 -10 8813 19QH45.3 800. 1 -10 8814 19QH45.4 800. 1 -10 8815 19QH45.6 800. 1 -10 8816 19QH45.7 800. 1 -10 8817 19QH45.6 800. 1 -10 8817 19QH45.6 800. 1 -10 8817 19QH45.7 800. 1 -10 8816 19QH45.6 800. 1 -10 890 890 1 1 -10 70 8816 19QH45.6 1	AX_EM_0016DP (Digital Output 16 Bits)		- **		Bit0	%QX45.0	BOOL		
** BH2 %QV45.2 BOOL ** BH3 %QV45.3 BOOL ** BH4 %QV45.4 BOOL ** BH5 %QV45.5 BOOL ** BH6 %QV45.6 BOOL ** BH6 %QV45.7 BOOL ** BH7 %QV45.7 BOOL ** BH7 %QV45.7 BOOL	SoftMotion General Axis Pool		- **		Bit1	%QX45.1	BOOL		
Pip Bit3 94Q45.3 BOOL Pip Bit4 94Q45.4 BOOL Pip Bit5 94Q45.6 BOOL Pip Bit6 94Q45.6 BOOL Pip Bit6 94Q45.6 BOOL Pip Bit6 94Q45.7 BOOL Pip Bit7 94Q45.8 BOOL Pip Bit7 94Q45.6 BOOL Pip Bit7 94Q45.7 BOOL Pip CuttlyHeD Reset Mapping Always update vanables Use parent devices Pip Create new variable Pip Mays update vanables Use parent devices			- **		Bit2	%QX45.2	BOOL		
-° Bit4 %QN45.4 BOOL -° Bit5 %QV45.5 BOOL -° Bit6 %QN45.6 BOOL -° Bit7 %QN45.6 BOOL -° Bit7 %QN45.6 BOOL - - Bit7 %QN45.6 BOOL - - Bit7 %QN45.7 BOOL - - - Bit7 %QN45.7 BOOL - - - - - - OutByte0 Reset Mapping Always update variables Use parent device r ° = * * -			- **		Bit3	%QX45.3	BOOL		
P BIS %QV45.5 BOOL P BIG %QV45.6 BOOL BIG %QV45.7 BOOL Outbyte0 Reset Mapping Always update variables Use perent device structure			- **		Bit4	%QX45.4	BOOL		
* Bits %QX45.6 BOOL * Bit2 %QX45.7 BOOL * Bit2 %Q45.7 BOOL OutByte0 Rieset: Mapping Always update variables Use parent device in the second device in t			- **		Bit5	%QX45.5	BOOL		
Top Bit? %QX45.7 BOOL OutByte0 Reset Mapping Always update variables Use parent device in the set of the set			- **		Bit6	%QX45.6	BOOL		
Duttryte0 Reset Mapping Always update variables Use parent devices Vig = Create new variable Vig = Map to existing variable			- * ø		Bit7	%QX45.7	BOOL		
🎼 = Create new variable 🌱 🍅 = Map to existing variable			OutByte0	Reset M	Bit7	%QX45.7 Always update	BOOL	Use pare	nt device settin
			🍫 = Create new varia	ble 🍫 = Ma	p to existing	variable			

Figure 4-13 Variable mapping of digital output module

2. After compiling, log in to download the project and run it.

4.6.3 Analog input module

EtherCAT remote extension module (AX-EM-Rcm-ET) is used to extend the analog input module (AX-EM-4AD) through the backplane. The instructions are as follows:

1. Right click **AX-EM-ECM-ET** in the device panel to add the analog input module (AX_EM_4AD). Control the module through the multiple sets of variables in the Module/IO mapping tab, as shown in the following figure.

Devices - 4 ×	PLC_PRG MAX_EM_	ADA MALEM_AAD X			
Untiled2	Startup Parameters	Find	Filter Show all		
⊨_mi Device (INVT AX7X) ⇒ mi PLC Logic	Module I/O Mapping	Variable Map	ping Channel Addr	ress Type Unit	Description
Application		*	CH0 %QW	/34 INT	CH0
- 📶 Library Manager	Module IEC Objects	₽- * ₽	CH1 %QW	/35 INT	CH1
PLC_PRG (PRG)	Tefermetica	B- *	CH2 %QW	/36 INT	CH2
🖻 🎆 Task Configuration	Information	÷-*•	CH3 %QW	/37 INT	CH3
EtherCAT_Task		÷- **	FP %QW	/38 INT	FP
🖹 🍪 MainTask			INO %IW	6 INT	INO
PLC_PRG		🕫 – 🏘	IN0_Fault_Code %IW	7 INT	IN0_Fault_Code
- A HIGH_PULSE_IO		10-10	IN1 %IW8	8 INT	IN1
EtherCAT_Master_SoftMotion (EtherCAT Master Soft	4	🕀 – 🏘	IN1_Fault_Code %IW9	9 INT	IN1_Fault_Code
AX_EM_ECM_ET (EtherCAT Slave Module)		10 - 10	IN2 %IW	10 INT	IN2
AX_EM_4DA (Analog Output 16 Bits)		18 - Ma	IN2_Fault_Code %IW	11 INT	IN2_Fault_Code
AX_EM_4AD (Analog Input 16 Bits)		10 - 10	IN3 %IW	12 INT	IN3
SoftMotion General Axis Pool		ii - ₩	IN3_Fault_Code %IW	13 INT	IN3_Fault_Code
		Re « Craste new variable » %	set Mapping Always update	variables Use parent	device setting
		•			
< >>	<pre></pre>				>

Figure 4-14 Variable mapping of analog input module

- 2. After compiling, log in to download the project and run it.
- Variable description: the following table uses channel 0 as an example to illustrate the use of all variables for channel 0.

Parameters				Valid bit	Variable name	Variable type
	sinc5+sinc1		00			
	sinc5+sinc1+enhance50/60		01	[4.0]	FP	
Filter	sinc3		10	[1:0]		
	Reserved					
	Enable	Enable	1	[0]		WORD
Channel O	channel 0	Disable	0	[U]		
configuration	Disconnection	Enable	1	[4]	CH0	
	detection	Disable	0	[1]		
	Conversion	0V–5V	000	[4:2]		

Table 4-4 Channel 0 variable description

Parameters				Valid bit	Variable name	Variable type
	mode	0V-10V	001			
		-5–5V	010			
		-10V–10V	011			
		-20mA–20mA	100			
		0mA–20mA	101			
		4mA–20mA	110			
		Enable	1	(5)		
	Over-limit flag	Disable	0	[5]		
	Over range detection	Enable	1	[6]		
	enable bit	Disable	0	[0]		
	Rese	erved		[15:7]		
Channel 0 data	Da	ata		[15:0]	IN0	
Channel 0 fault code (See Table 4-6 for details)	Indicates the information of the	current fault ne module.		[15:0]	IN0_Fault_Code	

Table 4-5 Mapping of rated range and actual input analog value

Туре	Input rated range	Mapped digital value		
Analog voltage input	-10V–10V	-10000-+10000		
	0V-10V	0–10000		
	-5V - +5V	- 5000–+5000		
	0V–5V	0–5000		
Analog current input	-20mA–20mA	-20000–20000		
	0mA–20mA	0–20000		
	4mA–20mA	4000–20000		

Table 4-6 Channel fault code

Channel 0	Meaning
AO	Channel 0 is disconnected.
A1	Channel 0 exceeds the limits (exceeds the range of -25V-+25V)
A2	Channel 0 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)
A3	Channel 0 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)

Channel 1	Meaning
A4	Channel 1 is disconnected.
A5	Channel 1 exceeds the limits (exceeds the range of -25V-+25V)
A6	Channel 1 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)
A7	Channel 1 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)

Channel 2	Meaning
A8	Channel 2 is disconnected.
A9	Channel 2 exceeds the limits (exceeds the range of -25V-+25V)
AA	Channel 2 exceeds the upper limit of the range (exceeds the upper limit of the

Channel 2	Meaning
	currently selected voltage range)
Ab	Channel 2 exceeds the lower limit of the range (exceeds the lower limit of the
	currently selected voltage range)

Channel 3	Meaning
AC	Channel 3 is disconnected.
Ad	Channel 3 exceeds the limits (exceeds the range of -25V-+25V)
AE	Channel 3 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)
AF	Channel 3 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)

4.6.4 Analog output module

EtherCAT remote extension module (AX-EM-Rcm-ET) is used to extend the analog output module (AX-EM-4DA) through the backplane. The instructions are as follows:

1. Right click **AX-EM-ECM-ET** in the device panel to add the analog output module (AX_EM_4DA). Control the module through the multiple sets of variables in the Module/IO mapping tab, as shown in the following figure.

Devices - 7 X	PLC_PRG AX_E	M_4DA X M AX_EM_4AD						
Untitled2	Startup Parameters	Find		Filter Show all				
Device (INVT AX7X)	Startup Parameters			Show an				
= 1 PLC Logic	Module I/O Mapping	Variable	Mapping	Channel	Address	Туре	Unit	Description
Application		B - 🍫		Configuration_CH0	%QW22	INT		Configuration_CH0
Library Manager	Module IEC Objects	8-**		Data_CH0	%QW23	INT		Data_CH0
PLC_PRG (PRG)		B- 🖗		Data_Default0	%QW24	INT		Data_Default0
Task Configuration	Information	B- *		Configuration_CH1	%QW25	INT		Configuration_CH1
- 😂 EtherCAT_Task		8-50		Data_CH1	%QW26	INT		Data_CH1
🖻 🥩 MainTask		B- *		Data_Default1	%QW27	INT		Data_Default1
PLC_PRG		10 - *		Configuration_CH2	%QW28	INT		Configuration_CH2
- A HIGH_PULSE_IO		8-**		Data_CH2	%QW29	INT		Data_CH2
🖻 🚮 EtherCAT_Master_SoftMotion (EtherCAT Master Soft		B- 🍫		Data_Default2	%QW30	INT		Data_Default2
AX_EM_ECM_ET (EtherCAT Slave Module)		8- *		Configuration_CH3	%QW31	INT		Configuration_CH3
AX_EM_4DA (Analog Output 16 Bits)		B- 🍫		Data_CH3	%QW32	INT		Data_CH3
AX_EM_4AD (Analog Input 16 Bits)		B-**		Data_Default3	%QW33	INT		Data_Default3
SoftMotion General Axis Pool		😟 - 🍫		INT0_Fault_Code	%IW2	INT		INT0_Fault_Code
		😟 – 🍫		INT1_Fault_Code	%IW3	INT		INT1_Fault_Code
		😟 - 🍫		INT2_Fault_Code	%IW4	INT		INT2_Fault_Code
		🗷 – 🍫		INT3_Fault_Code	%IW5	INT		INT3_Fault_Code
		K = Create new variat	Reset M	apping Always u p to existing variable	pdatevariable	s Use pa	arent dev	ice setting
< >	<							2

Figure 4-15 Variable mapping of analog output module

- 2. After compiling, log in to download the project and run it.
- 3. Variable description: the following table uses channel 0 as an example to illustrate the use of all variables for channel 0.

Parameters			Value	Valid bit	Variable name		
	Enable	Enable	1	[0]			
Channel 0 configuration	channel 0	Disable	0	[U]			
	Disconnectio n detection	Reserved		[1]	Configuration_CH0		
	Conversion	0V–5V	000	[4.0]			
	mode	0V–10V	001	[4:2]			

Table 4-7 Channel 0 variable description

Parameters			Value	Valid bit	Variable name
		-5V–5V	010		
		-10V–10V	011		
		4mA–20mA	100		
		0mA–20mA	101		
		Clear output	00		
	Output status	Keep output	01	[6,6]	
	after stop	Output preset value	10	[6:0]	
	Res	erved		[15:7]	
Channel 0 code value	D	lata		[15:0]	Data_CH0
Channel 0 output preset value	Output preset value			[15:0]	Data_Default0
Channel 0 fault code (See Table 4-9 for details)	Indicates th information		[15:0]	INT0_Fault_Code	

Table 4-8	Mapping	of rated	range and	actual	input	analog	value
						a	

Туре	Input rated range	Mapped digital value			
	-10V–10V	-10000-+10000			
Analog voltage output	0V-10V	0–10000			
	-5V–5V	-5000-+5000			
	0V–5V	0–5000			
	4mA–20mA	4000–20000			
Analog current output	0mA–20mA	0–20000			

Table 4-9 Channel fault code

Channel 0	Meaning					
B0	The current output of channel 0 is disconnected.					
B1	The voltage output of channel 0 is short-circuited.					

Channel 1	Meaning
B2	The current output of channel 1 is disconnected.
B3	The voltage output of channel 1 is short-circuited.

Channel 2	Meaning
B4	The current output of channel 2 is disconnected.
B5	The voltage output of channel 2 is short-circuited.

Channel 3	Meaning
B6	The current output of channel 3 is disconnected.
B7	The voltage output of channel 3 is short-circuited.

Output module power failure	Meaning
B8	The 24V power board of the output module is disconnected.

4.6.5 Temperature module

EtherCAT remote extension module (AX-EM-Rcm-ET) is used to extend the temperature module (AX-EM-4PTC) through the backplane. The instructions are as follows:

1. Right click **AX-EM-ECM-ET** in the device panel to add the temperature module (AX_EM_4PTC). Control the module through the multiple sets of variables in the Module/IO mapping tab, as shown in the following figure.

Untitled2	Startup Parameters	Find		Filter Show all				
E-ff Device (INVT AX7X)	Scarcup Parameters			Show an				
🖻 🛄 PLC Logic	Module I/O Mapping	Variable	Mapping	Channel	Address	Туре	Unit	Description
Application		· · · · ·		Config_Word0	%QW22	INT		Config_Word0
- 💼 Library Manager	Module IEC Objects	÷-*•		Config_Word1	%QW23	INT		Config_Word1
PLC_PRG (PRG)		· · · · · ·		Config_Word2	%QW24	INT		Config_Word2
🖹 🧱 Task Configuration	Information	÷-*•		Config_Word3	%QW25	INT		Config_Word3
- 🍪 EtherCAT_Task		÷-*•		Config_Word4	%QW26	INT		Config_Word4
🖮 🅪 MainTask				Config_Word5	%QW27	INT		Config_Word5
PLC_PRG		B - 🍫		Config_Word6	%QW28	INT		Config_Word6
- & HIGH_PULSE_IO		B- 10		Temperature0	%IW2	INT		Temperature0
EtherCAT_Master_SoftMotion (EtherCAT Master Soft		18 - M		Temperature 1	%IW3	INT		Temperature 1
AX_EM_ECM_ET (EtherCAT Slave Module)		B- 10		Temperature2	%IW4	INT		Temperature2
AX_EM_4PTC (Temperature Input 24Bits)		10 - 1 0		Temperature3	%IW5	INT		Temperature3
SoftMotion General Axis Pool		· · · · · · · · · · · · · · · · · · ·		Breakup	%IW6	INT		Breakup
			Reset Ma	apping Alwa	ys update vari	ables U	se parent	device setting
		-						

Figure 4-16 Variable mapping of temperature module

- 2. After compiling, log in to download the project and run it.
- 3. Variable description: the following tables describe the use of all variables for the four channels.

Table 4-10 Variable description

Note: The channel disconnection detection function and detection results are reserved.

Parameters		Value	Valid bit	Variable name				
Temperature of channel 0			[15:0]	Temperature0				
Temperature of channel 1			[15:0]	Temperature1				
Temperature	of channel 2		[15:0]	Temperature2				
Temperature	of channel 3		[15:0]	Temperature3				
Disconnection detection	Normal	00	[4, 0]					
result of channel 0	Disconnected	01	[1:0]					
Disconnection detection	Normal	00	[0.0]					
result of channel 1	Disconnected	01	[3:2]	Draelwr				
Disconnection detection	Normal	00	[0.0]	вгеакир				
result of channel 2	Disconnected	01	[9:8]					
Disconnection detection	Normal	00	[11.10]					
result of channel 3	Disconnected	01	[11.10]					
Enchle channel 0	Enable	1	[0]					
	Disable	0	[U]					
Dianlov mode	°C	0	[4]					
Display mode	°F	1	[1]					
	Internal cold junction	0						
Cold junction	compensation	0	[0]					
compensation method	External cold junction	1	[2]					
	compensation			Config_Word0				
Sensor disconnection	Enable	1	[3]					
detection	detection Disable		[5]					
Over-limit detection	Enable	1	[4]					
	Disable	0	ניין					
	В	000						
Sensor type	E	001	[11:8]					
	J	010						

Param	eters	Value	Valid bit	Variable name		
	К	011				
	Ν	100				
	R	101				
	S	110				
	Т	111				
	PT100	1000				
	PT500	1001				
	PT1000	1010				
	CU500	1011				
	2-Wire					
	3-Wire	00				
	4-Wire	01	[13:12]			
	(For RTD only)	10				
Filter time	0–100	0–100	[6:0]			
	Enable	1				
Enable channel 1	Disable	0	[8]			
	°C	0				
Display mode	°F	1	[9]			
	Internal cold junction					
Cold junction	compensation	0		Config Word1		
compensation method	External cold junction		[10]	<u>-</u>		
p	compensation	1				
Sensor disconnection	Enable	1				
detection	Disable	0	[11]			
	Enable	1				
Over-limit detection	Disable	0	[12]			
	В	000				
	Е	001				
	J	010				
	К	011				
	Ν	100				
	R	101				
	S	110	[3:0]			
-	т	111				
Sensor type	PT100	1000		Config_Word2		
	PT500	1001				
	PT1000	1010				
	CU500	1011				
	2-Wire	00				
	3-Wire	00	[5,4]			
	4-Wire	10	[5:4]			
	(For RTD only)	10				
Filter time	0–100	0–100	[14:8]			
	Enable	1	[0]			
	Disable	0	Įυj			
Diantau and	°C	0	[4]			
Display mode	°F	1	[1]	Config_Word3		
Cold junction	Internal cold junction		[0]			
compensation method	compensation	0	[2]			

Parameters		Value	Valid bit	Variable name
	External cold junction			
	compensation	1		
Sensor disconnection	Enable	1		
detection	Disable	0	[3]	
	Enable	1		
Over-limit detection	Disable	0	[4]	
	В	000		
	E	001		
	J	010		
	К	011		
	Ν	100		
	R	101		
	S	110	[11:8]	
•	Т	111		
Sensor type	PT100	1000		
	PT500	1001		
	PT1000	1010		
	CU500	1011		
	2-Wire			
	3-Wire	00	[40,40]	
	4-Wire	01	[13:12]	
	(For RTD only)	10		
Filter time	0–100	0–100	[6:0]	
Enable channel 3	Enable	1	[8]	
	Disable	0	[0]	
Display mode	°C	0	[0]	
	°F	1	[9]	
	Internal cold junction	0	[10]	
Cold junction	compensation			Config_Word4
compensation method	External cold junction	1	[]	
	compensation	-		
Sensor disconnection	Enable	1	[11]	
detection	Disable	0	[]	
Over-limit detection	Enable	1	[12]	
	Disable	0	[]	
	В	000		
	E	001		
	J	010		
	K	011		
	N	100		
	R	101	[3:0]	
Sensor type	5 -	110		Config_Word5
	I DT100	111		-
	PTTUU	1000		
	P1500	1001		
	P11000	1010		
		1011		
	2-wire	00	[5:4]	

Parameters			Valid bit	Variable name
	4-Wire	10		
	(For RTD only)			
Filter time	0–100	0–100	[14:8]	
Complian period of	250ms	01		
Sampling period of	500ms	10	[1:0]	
channel 0	1000ms	11		
Sampling period of channel 1	250ms	01		
	500ms	10	[3:2]	
	1000ms	11		Config WordC
Complian period of	250ms	01		Config_word6
Sampling period of	500ms	10	[5:4]	
channel 2	1000ms	11		
Complian period of	250ms	01		
Sampling period of	500ms	10	[7:6]	
channel 3	1000ms	11		

Table 4-11 Supported sensor types and measurement range

Item	Sensor name	Temperature range in Celsius	Temperature range in Fahrenheit
	PT100	-200.0°C-850°C	-328.0°F–1562.0°F
	PT500	-200.0°C-850°C	-328.0°F–1562.0°F
Thermal resistor type	PT1000	-200.0°C-850°C	-328.0°F–1562.0°F
	CU100	-50.0°C–150°C	-58.0°F–302.0°F
	В	200.0°C-1800°C	392.0°F-3272.0°F
	E	-270.0°C-1000°C	-454.0°F–1832.0°F
	Ν	-200.0°C–1300°C	-328.0°F–2372.0°F
The was a set up loss to me	J	-210.0°C–1200°C	-346.0°F–2192.0°F
i nermocoupies type	к	-270.0°C-1370°C	-454.0°F–2498.0°F
	R	-50.0°C–1765°C	-58.0°F–3209.0°F
	S	-50.0°C–1765°C	-58.0°F–3209.0°F
	Т	-270.0°C–400°C	-454.0°F–752.0°F

4.7 Distributed I/O module

The distributed I/O AE1420/AE2420 module is EtherCAT slave module with 32 digital inputs/32 digital outputs. Before using the module, you need to install the device profile **INVT_AE1420_AE2420_xxx.xml** for the module.

4.7.1 Creating a project for distributed I/O module

Create a distributed I/O module application, and install the device profile **INVT_AE1420_AE2420_xxx.xml** required by the module. Right click **EtherCAT_Master_SoftMotion** in the device panel to add the AE1420_AE2420 module.



Devices 🗸 🗸 🗙	AE1420_AE2420_100 X				•
EtherCAT Emercan Device (INVT AX7X)	General	Address		Additional	Ether
IDI PLC Logic A HIGH_PULSE_IO	Process Data	EtherCAT address	-1 1002	Enable expert settings	Luier
CANbus (CANbus)	Startup Parameters	Distributed Clock			
INVT_DA200_262 (DA200-N EtherCAT(CoE) Drive)	EtherCAT I/O Mapping	Select DC	DC-Synchron SM-Synchron	~	
AE 1420_AE 2420_100 (AE 1420_AE 2420 Module)	EtherCAT IEC Objects	Sync0:	DC-Synchron		
	Information	Enable Sync 0	x 1 ~	4000 ♠ Cycle time (μ	s)
		O User-defined		0 🗘 Shift time (µs	;)
		Sync1:			
		 Sync unit cycle 	x 1 ~	4000 🗘 Cycle time (µ	s)
		O User-defined		0 ♠ Shift time (µs	;)
		User-defined	x1 ~	4000 ¢ Cycle time (µs 0 ¢ Shift time (µs	s)

Click the AE1420_AE2420 module in the device panel, and configure the synchronization method in the General interface.

In the **EtherCAT I/O Mapping** interface, the output points are accessed via Y0–Y7/Y10–Y17/Y20–Y27/Y30–Y37, and the input points are accessed via X0–X7/X10–X17/X20–X27/X30–X37.

Biol Control Control <thcontrol< th=""> <thcontrol< th=""> <thcont< th=""><th>Devices 👻 🕈 🗙</th><th>AE1420_AE2420_100 ×</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></thcont<></thcontrol<></thcontrol<>	Devices 👻 🕈 🗙	AE1420_AE2420_100 ×							
Deface UNIT ADJ Address Type Unit Description * Into (PLOSE) DO FreeCAT Jeaster Softbook	EtherCAT	General	Find		Filter Show	w all			- + A
Backup CatMaxie Setup Parameters 1 4008.1 817 11 Behc/CAT Jester Softwoon Ether/CAT Lester Softwoon Ether/CAT Lester Softwoon 1 4008.2 817 12 Behc/CAT Jester Softwoon Ether/CAT LEC Objects 1 4008.3 817 13 Behc/CAT Jester Softwoon Ether/CAT LEC Objects 1 4008.5 817 16 Betc/CAT Jester Softwoon File 7 5008.6 17 16 AELESS JESTER Softwoon File 7 5008.7 817 17 Betc/CAT Jester Softwoon File 7 5008.7 817 17 Betc/CAT Jester Softwoon File 7 5008.7 817 17 Betc/CAT Jester Softwoon File 7 5008.7 817 17 Setue 10 10 4008.8 817 11 10 Information 10 5008.5 817 11 11 Information 11 5008.5 817 115 11 Information 12 5008.5 817 12		Process Data	Variable	Mapping	Channel Y0	Address %0X88.0	Туре	Unit	Description
BrbecAT_Master SofMoton (EberCAT (Go) Drw) EberCAT (U) Mapping '12 %2088.2 BTT '12 BrT_DADD_282 (0.020%) EberCAT (Sci) Drw) EberCAT (EC Objects '13 %2088.5 BTT '14 AE HAD_JE240D_100 (JE H2D_AE240D Module) EberCAT (EC Objects '14 %2088.6 BTT '13 Information '16 '17 %2088.6 BTT '13 Information '16 '17 %2088.6 BTT '13 '17 %2088.7 BTT '13 '14 </td <td>E GANhus (CANhus)</td> <td>Startup Parameters</td> <td> Ka</td> <td></td> <td>Y1</td> <td>%OX88.1</td> <td>BIT</td> <td></td> <td>Y1</td>	E GANhus (CANhus)	Startup Parameters	Ka		Y1	%OX88.1	BIT		Y1
Image: Proceeding of the second of the se	EtherCAT Master, SoftMotion (EtherCAT Master SoftMotion		- * ø		Y2	%OX88.2	BIT		Y2
Back of the control of the C	TINT DA200, 262 (DA200-N Ether CAT (CoE) Drive)	EtherCAT I/O Mapping	Kg		Y3	%OX88.3	BIT		Y3
 M 41420 AE2430 100 (AE1420 AE2430 Modele) Situat Information Information <liinformatio <="" linformation<="" td=""><td>M Still Drive ConsticDSD402 (SM Drive ConsticDSP#</td><td></td><td></td><td></td><td>Y4</td><td>%OX88.4</td><td>BIT</td><td></td><td>Y4</td></liinformatio>	M Still Drive ConsticDSD402 (SM Drive ConsticDSP#				Y4	%OX88.4	BIT		Y4
Status P <td>AE1420 AE2420 100 (AE1420 AE2420 Module)</td> <td>EtherCAT IEC Objects</td> <td></td> <td></td> <td>Y5</td> <td>%OX88.5</td> <td>BIT</td> <td></td> <td>Y5</td>	AE1420 AE2420 100 (AE1420 AE2420 Module)	EtherCAT IEC Objects			Y5	%OX88.5	BIT		Y5
Information P V(2083.7) BTT V10 P V10 V(2080.0) BTT V10 P V11 V(2080.0) BTT V11 P V11 V(2080.0) BTT V11 P V11 V(2080.0) BTT V12 P V11 V(2080.0) BTT V12 P V12 V(2080.0) BTT V13 P V14 V(2080.0) BTT V13 P V15 V(2080.0) BTT V14 P V16 V(2080.0) BTT V15 P V10 V(2080.0) BTT V16 P V10 V(2080.0) BTT V12 P V20 V(200.0) BTT V12 P V22 V(200.0) BTT V12 P V22 V(200.0) BTT V24 P V22 V(2000.0) BTT V24		Status	- *		Y6	%OX88.6	BIT		Y6
Information Yi0 Yu0800 BT Y10 Information Information Y11 Y0289.0 BT Y11 Information Y12 Y0289.2 BTT Y12 Information Y13 Y0289.2 BTT Y12 Information Y13 Y0289.2 BTT Y14 Information Y14 Y0289.2 BTT Y14 Information Y16 Y13 Y16 Y17 Information Y16 Y16 Y16 Y17 Y12 Y0289.7 BTT Y12 Information Y10 Y0289.7 BTT Y12 Y0280.0 BTT Y22 Information Y20 Y0280.0 BTT Y22 Y21 Y0280.0 BTT Y22 Information Y22 Y0280.0 BTT Y22 Y22 Y23 Y24	-				Y7	%OX88.7	BIT		Y7
- -		Information			Y10	%OX89.0	BIT		Y10
• 1 1 1 1 1 • 112 %Q898.2 BT 113 • 113 %Q898.3 BT 113 • 115 %Q898.5 BT 115 • 115 %Q89.6 BTT 115 • 115 %Q89.6 BTT 112 • 115 %Q89.6 BTT 112 • 115 %Q89.6 BTT 112 • 116 %Q89.0 BTT 127 • 120 %Q99.0 BTT 122 • 123 %Q90.0 BTT 122 • 123 %Q90.0 BTT 123 • 123 %Q90.0 BTT 123 • <td></td> <td></td> <td>- N</td> <td></td> <td>Y11</td> <td>%0189.1</td> <td>BIT</td> <td></td> <td>Y11</td>			- N		Y11	%0189.1	BIT		Y11
- 113 %2083.3 BT Y13 - Y13 %2083.4 BT Y14 - Y14 %2083.4 BT Y14 - Y15 %2083.6 BT Y15 - Y15 %2083.7 BT Y15 - Y17 %2083.7 BT Y15 - Y17 %2083.7 BT Y12 - Y21 %2080.2 BT Y22 - Y22 %2090.2 BT Y22 - Y23 %2090.3 BT Y23 - Y24 %2090.5 BT Y25 - Y25 %2090.5 BT Y25 - Y26 %2090.5 B			- <u>*</u>		Y12	%0189.2	BIT		Y12
No No No 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			1		Y13	%0189.3	BIT		Y13
• •			- N		Y14	%0189.4	BIT		Y14
• 1 1 1 1 1 1 • 1 1 1 1 1 1 1 • 1			- N		Y15	%0X89.5	BIT		Y15
• •			- N		Y16	%0X89.6	BIT		Y16
************************************					Y17	%OX89.7	BIT		Y17
100 100 100 100 100 100 100 100 100 100 100 100 121 40000.1 100 100 100 100 122 94090.2 100 100 100 100 123 94090.3 100 100 100 100 123 94090.5 100 100 100 100 125 94090.5 100 100 100 100 125 94090.5 100 100 100 100 125 94090.5 100 100 100 100 125 94090.5 100 100 100 100 126 127 100 100 100 100 100 127 100 100 100 100 100 100 100 100 130 94091.0 101 101 101 101 101 101 101 100 130 94091.1 101 101 103 <td< td=""><td></td><td></td><td></td><td></td><td>¥20</td><td>960390.0</td><td>BIT</td><td></td><td>¥20</td></td<>					¥20	960390.0	BIT		¥20
No. No. No. No. -					V21	%OX90.1	BIT		¥21
Name Name Name Name Name Y23 %Q903.3 BTT Y23 Name Y24 %Q903.4 BTT Y24 Name Y25 %Q905.6 BTT Y25 Name Y26 %Q905.6 BTT Y25 Name Y26 %Q905.6 BTT Y25 Name Y26 %Q905.6 BTT Y25 Name Y30 %Q905.7 BTT Y27 Name Y31 %Q915.1 BTT Y31 Name Y32 %Q915.1 BTT Y32 Name Y33 %Q915.8 BTT Y33 Name Y35 %Q915.6 BTT Y35 Name Y36 %Q915.6 BTT Y36 Name Y37 %QN5.6 BTT Y36 Name Y37 %QN5.6 BTT Y36 Name Y37 %QN5.6 BTT Y36 <			- N		¥22	%OX90.2	BIT		¥22
No No No No - 1/2 %Q/90.4 ETT Y24 - 1/2 %Q/90.4 ETT Y25 - 1/2 %Q/90.6 ETT Y25 - 1/2 %Q/90.6 ETT Y25 - 1/2 %Q/90.6 ETT Y26 - 1/2 %Q/90.6 ETT Y26 - 1/2 %Q/90.6 ETT Y30 - 1/2 %Q/91.0 ETT Y30 - 1/3 %Q/91.1 ETT Y31 - 1/3 %Q/91.1 ETT Y32 - 1/3 %Q/91.3 ETT Y33 - 1/3 %Q/91.4 ETT Y34 - 1/3 %Q/91.4 ETT Y35 - 1/3 %Q/91.7 ETT Y37 - 1/3 %Q/91.7 ETT Y37 - 1/3			- N		¥23	%OX90.3	BIT		¥23
Norm Norm Norm Norm Norm -			- N		¥24	%0190.4	BIT		¥24
No No No No * Y26 %Q90.6 ET Y26 * Y27 %Q90.6 ET Y27 * Y30 %Q90.6 ET Y27 * Y30 %Q90.6 ETT Y30 * Y30 %Q91.0 ETT Y31 * Y31 %Q91.1 ETT Y32 * Y33 %Q91.2 ETT Y33 * Y33 %Q91.4 ETT Y34 * Y33 %Q91.4 ETT Y34 * Y35 %Q91.5 ETT Y35 * Y35 %Q91.5 ETT Y35 * Y35 %Q91.6 ETT Y36 * Y36 Y37 %Q91.7 ETT Y37 * Y37 %Q91.5 ETT Y36 Y37 * Y37 %Q91.6 ETT Y37 Y37 Y37			- N		Y25	%0X90.5	BIT		¥25
No. No. No. No. - 1/2 %Q190.7 ETT Y27 - Y30 %Q10.0 ETT Y30 - Y31 %Q10.1 ETT Y31 - Y32 %Q191.3 ETT Y33 - Y33 %Q191.3 ETT Y33 - Y34 %Q191.4 ETT Y34 - Y35 %Q191.5 ETT Y35 - Y36 %Q191.7 ETT Y37 - Y36 %Q191.7 ETT Y37 - Y37 %Q191.7 ETT Y37 - Y36 %Q191.7 ETT Y37 - Y37 %Q191.7 ETT Y37 - Y37 %Q191.7 ETT Y37 - Y37 %Q191.7 ETT Y37 - Y38 %Q191.7 ETT Y37 - Y37			- N		¥26	%0X90.6	BIT		¥26
Ib Notestinit Ib -10 100 4020510 ETT 120 -10 131 9620510 ETT 131 -10 131 9620511 ETT 131 -10 132 9620512 ETT 132 -10 133 9620513. ETT 133 -10 133 9620514. ETT 133 -10 133 962051.5 ETT 135 -10 135 962051.5 ETT 135 -10 137 962051.5 ETT 135 -11 137 137 137 137 -11 137 137 137 137 -11 137 137 137 137 -11 <td< td=""><td></td><td></td><td></td><td></td><td>V27</td><td>%OV90.7</td><td>BIT</td><td></td><td>¥27</td></td<>					V27	%OV90.7	BIT		¥27
No No No * Y31 %QV91.1 BTT Y31 * Y32 %QV91.2 BTT Y32 * Y34 %QV91.4 BTT Y34 * Y34 %QV91.4 BTT Y34 * Y35 %QV91.4 BTT Y35 * Y35 %QV91.5 BTT Y35 * Y35 %QV91.6 BTT Y36 * Y36 %QV91.7 BTT Y37 * Y37 %QV91.5 BTT Y37 * Y36 %QV91.6 BTT Y37 * Y37 %QV91.7 BTT Y37 * Y37 %QV91.7			- <u>5</u>		Y30	%0X91.0	BIT		Y30
No. No. No. No. - 132 %QV91.2 BT Y32 - 133 %QV91.3 BT Y33 - 134 %QV91.4 BT Y34 - 135 %QV91.6 BT Y35 - 136 %QV91.6 BT Y35 - 136 %QV91.6 BT Y35 - 136 %QV91.7 BT Y35 - 137 %QV91.7 BT Y37 - 140 X0 %DX48.0 BT X37 - 140 X0 %DX48.0 BT X37			- <u>5</u>		Y31	%0X91.1	BIT		Y31
No. No. No. No. - (33) %Q/91.3 BIT Y33 - (34) %Q/91.4 BIT Y34 - (35) %Q/91.5 BIT Y35 - (35) %Q/91.5 BIT Y35 - (37) %Q/91.7 BIT Y36 - (37) %Q/91.7 BIT Y36 - (37) %Q/91.7 BIT Y37 - (37) %Q/91.7 BIT Y36 - (37) %Q/91.7 BIT Y36 - (30) %D/41.0 BIT X00 - (30) %D/41.0 BIT X1 - (31) %D/41.0 BIT X1			1 . <u>.</u>		V32	%OY91.2	BIT		V32
Non-Line					V33	960391.2	BIT		V33
Total Total Total Total Total -1 1 1 1 1 1 -1 1 1 1 1 1 1 -1 1 <td></td> <td></td> <td></td> <td></td> <td>V34</td> <td>960391.0</td> <td>BIT</td> <td></td> <td>V34</td>					V34	960391.0	BIT		V34
Total Total Total Total - * Y36 %Q/91.7 BIT Y36 - * Y37 %Q/91.7 BIT Y37 - * Y30 %D/48.0 BIT X0 - * Y36 Y37 Y37 Y37					V35	960391.5	BIT		V35
Total Total <th< td=""><td></td><td></td><td></td><td></td><td>V36</td><td>960391.5</td><td>BIT</td><td></td><td>V36</td></th<>					V36	960391.5	BIT		V36
** 1.37 746,95.17 b.11 1.37 -40 X0 95,04.0 BIT X0 -40 X1 95,04.0 BIT X0					V37	960191.7	BIT		V37
ψ Λυ γμλγκού δι1 Λυ - Μρ X1 γμλγκού δ11 Λυ			× ×		V0	76QX91.7	DIT		107
ψ Λ⊥ γε/μ(Mr).1 0.11 Λ⊥ Ψ ψ ψ μ/μ ψ μ/μ ψ					XU V1	761A-10.0	DIT		×1
					×2	961X49.2	DIT		×2

4.8 Priority setting of each module (recommended value)

4.8.1 Setting priority

If the created project contains multiple functional modules, create multiple tasks and set the task priority as follows. Table 4-12 shows the recommended values for task priority.

Devices 🗸 🗸 🗶	📄 PLC_PRG 🔹 Task 🖉 MainTask 🗙
- 🗿 Untitled2	Configuration
🖃 🍿 Device (INVT AX7X)	
PLC Logic	Priority (031): 0
🖃 🚫 Application	Type
Library Manager	(P) Cyclic Interval (e.g. ##200mp) 4
PLC_PRG (PRG)	incervar (e.g. t#200ms)
	Watchdog Enable Time (e.g. t#200ms) Sensitivity 1
SoftMotion General Axis Pool	🕂 Add Call 🗙 Remove Call 📝 Change Call 🕆 Move Up 🐥 Move Down 🔭
	POU Comment
	创 PLC_PRG

Figure 4-17 Example of task project priority settings

Table 4-12 Setting priority

Function module	Recommended priority
PlcCfg module	31
ModbusTCP	15–30
ModbusRTU	15–30
High-speed I/O	1–15
Analog input/output	1–15

Function module	Recommended priority
Temperature module	1–15
EtherCAT	0

4.8.2 Configuring sub-device bus cycle options

Under the **Controller settings** > **Bus cycle** > **Bus cycle task** of the AX7X device, the Bus cycle task list provides the tasks defined in the task configuration of the current valid project (such as "MainTask", "EtherCAT Master"). Select one of the tasks as the bus cycle of the current project, or select the option **<unspecified>**, which indicates that the shortest task cycle time or the fastest execution cycle will be applied. You can switch to another settings, but be sure to note the following.

Note: Before modifying the **<unspecified>** setting, be aware that it is a default action defined by the device description. By default, the task can be defined with a shortest cycle time or a longest cycle time. Please check this carefully before applying this setting.

To improve the stability of the system when using expansion modules and EtherCAT modules (especially the EtherCAT_Master_SoftMotion module), you should select the task corresponding to each module in **EtherCAT I/O Mapping** > **Bus Cycle Options**. The reference program is as follows.



Figure 4-18 Expansion module bus cycle task setting

5 Device Diagnosis

AX series equipment diagnostic information is reflected in three ways, namely fault indicator, digital tube and diagnostic code. Fault indicators show the system and bus error. Digital tubes display the fault code of a specific function module. Diagnostic codes further indicate the specific types of faults, which can be generally searched by upper computer software.

5.1 Fault indicator

The AX series fault indicator is mainly composed of two parts. The first part is mainly the system and bus indicator lights. The second part is mainly digital tube indicators.



Figure 5-1 Fault indicator diagram

5.1.1 System and bus fault indicator

Table 5-1 System and bus fault indicator

Fault indicator name	Error type	Error content
SF	System fault	Codesys is not started
BF	Bus communication fault	Modbus RTU/Modbus TCP/backplane bus fault
CAN	CAN bus fault	Reserved
ERR	Module fault	Reserved

Note: When connecting multiple programmable controllers, you can click the **Wink** button on the software platform to observe the simultaneous flashing of the SF, BF, CAN, and ERR indicators to identify the device.

5.1.2 High-speed input/output indicator

If the output/input of the port is valid, the indicator corresponding to the port is on, and if the output/input is invalid, the corresponding indicator is off.

5.2 Digital tube fault code

Digital tube Fault code	Module	Fault type	Solution
16#10		Error setting local new IP	Check the underlying network configuration file.
16#11		Error setting local new subnet mask	Check the underlying network configuration file.
16#12	CPU module	Failed to read the local IP and subnet mask	Check the underlying network configuration file.
16#13	PicCig	Abnormal time setting format	Check the time setting format.
16#14		Error setting motion controller time	Check the underlying code.
16#15		Error getting motion controller real time	The controller button battery voltage is insufficient, replace the battery.
16#20		Failed to open serial port COM1	Check whether the underlying serial port number corresponds to the hardware.
16#21		Baud rate setting failed	Check the baud rate setting of the slave node
16#22	COM1 485 ModbusRTU_Slave1	Data bit, stop bit or parity bit setting failed	Check the specific error code of Invtmatic Studio ErrorID. Data bit: ErrorID=3, check bit ErrorID=4, stop bit ErrorID=5.
16#23		Slave function enable failed	System error Err_Sym, or slave enable is turned on.
16#24		Slave read and write error	Check detailed parameter settings
16#25		Failed to open serial port COM1	Check whether the underlying serial port number corresponds to the hardware.
16#26		SlaveID setting failed	Check the SlaveID number settings of the master node.
16#27	COM1 485 ModbusRTU_Master 1	Data bit, stop bit or parity bit setting failed	Check whether the data bit setting value is 7 or 8, whether the check bit is 0, 1 or 2, and whether the stop bit is 1 or 2.
16#28		Master function enable failed	System error Err_Sym, or master enable is turned on.
16#29		One of the the following goes wrong: master read/write coil, read holding register, write a single register, write multiple registers	Check that the master-slave initialization parameter configuration is consistent and that the hardware connection is correct.
16#2A		Two function block enabled at the same time.	Ensure that only one of the function block is enabled in the program

Digital tube Fault code	Module	Fault type	Solution
16#30		Failed to open serial port COM2	Check whether the underlying serial port number corresponds to the hardware.
16#31		Baud rate setting failed	Check the baud rate setting of the slave node
16#32	COM2 485 ModbusRTU_Slave2	Data bit, stop bit or parity bit setting failed	Check the specific error code of Invtmatic Studio ErrorID. Data bit: ErrorID=3, check bit ErrorID=4, stop bit ErrorID=5.
16#33		Slave function enable failed	System error Err_Sym, or slave enable is turned on.
16#34		Slave read and write error	Check detailed parameter settings
16#35		Failed to open serial port COM2	Check whether the underlying serial port number corresponds to the hardware.
16#36		SlaveID setting failed	Check the SlaveID number settings of the master node.
16#37	COM2 485 ModbusRTU_Master 2	Data bit, stop bit or parity bit setting failed	Check whether the data bit setting value is 7 or 8, whether the check bit is 0, 1 or 2, and whether the stop bit is 1 or 2.
16#38		Master function enable failed	System error Err_Sym, or master enable is turned on.
16#39		One of the the following goes wrong: master read/write coil, read holding register, write a single register, write multiple registers	Check that the master-slave initialization parameter configuration is consistent and that the hardware connection is correct.
16#3A		Two function block enabled at the same time.	Ensure that only one of the function block is enabled in the program
16#60		Error configuring slave IP	Check the underlying corresponding configuration.
16#61		Port setting error	Check the port settings
16#62		Failed to listen to sockets (failed to create socket, failed to bind socket, failed to bind socket)	Check the corresponding configuration.
16#63	modbust CP_Slave	Failed to accept client	Check the corresponding configuration.
16#64		Failed to accept client data	Check the corresponding configuration.
16#65		Modbus reply error (modbus_reply)	Check the corresponding configuration.
16#66		Error setting slave IP or port	Check the IP setting or whether it is the default unit number.
16#67		Failed to set slave node	Check the parameter settings.
16#68	modbusTCP_Master	Failed to connect slave node	Check the parameter settings, such as slave IP or port
16#69	1	Write slave register failure	Check the parameter settings.

Digital tube Fault code	Module	Fault type	Solution				
16#6A		Read slave register failure	Check the parameter settings.				
16#A0		Channel 0 is disconnected.	Check whether the wires are connected properly.				
16#A1		Channel 0 exceeds the limits (that is, the voltage exceeds the range of -25V-+25V, and the current exceeds the range of -104mA- 104mA)	Check if the input voltage (current) is out of range.				
16#A2		Channel 0 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)	Reduce the input voltage (current) value, or use a wider range of conversion modes.				
16#A3		Channel 0 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)	Increase the input voltage (current) value, or use a wider range of conversion modes.				
16#A4		Channel 1 is disconnected.	Check whether the wires are connected properly.				
16#A5		Channel 1 exceeds the limits (that is, the voltage exceeds the range of -25V–+25V, and the current exceeds the range of -104mA– 104mA)	Check if the input voltage (current) is out of range.				
16#A6	Analog output module	Channel 1 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)	Reduce the input voltage (current) value, or use a wider range of conversion modes.				
16#A7	AX-EM-4AD	Channel 1 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)	Increase the input voltage (current) value, or use a wider range of conversion modes.				
16#A8		Channel 2 is disconnected.	Check whether the wires are connected properly.				
16#A9		Channel 2 exceeds the limits (that is, the voltage exceeds the range of -25V–+25V, and the current exceeds the range of -104mA– 104mA)	Check if the input voltage (current) is out of range.				
16#AA		Channel 2 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)	Reduce the input voltage (current) value, or use a wider range of conversion modes.				
16#Ab		Channel 2 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)	Increase the input voltage (current) value, or use a wider range of conversion modes.				
16#AC		Channel 3 is disconnected.	Check whether the wires are connected properly.				
16#Ad		Channel 3 exceeds the limits (that is, the voltage exceeds the range of -25V–+25V, and the current exceeds the range of -104mA–	Check if the input voltage (current) is out of range.				

Digital tube Fault code	Module	Fault type	Solution				
		104mA)					
16#AE		Channel 3 exceeds the upper limit of the range (exceeds the upper limit of the currently selected voltage range)	Reduce the input voltage (current) value, or use a wider range of conversion modes.				
16#AF		Channel 3 exceeds the lower limit of the range (exceeds the lower limit of the currently selected voltage range)	Increase the input voltage (current) value, or use a wider range of conversion modes.				
16#b0		The current output of channel 0 is disconnected.	Check whether the current channel is disconnected and reconnect it if it is				
16#b1		The voltage output of channel 0 is short-circuited.	Check whether the voltage channel is short-circuited. If so, restore it to normal.				
16#b2		The current output of channel 1 is disconnected.	Check whether the current channel is disconnected and reconnect it if it is				
16#b3		The voltage output of channel 1 is short-circuited.	Check whether the voltage channel is short-circuited. If so, restore it to normal.				
16#b4	Analog output module AX-EM-4DA	The current output of channel 2 is disconnected.	Check whether the current channel is disconnected and reconnect it if it is				
16#b5		The voltage output of channel 2 is short-circuited.	Check whether the voltage channel is short-circuited. If so, restore it to normal.				
16#b6		The current output of channel 3 is disconnected.	Check whether the current channel is disconnected and reconnect it if it is				
16#b7		The voltage output of channel 3 is short-circuited.	Check whether the voltage channel is short-circuited. If so, restore it to normal.				
16#b8		The 24V power board of the output module is disconnected.	Check whether the 24V power supply is normal and whether there is reverse connection.				
16#C0		Channel 0 exceeds the upper limit of range (the actual temperature exceeds the set upper limit)	Check whether the set temperature upper limit is greater than the actual value.				
16#C1		Channel 0 exceeds the lower limit of range (the actual temperature exceeds the set lower limit)	Check whether the set temperature lower limit is smaller than the actual value.				
16#C2	Temperature module AX-EM-4PTC	Channel 1 exceeds the upper limit of range (the actual temperature exceeds the set upper limit)	Check whether the set temperature upper limit is greater than the actual value.				
16#C3		Channel 1 exceeds the lower limit of range (the actual temperature exceeds the set lower limit)	Check whether the set temperature lower limit is smaller than the actual value.				
16#C4		Channel 2 exceeds the upper limit	Check whether the set				

Digital tube Fault code	Module	Fault type	Solution
		of range (the actual temperature exceeds the set upper limit)	temperature upper limit is greater than the actual value.
16#C5		Channel 2 exceeds the lower limit of range (the actual temperature exceeds the set lower limit)	Check whether the set temperature lower limit is smaller than the actual value.
16#C6		Channel 3 exceeds the upper limit of range (the actual temperature exceeds the set upper limit)	Check whether the set temperature upper limit is greater than the actual value.
16#C7		Channel 3 exceeds the lower limit of range (the actual temperature exceeds the set lower limit)	Check whether the set temperature lower limit is smaller than the actual value.
16#C8		Over-limit setting error (set upper limit is smaller than the lower limit)	Check whether the set temperature upper limit is greater than the lower limit.
16#C9		Channel 0 is disconnected. (Reserved)	
16#CA		Channel 1 is disconnected. (Reserved)	
16#CB		Channel 2 is disconnected. (Reserved)	
16#CC		Channel 3 is disconnected. (Reserved)	

6 Controller Program Structure and Execution

6.1 Program structure

The software model is represented by a hierarchical structure. Each layer implies many characteristics of the underlying layer. The software model describes the basic software elements and their interrelationships. These software elements contain: devices, applications, tasks, global variables, access paths, and application objects. Figure 6-1 shows their internal structure, which is consistent with the software model of the IEC 61131-3 standard.



Figure 6-1 Program hierarchical structure

6.2 Task

A program can be written in different programming languages. A typical program consists of a number of interconnected function blocks that can exchange data with each other. The execution of different parts of a program is controlled by "tasks". Tasks can be configured to cause a series of programs or blocks to execute periodically or to be triggered by a specific event to start execution.

The **Task Manager** tab in the device tree can be used to control the execution of other subprograms within the project, in addition to the specific controller_PRG program. A task is used to specify the properties of a program organization unit at run time. It is an execution control element with the ability to be called. Multiple tasks can be established in a task configuration, and multiple program organization units can be called in a task. Once the task is set, it can control the program to execute periodically or to be triggered by a specific event to start execution.

In the task configuration, define it with name, priority, and startup type of the task. This startup type can be defined either by time (cyclic, random) or by the timing of an internal or external trigger task, such as a rising edge of a Boolean global variable or a particular event in the system. For each task, you can set a sequence of programs to be started by the task. If this task is executed in the current cycle, these programs will be processed within one cycle. The combination of priority and conditions will determine the timing of task execution. The task setting interface is shown in Figure 6-2.

Devices 👻 🕈 🗙	MainTask 🗙 📆 EtherCAT_Master_SoftMotion 🔯 Task Configuration
Chittled2	Configuration
Choice22 Choice2 Priority (031): 0 Type Cyclic Interval (e.g. t#200ms) t#20ms Watchdog Finable	
	Time (e.g. t#200ms) t#200ms Sensitivity 1
	Add Call 🗙 Remove Call 🖉 Change Call 🕆 Move Up 🔹 Move Down 🎽 Open P
	POU Comment
	e PLC_PRG
POUs	
Project Settings	

Figure 6-2 Task configuration interface

The programmer must follow the following rules:

- 1. The maximum number of cyclic tasks is 100.
- 2. The maximum number of free running tasks is 100.
- 3. The maximum number of event-triggered tasks is 100.
- 4. Depending on the target system, the PLC_PRG may be executed as a free program under any circumstances, instead of being manually inserted into the task configuration.
- 5. Programs are processed and called in a top-down order within the task editor.

6.3 Program execution

The following figure describes in detail the complete process of program execution inside the AX series programmable controller. The main process includes input sampling, program execution and output refresh.



Figure 6-3 Controller execution

1) Input sampling

At the beginning of each scan cycle, the controller detects the state of the input device (such as switch, button) and writes the state to the input image register area. During program execution, the running system reads data from the input image area for program resolution. It is important to note that the input refresh only occurs at the beginning of a scan. During the scan, the input state will not change even if the output state changes.

2) Program execution

During the program execution phase of the scan cycle, the controller reads the status and data from the input image area or output image area and performs logical and arithmetic operations according to the commands. The operation results are stored in the corresponding unit in output image area. In this phase, only the contents in the input image registers remain unchanged, and the contents in other image registers will change with the execution of the program.

3) Output refresh

During the output refresh phase, also known as the write output phase, the controller transmits the state and data in the output image area to the output point, and isolates and amplifies the power in a certain way to drive the external load. The programmable controller completes not only the tasks of the above three phases, but also auxiliary tasks such as internal diagnosis, communication, public processing, and input/output services in a scan cycle.

The AX series programmable controller repeats the process of 1) to 3) above, and the time for each repetition is one work cycle (or scan cycle). It can be seen from the scanning method of the controller that the controller has a shorter scanning time to complete the control task to quickly respond to the change of input and output data, and the duty cycle is generally controlled within the order of ms. Therefore, it is necessary to develop a stable, reliable and fast-response real-time system for AX series programmable controller operation system.

Since the AX series programmable controller adopts a cyclic working mode, the input signal will only be refreshed at the beginning of each cycle, and the output will be concentrated at the end of each cycle. It will inevitably produce a lag between the output signal and the input signal. It takes a while for a signal input to change from the input of the AX series programmable controller to the output of the controller to respond to the change in the input signal. Lag time is an important parameter that should be understood when designing AX series programmable controller control system. Generally, the lag time is related to the following factors:

1. Filter time of the input circuit. It is determined by the time constant of the hardware RC filter circuit. The input lag time can be adjusted by changing the time constant. For example, Table 6-1 shows the technical parameters of the AX-EM-1600D digital input module, where "port filter time" indicates that the filter time of this input module is 10ms.

Item	Specifications
Input channel	16
Input connection mode	18-point terminal
Input voltage level	24V (up to 30V)
Input current (typical)	4.7mA
ON voltage	>15VDC
OFF voltage	<5VDC
Port filter time	10ms
Input resistance	5.4kΩ
Input signal form	Voltage DC input
Isolation method	Optocoupler
Input dynamic display	When the input is valid, the indicator is on.

Table 6-1 AX-EM-1600D Digital input module parameters

- 2. Lag time of the output circuit. It is related to the output circuit mode. Generally, the lag time of the relay output mode is about 10ms, and the lag time of the transistor output mode is less than 1ms.
- 3. Working mode of the controller cyclic scanning.
- 4. Arrangement of statements in the user program.

To allow readers to better understand the whole process, the following is a simple example of the ladder diagram program to show its input and output and how the lagging is produced The program logic is shown in Figure 6-4.



Figure 6-4 AX series programmable controller program

blnput has a hardware mapping relationship with the external input button. When the button is pressed, blnput is ON. bOutput has a hardware mapping relationship with the coil of the external relay. When bOutput is ON, the coil of the relay will also be energized. Within the AX series programmable controller, the handling relationship is shown in Figure 6-6. blnput is not immediately turned ON when the input button is pressed. Because the input sampling is only executed at the beginning of a cycle and the button signal has missed the sampling phase, it usually will be executed at the beginning of the next cycle. In the program in Figure 6-6, the state of blnput is assigned to bOutput. Since there is a certain program calculation during the program running, the bOutput needs a certain processing time of the program to be set to ON. Since the output refresh occurs at the end of the program process, it is at the end of the cycle that the bOutput passes its value to the actual hardware via the output refresh function before the coil is finally energized. The following figure is a relatively ideal state, with the final output having only one cycle of latency.



Figure 6-5 Fastest output case

In addition, we should also consider the worse situation. When a cycle of input sampling has just ended, the external input button is ON at this time. Since the input signal needs to be loaded into the input image area at the beginning of the next cycle and the actual output will not be loaded into the output image area until the end of the second cycle, the whole process is shown in Figure 6.7. In this case, the output delay is nearly 2 cycles, which is the output with longest delay time.



Figure 6-6 Slowest output case

6.4 Task execution type

At the top of the task configuration tree, there is a **Task Configuration** tab, which shows every defined task by their names. The call of POUs for specific tasks is not displayed in the task configuration tree. Each individual task can be edited and configured for the type of execution, which includes Cyclic, Event, Freewheeling, and Status. See Figure 6-7 for details.

Туре	
🕑 Cyclic	~
🕑 Cyclic	
🎸 Event	
🎸 External	
💪 Freewheeling	
Status	

Figure 6-7 Task execution type

1. Cyclic

The processing time of the program will vary depending on whether the commands used in the program are executed or not. Therefore, the actual execution time varies with each scan cycle. By using the cyclic mode, the program can be executed repeatedly for a certain cycle time. Even if the execution time of the program changes, the refresh interval can be maintained. It is recommended that you give priority to the cyclic start mode. For example, if you set the corresponding task to the Cyclic mode and set the interval to 10ms, the actual program execution timing is shown in Figure 6-8.



Figure 6-8 Cyclic execution sequence

If the actual execution time of the program is less than the set cyclic time, the remaining time is used for waiting. If there are low-priority tasks in the application that have not been executed, the remaining waiting time is used to execute these tasks. The priority of the task will be described in detail later.

2. Freewheeling

Tasks are processed as soon as the program starts running, and tasks will be automatically restarted in the next cycle after the end of a running cycle. This execution mode is not affected by the program scan cycle. That is to ensure that the last instruction of the program is executed each time before entering the next cycle. Otherwise, the program cycle will not end. Figure 6-9 shows the timing of freewheeling sequence.



Figure 6-9 Timing of freewheeling sequence

Since the freewheeling execution mode does not have a fixed task time, the execution time may be different each time. Therefore, the real-time performance of the program cannot be guaranteed, and this mode is seldom used in practical applications.

3. Event

If the variable in the event area gets a rising edge, the task begins.

4. Status

If the variable in the event area is TRUE, the task begins. The Status mode is similar to the Event mode, except that the task will be executed when the trigger variable of status triggering is TRUE, and will not be executed when it is FALSE. The event trigger only collects the effective signal of the rising edge of the trigger variable. Figure 6-10 compares the event and status trigger modes, and the green solid line is the Boolean variable status selected by the two modes. Table 6-2 shows the comparison result.



Figure 6-10 Task input trigger signal

Different types of tasks showed different responses at sampling points 1–4 (purple). The trigger condition of Status mode is fulfilled when a specific event is TRUE, but an event-driven task requires the event to change from FALSE to TRUE. If the sampling frequency of the task is too low, the rising edge of the event may not be detected.

T 1 1 0 0 0		. – .	1011	
Table 6-2 Com	parison result b	etween Event and	d Status trigger n	nodes

Execution point	1	2	3	4			
Event	No execute	Execute	Execute	Execute			
Status	No execute	Execute	No execute	No execute			

6.5 Task priority

1. Task priority setting

You can set the priority of the task, with a total of 32 levels (a number from 0 to 31, with 0 the highest priority and 31 the lowest priority). When a program is executing, tasks with high priority takes precedence over tasks with low priority. A task with high priority 0 can interrupt the execution of lower priority programs in the same resource, so that the execution of the program with low priority is slowed down.

Note: When assigning task priority levels, do not assign tasks with the same priority. If there are other task views that precede tasks with the same priority, the result may be uncertain and unpredictable.

If the task type is "Cyclic", it will be executed in a cycle according to the time set in "Interval". The specific settings are shown in Figure 6-11.

Configuration			
Priority (031):	1		
Туре			
🕑 Cyclic	~	Interval (e.g. t#200ms)	t#20ms

Figure 6-11 Cyclic mode configuration

Example: Suppose there are 3 different tasks with three different priority levels, the specific assignments are as follows.

: Task 1 with **Priority** set to 0 and **Interval** to 10ms

Task 2 with Priority set to 1 and Interval to 30ms

: Task 3 with Priority set to 2 and Interval to 40ms

Inside the controller, the timing relationship of each task is shown in Figure 6-13, and the specific description is as follows:

0–10ms: Execute Task 1 first (highest priority), and if the program is finished within this cycle, the remaining time will be used to execute the Task 2 program. However, if Task 2 has not been fully executed after10ms, Task 2 will be interrupted because Task 1 is executed every 10 milliseconds and has a highest priority.

10–20ms: Execute the programs in Task 1 first. If there is any time left, execute the unfinished Task 2 in the previous cycle.

20–30ms: Since Task 2 is executed every 30ms and Task 2 has been finished within 10–20ms, there is no need to execute task 2 at this time, just execute Task 1 once.

30–40Ms: Similar to before.

40–50ms: Task 3 appears at this time. Since Task 3 has the lowest priority, Task 3 can only be executed after ensuring that Task 2 has been thoroughly executed.



Figure 6-12 Task interrupt execution order

2. AX7x task priority configuration

When the upper computer software of AX series controller creates a new standard project, MainTask is created by default in the task configuration with a priority of 1. The priority of newly created tasks is also 1 by default, but to ensure that important tasks such as motion control are prioritized, the performance of the controller can be used appropriately in some applications that require high-performance motion control (MC). The following table shows the recommended task priority order setting (if there is only one task, the task priority can be set at will):

Task Type	Recommended Priority
PlcCfg module	31
ModbusTCP	15–30
ModbusRTU	15–30
High-speed I/O	1–15
Analog input/output	1–15
Temperature module	1–15
EtherCAT	0

The smaller the priority value, the higher the priority. POU with a higher priority can interrupt the execution of POU with a lower priority, as shown in Figure 6-13, where ECT stands for EtherCAT.

Controller Program Structure and Execution

ETC cycle (priority0) ETC cycle						ETC cycle				ETC cycle				\downarrow	ETC cycle									
IO U	UPR G	M C	Exe	ecution	10	UPR G	M C	Executi	on te Task cycle (priori			M C	Execution complete (16)	IO UPR M G C			Execution complete			IO UPR M G C			Execution complete	
	Pause		10	UPRG		Pause		UPRG	Execu comp	ution lete		Та	sk cycle (priority	17)						Pause		Ю	UPRG	
	Pause						UPR G		Pause		UPRG		Pause		UPR G	Exec com	cution plete				;			



As shown in Figure 6-13,

When the controller executes a task, there is a time alignment point that is not observed by the user, as shown on the left side of the figure above. Starting at this point, the execution will start in the order of highest priority -> second highest priority -> lowest priority.

A low-priority task may be interrupted by a high-priority task while it is being executed, and when the execution of the high-priority task is complete, the interrupted task with low-priority will continues.

The EtherCAT task is the highest priority task, which is entered according to the EtherCAT cycle, and all POUs within the task are executed once before executing the lower priority task.

3. Requirements for execution cycle setting in task configuration

The AX series system upper computer software uses multitasking to execute the "tasks" of the user program, and each "task" is assigned a different execution cycle. Some global variables may be accessed and modified in different POUs, so the interactive synchronization of global variables should be carried out at the "time alignment point" of the task. For the cycle of a cyclic task setting, the cycle time of different cyclic task types is an integer multiple.

For example, the EtherCAT task cycle time is set to 4ms, 8ms, while the normal cycle is set to 400ms, and the cycle of lower priority is set to 100ms or 200ms. Do not set the EtherCAT task cycle to 5ms, 7ms, 9ms and so on, which may cause non-integer multiple of 2.

4. Configuring sub-device bus cycle options

Under the **Controller settings** > **Bus cycle** > **Bus cycle task** of the controller device, the Bus cycle task list provides the tasks defined in the task configuration of the current valid project (such as "MainTask", "EtherCAT Master"). Select one of the tasks as the bus cycle of the current project, or select the option **<unspecified>**, which means that the shortest task cycle time or the fastest execution cycle will be applied. You can switch to another settings, but be sure to note the following.

Before modifying the <unspecified> setting, be aware that it is a default action defined by the device description. By default, the task can be defined with a shortest cycle time or a longest cycle time. Please check this carefully before applying this setting.

Therefore, select the task corresponding to each module in EtherCAT I/O when using expansion modules and EtherCAT modules (especially the EtherCAT_Master_SoftMotion module) to improve the stability of the system. The reference program is shown in Figure 6-14.

Untitled2.project* - Invtmatic Studio		
File Edit View Project Build Online Debug	Tools Window Help	
◎ ☞ ■ ● ∽ ∼ 以 № 池 × 巻 弦 巻 な」	貝 刻 刻 洵 陶 陶- 🕜 圏 Appli	cation [Device: PLC Logic] • 🐝 🖏 $\models = \overset{\bullet}{=} \overset{\bullet}{=} [\downarrow_{\Xi} \circ_{\Xi} \circ_{\Xi} \Rightarrow_{\Xi} & \Rightarrow = $
Devices 👻 🛱 🗙	EtherCAT_Master_SoftMotion X	POU 🥸 Task 🕸 Task_1 🗸 🗸
Device (DVT AX7X)	General Bus G Sync Unit Assignment Log EtherCAT I/O Mapping EtherCAT IEC Objects Status Information	yde task. EfterCAT_Task. Use parent bus cycle setting EtherCAT_Task. Task. Task. Task.
< POUs POUs POUs POUs Project Settings roject Settings Project Se	Key Control (a) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c	ge(s) • 0 0 error(s) ● 0 warning(s) ● 3 message(s) × ¥ Project Object
		Lasthuide 🔿 0 🔿 0 Draconnela 🦯 🔐 Draiacturan (as

Figure 6-14 EtherCAT bus cycle task setting

6.6 Operation of multiple subprograms

In practical projects, the program can usually be divided into many subprograms according to the control flow or the object of the equipment. The designer can program each processing unit separately. As shown in Figure 6-15, the main program is divided into multiple subprograms with different processes through the control flow. The main purpose of the division is to make the main program clearer and facilitate future debugging.



Figure 6-15 Split in multiple subprograms by process

The right part of Figure 6-15 displays the subprograms PRG1, PRG2...PRGn classified by the flow. The left part of the figure displays the main program PLC _PRG. The PRG1...PRGn subprograms can be called separately in the main program. There are two ways to run multiple subprograms. One is to add subprograms in the task configuration. The other is to call subprograms from the main program, which is more common and flexible.

1. Add subprograms in task configuration

Users can add subprograms in the task configuration page to realize the operation of multiple programs. Click Add Call to

add subprograms in the order in which they are executed. As shown in Figure 6-16, after adding subprograms, the tasks will be executed in the top-to-bottom order specified by the user, or you can edit the order manually by using the **Move Up** and **Move Down** functions.

🕂 Add Call 🔀 Remove Call	🗹 Change Call 🔄 Move Up 🐥 Move Down 🛛 → 🔤 Open POU
POU	Comment
PLC_PRG	
POU POU	
POU_1	

Figure 6-16 Add subprograms in a task

2. Call subprograms from main program PLC_PRG

PLC_PRG is the default main program of the system. In a sense, it can be understood as the battery of a car. In the production of a car, each part is assembled, which is equivalent to the writing of subprograms. When the car is assembled, it is necessary to check whether the car is usable. If you want to start the car, you must start the engine, lights and other parts through the battery which is equivalent to the entry point for starting the car. By calling the program in this way, the program becomes more operable and flexible. You can add judgment statements and use nesting in the program.

PLC_PRG is a special POU that runs by default with a coasting mode. This POU is called every control cycle by default without any additional task configuration. The configuration of the POU can be found in the task configuration. It can be used to call other subprograms and add necessary condition selection at the time of the call, or nest subprograms to make program calling more flexible. To implement the call relationship in Figure 6-17, write the following code in the main program PLC _PRG.



Figure 6-17 POU calling sequence

As shown in the Figure 6-17, the main program is PLC_PRG, which uses structured text programming language, and the program content is POU_1(); POU_2();.

The main function of the above programs is to call and execute POU_1 and POU_2 subprograms respectively. And POU_1 calls POU_3 and POU_4 respectively. The AX series programmable controller actually executes the programs in the following order:

- A. AX series programmable controller program executes POU_1 first.
- B. Since POU_3 and POU_4 are called sequentially in POU_1, POU_3 is executed first.
- C. Execute POU_4 to complete POU_1.
- D. Finally execute POU_2 to complete a full task cycle.

Repeating the above steps A to D is the internal execution sequence of the AX series programmable controller.

7 EtherCAT Bus Motion Control

7.1 EtherCAT operation principle

7.1.1 Protocol introduction

EtherCAT overcomes the inherent limitations of other Ethernet solutions. : On the one hand the Ethernet packet is no longer received then interpreted and process data then copied at every device, but the EtherCAT slave devices read the data addressed to them while the frame passes through the node. Similarly, input data is inserted while the telegram passes through. In the whole process, the frames are only delayed by a few nanoseconds.

The frame send by the master is passed through to the next device until it reaches the end of the segment (or branch). The last device detects an open port and therefore sends the frame back to the master. On the other hand, an EtherCAT frame comprises the data of many devices both in sending and receiving direction within one Ethernet frame. The usable data rate increases to over 90 %. The full-duplex features of 100 Mb/s TX are fully utilized, so that effective data rates of > 100 Mb/s (> 90 % of 2 x 100 Mb/s) can be achieved.

The EtherCAT master uses standard Ethernet Medium Access Controllers (MACs) without extra communication processors. Thus an EtherCAT master can be implemented on any equipment controller that provides an Ethernet interface, independently of the operating system or application environment. The EtherCAT slave uses an EtherCAT Slave Controller (ESC) for processing the data on-the-fly. Thus the performance of the network is not determined by the microcontroller performance of the slave but is handled complete in hardware. A process data interface (PDI) to the slave's application offers a Dual-Port-RAM (DPRAM) for data exchange.

Precise synchronization is particularly important in a wide range of distribution processes that require simultaneous actions, such as when several servo axes are performing simultaneous tasks. Precise calibration of distributed clocks is the most effective solution for synchronization. In the communication system, the stepwise calibration clock has the tolerance of error delay to a certain extent, compared with the fully synchronous communication.

7.1.2 Work counter WKC

The end of each EtherCAT message has a 16-bit working counter, WKC. WKC is a working counter used to record the number of reads and writes to the EtherCAT slave device. The EtherCAT slave controller calculates WKC in the hardware. The master receives the return data and checks the WKC in the sub-message. If WKC is not equal to the expected value, the sub-message has not been processed correctly. When a sub-message passes through a certain slave node, WKC will be increased by 1 if it is a single read or write operation. If it is a read and write operation, WKC will be increased by 1 upon read success, by 2 upon write success and by 3 upon complete. WKC is the accumulation of the processing results of each slave. The description of WKC increment is shown in Table 7-1.

Command	Data type	Increment
Deed	Read failed	_
Read	Read succeeded	+1
Write	Write failed	_
	Write succeeded	+1
	Failed	_
	Read succeeded	+1
Read/write	Write succeeded	+2
	Read and write succeeded	+3

Table 7-1 WKC	increment
---------------	-----------

7.1.3 Addressing mode

EtherCAT communication is realized by the master sending EtherCAT data frames to read and write the internal storage area of the slave device. EtherCAT messages use multiple addressing modes to operate the ESC internal storage area for multiple communication services. The addressing mode of EtherCAT is shown in Figure 7-1. An EtherCAT network segment is equivalent to an Ethernet device. The master first uses the MAC address of the Ethernet data frame header to address the network segment, and then uses the 32-bit address in the EtherCAT sub-message header to address the device in the segment. There are two ways to achieve in-segment addressing: device addressing and logical addressing. Device addressing performs read and write operations for a certain slave node. Logical addressing is oriented to process data and can be multicast. The same sub-message can read and write multiple slave devices.



Figure 7-1 Addressing mode of EtherCAT

7.1.3.1 Segment addressing

Depending on how the EtherCAT master and its segment are connected, the segment can be addressed in two ways.

1. Direct connection mode

An EtherCAT segment is directly connected to the standard Ethernet port of the master device, as shown in Figure 7-2. In this case, the master uses the broadcast MAC address and the EtherCAT data frame is shown in Figure 7-3.



Figure 7-2 EtherCAT segment in direct connection mode

6 bytes	6 bytes	2 bytes	2 bytes	44-1498 bytes	4 bytes
Destination address: FF FF FF FF FF FF	Source address: FF FF FF FF FF FF	Frame type (0x88A4)	EtherCAT message header	EtherCAT data	PCS



2. Open mode

EtherCAT segment is connected to a standard Ethernet switch, as shown in Figure 7-4. In this case, a segment needs a MAC address and the address in the EtherCAT data frame sent by the master is the MAC address of the segment it controls, as shown in Figure 7-5. The first slave device in the EtherCAT segment has an ISO/IEC 8802.3 MAC address, which represents the entire segment. This slave is called a segment address slave, which can exchange the destination address area and source address area in the Ethernet. If EtherCAT data frame is sent over UDP, the device will also exchange the source and destination IP addresses and the source and destination UDP port numbers, making the response frame fully complied with the UDP/IP protocol.



Figure 7-4 EtherCAT segment in open mode

6 bytes	6 bytes 2 bytes 2 bytes		44-1498 bytes	4 bytes	
Destination address: Segment MAC address	Source address: Master MAC address	Frame type (0x88A4)	EtherCAT message head	EtherCAT data	PCS

Figure 7-5 Addressing mode of EtherCAT in open mode

7.1.3.2 Device addressing

During device addressing, the 32-bit address in the EtherCAT sub-message header is divided into a 16-bit slave device address and a 16-bit slave device internal physical storage space address, as shown in Figure 7-6. The 16-bit slave device address can address 65535 slave devices, and each device can have up to 64 local address spaces.

Only one unique slave device is addressed per message in the device addressing mode, but there are two different mechanisms for addressing devices.



Figure 7-6 EtherCAT device addressing structure

3. Sequential addressing

For sequential addressing, the address of a slave is determined by its connection location within the segment, with a

negative number indicating the location of each slave within the segment as determined by the wiring sequence. When the sequential addressing sub-message passes through each slave device, its sequential address is increased by 1. When the slave receives a message, the message with a sequential address of 0 is the message addressed to it. This mechanism is also known as "automatic incremental addressing" because it updates the device address as the message passes through.

In Figure 7-7, there are three slave devices in the segment that are sequentially addressed as 0, -1, -2, and so on. When the master uses sequential addressing to access the slave, the address change of the sub-message is shown in Figure 7.8. The master station sends 3 sub-messages to address 3 slave nodes, where the addresses are 0, -1 and -2 respectively, and the data frame is 1 as shown in the figure. When the data frame reaches the slave ①, the slave ① checks that the address in sub-message 1 is 0, thus knowing that sub-message 1 is the message addressed to itself. After the data frame passes through the slave ①, all sequential addresses are increased by 1, called 1, 0 and -1, as shown in the data frame 2 in Figure 7-8. When the data frame reaches the slave ②, the slave ② finds that the address in sub-message. Similarly, subsequent slave nodes are addressed in this way. As shown in Figure 7.7, in actual engineering applications, sequential addressing is mainly used in the startup phase, and the master node configures a site address for each slave node. After that, the slave node can be addressed using a site address that is independent of their physical location. The sequential addressing mechanism can be used to automatically address the slave node, as shown in Figure 7-8.



Figure 7-7 Sequentially addressed slave address



Figure 7-8 Change of sub-message address during sequential addressing

4. Setting addressing

When setting addressing, the slave node address is independent of its sequential order within the network segment. As shown in Figure 7-9, the address can be configured by the master to the slave in the data link start-up phase, or loaded by the configuration data of the slave in the power-on initialization phase, and then read by the master in the link start-up phase using the sequential addressing mode to set the address of each slave node. Its message structure is shown in Figure 7-10.



Figure 7-9 Slave address in setting addressing mode

	Sub-message 1		e 1	Sub-message 2		Sub-message 3	
Data frame 1		1000		1234		5678	

Figure 7-10 Message structure in setting addressing mode

5. Logic addressing

For logical addressing, the slave address is not defined separately, but using a section of the 4GB logical address space in the addressing section. The 32-bit address area within the message is used as the overall data logical address to complete the logical addressing of the device. The logical addressing mode is implemented by the Fieldbus Memory Management Unit (FMMU). The FMMU function is located inside each ESC and maps the local physical storage address of the slave to the logical address of the segment. The schematic diagram is shown in Figure 7-11.



Figure 7-11 FMMU operating Principle

When receiving an EtherCAT sub-message of data logic addressing, the slave device will check for an FMMU unit address match. If the match exists, the slave device will insert the input type data into the corresponding position in the EtherCAT sub-message data area, and extracts the output type data from the corresponding position in the EtherCAT sub-message data area.

7.1.4 Distributed clocks

7.1.4.1 Concepts

In applications with spatially distributed processes requiring simultaneous actions, exact synchronization is particularly important. For example, this is the case for applications in which multiple servo axes execute coordinated movements. With this mechanism, the slave device clocks can be precisely adjusted to this reference clock. The first slave connected to the master with distributed clocking functions acts as a reference clock to synchronize the slave clocks of the other devices and the master. To achieve precise clock synchronization control, it is necessary to measure and calculate the data transmission delay and local clock offset, and to compensate for the drift of the local clock. The following 6 concepts are involved in the synchronization of the clock.

1. System time

The system time is the system timing used by the distributed clock. It starts at 0:00 on January 1, 2001, and is expressed in a 64-bit binary variable in nanoseconds (ns) and can be timed for up to 500 years. It can also be expressed as a 32-bit binary variable with a maximum of 4.2s, which is usually used for communication and time stamping.

2. Reference clock and slave clock

The EtherCAT protocol defines the first slave connected to the master with distributed clocking functions acts as a reference clock, and the clocks of other slave nodes are called slave clocks. The reference clock is used to synchronize the slave clocks and the master clock of other slave devices. The reference clock provides the EtherCAT system time.

3. Master clock

The EtherCAT master station also has a timing function, which is called the master clock. The master clock can be synchronized as a slave clock in a distributed clock system. In the initialization phase, the master can send the master clock to the reference clock slaves in system time format, which enables the distribution clocks to be timed using system time.

4. Local clock, initial offset and clock drift

Each DC slave has a local clock, which runs independently and is timed using the local clock signal. When the system starts, there is a certain difference between the local clock and the reference clock of each slave, which is called the initial clock offset. During operation, due to the fact that the reference clock and the DC slave clock use their own clock sources, their timing cycles drift to a certain extent, which will lead to the clock running out of sync and the local clock drifting. Therefore, the initial clock offset and clock drift must be compensated.

5. Local system time

The local clock of each DC slave generates a local system time after compensation and synchronization. The distributed clock synchronization mechanism is to keep the local system time of each slave consistent. The reference clock is also the local system clock of the corresponding slave.

6. Transmission delay

There will be a certain delay when data frames are transmitted between slaves, which includes device internal and physical connection delays. Therefore, when synchronizing slave clocks, the transmission delay between the reference clock and multiple slave clocks should be considered.

7.1.4.2 Clock synchronization process

Clock synchronization consists of the following three steps:

• Transmission delay measurement

When the distributed clock is initialized, the master will initialize the transmission delay for slave nodes in all directions, calculate the deviation value between the slave clocks and the reference clock, and write it into the slave clock.

• Reference clock offset compensation (system time)

The local clock of each slave will be compared with the system time, and then different comparison results will be written into different slaves, so that all slaves will get the absolute system time.

Reference clock drift compensation

Clock drift compensation and local time are used to periodically compensate for local clock errors and fine-tuning. The following figure illustrates two application cases of compensation calculation. Figure 7-12 shows a case where the system time is less than the slave local clock. Figure 7-13 shows a case where the system time is greater than the slave local clock.

1. System time < local time



Figure 7-12 Clock synchronization: system time < local time

2. System time > local time



Figure 7-13 Clock synchronization: system time > local time

With EtherCAT, data exchange is completely hardware-based. Due to the logic ring structure of communication (with the help of the physical layer of full-duplex fast Ethernet), the master clock can simply and accurately determine the delay offset of slave clock propagation, and vice versa. The distributed clocks are adjusted based on this value, which indicates that a very precise deterministic synchronization error time base (less than 1 microsecond) can be used across the network. Its structure is shown in Figure 7-14.


Figure 7-14 Clock synchronization principle

For example, there is a difference of 300 nodes between the two devices, and the cable length is 120 meters. Use an oscilloscope to capture the communication signal, and the result is shown in Figure 7-15.



Figure 7-15 Performance test of clock synchronization

This function is very important for motion control. In such applications, velocity is typically derived from the measured position. Even very small jitter in the position measurement timing can translate to larger inaccuracies in the calculated velocity, especially relative to short cycle times. In EtherCAT, the introduction of time-stamped data types as a logical extension allows high-resolution system times to be added to the measured value, which is made possible by the huge bandwidth that Ethernet provides.

7.1.5 EtherCAT cable redundancy

Increasing demands in terms of system availability are catered for with optional cable redundancy that enables devices to be exchanged without having to shut down the network. Adding redundancy is very inexpensive: the only additional hardware is another standard Ethernet port (no special card or interface) in the master device and the single cable that turns the line topology into the ring. Switchover in case of device or cable failure only takes one cycle, so even demanding motion control applications survive a cable failure without problems.

EtherCAT also supports redundant masters with hot standby functionality. Since the EtherCAT Slave Controllers immediately return the frame automatically if an interruption is encountered, failure of a device does not necessarily lead to the complete network being shut down. For example, the standard EtherCAT topology is shown in Figure 7-16 a). If there is a network interruption between Slave2 and SlaveN-2 in this topology (the red part in the figure), all slave communication after Slave N-2 is interrupted accordingly. This is also the disadvantage of the standard topology.



a) Standard EtherCAT topology

b) EtherCAT redundant topology



Figure 7-16 b) shows the topology structure of the EtherCAT redundancy mode. Only two standard network ports are needed for the master to realize the topology. With these two ports, all slave nodes can form a loop. Even if the network is interrupted while in use, such as the disconnected red part in Figure 7-16, the master node will detect the error immediately and automatically divide the communication into two channels, and all the slave nodes can continue to communicate to ensure the stable operation of the system.

7.2 EtherCAT communication mode

In actual automation control systems, there are usually two forms of data exchange between applications: time-critical and time-non-critical. Time critical indicates that a specific action must be completed within a certain time window. If the communication cannot be completed within the required time window, it may cause control failure. Time-critical data is usually sent periodically, which is called periodic process data communication. Non-time-critical data can be sent out of cycle, and non-periodical mailbox data communication is used in EtherCAT.

7.2.1 Periodic process data communication

The master node can use logical read, write or read and write commands to control multiple slaves at the same time. In the periodic data communication mode, the master and the slave have multiple synchronous operation modes.

1) Slave device synchronization mode

♦ Free running

In free-run mode, the local control cycle is generated by a local timer interrupt. The cycle time can be set by the master, which is an optional feature of the slave. The local cycle in free-running mode is shown in Figure 7-17. In the figure, T1 is the time for the local microcontroller to copy data from the EtherCAT slave controller and calculate the output data; T2 is the output hardware delay, and T3 is the input latch offset time. These parameters reflect the time response performance of the slave.



Figure 7-17 Local cycle in free-running mode

♦ Synchronization to data or output events

The local cycle is triggered on the occurrence of a data input or output event, as shown in Figure 7-18. The master can write the sending cycle of the process data frame into the slave. The slave will check if this cycle time is supported or if the cycle time is optimized locally. The slave can choose to support this feature. It is usually synchronized to the data output event. If the slave only has input data, the data is synchronized to the input event.



Figure 7-18 Local cycle of synchronization to data input or output events

♦ Synchronization to distributed clock synchronization event

The local cycle is triggered by the SYNC event, as shown in Figure 7-19. The master must complete the transmission of the data frame before the SYNC event. For this reason, the master clock must also be synchronized with the reference clock.



Figure 7-19 Local cycle of synchronization to SYNC event

To further optimize slave station synchronization performance, the master should copy the output information from the received process data frame when a data transmission and reception event occurs. After the SYNC signal arrives, continue the local operation. As shown in Figure 7-20, the data frame must arrive T1 time earlier than the SYNC signal. The slave has completed data exchange and control calculations before the SYNC event and can perform the output operation immediately after receiving the SYNC signal, further improving synchronization performance.



Figure 7-20 Local cycle of the optimized synchronization to SYNC event

2) Master device synchronization mode

There are two synchronization modes for the master.

♦ Cyclic mode

In cyclic mode, the master periodically sends process data frames. The master's cycle is usually controlled by a local timer. The slave node can run in free-running mode or in synchronization to received data event mode. For the slave in synchronization mode, the master should check that the cycle time of the corresponding process data frame is greater than the minimum cycle time supported by the slave.

The master can send a variety of periodic process data frames at different cycle times to get the most optimized bandwidth. For example, a shorter cycle is used to send motion control data and a longer cycle is used to send I/O data.

♦ DC mode

The master runs in DC mode similarly to cyclic mode, except that the local cycle of the master should be synchronized with the reference clock. The master's local timer should be adjusted based on the ARMW message that publishes the reference clock. After the ARMW message used to dynamically compensate clock drift is returned to the master, the master clock can be adjusted based on the read back reference clock time to be roughly synchronized with the reference clock time.

In DC mode, all DC-enabled slaves should be synchronized to the DC system time. The master should also synchronize the other communication cycles with the DC reference clock time. Figure 7-21 shows how the local cycle is synchronized with the DC reference clock.



Figure 7-21 Master DC mode

The master local run is started by a local timer. The local timer should have an advance over the DC reference clock timing, which is the sum of the following times.

- 1. Control program execution time
- 2. Data frame transmission time
- 3. Data frame transmission delay (D)
- 4. Additional offset (U) (Related to the jitter value of the delay time of each slave and the jitter value of the control program execution time, used for the adjustment of the master cycle)

7.2.2 Non-periodic mailbox data communication

The non-periodical data communication in the EtherCAT protocol is called mailbox data communication, which can be carried out in both directions, i.e. from the master to the slave and from the slave to the master. It supports full duplex, two-way independent communication and multi-user protocols. The slave-to-slave communication is managed by the master as a router. The mailbox communication data header includes an address field that enables the master to resend mailbox data. Mailbox data communication is a standard way of realizing parameter exchange, and is used if periodic process data communication or other non-periodic services need to be configured.

The mailbox data message structure is shown in Figure 7-22. Usually the mailbox communication value corresponds to a slave station, so the device addressing mode is used in the message. The data elements in its data header are listed in Table 7-2.



Figure 7-22 Mailbox data unit structure

Data element	Bit	Description				
Length	16 bits	Length of the followed mailbox service data				
Address	16 bits	Slave address of data source for master-to-slave communication Slave address of data destination for master-to-slave communication				
Channel	6 bits	Reserved				
Priority	2 bits	Reserved				
Туре	4 bits	Mailbox type, i.e. type of subsequent protocol. 0: Mailbox communication error 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)				
Counter (Ctr)	4 bits	Sequence number used for repeated detection, increasing by 1 for each new mailbox service (Only 1 to 7 is used for compatibility with older versions)				

Table 7-2 Mailbox data header

♦ Master-to-slave communication – write mailbox command

The master sends the write data area command to send mailbox data to the slave. The master will check the work counter WKC in the slave's answer message of mailbox command. If the work counter is 1, the write command is successful. Conversely, if the work counter is not increased, which is usually because the slave did not finish reading the previous command, or did not respond within a limited time, the master must resend the write mailbox data command.

♦ Master-to-slave communication – read mailbox command

To be sent from the slave to the master, the data must first be written to the input mailbox cache and then read by the master. If there is valid data waiting to be sent from the slave ESC input mailbox data area, the master will send the appropriate read command to read the slave data as soon as possible. There are two ways for the master to determine whether the slave has filled the mailbox data into the input data area. One is to use FMMU to periodically read a certain flag bit. Logical addressing can be used to read the flags of multiple slave s, but the disadvantage is that each slave requires an FMMU unit. The other way is to input a simple rotation training ESC into the input area of the mailbox. An increase of 1 in the work counter of the read command indicates that the slave has populated the input data area with new data.

7.3 EtherCAT state machine

EtherCAT State Machine (ESM) coordinates the state of the master and slave applications at initialization and runtime.

The EtherCAT device must support four states and an optional state.

- ♦ Init: initialization, abbreviated as I.
- Pre-Operational: abbreviated as P.
- ♦ Safe-Operational: abbreviated as S.
- ♦ Operational: abbreviated as O.
- ♦ Boot-Strap: (Optional) abbreviated as B.

The conversion relationship between the above states is shown in Figure 7-23. When the state is converted from the initialization state to the operational state, the conversion must be done in the order of "Init > Pre-Operational > Safe-Operational > Operational > Boot-Strap". The leapfrog conversion is only available when returning from the Operational state. The Boot-Strap state is optional and is only allowed to convert to and from the Init state. All state

changes are initiated by the master node, which sends a state control command to the slave to request a new state, and the slave responds to this command by performing the requested state conversion and writing the result to the slave state indicator variable. If the requested state conversion fails, the slave will give an error flag. Table 7-3 shows the summary of state conversions.



Figure 7-23 EtherCAT state conversion

♦ Init

The initialization state defines the initial communication relationship between the master and the slave at the application layer. At this time, the master and the slave cannot communicate directly at the application layer, and the master uses the initialization state to initialize some configuration registers of the ESC. If the master supports mailbox communication, configure the mailbox communication parameters.

Pre-Operational

In Pre-Operational state, mailbox communication is activated. The master and slave can use mailbox communication to exchange application-related initialization operations and parameters. Process data communication is not allowed in this state.

♦ Safe-Operational

In Safe-Operational state, the slave application reads the input data, but does not generate an output signal. The device has no output and is in a "safe state". In this case, mailbox communication is still available.

♦ Operational

In Operational state, the slave application reads data, the master application sends out output data, and the slave device generates an output signal. In this case, mailbox communication is still available.

♦ Boot-Strap

The function of the boot strap state is to download the device firmware program. The master can download a new firmware program to the slave using FoE protocol mailbox communication.

State and state conversion	Description
Init	There is no communication at the application layer, and the master can
i int	only read and write ESC registers.
	The master configures the slave site address register.
Init to Pre-OP	Configure mailbox channel parameters if mailbox communication is
	supported.
(IP)	Configure DC related registers if distributed clocks are supported.
	The master writes state control register to request "Pre-Op" state.
Pre-Operational	Mailbox data communication at application layer

Table 7-3 State conversion of EtherCAT state machine

State and state conversion	Description		
	The master uses mailboxes to initialize process data mapping.		
Dro On to Safa On (DS)	The master configures the SM channel used for data communication.		
	The master configures FMMU.		
	The master writes state control register to request "Safe-Op" state.		
Sofo Operational	The master sends valid output data.		
Sale-Operational	The master writes state control register to request "Op" state.		
Operational	All inputs and outputs are valid.		
Operational	Mailbox communication is still available.		

7.4 EtherCAT servo drive controller application protocol

IEC 61800 standard series is a general specification for variable speed electronic power drive systems. IEC 61800-7 defines the standard of communication interface between control system and power drive system, including network communication technology and application profile, as shown in Figure 7-24. EtherCAT, as a network communication technology, supports the profile CiA 402 in the CANopen protocol and the application layer of the SERCOS protocol, which are called CoE and SoE respectively.



Figure 7-24 IEC 61800-7 architecture

7.4.1 EtherCAT-based CAN application protocol (CoE)

CANopen device and application profiles are available for a wide range of device classes and applications, ranging from I/O components, drives, encoders, proportional valves and hydraulic controllers to application profiles for plastic or textile machinery, for example. EtherCAT can provide the same communication mechanisms as the familiar CANopen mechanisms: object dictionary, PDO (process data objects) and SDO (service data objects) – even the network management is comparable. EtherCAT can thus be implemented with minimum effort on devices equipped with CANopen. Large parts of the CANopen firmware can be reused. Objects can optionally be expanded in order to account for the larger bandwidth offered by EtherCAT.

The EtherCAT protocol supports the CANopen protocol at the application level and is supplemented by the following main features:

- Network initialization by accessing the CANopen object dictionary and objects using mailbox communication
- Network management by using CANopen application objects and optional time-driven PDO messages.
- Mapping process data, cyclic transmission command data and state data by object dictionary.

Figure 7-25 shows the CoE device structure whose communication modes mainly include periodic process data communication and non-periodic data communication. The following section will introduce the differences between both modes in practical applications.



Figure 7-25 CoE device structure

7.4.1.1 CoE object dictionary

The CoE protocol fully complies with the CANopen protocol and has the same object dictionary definition as shown in Table 7-4.

Table 7-5 lists the CoE communication data objects, which extend the relevant communication objects 0x1C00–0x1C4F for EtherCAT communication to set the type of storage synchronization manager, communication parameters and PDO data allocation.

Index number range	Description		
0x0000-0x0FFF	Data type description		
0x1000–0x1FFF	Communication objects include: device type, identifier, PDO mapping, CANopen-compatible data object for CANopen. EtherCAT extension data object is reserved in EtherCAT.		
0x2000–0x5FFF	Manufacturer definition object		
0x6000-0x9FFF	Profile definition data object		
0xA000-0xFFFF	Reserved		

Table 7-4 CoE object dictionary definition

Table 7-5 CoE c	communication data object
-----------------	---------------------------

Index	Description		
0x1000	Device type		
0x1001	Error register		
0x1008	Vendor device name		
0x1009	Manufacturer hardware version		
0x100A	Manufacturer software version		
0x1018	Device identifier		
0x1600–0x17FF	RxPDO mapping		
0x1A00–0x1BFF	TxPDO mapping		
0x1C00	Sync manager communication type		
0x0x1C10-0x1C2F	Process data communication sync manager PDO assignment		
0x0x1C30-0x1C4F	Synchronization management parameters		

7.4.1.2 CoE periodic process data communication (PDO)

In periodic data communication, the process data can contain multiple PDO mapping data objects. The data objects 0x1C10 to 0x1C2F used by the CoE protocol define the corresponding PDO mapping channels. Table 7-6 shows the specific structure of the communication data in the EtherCAT protocol.

Table 7-6 CoE	communication data object
---------------	---------------------------

Index	Object type	Description	Туре
0x1C10	Array	SM0 PDO assignment	Unsigned integer 16-bit
0x1C11	Array	SM1 PDO assignment	Unsigned integer 16-bit
0x1C12	Array	SM2 PDO assignment	Unsigned integer 16-bit
0x1C13	Array	SM3 PDO assignment	Unsigned integer 16-bit
0x1C2F	Array	SM31 PDO assignment	Unsigned integer 16-bit

The following uses the allocation for SM2 PDO (0x1C12) as an example and Table 7-7 lists its value. If two data are mapped in PDO0, the first communication variable will be the control word with the corresponding mapped index and sub-index address 0x6040:00, and the second communication variable is the target position value with the corresponding mapped index and sub-index address 0x607A:00.

0X1C12	Numeric	PDO data object mapping			
Sub-index	value	Sub-index	Numeric value	Bytes	Description
0	3			1	Number of PDO mapping objects
	PDO0	0	2	1	Number of data mapping data objects
1	0x1600	1	0x6040: 00	2	Control word
		2	0x607A: 00	4	Target position

Table 7-7 Example of SM2 channel PDO assign object data 0x1C12

0X1C12	Numeric	PDO data object mapping				
Sub-index	value	Sub-index	Numeric value	Bytes	Description	
1	PDO1 0x1601	0	2	1	Number of data mapping data objects	
		1	0x6071: 00	2	Target torque	
		2	0x6087: 00	4	Target ramp	
1	PDO2 0x1602	0	2	1	Number of data mapping data objects	
		1	0x6073: 00	2	Max. current	
		2	0x6075: 00	4	Motor rated current	

There are several PDO mapping modes:

- 1. Simple devices do not require mapping protocols
- Use simple process data
- Read in the EEPROM of the slave
- 2. Readable PDO mapping
- Fix process data mapping
- Read with SDO communication
- 3. Selectable PDO mapping
- Multiple fixed PDO groups are selected by object 0x1C1X
- Read through SDO communication
- 4. Variable PDO mapping
- Configure through CoE communication

7.4.1.3 CoE non-periodic process data communication (SDO)

The EtherCAT master enables non-periodic data communication via reading and writing mailbox data SM channels. The CoE protocol mailbox data structure is shown in Figure 7-26.

8 bytes	2 bytes			1478 bytes at most
Mailbox data header type=3(CoE)	CoE command			Command-related data
	9 bit	3 bit	4 bit	
	Number	Reserved	Туре	

Figure 7-26 CoE data header

The numbered part in Figure 7-26 is explained in detail in Table 7-8.

Table	7-8 Col	command	definition
Table	1 0 000		acminuon

CoE command field	Description
No.	Number when PDO is sent
	Message type:
	0: Reserved
	1: Emergency information
	2: SDO request
	3: SDO response
Туре	4: TxPDO
	5: RxPDO
	6: Remote TxPDO send request
	7: Remote RxPDO send request
	8: SDO information
	9–15: Reserved

♦ SDO service

CoE communication service types 2 and 3 are SDO communication services, and the SDO data structure is shown in Figure 7-27.



Standard CANopen data frame

Figure 7-27 SDO data frame format

SDO usually has three transmission modes. Table 7-9 shows the specific content of the SDO data frame. Its structure is shown in Figure 7-28:

Fast transmission service: As with the standard CANopen protocol, only 8 bytes are used and up to 4 bytes of valid data can be transmitted.

Regular transmission service: More than 8 bytes can be used to transmit more than 4 bytes of valid data. The maximum valid data that can be transmitted depends on the storage area capacity managed by the mailbox SM.

Segmented transmission service: Use this service when the capacity of the mailbox is exceeded.

SDO control	Standard CANopen SDO service	
Index	Device object index	
Sub-index	Sub-index	
Data	Data in SDO	
Data (Optional)	There are four bytes of optional data that can be added to the data frame.	

Table 7-9 CoE data frame content

	Fast transmission	Regular tranmission	Segmented transmission
	Mailbox data header	Mailbox data header	Mailbox data header
	CoE	CoE	CoE
Mailbox storage capacity	Data < 4 bytes	4 bytes < Data < Mailbox size	Data > Mailbox size
			Mailbox data header
			CoE
			Mailbox data header
			CoE
			Mailbox data header
			CoE



If the data to be transmitted is larger than 4 bytes, the regular transmission service is used. In regular transmission, the 4 data bytes in the fast transmission mode will be used to indicate the full size of the data to be transmitted. The valid data is transmitted in the extended data section. The maximum size of the valid data is the mailbox capacity minus 16.

7.4.2 Servo drive profile according to IEC 61800-7-204 (SERCOS)

SERCOS is known as a real-time communication interface, especially for motion control applications. The SERCOS profile for servo drives is included in the international standard IEC61800-7-204. The mapping of this profile to EtherCAT is defined in section 304 of the standard. The service channel, including access to all drive-internal parameters and functions, is based on the EtherCAT mailbox. Here too, the focus is on compatibility with the existing protocol (access to value, attribute, name, units of the IDNs) and expandability with regard to data length limitation. The process data, with SERCOS in the form of AT and MDT data, are transferred using EtherCAT device protocol mechanisms. The mapping is similar to the SERCOS mapping. The EtherCAT slave state machine can also be mapped easily to the phases of the SERCOS protocol.

7.4.2.1 SoE state machine

A comparison between the communication phase of the SERCOS protocol and the EtherCAT state machine is shown in the Figure 7-29. The SoE state machine is featured as follows:

- 1. SERCOS protocol communication phase 0 and 1 are overwritten by EtherCAT initialization state.
- 2. Communication phase 2 corresponds to the operational state, allowing the use of mailbox communication to implement the service channel and operate IDN parameters.
- 3. Communication phase 3 corresponds to the safe operational state and starts transmitting periodic data, where only input data is valid and output data is ignored, implementing clock synchronization.
- 4. Communication phase 4 corresponds to the operational phase, where all inputs and outputs are valid.
- 5. Phase switching process commands S-0-0127 (communication phase 3 switchover check) and S-0-0128 (communication phase 4 switchover check) that do not use the SERCOS protocol are replaced by PS and SO state conversion respectively.
- 6. The SERCOS protocol only allows switching down from the advanced communication phase to communication phase 0, whereas EtherCAT allows any state switching down (as shown in a) in Figure 7-29. For example, switching from the operational state to the safe operational state or from the safe operational state to the pre-operational state. The SoE should also support this switchover as shown in b) in Figure 7-29. If the slave does not support this switchover, set the error bit in the EtherCAT AL state register.



Figure 7-29 SoE state machine

7.4.2.2 IDN inheritance

The SoE protocol inherits the DIN parameter definition of the SERCOS protocol. Each IDN parameter has a unique 16-bit IDN, which corresponds to a unique data block that holds all information about the parameter. The data block consists of 7 elements, as listed in Table 7-10. The IDN parameters are divided into standard data and product data, and each part consists of eight parameter groups with different IDN, as listed in Table 7-11.

No.	Name
Element 1	IDN
Element 2	Name
Element 3	Attribute
Element 4	Unit
Element 5	Minimum allowable value
Element 6	Maximum allowable value
Element 7	Data value

Table 7-11 IDN number definition

Bit	15	14-12	11-0
Meaning	Classification	Parameter group	Parameter number
Value	0: Standard data (S) 1: Product data (P)	0–7: 8 parameter groups	0000-4095

When using EtherCAT as a communication network, some IDNs in the SERCOS protocol for communication interface control have been deleted, as listed in Table 7-12. And some IDN has been modified, as listed in

Table 7-13.

Table 7-12 Deleted IDN

IDN	IDN description
S-0-0003	Minimum start time of AT sending
S-0-0004	Time between sending and receiving state switching
S-0-0005	Minimum feedback sampling lead time
S-0-0009	Start address in the master data message

IDN	IDN description
S-0-0010	Master data message length
S-0-0088	Recovery time required for receiving MSTs after receiving MDTs
S-0-0090	Command processing time
S-0-0127	Communications phase 3 switchover check
S-0-0128	Communications phase 4 switchover check

Table 7-13 Modified IDN

IDN	Original description	Updated description
S-0-0006	Start time of AT sending	Time offset in which an application writes AT data to ESC memory after a synchronization signal within the slave.
S-0-0014	Communication interface state	Map slave DL state and AL state code.
S-0-0028	MST error technology	Map the slave RX error counter to the loss counter.
S-0-0089	Start time of MDT sending	Time offset of obtaining MDT data from ESC memory after a synchronization signal within the slave.

7.4.2.3 SoE periodic process data

Output process data (MDT data content) and input process data (AT data content) are configured by S-0-0015, S-0-0016 and S-0-0024. The process data only includes periodic process data, but not service channel data. The output process data includes servo control words and command data, while the input process includes status words and feedback data. S-0-0015 sets the type of periodic process data, as listed in Table 7-14, and the definition of parameters S-0-0016 and S-0-0024 are listed in Table 7-15. The master writes these three parameters via mailbox communication during the Pre-Operational phase to configure the contents of the periodic process data.

S-0-0015	Command data	Feedback data
0: Standard type 0	None	No feedback data
1: Standard type 1	Torque command S-0-0080 (2 bytes)	No feedback data
2: Standard type 2	Speed command S-0-0036 (4 bytes)	Speed feedback S-0-0053 (4 bytes)
3: Standard type 3	Speed command S-0-0036 (4 bytes)	Position feedback S-0-0051 (4 bytes)
4: Standard type 4	Position command S-0-0047 (4 bytes)	Speed feedback S-0-0053 (4 bytes)
5: Standard type 5	Position command S-0-0047 (4 bytes) Speed command S-0-0036 (4 bytes)	Position feedback S-0-0051 (4 bytes) Or speed feedback S-0-0053 (4 bytes) + Position feedback S-0-0051 (4 bytes)
6: Standard type 6	Speed command S-0-0036 (4 bytes)	No feedback data
7: Custom	S-0-0024 configuration	S-0-0016 configuration

Data word	S-0-0024 definition	S-0-0016 definition
0	Maximum length of output data (Word)	Maximum length of input data (Word)
1	Actual length of output data (Word)	Actual length of input data (Word)
2	First IDN of command data mapping	First IDN of feedback data mapping
3	Second IDN of command data mapping	Second IDN of feedback data mapping

Table 7-15 Definition of parameters S-0-0016 and S-0-0016

7.4.2.4 SoE non-periodic service channels

The EtherCAT SoE Service Channel (SSC) is done by the EtherCAT mailbox communication function, which is used for non-periodic data exchange, such as reading and writing IDNs and their elements. The SoE data header format is shown in Figure 7-30.



Figure 7-30 SoE data header format



Data area	Description					
	Command type:					
	0x01: Read request					
	0x02: Read response					
Command	0x03: Write request					
Commanu	0x04: Write response					
	0x05: bulletin					
	0x06: Slave information					
	0x07: Reserved					
	Subsequent data signal:					
Subsequent data	0x00: No subsequent data frame					
	0x01: Transmission incomplete, with subsequent data frame					
	Error signal:					
Error	0x00: No error					
	0x01: Error occurred, 2-byte error code in data area					
Address	Specific address of the slave device					
Operation element	Element selection for single element operation, defined by bit, with each bit					
identification	corresponding to one element.					
	Number of elements for addressing constructs					
IDN	IDN number of the parameter, or the remaining segments during the segment operation					

Commonly used SSC operations include SSC read operations, SSC write operations, and process commands.

- SSC read operation: The master initiates the SSC read operation and writes the SSC request to the slave. After
 receiving the read operation request, the slave responds with the requested IDN number and data value. The master
 can read multiple elements at the same time, so the slave should answer multiple elements. If the slave only
 supports single element operation, it should respond with the first element requested.
- SSC write operation: This operation is used to download data from the master to the slave, which should answer with the result of the write operation. Segment operation consists of one or more segmented write operations and an SSC write response service.
- SSC process command: A process command is a special non-periodic data. Each process command has a unique IDN and specified data elements, which are used to start certain specific functions or processes of the servo device. It usually takes a while to execute these functions or processes. The process command only triggers the start of the process, so after that, the service channel it occupies will become immediately available for the transfer of other non-periodic data or process commands. There is no need to wait until the triggered functions or processes to complete their execution.

8 Application Programming

8.1 Single axis control

8.1.1 Single axis control programming description

The motion control of the AX series controller with the servo axis (such as DA200) is implemented based on the EtherCAT bus network. Each EtherCAT bus cycle will perform a calculation and issue a control command to control the servo. Different from the previous pulse control mode, EtherCAT bus is entirely based on the software. Pay attention to the following points when applying:

- MC-related POUs should be configured to execute under the EtherCAT task. Most MC function blocks cannot run normally when placed in the POU of the low-priority Main tasks.
- The PDO configuration table needs to be configured with relevant data objects. Otherwise the servo will not be able to run due to the missing communication data object configuration. No error alarm will be generated for this case, making it more difficult to troubleshoot.
- The controller can set the parameters of the servo by configuring SDO.
- MC function block instance can only be used for a unique servo axis control. Error occurs if it is used for multiple servo axis controls.
- MC function block must be used to monitor the running servo axis to avoid error caused by program logic jump without MC function block monitoring. Such error is usually difficult to detect.
- Pay attention to the safe handling of the debugging, and ensure that the signal configuration is consistent with the practical application. If the servo system uses incremental encoder, zeroing is required prior to normal operation. For movements within a limited range (e.g. a screw), limit and safety signals should be set.

8.1.2 MC function blocks commonly used for single-axis control

MC function block (FB) is also known as MC command. In fact, the object instance of MC function block is used in the user program, and the servo axis is controlled by MC object instance, for example:

MC Power1: MC Power;//Statement instance MC_Power1

MC_Power1 (Axis=Axis1,)

Single-axis control is generally used for positioning control, that is, the servo motor drives the external mechanism to move to the specified position. Sometimes the servo is required to run at a specified speed or torque. In single-axis control, the following MC function blocks are commonly used:

Control operation	Required MC command	Description
Enable servo	MC_Power	Run this command to enable the servo axis to perform subsequent running control.
Absolute positioning	MC_MoveAbsolute	Command the servo to run to a specified coordinate point.
Relative positioning	MC_MoveRelative	Runs the specified distance with the current location as a reference.
Servo jog operation	MC_Jog	The jog operation of the servo motor is often used for low-speed test runs to inspect equipment or adjust the position of the servo motor.

Table 8-1 MC function blocks commonly used for single-axis control

Control operation	Required MC command	Description
Relative superposition	MC_MoveAdditive	Based on the current running command of the servo, run the specified distance relatively.
Speed control	MC_MoveVelocity	Command the servo runs at the specified speed.
Servo suspend	MC_Halt	Command the servo to suspend operation. If MC_Movexxx is triggered again, the servo can run again.
Emergency stop	MC_Stop	Command the servo to stop. The servo can run again only after the stop command is reset and MC_Movexxx is triggered.
Alarm reset	MC_Reset	When the servo stops with an alarm, this command is used to reset the servo.
Servo homing	MC_Home	Command the servo to start homing operation. Both the home signal of the application system and the limit signals on both sides are connected to the DI port of the servo.
Controller homing	MC_Homing	Command the control system to start homing operation. Both the home signal of the application system and the limit signals on both sides are connected to the DI port of the controller.

8.2 Cam synchronization control

Electronic cam (abbreviation ECAM) utilizes the constructed cam curves to simulate the mechanical cam to meet the relative motion software system between main shaft and camshaft system the same to mechanical cam system. Electronic cams can be applied to various fields, such as automobile manufacturing, metallurgy, machining, textiles, printing, and food packaging. The electronic cam curve is a function curve with the main shaft pulse (active shaft) input as X and the corresponding output of the servo motor (camshaft) as Y=F(X).



Figure 8-1 Electronic cam diagram

The AX series programmable controller electronic cam function has the following features.

- CAM curves are easy to draw: Cams can be described by cam chart, CAM curves or array. It supports multiple cam chart selection and dynamic switching during running.
- CAM curves are easy to correct: The running cam table can be modified dynamically.
- Support one master and multiple slaves: one main shaft can have multiple slave shafts corresponding to it.
- Cam lifter: multiple cam lifters and multiple setting intervals are allowed.
- Cam clutch: It can make the cam enter and exit the cam running through the user program.

• Special functions: Virtual main shaft, phase offset and output superposition are supported.

Note: "online modification of CAM curve" refers to the modification of the key point coordinates of the CAM curve according to the needs of control characteristics during the execution of the program written by the user. The content to be modified is generally the key point coordinates, but it can also be the number of key points, the distance range of the main axis.

The AX series programmable controller electronic cam function contains three control elements:

- 1. Main shaft: Reference for synchronous control.
- 2. Slave shaft: a servo axis that follows the movement of the main shaft according to the non-linear characteristics.
- 3. Cam table: Data table or cam curve describing the relative position, range, periodicity of the master-slave shafts.

The commonly used function blocks related to electronic cam are listed in the following table.

MC Command	Description
MC CamTableSelect	Run this command to associate the main shaft, slave shaft and
MO_CannableCelect	cam table.
MC_CamIn	Let the slave shaft enter the cam running
MC_CamOut	Let the slave shaft exit the cam running
MC_Phasing	Main shaft phase modification

Table 8-2 Commonly used electronic cam function blocks

8.2.1 Periodic mode of the cam table

1. Single cycle mode (Periodic:=0): After the cam table cycle is completed, the slave shaft leaves the cam running state, as shown in Figure 8-2.



Figure 8-2 Single cycle mode

2. Periodic mode (Periodic:=1): After the cam table cycle is completed, the slave shaft will start the next cam cycle until the user program commands it to exit the cam running state, as shown in Figure 8-3.



Figure 8-3 Periodic mode

8.2.2 Input method of cam table

- 1. When creating a new cam table, the system will automatically generate the simplest cam curve, on which the user can edit and customize the CAM curve table.
- 2. User can increase or decrease the number of key points in the cam curve or change the coordinates of the key points.
- 3. The line pattern between the two key points of the cam curve can be set to a straight line or a quantic polynomial, and the system will optimally optimize each curve to minimize sudden changes in speed and acceleration.



Figure 8-4 CAM curve

8.2.3 Data structure of cam table

Invtmatic Studio contains data structure for each CAM table that describes the feature data of the CAM table. The following figure describes the data structure of the "CAM0" cam table. Please note the names of the variables in the structure.

Cam	Cam table	Tappets	Tappet table							
	Х	Y	V	А	J	Segment Type	min(Position)	max(Position)	max(Velocity)	max(Acceleration)
	0	0	0	0	0					
0						Poly5	0	120	1.51200000000	0.032835282941414
1	120	120	1	0	0					
0						Poly5	120	240	1	0
1	240	240	1	0	0					
0						Poly5	240	360	1.512	0.032835282941414
	360	360	0	0	0					

Figure 8-5 Data structure of cam table

Invtmatic Studio has an internal data structure to characterize the CAM table. We can also write a CAM table manually, or modify the CAM feature data by accessing the data structure.

Note: When we state the CAM0 cam table, the system automatically states the CAM0 data structure of the global variable type by default, along with the CAM0_A[i] array. For example, modify the number of key points or coordinates of the CAM0 cam table in the user program.

CAM0. nElements:=10; // Change the number of key points to 10.

CAM0. xEnd:=300; // Change the end point of the main shaft to 300.

//For example, modify the coordinates of two key points in the user program.

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 CAM table reference and switch

CAM table is stored in the controller with an array, which can be pointed to by specific MC_CAM_REF variable type, such as statement:

CAM table q: MC CAM REF;

You can assign a value to this variable, namely pointing it to a specific CAM table:

CAM table q:= Cam0; // Point to the required CAM table. CAM table q: MC CAM REF; // Cam table pointer; TableID: uint; // Cam table selection command that can be set by HMI; Case TableID of 0: CAM table q: = CAM table A; 1: CAM table q: = CAM table B; 2: CAM table q: = CAM table C; End_case MC_CamTableSelect_0 (//CAM relationship Master:= Virtual main shaft, Slave:= CAM slave shaft, CamTable:= CAM table q, Execute:= bSelect, // Rising edge triggers CAM table selection. Periodic:= TRUE, MasterAbsolute:=FALSE,

SlaveAbsolute:= FALSE);

In the above example, the assignment operation of the MC_CAM_REF variable can be used to switch multiple CAM tables.

Appendix A Project Instance

A.1 Controller and Goodrive20 Series VFD Configuration Example

The AX Series controller is now set up as the master and a Goodrive 20 Series VFD is set up as the slave. The controller uses the Modbus/RTU communication protocol with a two-wire RS485 physical layer and communicates with the VFD via the COM2 port. Let's write a small program that reads and writes the functional parameters of the Goodrive20 VFD with the upper computer.

 Select File > New Project from the menu to create a new standard project. Set the device to INVT AX7X, and select Structured Text (ST) as the programming language. Edit the project information as needed, as shown in the following figure.

	管 New Pr	oject						×	
	<u>C</u> ategories	s		Templates					
	Pr	Projects			HMI project	Standard project	Standard project w		
	A project o	ontaining (one device, one ap	plication, and an e	empty implement	ation for PLC_	PRG		
	Nama	Coodrin	- 20l						
	Location	D:\Invtr	natic Studio (Projec	t			~ .		
						OK	Cancel		
Stand	ard Projec	:t						×	
6	You are about to create a new standard project. This wizard will create the following objects within this project: - One programmable device as specified below - A program PLC_PRG in the language specified below - A cyclic task which calls PLC_PRG - A reference to the newest version of the Standard library currently installed.								
	Devie	ce	INVT AX7X (Sh	nenzhen INVT E	lectric Co., Lt	d.)		~	
	PLC_	PRG in	Structured Tex	xt (ST)				~	
						ОК		Cancel	

oject Information		
ile Summary Prope	erties Statistics Licensing Signing	
Company	INVT	
Title	RTU mode Goodrive20 Communication	
Version	1.0.0.0 🗌 Release	d
Library Categories		
Default namespace		
Author	232	
Description	RTU mode Goodrive20 Communication	~
		/
Library compatibility	Invtmatic Studio V1.0.2	~
The fields in bold lett	ers are used to identify a library.	
Automatically generate	Library Information' POUs	
Automatically generate	Project Information' POUs	
	OK Cancel	

2. Select **Tool** > **Library Repository** from the menu, and install the library file **CmpModbusRTU_Master2_1.0.0.3.library**, as shown in the following figure.

🎁 Library R	epository		×
Location	System (C:\ProgramData\Invtmatic Studio\Managed Libraries)	\sim	Edit Locations
Installed lib	raries:		Install
Company	INVT	\sim	Uninstall
	tiscellaneous) CmpHSIO_C JW/T CmpHSIO_M JW/T CmpModbusRTU_Master1 JW/T CmpModbusRTU_Master2 JW/T 	^	Export
	CmpModbusTCP_Master IN/T CmpModbusTCP_Slave IN/T CmpModbus_RTU_Slave1 IN/T CmpModbus_RTU_Slave2 IN/T	*	Find Details Trust Certificate
Group t	by category		Dependencies
Library Pr	ofiles		Close

3. Select Library Manager > Add Library to add the installed library to the application, as shown in the following figure.

1 Library Manager 🗙 📄 PLC_PRG		
Add Library 🗙 Delete Library 🛛 🚰 Properties 📷 Details 🛛 🗐 Placeholders 🛛 🎁 Libra	ary Repository 🕕 Icon le	gend
lame	Namespace	Effective version
🗉 💟 3SLicense = 3SLicense, 3.5. 14.0 (3S - Smart Software Solutions GmbH)	_3S_LICENSE	3.5.14.0
- 🤚 BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH)	BPLog	3.5.5.0
CAA Device Diagnosis = CAA Device Diagnosis, 3.5.15.0 (CAA Technical Workgroup)	DED	3.5.15.0
CmpModbusRTU_Master2, 1.0.0.3 (INVT)	CmpModbusRTU_Master2	1.0.0.3
i- 📒 IoStandard = IoStandard, 3.5.15.0 (System)	IoStandard	3.5.15.0
- E SM3_Basic = SM3_Basic, 4.5.1.0 (3S - Smart Software Solutions GmbH)	SM3_Basic	4.5.1.0
- 📒 SM3_CNC = SM3_CNC, 4.5.1.0 (3S - Smart Software Solutions GmbH)	SM3_CNC	4.5.1.0
- 🧾 SM3_Robotics = SM3_Robotics, 4.5.1.0 (3S - Smart Software Solutions GmbH)	SM3_Robotics	4.5.1.0
- 📙 SM3_Robotics_Visu = SM3_Robotics_Visu, 4.5.1.0 (3S - Smart Software Solutions GmbH)	SM3_Robotics_Visu	4.5.1.0
الله SM3_Transformation = SM3_Transformation, 4.5.1.0 (3S - Smart Software Solutions GmbH)	TRAFO	4.5.1.0
📒 Standard = Standard, 3.5.15.0 (System)	Standard	3.5.15.0
 SM3_Transformation = SM3_Transformation, 4.5.1.0 (3S - Smart Software Solutions GmbH) Standard = Standard, 3.5.15.0 (System) 	TRAFO Standard	4.5.1.0 3.5.15.0
CmoModhus871/ Master2, 1,0,0,3 (IW/7)		

4. Double-click the PLC_PRG and enter the following codes on the statement editor:

```
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 Enter the following code in the main code editor:

```
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=> ,
    Error1D2=> );
```

Here are some descriptions of the program. The program calls two function blocks of the CmpModbusRTU_Master2 library, ModbusRTU_Master_Init_COM2 and ModbusRTU_Master_Fun_COM2. ModbusRTU_Master_Init_COM2 is used to initialize the RTU Master2, where the baud rate is set to 19200, the data bit is 8, the stop bit is 1, the check bit is even check, and the timeout time is 1000ms. ModbusRTU_Master_Fun_COM2 is the enablement and specific application of the function module. The variable Fun_Code2 is the standard Modbus function code, Addr2 is the address of the VFD Goodrive20 function. For the address of other MODBUS functions, refer to the INVT Goodrive20 Series VFD product manual. Slave2 indicates the VFD slave address, which is set to 1 here.

Connect the VFD and the controller with the two-wire RS485, and then start the VFD. Set the function code P00.01 to 2 through the VFD keypad, so that the running command can be controlled by the upper computer through communication modes. Set P00.06 to 8 to select the MODBUS communication mode. Set the serial communication parameters of group P14 to make it consistent with the initial parameter settings of the upper computer, including baud rate, data bit, parity bit, slave address, timeout time.

Click the button on the toolbar to compile the code. After compiling, click the button on the toolbar to log in to the controller. Check that the controller digital tube has no error, the VFD Goodrive20 is connected to the controller smoothly, and the communication is normal. The upper computer interface is shown in the figure.

	👔 Library Manager 🛛 📄 PLC_PI	RG 🗙 💮 Device					
	Device.Application.PLC_PRG						
x	xpression	Туре	Value	Prepared value	Address	Comment	
Đ	ModbusRTU_Master_Fun_COM2	ModbusRTU_Master_Fun_COM2					
H	ModbusRTU_Master_Init_COM2	ModbusRTU_Master_Init_COM2					
ŧ	Ø DatePtr2	ARRAY [00] OF INT					
÷	input_registers_Ptr2	ARRAY [09] OF INT					
H	Ø CoilDataPtr2	ARRAY [09] OF BOOL					
÷	<pre>input_bits_Ptr2</pre>	ARRAY [09] OF BOOL					
	CoilSingleData2	INT	0				
	Fun Code2	INT	0				
	Addr2	UINT	0				
	DataCount2	LIINT	1				
	*	Carri	-				
							>
	8 bDone2=>, 9 Error2=>,	.= 1000,					
	10 ErrorID2=>); 11 ModbusRTU_Master_Fun_C	OM2 (
	<pre>10 ErrorID2=>); 11 ModbusRTU_Master_Fun_C 12 xExecute2 TAUE 13 Fun_Code2 0 14 Addr2 0 := A 15 Slave2 1 := 1, 16 DataCount2 1 17 CoilDataFraGoo</pre>	OM2(i= 1, := Fun_Code2_0, ddr2_0, := DataCount2_1, 00020557E0000S4ADR(CoilDataPtr:	2),				
	<pre>10 ErrorID2=>); 11 ModbusRIU_Master_Fun_C 12 xExecute2Imue 13 Fun_Code2 0 14 Addr20 == A 15 Slave21 == 1 16 DataCount21 17 CoilDataPtra@0 18 CoilSingleData 19 input_bisp_Fut 20 input_registers_Pt</pre>	OM2(:= 1, := Fun_Code2_0, := DataCount2_1, := DataCount2_1, := CoilSingleData2 @coccoseSFE003es ADR(input_b: re@coccoseSFE003esADR(input_ret)	2) , 0, its_Ptr2), egisters_Ptr2),			
	<pre>10 ErrorID2=>); 11 ModbusRIU_Master_Fun_C 12 xExecute2ITHE 13 Fun_Code2_0 14 Addr2_0_:= A 15 Slave2_1_:= 1, 16 DataCount2_1 17 CoilDataPtra00 18 CollSingleData 19 input_ister_Pt 20 input_registers_Pt 21 DataPtra0000000</pre>	OM2(:= 1, := Fun_Code2_0, ddr2_0, := DataCount2_1, code557E00304ADR(CoilDataPtr) 20 := CoilSingleData2 code557E003054ADR(input_b) ref0000020557E003054ADR(input_b) ref0000020557E003054ADR(input_b)	2) , 0 , its_Ptr2), egisters_Ptr2	1) ,			
	<pre>10 ErrorID2->); 11 ModbusRIU_Master_Fun_C 12 xExecute2IRUE 13 Fun_Code2_0 14 Addr2_0_:= A 15 Slave2_1_:= 1, 16 DataCount2_1 17 CoilDataPtr900 18 CoilSingleData 19 input_registers_Pt 20 input_registers_Pt 21 DataPtr9000000 22 Done2-> , 23 Frror2-></pre>	OM2(:= 1, := Fun_Code2_0, ddr2_0, := DataCount2_1, 00000057E00004ADR(CoilDataPtr: 2_0_:= CoilSingleData2 0_:= CoilSingleData2 ref0000020557E0000;seADR(input_b: ref00000205500;seADR(input_ref); 57E000;E&ADR(DatePtr2),	2) , 0, its_Ptr2), egisters_Ptr2) ,			
3	<pre>10 ErrorID2=>); 11 ModbusRTU_Master_Fun_C 12 xExecute2TRUE 13 Fun_Code2 0 14 Addr2 0 == A 15 Slave2 1 == 1, 16 DataCount2 1 17 CoilDataPtrac 19 input_bits_Ptr 20 input_registers_Pt 21 DataPtrac 22 Done2=> , 23 Error2=> , 24 ErrorID2=>);</pre>	OM2(:= 1, := Fun_Code2_0, ddr2_0, := DataCount2_1, := DataCount2_1, := CoilSingleData22 #000002057E00300ADR(input_b: := for the constant of the const	2) , 0_, its_Ptr2), egisters_Ptr2	ı) ,			

Now we take an example of the read operation. Write the value to the variable in the login state. Write 3 to the Fun_Code, which means 03H function code Read Holding Registers. Write 16#3002 to the Addr, which means that one address is read from 3002H. The value 3335 can be read from the array DataPtr2 (i.e. 3002H address), which means the bus voltage is 333.5V with reference to the VFD product manual. Similarly, write 3 to the Fun_Code, which means 03H function code Read Holding Registers. Write 16#2100 to the Addr. The value 3 can be read from the array DataPtr2 (i.e. 2100H address), which means the VFD is down with reference to the VFD product manual.

	· · · · · · · · · · · · · · · · · · ·							
Œ	ModbusRTU_Master_Fun_CO	ModbusRTU_Master						
æ	ModbusRTU_Master_Init_CO					=		
	Ø DatePtr2	ARRAY [00] OF INT						
	Ø DatePtr2[0]	INT	3335					
۰	input_registers_Ptr2	ARRAY [09] OF INT						
۰	Ø CoilDataPtr2	ARRAY [09] OF BO						
۰	input_bits_Ptr2	ARRAY [09] OF BO						
	A alter Laira						· · · · ·	
8	9 Error2=>, 10 Error1D2=>); 11 ModbusRTU_Master_Fun_COM2(xFrequer2[5006]= 1.							
	13 Fun_Code2 3	= Fun_Code2 3						
	15 Slave2 1:= 1,							E
	16 DataCount2 1 := DataCount2 1,							
	17 CoilDataPtr2 16#	B60AE9A8 :=ADR(Coil	DataPtr2) ,					
	18 CoilSingleData2	0 := CoilSing	leData20,					
	19 input_bits_Ptr2	16#B60AE9B2 := ADR (input_bits_Ptr	2),			100 %	- 1
L	201 input nogistons	DEWOL KOMPONEDDA	ADD (input modi	atoma Dtm?)				-

AX series programmable controller software manual

Ξ	ModbusR1	U_Master_Fun_CO	ModbusRTU_Master					
E	🚸 ModbusR1	'U_Master_Init_CO	ModbusRTU_Master					=
	DatePtr2		ARRAY [00] OF INT					-
	DatePt	r2[0]	INT	3				
Ð	input_regi	sters_Ptr2	ARRAY [09] OF INT					
۰	< CoilDataPt	r2	ARRAY [09] OF BO					
۰	input_bits	_Ptr2	ARRAY [09] OF BO					_
	A conter i tr			•				
	9 10	Error2=> , ErrorID2=>);						^
	11 🕘 Modb	usRTU_Master_Fu	in_COM2 (
	12	xExecute2 TRUE :	= 1,					
	13	Fun_Code2 3	:= Fun_Code2 3	,				
	14	14 Addr2 8448 := Addr2 8448 ,						
	15 Slave2 1 := 1,							
	16 DataCount2 1 := DataCount2 1,							
	<pre>17 CoilDataPtr2 16#B60AE9A8 :=ADR(CoilDataPtr2) ,</pre>							
	18	CoilSingleData2	0 := CoilSing	leData2 0 ,				
	19	input_bits_Ptr2	16#B60AE9B2 := ADR (input_bits_Ptr	2),			100 %
	20	input registers Dtr2 (SECONDOL - DD2 (input registers Dtr2)						

Now we take an example of the write operation. Write the value to the variable in the login state. Write 6 to the Fun_Code, which means 06H function code Write Single Register. Write 16#0003 to the Addr, which means to write a value to the address 0003H. Referring to the VFD product manual, 0003H is the address of the maximum output frequency of the VFD with a default value of 50.00 HZ. Before writing the value of the address, the value of the address 0003H in the upper computer is 5000 which is obtained by 50.00Hz multiplied by the scale value of 100. If the maximum output frequency of the VFD is set to 100Hz, write the 0003H with value 100Hz*100, that is, 10000. After that, the value of P00.03 will change from 50.00 to 100.00, indicating that the controller wrote successfully to the VFD. See the figure.

÷ 🔌	ModbusRTU_Master_Fun_CO	ModbusRTU_Master				
i 🧳	ModbusRTU_Master_Init_CO	ModbusRTU_Master				
3	DatePtr2	ARRAY [00] OF INT				
	DatePtr2[0]	INT	5000			
E 🧳	input_registers_Ptr2	ARRAY [09] OF INT				
E 🧳	CoilDataPtr2	ARRAY [09] OF BO				
± 🥠	input_bits_Ptr2	ARRAY [09] OF BO				
	e de l'este					
	9 Error2=> ,					
1	<pre>0 Error1D2=>);</pre>					
1	 ModbusRTU_Master_Fu 	n_COM2 (
1	2 xExecute2 TRUE :	= 1,				
1	3 Fun_Code2 6	= Fun_Code2 6				
1	4 Addr2 3 =Add	r2 <u>3</u> ,				
1	5 Slave2 1 := 1,					
1	6 DataCount2 1	:= DataCount2 1				
1	7 CoilDataPtr2 16#	B60AE9A8 :=ADR(Coil	DataPtr2) ,			
	8 CoilSingleData2	0 := CoilSing	leData2 0,			
1						
1	9 input_bits_Ptr2	16#B60AE9B2 := ADR (input_bits_Pt	2),		

æ	Ø Modbus	sRTU_Master_Fun_CO	ModbusRTU_Master					
æ	Ø Modbus	sRTU_Master_Init_CO	ModbusRTU_Master				=	
	DatePtr	2	ARRAY [00] OF INT					
	🧼 Dat	ePtr2[0]	INT	10000				
±	<pre>input_r</pre>	egisters_Ptr2	ARRAY [09] OF INT					
±	Ø CoilDat	aPtr2	ARRAY [09] OF BO					
Đ	<pre>input_b</pre>	its_Ptr2	ARRAY [09] OF BO					
	A DOMESTIC:	1.0.1.0					· · · · ·	
	9 10 11 • Mo 12	Error2=> , ErrorID2=>); dbusRTU_Master_Fu xExecute2 <mark>TRUE</mark> :	n_COM2(= 1,					•
	13 14 15 16 17	Fun_Code2 6 Addr2 3 :=Add Slave2 1 := 1, DataCount2 1 CoilDataPtr2 16#	:= Fun_Code2 6 r2 3 , := DataCount2 1 BEDAE5A8 :=ADR (Coil	, DataPtr2) .			E	
	18 19 20	CoilSingleData2 input_bits_Ptr2	0 := CoilSing: 16#B60AE9B2 := ADR (leData20 input_bits_	Ptr2),		100 % 🕅 -	

A.2 Controller and DA200 Series Servo Drive Configuration Example

In this section, we will write a program to control four DA200 series servo drives to drive four motor axes for constant forward and reverse motion.

 Select File > New Project from the menu to create a new standard project. Set the device to INVT AX7X, and select Structured Text (ST) as the programming language. Edit the project information as needed, as shown in the following figure.

	管 New Pro	ject						\times	<
	Categories			Templates				× in	
	Pro	raries jects		Empty project	HMI project	t Standard project	Standard project w		
				,,			2		
	Name	DA200							
	Location	E: \codes	ys				~ .		
						OK	Cancel		
Standa	rd Project							×	<
F T	You are object: - One p - A pro - A cyc - A refe	e about f s within f gram PL lic task v erence to	to create a ne this project: nable device C_PRG in the vhich calls PL the newest v	w standard p as specified b language spe C_PRG ersion of the	roject. This selow scified belo Standard l	s wizard will c w ibrary current	reate the fo	llowing	
	Device	1	INVT AX7X (SH	enzhen INVT (Electric Co.	, Ltd.)		~	1
	PLC_PF	RG in	Structured Tex	ct (ST)				~	i
						0	K	Cancel	
	Project	Informat	ion				×		
	File	Summary	Properties Sta	tistics Licensing	Signing				
	Cor	mpany	INVT						
	Tit	le	DA200 C	ontrol 4 motor axi	s				
	Ver	rsion	1.0.0.0				Released		
	Lib	rary C-+							
	Def	ault names	pace						
	Aut	hor	ZJZ						
	Des	scription	DA200 C	ontrol 4 motor ax	is				
	Lib	rary compa	tibility Invtmatic	Studio V1.0.2			~		
	The	fields in b	old letters are use	d to identify a libr	ary.				
	Auto	matically ge	nerate 'Library Info	ormation' POUs					
	Auto	matically ge	nerate 'Project Inf	ormation' POUs	_				
					L	OK	Cancel		

2. Right click the device from the device panel and select **Add Device** to add the EtherCAT master. Select **EtherCAT Master SoftMotion** with a version of 3.5.15.0, as shown in the following figure.

f Add Device					×
Name EtherCAT_Master_SoftMotion Action Append device Insert device Plug de	evice Ol	Jpdate device			
String for a fulltext search	Vendor	<all vendors=""></all>			~
Name Miscellaneous Fieldbuses GAN CANbus Miscellaneous Fieldbuses Can CANbus Can CANbus Can CANbus	Vendo	r	Version	Description	^
Burger Master EtherCAT Master EtherCAT Master	3S - Sm 3S - Sm	art Software Solutions GmbH	3.5.15.0	EtherCAT Master	
	or experts o	univ) 🗌 Disolav outdated ver	sions		~
Name: EtherCAT Master SoftMotion Vendor: 33 - Smart Software Solutions Gmt Categories: Master Version: 3.5.15.0 Order Number: Description: EtherCAT Master SoftMotion.	он			Ż	
Append selected device as last child of Device (You can select another target node in the n	avigator w	hile this window is open.)			
				Add Device	Close

3. Right click the device **EtherCAT Master SoftMotion** from the device panel and select **Add Device** to add 4 servo drives. Select **INVT_DA200_171**, as shown in the following figure.

Vame INVT DA200 262				1
Action				1
Append device Insert device Plug		Jodate device		
String for a fulltext search	Vendor	<all vendors=""></all>		~
Name			Vendor	^
🖃 👔 Fieldbuses				
Brow EtherCAT				
🖮 🔐 Budd Slave				
🖲 🖓 Delta Electronics, Inc Serv	o Drives			
🍺 🚞 ifm electronic - ifm electronic	EtherCAT De	vices		
INVT				
🖨 - 🚞 INVT INDUSTRIAL				
🖮 🚞 Servo Drives				
DA200-N EtherCAT(CoE) Drive		INVT INDUSTRIAL	
🗷 词 Panasonic Corporation, Appl	iances Compa	any - AC Servo Driver		
🗍 🗁 Darkar Hansifin - Darkar Can	io Drivo 1M		×	×
Crown by category Display all versions	(for experts)			-
	(ior experts)			
Name: DA200-N EtherCAT(CoE) Drive			<u>^</u>	
Vendor: INVT INDUSTRIAL				
Version: Revision=16#000000AB				
Order Number: INVT_DA200_262			~	
Description: EtherCAT Slave imported fi	rom Slave XM	L: INVT_DA200_EtherCAT_V262_200313.xml Device:	:	
DA200-N EtherCAT(CoE) Drive			¥	
Append selected device as last child of EtherCAT_Master_SoftMotion				
• (You can select another target node in the	e navigator v	while this window is open.)		
			Add Device Close	

4. Right click an **INVT_DA200_171** device in the device panel and select **Add SoftMotion CiA402 Axis**. Preform the same procedure for the remaining 3 INVT_DA200_171 devices, as shown in the figure.



5. Double-click the PLC_PRG and enter the following codes on the statement editor:

```
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. Enter the following code in the main code editor:
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 , );
```

```
AX series programmable controller software manual
                                                                             Project Instance
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

The main body of the program takes the form of a state machine that determines which part of the code to execute through the value of iStatus. When the program starts, the iStatus value is 0 and the program initializes the MC_Power function block and enables the corresponding motor shaft. If the enabling is successful, the iStatus value is 1 and the program enters the next state. When the iStatus value is 1, the MC_MoveAbsolute function block is executed, and the motor rotates to the specified position at the specified speed. If the motor moves normally to the specified position, the iStatus value is increased by 1, and the motor enters the next state. When the iStatus value is 2, execute the MC_MoveAbsolute function block in the other direction. The motor continues to rotate to the specified position at the speed specified by the function block. If the motor moves normally to the specified position, the iStatus value is reset to 1. The procedure is executed repeatedly to implement the forward and reverse movement of the motor.

Double-click EtherCAT Master SoftMotion from the device panel and click Browse to select the EtherCAT communication network eth0. Select the distributed clock as needed. In this example, select 4000us for the cycle time. See the figure.

	•
General Autoconfig Master/Slaves EtherCAT	
Sync Unit Assignment EtherCAT NIC Setting	_
Log Destination address (MAC) FF-FF-FF-FF Broadcast Enable redundancy	
EtherCAT I/O Mapping Source address (MAC) B0-7E-11-3D-81-5C Browse	
EtherCAT IEC Objects	
Status	
Information Cycle time 4000	
<	>

button on the toolbar to compile the code. After compiling, click the 🥵 button on the toolbar to log in Click the to the controller. The servo starts normally, the motor runs smoothly, and the upper computer interface is shown in the following figure.



	_PRG X EtherCAT_M	laster_SoftMotion				
Device.Application.PLC_PRG						
Expression	Туре	Value	Prepared value	Address	Comment	^
🞓 iStatus	INT	1				
# Ø 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					
A MAR MALIARE AND A						
						>
<pre>7 IF MC_Power_0.Status Haus AND MC_Power_1.Status 8</pre>	TRUE AND MC_Power_2.:	Status TRUE AN	ND MC_Power_3.Statu	IS TRUE THEN		
11 MC MoveBheolute 0(BvisteSM Drive GenericDSB402						
	Evecute TRUE := TRUE	Pogition	50 :=50 Veloc	itu 3		Dec
12 MC MoveAbsolute 1(Axis:=SM Drive GenericDSP402	<pre>, Execute TRUE := TRUE, 1, Execute TRUE := TRUE</pre>	, Position	50 :=50 , Veloc	ity 3 city 3	=3 , Acceleration 2 := 2	, Deci 2, Dec
12 MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_ 13 MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_	<pre>, Execute TRUE := TRUE, 1, Execute TRUE := TRUE 2, Execute TRUE := TRUE</pre>	, Position 2, Position 2, Position	50 :=50 , Veloc 50 :=50 , Veloc 50 :=50 , Veloc	city 3 city 3 city 3	:=3 , Acceleration 2 := 2 :=3 , Acceleration 2 := :=3 , Acceleration 2 :=	, Dec 2, De 2, De
12 MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_ 13 MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_ 14 MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_	, Execute TRUE := TRUE 1, Execute TRUE := TRUE 2, Execute TRUE := TRUE 3, Execute TRUE := TRUE	, Position 2, Position 2, Position 2, Position	50 :=50 , Veloc 50 :=50 , Veloc 50 :=50 , Veloc 50 :=50 , Veloc	city 3 ocity 3 ocity 3 ocity 3	:=3 , Acceleration 2 := 2 :=3 , Acceleration 2 := :=3 , Acceleration 2 := :=3 , Acceleration 2 :=	, Dec 2, De 2, De 2, De 2, De
12 MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_ 13 MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_ 14 MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSP402_ 15 If MC_MoveAbsolute_0.DoneFxAsg_AND MC_MoveAbsol	<pre>, Execute TRUE := TRUE, 1, Execute TRUE := TRUE, 2, Execute TRUE := TRUE 3, Execute TRUE := TRUE ute_1.Done TRUE AND MM</pre>	, Position 2, Position 2, Position 2, Position 2 MoveAbsolute	50 :=50 , Veloc 2.Done TRUE AND M	ty 3 ocity 3 ocity 3 ocity 3 C_MoveAbsolut	<pre>[:=3 , Acceleration 2 := 2 [:=3 , Acceleration 2 := :=3 , Acceleration 2 := :=3 , Acceleration 2 := e_3.Done TRUE THEN</pre>	, Dec 2, De 2, De 2, De 2, De
12 WC_MoveAbsolute_1(Axis=SM_Drive_GenerioDSF402_ 13 WC_MoveAbsolute_2(Axis=SM_Drive_GenerioDSF402_ 14 WC_MoveAbsolute_3(Axis=SM_Drive_GenerioDSF402_ 15 UF_MC_MoveAbsolute_0(Axis=SM_Drive_GenerioDSF402_ 16 MC_MoveAbsolute_0(Axis=SM_Drive_GenerioDSF402_ 17 USAN DRIVENT DRIV	<pre>, Execute TRUE := TRUE, 1, Execute TRUE := TRUE 2, Execute TRUE := TRUE 3, Execute TRUE := TRUE 3, Execute TRUE := TRUE ute_1.Done TRUE AND M nericDSP402, Execute</pre>	, Position 2, Position 2, Position 2, Position 3 4 5 5 6 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7	50 :=50 , Veloc 2. Done TRUE AND M); ;	sity 3 ocity 3 ocity 3 ocity 3 C_MoveAbsolut	<pre>!=3 , Acceleration 2 := 2 :=3 , Acceleration 2 := :=3 , Acceleration 2 := :=3 , Acceleration 2 := e_3.Done TRUE THEN</pre>	, Dec 2, De 2, De 2, De 2, De
12 MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericDSF402_ 13 MC_MoveAbsolute_2 (Axis:=SM_Drive_GenericDSF402_ 14 MC_MoveAbsolute_0 (Axis:=SM_Drive_GenericDSF402_ 15 IF MC_MoveAbsolute_0.0cme[AXES] AND MC_MoveAbsolute_0 16 MC_MoveAbsolute_0.1cme[AXES] AND MC_MoveAbsolute_0 17 MC_MoveAbsolute_0 (Axis:=SM_Drive_GenericDSF402_	<pre>, Execute TRUE := TRUE, 1, Execute TRUE := TRUI 2, Execute TRUE := TRUI 3, Execute TRUE := TRUI ute_1.Done TRUE AND M nericDSP402 , Execute nericDSP402_1 , Execute</pre>	. Position 5. Position 5. Position 5. Position 5. MoveAbsolute TRUE := FALSE, 5. TRUE := FALSE,	50 :=50 , Veloo 50 :=50 , Velo 50 :=50 , Velo 50 :=50 , Velo 50 :=50 , Velo 2.Done TRUE AND &); (E,);	eity 3 ocity 3 ocity 3 ocity 3 C_MoveAbsolut	<pre>[:=3 , Acceleration 2 := 2 [:=3 , Acceleration 2 := [:=3 , Acceleration 2 := [:=3 , Acceleration 2 := [:=3 , Acceleration 2 := e_3.Done TRUE THEN</pre>	, Dec 2, De 2, De 2, De
11 MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_ 13 WC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSP402_ 14 WC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSP402_ 15 IF MC_MoveAbsolute_0.DoneFAME_AND_MC_MoveAbsol 16 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 17 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 18 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 19 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 10 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 11 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 12 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 13 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 14 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 15 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 16 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 16 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 17 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_ 18 MC_MoveAbsolute_0(Axis:=SM_Drive_GenericDSP402_	<pre>, Execute TAUE := TRUE, 1, Execute TAUE := TRUE, 2, Execute TAUE := TRUE 3, Execute TAUE := TRUE ute_1.Done TAUE = TRUE ute_1.Done TAUE = AND M nericDSP402 , Execut nericDSP402_1 , Execut</pre>	C. Position C. Position C. Position C. Position C. MoveAbsolute TRUE := FALSE, RETRUE := FALSE, RETRUE := FALSE	50 :=50 , Veloc 50 ;=50 , Veloc 50 ;; (E,);	sity 3 ocity 3 ocity 3 ocity 3 ocity 3 C_MoveAbsolut	-3 , Acceleration 2 = 2 =3 , Acceleration 2 = 2 =5 , Acceleration 2 = =3 , Acceleration 2 = =3 , Done #AUE THEN	, Dec 2, De 2, De 2, De 2, De
 MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_2(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_3(Axis:=SM_Drive_GenericDSF402_ IF MC_MoveAbsolute_0.DoneFXDEF AND MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ MC_MoveAbsolute_1(Axis:=SM_Drive_GenericDSF402_ 	<pre>, Execute TAUE := TRUE, := 1, Execute TAUE := TRUE 2, Execute TAUE := TRUE 3, Execute TAUE := TRUE ute 1.Done TAUE AND M nericDSP402 A. Execute nericDSP402_1 . Execut nericDSP402_3 . Execut</pre>	2. Position 2. Position 2. Position 2. MoveAbsolute TRUE := FALSE, 2. TRUE := FALSE 2. TRUE := FALSE 2. TRUE := FALSE	50 :=50 , Veloc 50 :=50 ; Veloc 50 :=5	sity 3 city 3 city 3 city 3 city 3 KC_MoveAbsolut	-3 , Acceleration 2 = 2 -3 , Acceleration 2 = 2 -5 , Acceleration 2 = -3 , Acceleration 2 = -3 , Acceleration 2 = -3 , Done TRUE THEN	, Deci 2, Dec 2, Dec 2, Dec
 MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_2 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_3 (Axis:=SM_Drive_GenericD5F402_ IF MC_MoveAbsolute_0.Done[XMSH_AND_MV_MoveAbsolute] IF MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_1 (Axis:=SM_Drive_GenericD5F402_ MC_MoveAbsolute_2 (Axis:=SM_Drive_GenericD5F402_ MD_TF 	, Execute TAUE := TRUE, 1. Execute TAUE := TRUE 2. Execute TAUE := TRUE 3. Execute TAUE := TRUE ute_1.DomeTAUE AND mericDSP402 , Execute nericDSP402_2 , Execut nericDSP402_2 , Execut nericDSP402_3 , Execut	2. Position 2. Position 2. Position 2. Position 2. MoveAbsolute TRUE := FALSE, 3. TRUE := FALSE 3. TRUE := FALS 3. TRUE := FALS	50 :=50 , Veloc 50 ;=50 ; ;50	sity 3 city 3 city 3 city 3 city 3 KC_MoveAbsolut	<pre> -3 , Acceleration 2 = 2 =3 , Acceleration 2 = = =3 , Acceleration 2 = =3 , Acceleration 2 = =3. Done HULE THEN</pre>	, Deci 2, Dec 2, Dec 2, Dec

Double-click **INVT_DA200_171** from the device panel to view or set the current motor running parameters in the I/O mapping interface. See the figure.

Data	Variable	Mapping	Channel	Address	Туре	Current Value	Prepared Value	Unit	Description
Data			Control Word	%QW22	UINT	15			Control Word
arameters	±*•		Target Position	%QD12	DINT	205891400			Target Position
			Target Velocity	%QD13	DINT	-196608			Target Velocity
I/O Mapping			Mode of Operation	%QB56	SINT	8			Mode of Operation
TEC Objects	÷-**		Target torque	%QW29	INT	0			Target torque
ice objects	· · · · · · · · · · · · · · · · · · ·		Touch probe control	%QW30	UINT	0			Touch probe control
	⊞ - ^K ¢		Positive torque limit	%QW31	UINT	0			Positive torque limit
			Negtive torque limit	%QW32	UINT	0			Negtive torque limit
on	i⊛ - * ∳		Max profile velocity	%QD17	UDINT	0			Max profile velocity
	i≣¥≱		Status Word	%IW2	UINT	4919			Status Word
	📃 😐 🧤		Position Actual Value	%ID2	DINT	205895807			Position Actual Value
	😟 🧤		Speed Actual Value	%ID3	DINT	-197133			Speed Actual Value
	🖶 - 🍫		Torque Actual Value	%IW8	INT	-46			Torque Actual Value
	😐 🍫		Operation Mode Display	%IB18	SINT	8			Operation Mode Display
	😟 - 🦄		Current Actual Value	%IW10	INT	12			Current Actual Value
	😟 🏘		Touch Probe Status	%IW11	UINT	0			Touch Probe Status
	🕸 🍫		Touch Probe Value	%ID6	DINT	0			Touch Probe Value
	😟 🧤		Digital outputs	%ID7	UDINT	0			Digital outputs
	😟 - 🍫		Digital inputs	%ID8	UDINT	0			Digital inputs

Select **Device** > **PLCShell**. Click the button at the bottom right corner and select **prcload**. Then the CPU load rate of the current controller will be shown as follows.

EtherCAT_Task	n# SM_Drive_GenericdSP402
Communication Settings	version
Applications	u-boot V200 kernel V200 rootfs V202
Backup and Restore	
Files	picload
Log	PLC load average: 304 CoveTD: 0
PLC Settings	PLC Core load: 30%
PLC Shell	
Users and Groups	
Access Rights	
Symbol Rights	
IEC Objects	
Task Deployment	
Status	
Information	
	pldoad V

To observe the operation of the motor shaft in an intuitive way and track the actual position of the shaft, create a new trace. Right click **Application** and select **Add Object** > **Trace**. Set the task attribute to **EtherCAT_Task**, and add **PLC_PRG.MC_Power_0.Axis.fActPosition** and **PLC_PRG.MC_Power_0.Axis.fActVelocity** variables in **Trace**. Adjust the display properties of the coordinates appropriately. Right click the graph and select **Download Trace** to track the actual position and actual speed of the motor, as shown in the following figure.



A.3 Controller and DA200 Series Servo CANopen Configuration Example

In this section, we will write a program to connect to the DA200 series servo drive through CANopen communication.

1. Refer to section 2.4 "Project creation" to creat a project, as shown in the following figure.



2. In the **Tool** bar, select **Tools** > **Device Repository**, click **Install** to select the device profile **INVT_DA200_CANopen.eds**, and click **Open**. Then the DA200 CANopen device profile can be added successfully.

		▼ # X 📄 PLC_PRG	CANbus CANopen_Manager	×		
CANOpen_test	🌋 Device	Repository				×
B-B PLC Logic B-O Application	<u>L</u> ocation	System Repository (C:\ProgramData\Invtmatic Studio\Devices)			~	Edit Locations
PLC_PRG (Installed d	e <u>v</u> ice descriptions				
🖃 🎃 Task Cornig	String for	a fulltext search Vendor: <all vendors=""></all>			\sim	<u>I</u> nstall
HIGH PULSE IO	Name	a	Vendor	Version	^	<u>U</u> ninstall
CANbus (CANbus)		CMMP-AS-C5-11A-P3-M3_SoftMotion GMMP-AS-C5-3A-M0_SoftMotion	Festo AG & Co. KG	4.3.0.0		Export
🖻 👚 CANopen_Man		CMMP-AS-C5-3A-M3 SoftMotion	Festo AG & Co. KG	4.3.0.0		
DA200_Dri		- 🗐 DA200 Drive	DA200,invt	Revision=16#000	0010	
SoftMotion General		DA300 and DA180 Drive	DA300 and DA180, invt	Revision=16#000	0003	
		- II DIS-2_SoftMotion	Metronix GmbH	4.4.0.0		
		ECOSTEP 200_SoftMotion	Jenaer Antriebstechnik GmbH	4.4.0.0		
		ECOVARIO 114/214/414_SoftMotion	Jenaer Antriebstechnik GmbH	4.4.0.0		
		EMCA-EC-67CO(DS402)	Festo AG & Co. KG	4.0.0.0	~	
	<				>	Details

3. Right click **Device** in the device panel, and select **Add Device** > **CANbus**. Click **Add Device**, and **CANbus** is added successfully.

CANOpen_test.project* - Invtmatic Studio			-	
<u>File Edit View Project Build Online Debug Tools Wi</u>	ndow <u>H</u> elp			T
管 📽 🖬 番 い つ よ 助 逸 🗙 構 協 🍓 協 則 別 別 別	📾 🎦 • 📑 🎬 Application [[Device: PLC Logic] - 🧐 🧐 👌 = '	≰ (≣ = = = \$	¢ ∭ ≓ ∎
Devices - 4 ×	Add Device		×	-
CANOpen_test				B
= (j) Device (INVT AX7X)	Name CANbus			
🖶 🗐 PLC Logic	Action			
🖹 🔘 Application	Append device	Plug device Update device		
Library Manager	String for a fulltext search	Vendor <all vendors=""></all>	~	
🖃 🎆 Task Configuration	Name	Vendor	/ersion Descriptic ^	
🖹 🍪 MainTask				100 %
PLC_PRG	Fieldbuses			
- A HIGH_PULSE_IO	CANbus			
SoftMotion General Axis Pool	🔟 CANbus	3S - Smart Software Solutions GmbH 3	.5.15.0 Needed for a	
	NetX CANbus	3S - Smart Software Solutions GmbH 3	.5.15.0 CANbus on a	
	EtherCAT		~	
	<		>	
	Group by category Display all	versions (for experts only) Display outo	lated versions	
	Name: CANbus	^		
	Vendor: 3S - Smart Software S	olutions GmbH		
	Categories: CANbus		<u></u>	
	Version: 3.5.15.0 Order Number:		× 1	
	order number.			
	Description: Needed for all Be	dbusses which communicate over the		
	Append selected device as last chi Device	ld of		
Devices POUs	(You can select another target n	ode in the navigator while this window is op	en.)	100 %
Messages - Total 0 error(s), 0 warning(s), 0 message(s)		Add	Device Close	() ()

4. Right click **CANbus**, and select **Add Device** > **CANopen_Manager**. Click **Add Device**, and **CANopen_Manager** is added successfully.

vices	• • • • • • • • • • • • • • • • • • •	×	· pms	
CANCiper, Lest CANCiper, Lest Concert, Extra X7(3) Concert, Extr	Name CAlvopen Manager Action Ac		100 %	A
	Append selected device as last child of CAttyper Manager Append selected device as last child of CAttyper Manager x		10	0.96
5. Right click **CANopen_Manager** and select **Add Device** > **DA200 Drive**. Click **Add Device**, and **DA200 Drive** is added successfully.

CANOpen_test.project* - Invtmatic Studio		-	\Box \times
<u>File Edit View Project Build Online Debug Tools</u>	Window Help		
🛅 😅 🔚 🚳 너 더 공 ங 🛍 🗙 🛤 😘 🐴 🌿 제 🦄 🦄	🐐 🔚 🔚 - 🗂 🔛 Application [Device: PLC Logic] 🔹 😂 📦 💼 🍕	C∃ 93 63 *3 \$ \$	=
Devices 👻 🕂	Add Device	×	-
CANOpen_test	•		B)
Device (INVT AX7X)	Name DA200_Drive		
■ III PLC Logic	Action		
🖹 🚫 Application	Append device Insert device Plug device Update device		
- 🛗 Library Manager	China for a fellow because		
PLC_PRG (PRG)	string for a fullext search Vendor All Vendors>	Y	
E I Task Configuration	Name	Vendor ^	00 % 🙉
AinTask	CMMP-AS-C5-3A-M0_SoftMotion	Festo AG & Co. KG	
di PLC_PRG	CMMP-AS-C5-3A-M3_SoftMotion	Festo AG & Co. KG	
A HIGH_PULSE_IO	DA200 Drive	DA200,invt	
CANbus (CANbus)	DA300 and DA180 Drive	DA300 and DA180, invt	
Carvopen_manager (Carvopen_manager)	DIS-2_SoftMotion	Metronix GmbH	
a solution deleta Axis Pool	Init ECOSTEP200 SoftMotion	Jenaer Antriebstechnik (*	
	Group by category Display all versions (for experts only) Display outda	ated versions	
	Vendor: DA200 Unive		
	Categories: Remote Device	<u> </u>	
	Version: Revision = 16#00000104, FileVersion = 1.1	1	
	Order Number: DA200 Description: Imported from INVT. DA200. CANopen. V2.60.eds	~	
	v		
	Annead colorted during as bot shild of		
	CANopen_Manager		
	You can select another target node in the navigator while this window is ope	en.)	100 % 🙊
The services in the services i			>
Messages - Total 0 error(s), 0 warning(s), 0 message(s)	Add D	levice Close	
			😯:

- In the CANbus overview interface, set the baud rate to be the same as that of the DA200 CANopen servo (DA200 P4.02). In the DA200_Drive overview interface, set the node ID to be the same as that of the DA200 CANopen servo (DA200 P4.05).
- 7. After completing the physical connection of the device, download the program, log in the device, and you can see that the CANopen is connected to DA200 communication successfully.

CAN_OPEN_test.project* - Invtmatic Studio						-		×
File Edit View Project Build Online Debug Tools Window	Help							₹
10 単日の「へるもの×」Aは45411111111111111111111111111111111111	• 1 HApplication (Device:	PLC Logic] + 🔍 💜 🕟 💼 🍕	(D 91 41 42 8	* 종 ㅠ 장				
Devices - # X	DA200_Drive X							•
B 🔞 CAN_OPEN_test								_
=-G 🕤 Device [connected] (INVT AX7X)	General	CANopen :		n/a				
R I PLC Logic	PDOs	Last Diagnostic Message						
O Application [run]								_
1 Lbrary Manager	SDOs	Diagnosis Info						-
E_PRC_PRG (PRG)	Log	 Current Network ID 	0	Currently used I	Network ID. Can b	e changed at runtime by De	viceDiagno	şi
- C C MainTark		 Current Baudrate 	500	Currently used	Baudrate. Can be	changed at runtime by Devic	ceDiagnosis	Æ
an an file	CANopen I/O Mapping	Bus state	ACTIVE	Only few CAN b	us errors so far. T	he error counters of the chip	p are below	i -
	CANopen IEC Objects	- Bus Alarm	FALSE	If CAN driver sig	gnals a bus alarm a	critical bus error occurred.	The fieldbu	s =
- O fil CANbus (CANbus)		Bus Error counter	0	Number of occu	rred bus errors. W	ill be reset if diagnosis will be	e acknowle	d
CANopen Manager (CANopen Manager)	Status	- Last Driver Open Error	No error occurred					
DA200_Drive (DA200 Drive)	Information	Driver Instances	1	Number of drive	r instances curren	tly opened by underlying fie	dbus stad	s
😔 🚡 SoftMotion General Axis Pool		- Bus Load	0	Bus load in perc	ent. If CAN driver	does not support bus load m	neasureme	1
		- Tx Counter	362	Number of suco	essful sent CAN te	learams		
		Ty Error Counter	0	Value of tracers	it error counter pr	ouided by CAN chip		
		Du Counter	201	Value of earland	a control counter pro	billed by CAN dip.		
		<		i Number of soco	essiu received CA	ry telegrams.	,	
		<						
Ser Devices 1 POLIS							-	>
Watch 1		A . R . C	-	14 I		e	• •	×
Expression		Application	Type	Value	Prepared value	Execution point		Ac
<								>
😹 Watch 1 🕢 Breakpoints								
Messages - Total 0 error(s), 0 warning(s), 5 message(s)								
Device user: Anonymous Last build: 🗿 0 🚸 0	Precomple 🗸 🤷 RUN	Program load	ed	Program unchang	ed	Project user: (nobody)	()

Note:

- If there is a high requirement for real-time data, the CAN bus load shall be less than 30% in order to avoid a small delay in data sending and receiving due to bus competition.
- For CAN buses with synchronization requirements, the set value of the window length in the bus synchronization message is slightly less than the cycle period.

✓ Enable SYNC producing
 COB ID (Hex) 16# 80 ♀
 Cycle period (µs) 60000 ♀
 Window length (µs) 58000 ♀

Enab	le SYNC	consuming
------	---------	-----------

- The task period of the task where CANopen is located shall be slightly longer than the actual execution time of the task.
- To ensure that the master monitors the slave properly, the **Enable heartbeat producing** option in the slave shall be checked.

Guarding ———					
Enable nodeguar	ding		🗹 Enable heartbeat p	oroducing	
Guard time (ms)	0	*	Producer time (ms)	200	+
Life time factor	0	*	Heartbeat cons	uming (1/1 active)	

Appendix B SMC_ERROR description

Error number	Function block	ENUM value	Description
0	All function blocks	SMC_NO_ERROR	No error
1	DriveInterface	SMC_DI_GENERAL_COMMUNIC ATION_ERROR	Communication error. For example, sercos ring has broken
2	DriveInterface	SMC_DI_AXIS_ERROR	Axis error
10	DriveInterface	SMC_DI_SWLIMITS_EXCEEDED	Position outside of permissible range of SWLimit
11	DriveInterface	SMC_DI_HWLIMITS_EXCEEDED	Hardware end switch is active
13	DriveInterface	SMC_DI_HALT_OR_QUICKSTOP _NOT_SUPPORTED	Drive status Halt or Quickstop is not supported
14	DriveInterface	SMC_DI_VOLTAGE_DISABLED	Drive has no power
15	DriveInterface	SMC_DI_IRREGULAR_ACTPOSIT	Current position given from the drive seems to be irregular. Check the communication.
16	DriveInterface	SMC_DI_POSITIONLAGERROR	Position lag error. Difference between set and current position exceeds the given limit
20	All motion generating function blocks	SMC_REGULATOR_OR_START_ NOT_SET	Controller enable not done or brake applied
21	Axis in wrong controller mode	SMC_WRONG_CONTROLLER_M ODE	Axis in wrong controller mode
30	DriveInterface	SMC_FB_WASNT_CALLED_DURI NG_MOTION	Motion creating module has not been called again before end of the motion.
31	All function blocks	SMC_AXIS_IS_NO_AXIS_REF	Type of given AXIS_REF variable is not AXIS_REF
32	Axis in wrong controller mode	SMC_AXIS_REF_CHANGED_DU RING_OPERATION	AXIS_REF variable has been exchanged while the module was active.
33	DriveInterface	SMC_FB_ACTIVE_AXIS_DIABLE D	Axis disabled while being moved. MC_Power.bRegulatorOn
34	All motion generating function blocks	SMC_AXIS_NOT_READY_FOR_ MOTION	Axis in its current state cannot execute a motion command, because the axis doesn't signal currently that it follows the target values
40	VirtualDrive	SMC_VD_MAX_VELOCITY_EXCE EDED	Maximum velocity fMaxVelocity exceeded
41	VirtualDrive	SMC_VD_MAX_ACCELERATION_ EXCEEDED	Maximum acceleration fMaxAcceleration exceeded
42	VirtualDrive	SMC_VD_MAX_DECELERATION_ EXCEEDED	Maximum deceleration fMaxDeceleration exceeded
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VA LUES	Invalid velocity or acceleration values
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLI MIT	Mode requests for safety reasons use of end switches

Error number	Function block	ENUM value	Description
70	SMC_SetControllerM ode	SMC_SCM_NOT_SUPPORTED	Mode not supported
71	SMC_SetControllerM ode	SMC_SCM_AXIS_IN_WRONG_ST ATE	In current mode, controller mode cannot be changed
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLE R_MODE	Axis not in correct controller mode
80	SMC_ResetAxisGrou p	SMC_RAG_ERROR_DURING_ST ARTUP	Error at startup of the axis group
90	SMC_ChangeGearin gRatio	SMC_CGR_ZERO_VALUES	Invalid values
91	SMC_ChangeGearin gRatio	SMC_CGR_DRIVE_POWERED	Gearing parameters must not be changed as long as the drive is under control
92	SMC_ChangeGearin gRatio	SMC_CGR_INVALID_POSPERIO D	Invalid position period (<= 0)
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	Axis contain no information on cycle time (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	Axis without error
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSW	Axis does not perform error-reset
122	MC_Reset	SMC_R_ERROR_NOT_RESETTA BLE	Error could not be reset
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSW ER_IN_TIME	Communication with the axis did not work
	MC_ReadParameter,		
130	MC_ReadBoolParam eter	SMC_RP_PARAM_UNKNOWN	Parameter number unknown
131	MC_ReadParameter, MC_ReadBoolParam eter	SMC_RP_REQUESTING_ERROR	Error during transmission to the drives. See error number in function block instance ReadDriveParameter, Library SM_DriveBasic
140	MC_WriteParameter, MC_WriteBoolParam eter	SMC_WP_PARAM_INVALID	Parameter number unknown or writing not allowed
141	MC_WriteParameter, MC_WriteBoolParam eter	SMC_WP_SENDING_ERROR	See error number in function block instance WriteDriveParameter, Library Drive_Basic
170	MC_Home	SMC_H_AXIS_WASNT_STANDST ILL	Axis has not been in standstill state
171	MC_Home	SMC_H_AXIS_DIDNT_START_HO MING	Error at start of homing action
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	Communication error
173	MC_Home	SMC_H_ERROR_WHEN_STOPPI NG	Error at stop after homing. Deceleration may not be set
180	MC_Stop	SMC_MS_UNKNOWN_STOPPIN G_ERROR_	Unknown error at stop
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VA	Invalid velocity or acceleration

Error number	Function block	ENUM value	Description
182	MC_Stop	SMC_MS_DIRECTION_NOT_APP	Direction = shortest not applicable
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	Axis is in error stop status. Stop cannot be executed
184	MC_Stop	SMC_BLOCKING_MC_STOP_WA SNT_CALLED	Instance of MC_Stop blocking the axis by Execute = TRUE has not been called yet. MC_Stop (Execute = FALSE) has to be called.
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VAL UES	Invalid velocity or acceleration values
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	Direction error
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VAL UES	Invalid velocity or acceleration values
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	Direction error
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VA LUES	Invalid velocity or acceleration values
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	Direction error
276	MC_MoveSuperImpo sed	SMC_MSI_INVALID_VELACC_VA LUES	Invalid velocity or acceleration values
277	MC_MoveSuperImpo sed	SMC_MSI_INVALID_DIRECTION	Direction error
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VA	Invalid velocity or acceleration values
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APP	Direction = shortest/fastest not applicable
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	Erroneous array size
326	MC_PositionProfile	SMC_PP_STEP0MS	Step time = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	Erroneous array size
351	MC_VelocityProfile	SMC_VP_STEP0MS	Step time = t#0s
375	MC_AccelerationProfi le	SMC_AP_ARRAYSIZE	Erroneous array size
376	MC_AccelerationProfi le	SMC_AP_STEP0MS	Step time = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Trigger already active
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WIND OW	DriveInterface does not support the window function
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Communication error
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIE	Trigger already de-allocated
426	SMC_MoveContinuo usRelative	SMC_MCR_INVALID_VELACC_V ALUES	Invalid velocity or acceleration values
427	SMC_MoveContinuo usRelative	SMC_MCR_INVALID_DIRECTION	Invalid direction
451	SMC_MoveContinuo usAbsolute	SMC_MCA_INVALID_VELACC_VA	Invalid velocity or acceleration values

Error number	Function block	ENUM value	Description
452	SMC_MoveContinuo usAbsolute	SMC_MCA_INVALID_DIRECTION	Invalid direction
453	SMC_MoveContinuo usAbsolute	SMC_MCA_DIRECTION_NOT_AP PLICABLE	Direction= fastest not applicable
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	CAM does not contain any tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet group ID exceeds MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACC ESSES	More than 32 accesses on one CAM_REF
625	MC_CamIN	SMC_CI_NO_CAM_SELECTED	No cam selected
626	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCA LE	Master axis out of valid range
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_VELA CC_VALUES	Velocity and acceleration values must be specified for ramp_in function
628	MC_CamIN	SMC_CI_SCALING_INCORRECT	Scaling variables fEditor/TableMasterMin/Max are not correct
640	SMC_CAMBounds, SMC_CamBounds_P os	SMC_CB_NOT_IMPLEMENTED	Function block for the given cam format is not implemented
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator=0
676	MC_GearIn	SMC_GI_INVALID_ACC	Acceleration invalid
677	MC_GearIn	SMC_GI_INVALID_DEC	Deceleration invalid
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	Velocity and acceleration/deceleration values invalid
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD 0	Rotation axis with fPositionPeriod = 0
750	All modules using MC_CAM_REF as input	SMC_NO_CAM_REF_TYPE	Type of given cam is not MC_CAM_REF
751	MC_CamTableSelect	SMC_CAM_TABLE_DOES_NOT_ COVER_MASTER_SCALE	Master area, xStart and xEnd, from CamTable is not covered by curve data
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION _CHANGE	During coupling of slave axis, master axis has changed direction of rotation
800	SMC_BacklashComp ensation	SMC_BC_BL_TOO_BIG	Gear backlash fBacklash too large (> position periode/2)
1000	CNC function blocks which are supervising the licensing	SMC_NO_LICENSE	Target is not licensed for CNC
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Path cannot be processed because set velocity = 0
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Last path object has Vel_End > 0
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Warning: GEOINFO-List processed in DataIn but end of list not reached.

Error number	Function block	ENUM value	Description
			Reason: EndOfList of the queue in DataIn not be set. SMC_Interpolator faster than path generating function blocks.
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_S TOP	Velocity at Stop > 0
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSI ONS	Too much SMC_Interpolator recursions. SoftMotion-Error
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIE	SMC_CHeckVelocities is not the last processed function block, that accesses to the OutQueue-data by poqDataIn
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Internal or numeric error
1008	SMC_Interpolator	SMC_INT_VEL_ACC_DEC_ZERO	Velocity and acceleration/ deceleration is null or to low
1009	SMC_Interpolator	SMC_INT_DWIPOTIME_ZERO	FB called with dwlpoTime = 0
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_S MALL	Data buffer too small
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_ IN_QUEUE	Path does not go completely in queue
1100	SMC_CheckVelocitie s	SMC_CV_ACC_DEC_VEL_NONP OSITIVE	Velocity and acceleration/ deceleration values non-positive
1120	SMC_Controlaxisbyp os	SMC_CA_INVALID_ACCDEC_VAL UES	Values of fGapVelocity /fGapAcceleration/fGapDeceleratio n non-positive
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Acceleration value impermissible
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Deceleration value impermissible
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_E MPTY	Data underrun. Queue has been read and is empty
1203	SMC_NCDecoder	SMC_DEC_JUMP_TO_UNKNOW N_LINE	Jump to line cannot be executed because line number is unknown
1204	SMC_NCDecoder	SMC_DEC_INVALID_SYNTAX	Syntax invalid
1205	SMC_NCDecoder	SMC_DEC_3DMODE_OBJECT_N OT_SUPPORTED	Objects are not supported in 3D mode
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMAL	Buffer too small
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_T YPE	Buffer elements have wrong type
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LIN E	Current line of the Interpolator could not be found
1500	All function blocks using SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	Given CNC program is not of type SMC_CNC_REF
1501	All function blocks using SMC_OUTQUEUE	SMC_NO_OUTQUEUE_TYPE	Given OutQueue is not of type SMC_OUTQUEUE
1600	CNC function blocks	SMC_3D_MODE_NOT_SUPPORT	Function block only works with 2D paths

Error number	Function block	ENUM value	Description
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXI ST	File does not exist
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	No buffer allocated
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMA LL	Buffer too small
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Data underrun. Buffer has been read, is empty
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE _REPLACED	Placeholder variable could not be replaced
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	Input pvI does not point to a SMC_VARLIST object
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXI ST	File could not be opened
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	No buffer defined
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMA LL	Buffer too small
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	Unexpected end of file
2100	SMC_AxisDiagnostic Log	SMC_ADL_FILE_CANNOT_BE_O PENED	File could not be opened
2101	SMC_AxisDiagnostic Log	SMC_ADL_BUFFER_OVERRUN	Buffer overrun. WriteToFile must be called more frequently
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXI ST	File could not be opened
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	Saved cam too big
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE _TYPE	Wrong compilation mode
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	File has wrong version
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	Unexpected end of file
3001	SMC_WriteDrivePara msToFile	SMC_WDPF_CHANNEL_OCCUPI ED	SMC_WDPF_TIMEOUT_PREPARI NG_LIST
3002	SMC_WriteDrivePara msToFile	SMC_WDPF_CANNOT_CREATE_ FILE	File could not be created
3003	SMC_WriteDrivePara msToFile	SMC_WDPF_ERROR_WHEN_RE ADING_PARAMS	Error at reading the parameters
3004	SMC_WriteDrivePara msToFile	SMC_WDPF_TIMEOUT_PREPARI NG_LIST	Timeout during preparing the parameter list
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	Nominator of the conversion factor dwRatioTechUnitsDenom of the Encoder reference is 0
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHER FB	Other module trying to process motion on the Encoder axis
5002	DriveInterface	SMC_ENC_FILTER_DEPTH_INVA	Filter depth is invalid



Service line:86-755-23535967 E-mail:overseas@invt.com.cn Website:www.invt.com

The products are owned by Shenzhen INVT Electric Co.,Ltd.

Two companies are commissioned to manufacture: (For product code, refer to the 2nd/3rd place of S/N on the name plate.)

INVT Power Electronics (Suzhou) Co.,Ltd. (origin code: 06) Shenzhen INVT Electric Co.,Ltd. (origin code: 01) Address: INVT Guangming Technology Building, Songbai Road, Address: No. 1 Kunlun Mountain Road, Science & Technology Matian, Guangming District, Shenzhen, China Town, Gaoxin District, Suzhou, Jiangsu, China Industrial Automation: HMI ■ PLC VFD Servo System Elevator Intelligent Control System Rail Transit Traction System DCIM Energy & Power: **■**UPS Solar Inverter SVG ■ New Energy Vehicle Powertrain System ■ New Energy Vehicle Charging System New Energy Vehicle Motor



Copyright© INVT. Manual information may be subject to change without prior notice.