DELTA

Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan City,
Taoyuan County 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Jiangsu) Ltd.
Wujiang Plant 3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province, P.R.C. 215200
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Pudong District,
ShangHai, P.R.C. 201209
TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
Tokyo Office
2-1-14 Minato-ku Shibadaimon,
Tokyo 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
1511, Byucksan Digital Valley 6-cha, Gasan-dong,
Geumcheon-gu, Seoul, Korea, 153-704
TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int’l (S) Pte Ltd.
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas

Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A.
Sao Paulo Office
Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
01332-000-São Paulo-SP-Brazil
TEL: 55 11 3568-3855 / FAX: 55 11 3568-3865

Europe

Delta Electronics (Netherlands) B.V.
Eindhoven Office
De Witbogt 20, 5652 AG Eindhoven, The Netherlands
TEL : +31 (0)40-8003800 / FAX : +31 (0)40-8003898

*We reserve the right to change the information in this manual without prior notice.

Delta EtherCAT Programming Guide

Delta EtherCAT Programming Guide

www.deltaww.com

DELTA

Smarter. Greener. Together.

Table of Contents

1 Introduction to API Library

2 EtherCAT Introduction

3 EtherCAT Operation Example

4 API List of Dynamic-Link Library

5 EtherCAT Master Configuration

6 Master Initialization

9 Cyclic Synchronous Position Mode (CSP)

15 Inverter Motion Control

16 EtherCAT Profile Torque Mode (PT)

17 Group Motion Control

18 Operation of DI/DO Module

19 Operation of AI/AO Module

20 Operation of Pulse Module (For R1-EC5621D0 Series)

25 Auto Recording Function of Motion Axis

26 Operation of Local Digital I/O

27 High-Speed Pulse Compare Function

(This page is intentionally left blank.)

Introduction to API Function Library

1

This chapter introduces APIs of EtherCAT dynamic-link library (DLL). Users can perform various functions through calling these API libraries. The contents below provide instructions on how to import API libraries into your developing environment.

1.1 How to use function libraries?

When the installation program is completed, two function libraries will be found in the folder named “lib”. They can be used in Visual Studio C and Borland C++ respectively.

Function library	Development environment
EtherCatDll.lib	Visual Studio C++
BCBEtherCAT_DLL.lib	Borland C++Builder 6

1.2 Start a new project

1.2.1 Using VC

(1) Place following instructions in the user-built project:
#include “EtherCat_DLL.h”
#include “EtherCat_DLL_Err.h”

(2) Select **Project / Setting / Link** in Visual C development environment.
Then, type “..\lib\EtherCat_DLL.lib” in **Object / Library** modules

(3) Setup completed. Users can start to operate EtherCAT DLL with API.

1.2.2 Using VB

Place “EtherCat_DLL.bas” and “EtherCat_DLL_Err.bas” in the project created by users to control EtherCAT DLL with API.

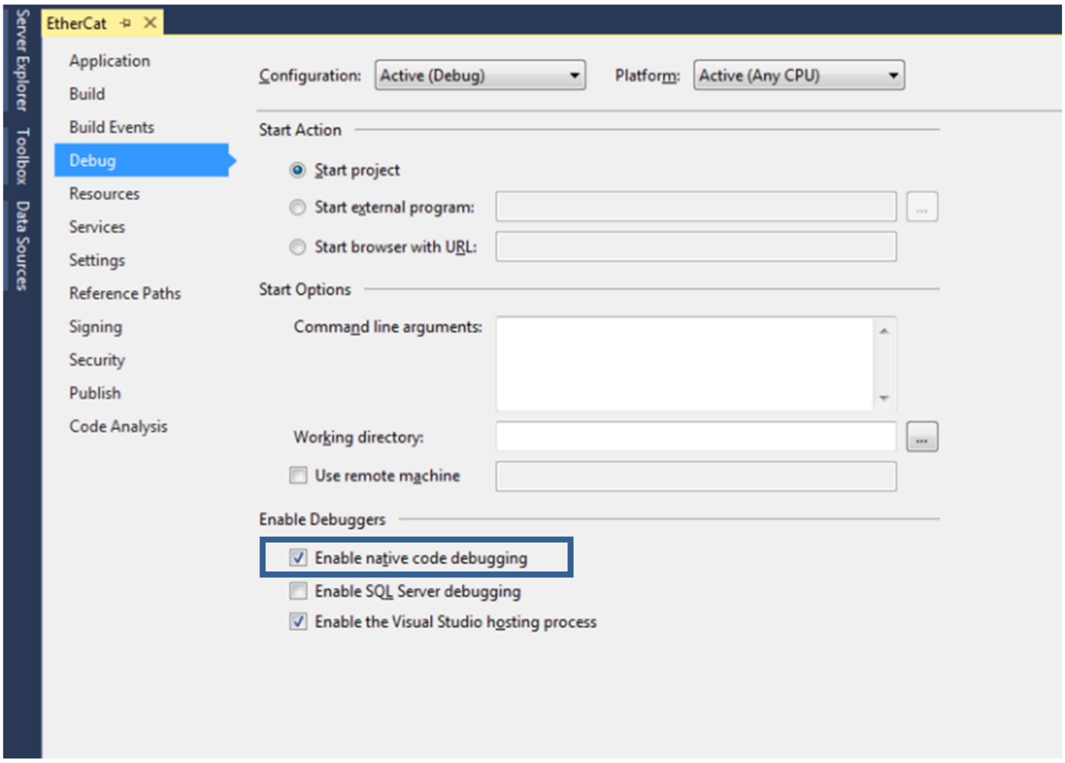
1.2.3 Using VB.Net

Place “EtherCat_DLL.vb” and “EtherCat_DLL_Err.vb” in the project created by users to control EtherCAT DLL with API.

1.2.4 Using C#

Place “EtherCat_DLL.cs” and “EtherCat_DLL_Err.cs” in user-built project to control EtherCAT DLL with API.

Note: For C# projects, please tick the **Enable native code debugging** option in the Debug tab. See figure below. Serious errors (i.e. blue screen) will occur if this option is not ticked.



Configuration: Active (Debug) Platform: Active (Any CPU)

Start Action

● Start project

○ Start external program:

○ Start browser with URL:

Start Options

Command line arguments:

Working directory:

□ Use remote machine

Enable Debuggers

☑ Enable native code debugging

□ Enable SQL Server debugging

☑ Enable the Visual Studio hosting process

Figure 1.2.4.1 Settings for C# project

(This page is intentionally left blank.)

EtherCAT Introduction 2

This chapter introduces the setting for applying Delta EtherCAT function library, including the suggested maximum slave number for connection, initialization of RTX operating system and the description on how to check if the EtherCAT associated DLL can operate normally in RTS. Please see the contents below for more information.

2.1 Maximum number of the slave device

Delta EtherCAT master has two types of operation system, RTX real-time operating system from PAC and EtherCAT PCI motion card. Since the operational efficiency among both differs from one another, the connected slave number is different.

When EtherCAT master runs in RTX real-time operating system, the default of its communication cycle is 1 ms (1K) and can connect up to remote modules and 64 motion axes, including servo drives and pulse modules. There is no limitation on quantity of the device that connects to EtherCAT master. However, if the communication cycle is shorter, the suggested maximum number of the connected slave will be reduced in accordance with proportion. Please refer to table 2.1.1 below for the suggested maximum slave quantity to avoid communication instability.

When EtherCAT master runs in PCI motion card, it can connect 64 remote modules and 32 motion axes at most with the default communication cycle 1 ms (1K). Please refer to table 2.1.2.

EtherCAT master runs in RTX real-time operating system of Delta PAC

Communication cycle (ms)	Max. number of the connected remote module (1 ms)	Max. number of the connected motion axis (1 ms)
4 ms	100	64
2 ms	100	64
1 ms	100	64
0.5 ms	50	32
0.25 ms	25	16
0.125 ms	22	8

Table 2.1.1 Suggested number of the slave

EtherCAT master runs in Delta PCI EtherCAT motion card

Communication cycle (ms)	Max. number of the connected remote module (1 ms)	Max. number of the connected motion axis (1 ms)
4 ms	64	32
2 ms	64	32
1 ms	64	32
0.5 ms	32	16
0.25 ms	16	8
0.125 ms	8	4

Table 2.1.2 Suggested number of the slave

2.2 Initialize RTX runtime environment

For running the real-time operating system uploaded by synchronous function of EtherCAT communication, RTX has one CPU and one Ethernet communication port. If you are applying a Delta EtherCAT motion card, such as PCI-L221PPI, then there is nothing to do with RTX system. When the host computer has started up, the RTX runtime environment will not be automatically loaded into the system. It will start running only after instructions related to EtherCAT initialization API are executed. Or, you can also manually enable RTX. See the steps below.

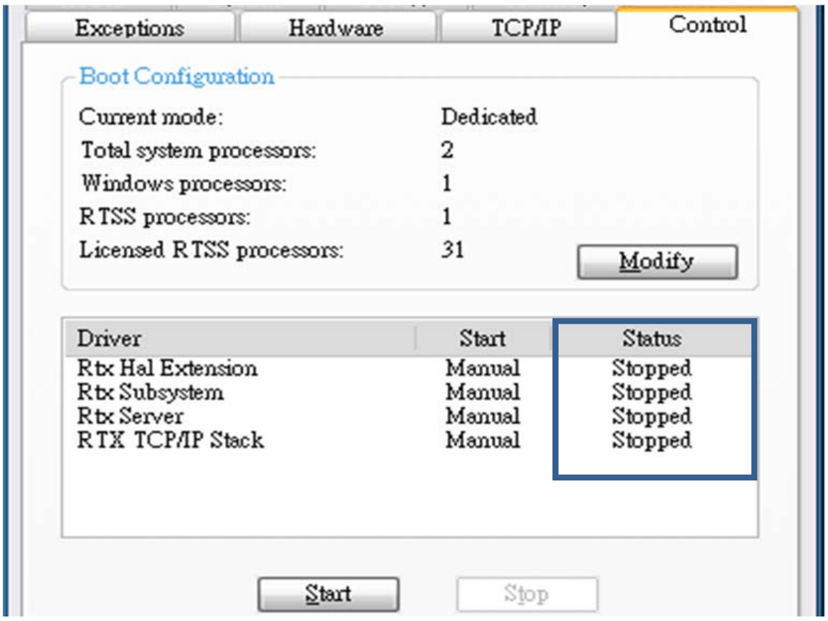
Once RTX is enabled, you can start to use other related functions and APIs in EtherCAT dynamic link library.

(1) Open [RTX Properties]

File location: [Start] > [All programs] > [IntervalZero] > [RTX 2012] > [RTX Properties]

(2) Enable RTX devices

After opening [RTX Properties], select the Control tab and check the Driver status. If the driver status shows “Stopped”, press the **Start** key to have the RTX devices start running.



Boot Configuration

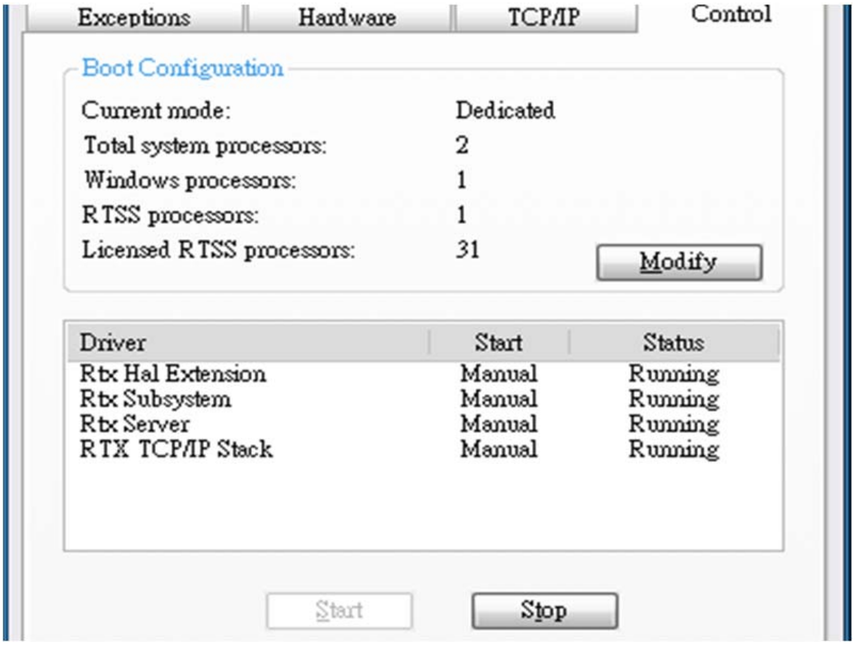
Current mode: Dedicated
Total system processors: 2
Windows processors: 1
RTSS processors: 1
Licensed RTSS processors: 31

Driver	Start	Status
Rtx Hal Extension	Manual	Stopped
Rtx Subsystem	Manual	Stopped
Rtx Server	Manual	Stopped
RTX TCP/IP Stack	Manual	Stopped

Figure 2.2.1 RTX devices in Stopped status

(3) Check the status of RTX devices

When the status of RTX devices shows “Running”, the user can start to use EtherCAT related functions.



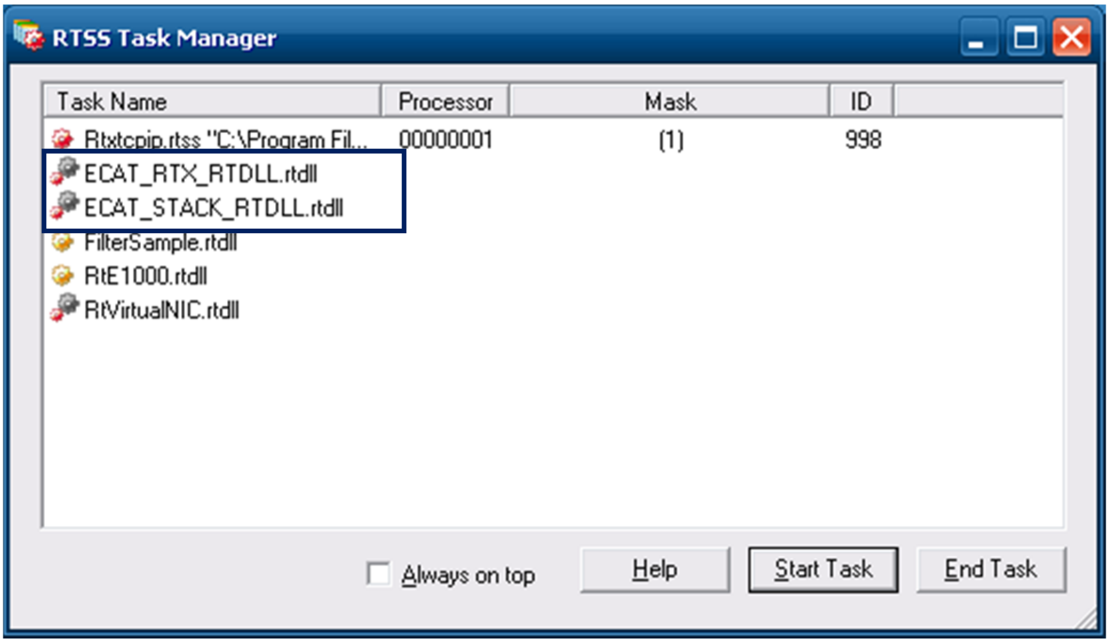
Current mode: Dedicated
Total system processors: 2
Windows processors: 1
RTSS processors: 1
Licensed RTSS processors: 31

Driver	Start	Status
Rtx Hal Extension	Manual	Running
Rtx Subsystem	Manual	Running
Rtx Server	Manual	Running
RTX TCP/IP Stack	Manual	Running

Figure 2.2.2 RTX devices in Running status

2.3 Introduction to RTSS Task Manager

RTSS Task Manager can be used to check whether the EtherCAT associated files (ECAT_RTX_RTDLL.rtdll, ECAT_STACK_RTDLL.rtdll) are operating normally in RTX.

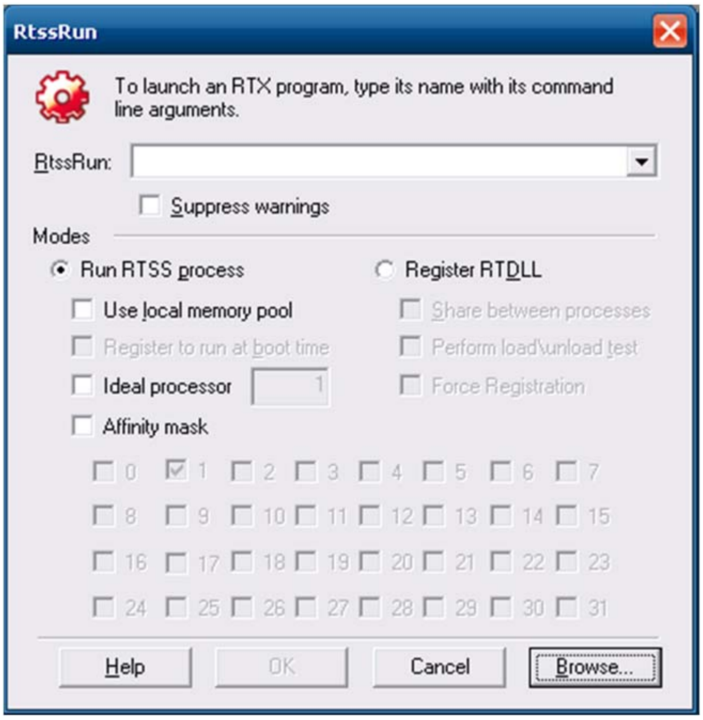


Task Name	Processor	Mask	ID
Rtxtcpip.rtss "C:\Program Fil...	00000001	(1)	998
ECAT_RTX_RTDLL.rtdll			
ECAT_STACK_RTDLL.rtdll			
FilterSample.rtdll			
RtE1000.rtdll			
RtVirtualNIC.rtdll			

Figure 2.3.1 Interface of RTSS Task Manager

If there is any file missing in Task Manager, users can reload the missing file.

(1) Click the **Start Task** key, and the RtssRun window will pop out.


RtssRun

To launch an RTX program, type its name with its command line arguments.

RtssRun:

□ Suppress warnings

Modes

● Run RTSS process ○ Register RTDLL

□ Use local memory pool □ Share between processes

□ Register to run at boot time □ Perform load\unload test

□ Ideal processor 1 □ Force Registration

□ Affinity mask

□ 0 ☑ 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7

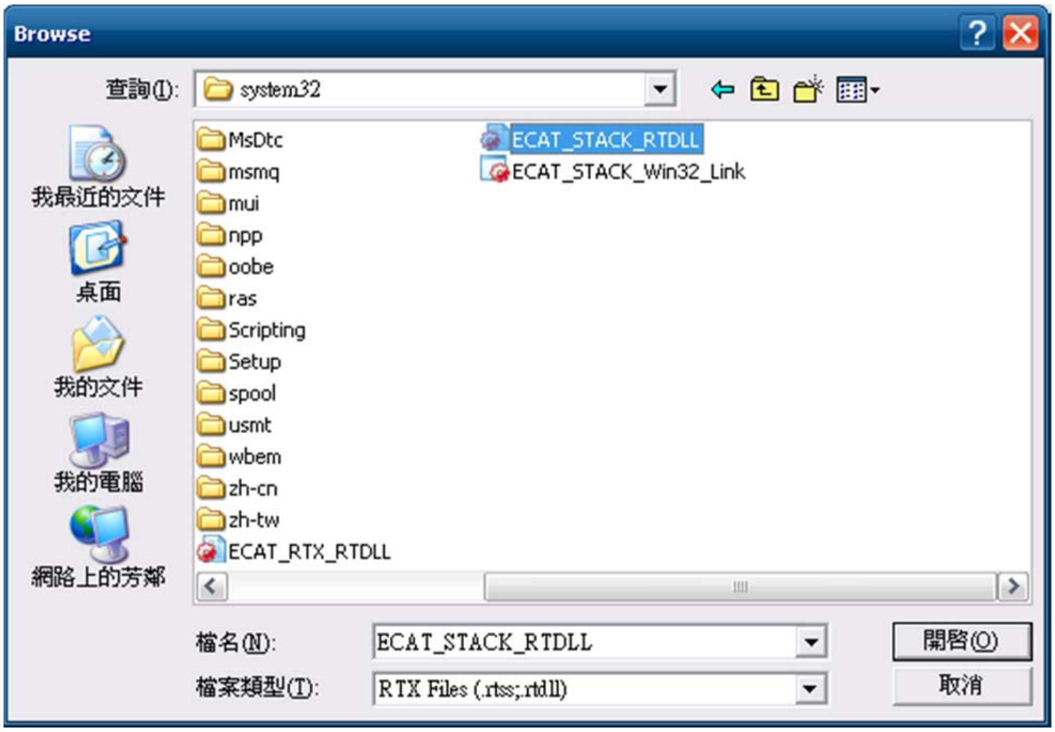
□ 8 □ 9 □ 10 □ 11 □ 12 □ 13 □ 14 □ 15

□ 16 □ 17 □ 18 □ 19 □ 20 □ 21 □ 22 □ 23

□ 24 □ 25 □ 26 □ 27 □ 28 □ 29 □ 30 □ 31

Figure 2.3.2 RtssRun window

(2) Click **Browse** and select the missing file, which is placed in C:\Windows\system32\.


Browse

查詢(I): system32

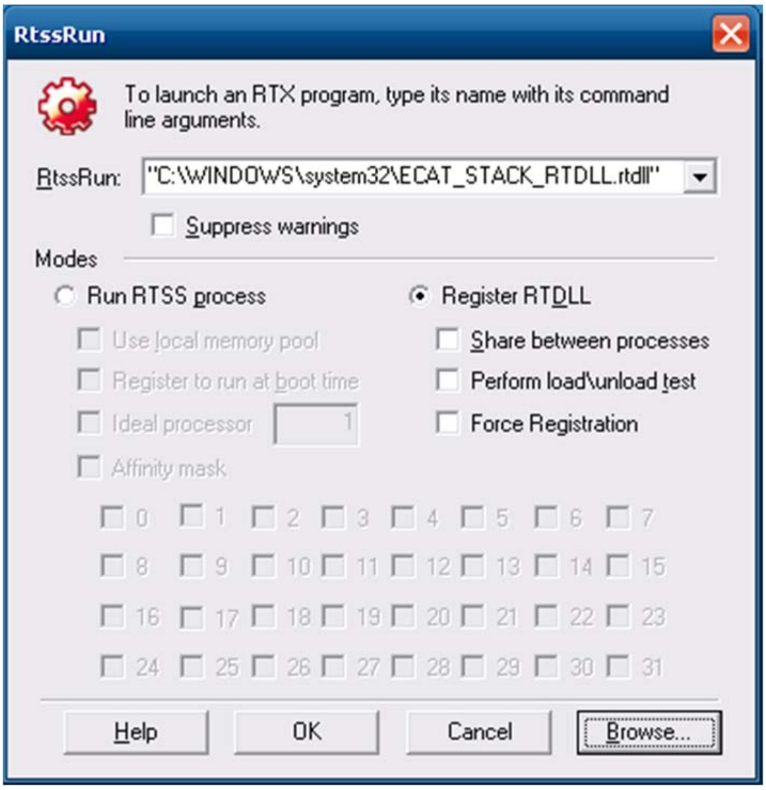
MsDtc
msmq
mui
npp
oobe
ras
Scripting
Setup
spool
usmt
wbem
zh-cn
zh-tw
ECAT_RTX_RTDLL
ECAT_STACK_RTDLL
ECAT_STACK_Win32_Link

檔名(N): ECAT_STACK_RTDLL

檔案類型(T): RTX Files (.rtss;.rtdll)

Figure 2.3.3 Select the missing file

Then, while the other options remain unchanged, press **OK** and the file will be loaded into RTX system.



RtssRun

To launch an RTX program, type its name with its command line arguments.

RtssRun: "C:\WINDOWS\system32\ECAT_STACK_RTDLL.rtdll"

□ Suppress warnings

Modes

○ Run RTSS process ● Register RTDLL

□ Use local memory pool □ Share between processes

□ Register to run at boot time □ Perform load\unload test

□ Ideal processor 1 □ Force Registration

□ Affinity mask

Figure 2.3.4 Load the missing file

EtherCAT Operation Example

3

This chapter provides the C/C++ examples of EtherCAT dynamic-link library, including EtherCAT initialization, homing procedure, PT, PV, PP, CSP modes, remote digital input/output module, analog input/output, and high-speed pulse compare function.

3.1 EtherCAT initialization

3.1.1 Function list

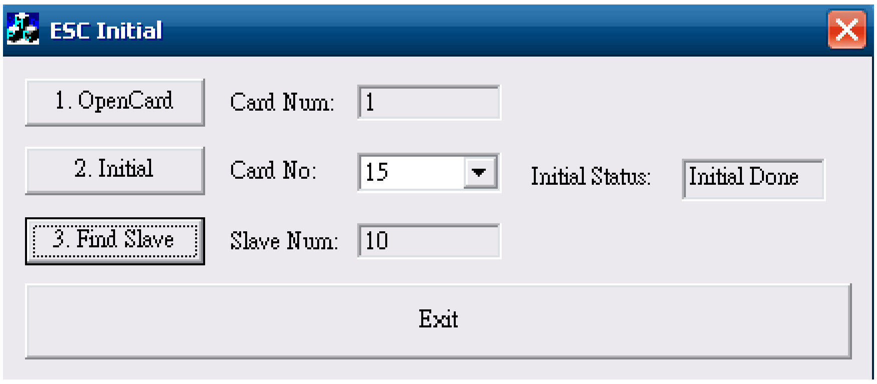
Function name
_ECAT_Master_Open
_ECAT_Master_Get_CardSeq
_ECAT_Master_Initial
_ECAT_Master_Get_SlaveNum
_ECAT_Master_Reset
_ECAT_Master_Close
_ECAT_Master_Check_Initial_Done

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.1.2 Application examples

Program interface



Card Num: 1

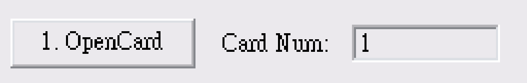
Card No: 15

Initial Status: Initial Done

Slave Num: 10

Figure 3.1.2.1

(1) Activate interface card



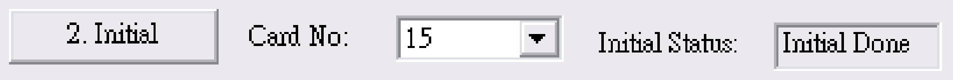
Card Num: 1

Figure 3.1.2.2

Press the **OpenCard** key to execute the following program:

```
RetCode = _ECAT_Master_Open(&gESCExistCards);
/* The variable, gESCExistCards, will return EtherCAT motion card number. */
```

(2) Initialize interface card



Card No: 15

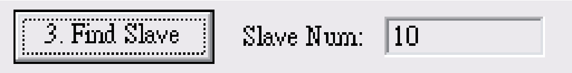
Initial Status: Initial Done

Figure 3.1.2.3

Press the **Initial** key to execute the following program:

```
for(i=0; i< gESCExistCards; i++)
{
        RetCode = _ECAT_Master_Get_CardSeq(i, &CardNo);
        /* Get the card No. of the PC interface Card i. This card No. is the dip switch value.
            EtherCAT card number in RTX version is 16. */

        RetCode = _ECAT_Master_Initial(CardNo);
        /* Start to initialize the interface card. */

        if(RetCode != 0)
        {
                strMsg.Format("_ECAT_Master_Initial, RetCode = %d", RetCode);
                MessageBox(strMsg);
        }
}
RetCode = _ECAT_Master_Check_Initial_Done(gESCCardNo, &InitialDone);
/* Get the Initial status. */
// Display the Initial Status:
// InitialDone = 1: Display “Pre Initial”
// InitialDone = 0: Display “Initial Done”
// InitialDone = 99: Display “Initial Error”
```

(3) Set the information of the connected modules



Slave Num: 10

Figure 3.1.2.5

Press the **Find Slave** key to execute the following program:

```
RetCode = _ECAT_Master_Get_SlaveNum(gESCCardNo, &SlaveNum);
// Get the number of the connected modules.
```

When the above program completes, the number of the found Slave devices will be displayed in **Slave Num** field.

(4) Exit program

Figure 3.1.2.6

Press the **Exit** key to execute the following program:

```
for(i=0; i< gESCExistCards; i++)
{
        _ECAT_Master_Reset(gpESCCardNoList[i]);
        // Reset the interface card.
}
_ECAT_Master_Close();
// End the operation of motion control card.
```

3.2 Motion control of homing

3.2.1 Function list

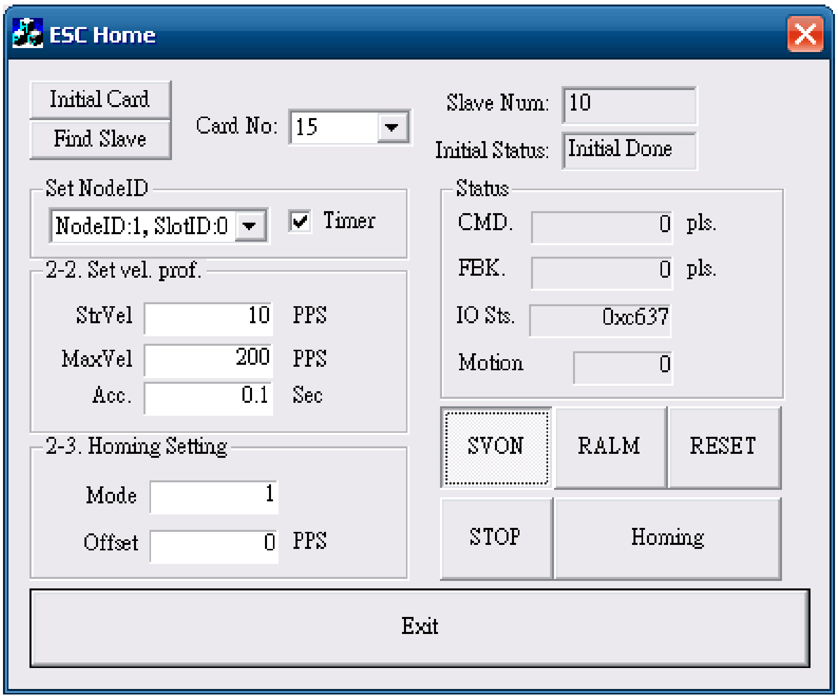
Function name
_ECAT_Slave_Home_Config
_ECAT_Slave_Home_Move
_ECAT_Slave_Motion_Sd_Stop

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.2.2 Application examples

Program interface



Card No: 15

Slave Num: 10

Initial Status: Initial Done

Set NodeID: NodeID:1, SlotID:0 ☑ Timer

2-2. Set vel. prof.

StrVel 10 PPS

MaxVel 200 PPS

Acc. 0.1 Sec

2-3. Homing Setting

Mode 1

Offset 0 PPS

Status

CMD. 0 pls.

FBK. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.2.2.1

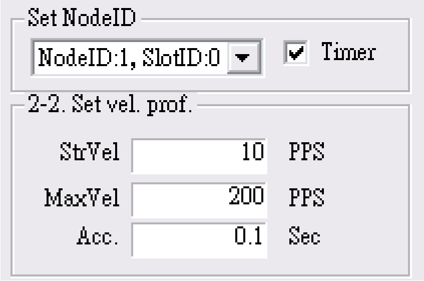
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.2.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.2.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Enter the parameter for motion control



Set NodeID

NodeID:1, SlotID:0 ☑ Timer

2-2. Set vel. prof.

StrVel 10 PPS

MaxVel 200 PPS

Acc. 0.1 Sec

Figure 3.2.2.2

Select Node ID and Slot ID and check the Timer box to display the motion status.

Set NodeID: Specify the Node ID to be executed. The parameters “AxisNo” and “SlotNo” in the API function.

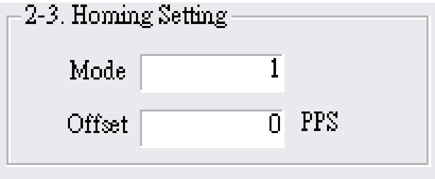
Timer: Check the Timer box to display the current motion status.

StrVel.: Input motion speed for homing (pulse sent per second). The parameter “FirstSpeed” in the API function.

MaxVel.: Input the motion speed after homing to the next index pulse (pulse sent per second). The parameter “SecondSpeed” in the API function.

Acc.: Input the duration to accelerate to the target speed. The parameter “Tacc” in the API function.

(3) Set the parameters for homing (homing mode and offset value)



2-3. Homing Setting

Mode 1

Offset 0 PPS

Figure 3.2.2.3

Mode: Homing mode 1 ~ 35. The parameter “Mode” in the API function.

Offset: Homing offset. The parameter “Offset” in the API function.

(4) Set the servo motor to ON/OFF state (servo on/servo off)

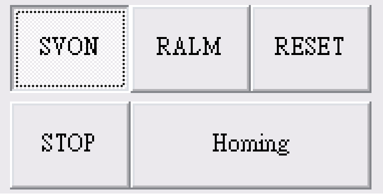


Figure 3.2.2.4

Press the **SVON** key (as shown in figure 3.2.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID,ON_OFF);
// ON_OFF
// 0: Servo OFF
// 1: Servo ON
```

(5) Homing procedures

Press the **Homing** key (as shown in figure 3.2.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Home_Config(gESCCardNo, gNodeID, gSlotID, Mode, Offset, StrVel, MaxVel, Tacc);
/* Set homing mode: 1 ~ 35, offset and speed parameters, but the servo will not operate now. */
RetCode = _ECAT_Slave_Home_Move(gESCCardNo, gNodeID, gSlotID);
/* Start homing according to the set parameters. */
```

(6) Stop homing

To stop homing, press the **STOP** key (as shown in figure 3.2.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Sd_Stop(gESCCardNo, gNodeID, gSlotID, Tdec);
/* Interrupt homing process. */
```

(7) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.3 Torque control

3.3.1 Function list

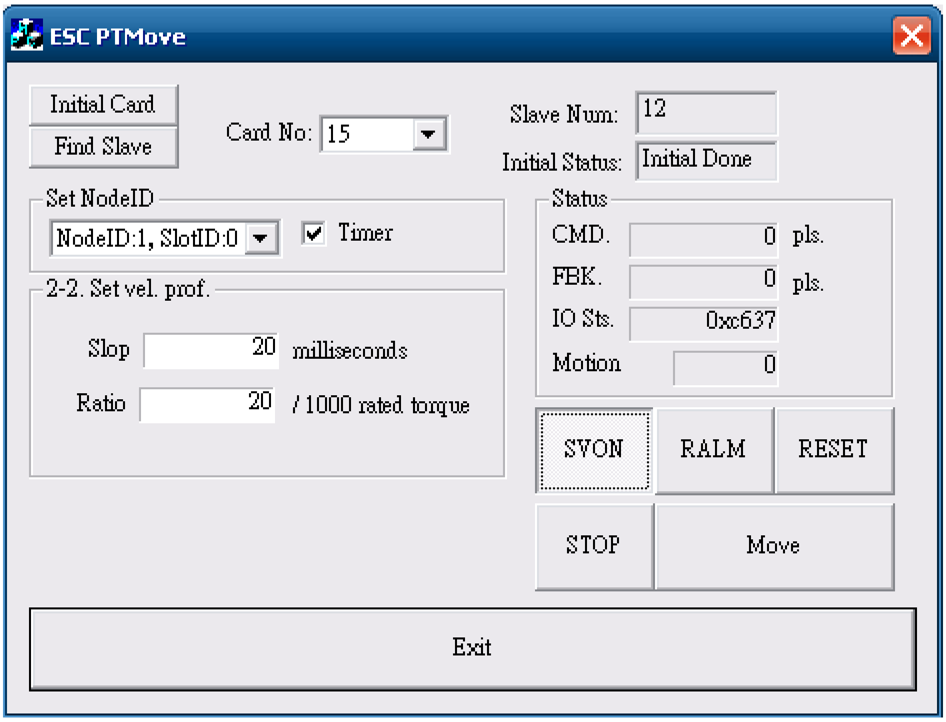
Function name
_ECAT_Slave_PT_Start_Move
_ECAT_Slave_Motion_Emg_Stop

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.3.2 Application examples

Program interface



Card No: 15

Slave Num: 12

Initial Status: Initial Done

NodeID:1, SlotID:0 ☑ Timer

Slop 20 milliseconds

Ratio 20 / 1000 rated torque

CMD. 0 pls.

FBK. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.3.2.1

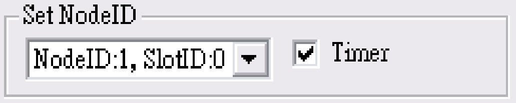
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.3.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.3.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID and Slot ID for the servo drive and enable motion status display



Set NodeID

NodeID:1, SlotID:0

☑ Timer

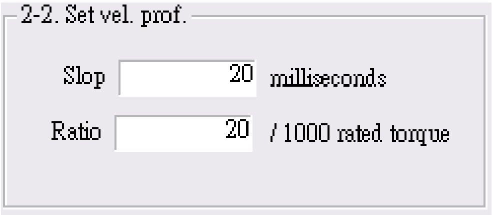
Figure 3.3.2.2

Select Node ID and Slot ID and check the Timer box to display the motion status.

Set NodeID: Specify the Node ID to be executed. The parameters "AxisNo." and "SlotNo" in the API function.

Timer: Check the Timer box to display the current motion status.

(3) Enter the Slop and Ratio values



2-2. Set vel. prof.

Slop 20 milliseconds

Ratio 20 / 1000 rated torque

Figure 3.3.2.3

Slop: Input the required time for the rated torque. (unit: ms)

Ratio: Input the permillage of the rated torque value. For example, if the ratio value is 20, the rated torque will be 2%.

(4) Set the servo motor to ON/OFF state (servo on/servo off)

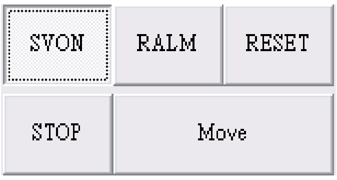


Figure 3.3.2.4

Press the **SVON** key (as shown in figure 3.3.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID, ON_OFF);
// ON_OFF:
// 0: Servo OFF.
// 1: Servo ON.
```

(5) Torque control

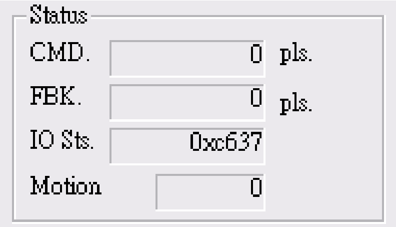
Press the **Move** key (as shown in figure 3.3.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_PT_Start_Move(gESCCardNo, gNodeID, gSlotID, Torque, Slope);
/* Set torque parameters and enable torque control. */
// The motor will run in forward direction if the torque value is greater than 0, and run in reverse direction if the value is smaller than 0.
```

Press the **STOP** key (as shown in figure 3.3.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
/* Stop torque control of the motor. */
```

(6) Status display



Status

CMD. 0 pls.

FBK. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.3.2.5

Command values of the motion:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID, gSlotID,
&Cmd);
// Get the command value (CMD. field).
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID, &Pos);
// Get the feedback value of the command (FBK. field).
```

Motion status:

```
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID, gSlotID,
&Status);
// Get the current motion status (IO Sts. field).
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone);
// Get the current status of the motor (Motion field).
```

(7) Reset feedback and clear alarm

Press the **RESET** key (as shown in figure 3.3.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
// Clear feedback first (Value showed in servo drive panel will be set to 0).
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo,gNodeID,gSlotID,0);
// Then, clear the command.
```

Press the **RALM** key (as shown in figure 3.3.2.4) to execute the alarm clearing command:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
// Clear the alarm of slave station.
```

(8) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.4 Constant speed control

3.4.1 Function list

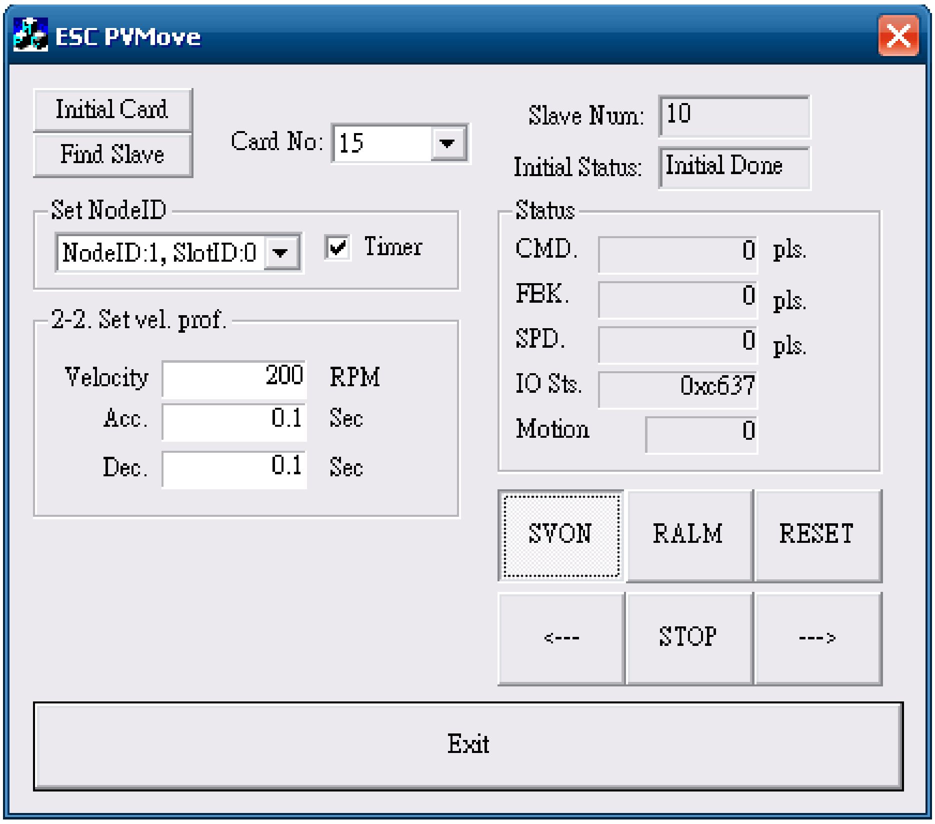
Function name
_ECAT_Slave_PV_Start_Move
_ECAT_Slave_Motion_Sd_Stop

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.4.2 Application examples

Program interface



Card No: 15

Slave Num: 10

Initial Status: Initial Done

NodeID:1, SlotID:0

☑ Timer

Velocity 200 RPM

Acc. 0.1 Sec

Dec. 0.1 Sec

CMD. 0 pls.

FBK. 0 pls.

SPD. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.4.2.1

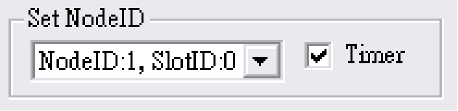
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.4.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.4.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID and Slot ID for the servo drive and enable motion status display



Set NodeID

NodeID:1, SlotID:0

☑ Timer

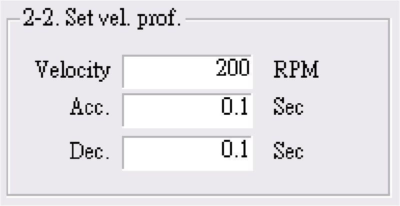
Figure 3.4.2.2

Select Node ID and Slot ID and check the Timer box to display the motion status.

Set NodeID: Specify the Node ID to be executed. The parameters “AxisNo.” and “SlotNo” in the API function.

Timer: Check the Timer box to display the current motion status.

(3) Enter the acceleration/deceleration time and rotation speed per minute (rpm).



2-2. Set vel. prof.

Velocity 200 RPM

Acc. 0.1 Sec

Dec. 0.1 Sec

Figure 3.4.2.3

Velocity: Input the target speed. The parameter “RPM” in the API function. *The actual rpm value is 0.1 time of the variable Velocity.

Acc: Input the duration from current speed to target speed. The parameter “Tacc” in the API function.

Dec: Input the duration to decelerate from current speed to 0. The parameter “Tdec” in the API function.

(4) Set the servo motor to ON/OFF state (servo on/servo off)

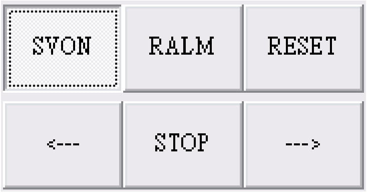


Figure 3.4.2.4

Press the **SVON** key (as shown in figure 3.4.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID, ON_OFF);
// ON_OFF:
// 0: Servo OFF.
// 1: Servo ON.
```

(5) Speed control

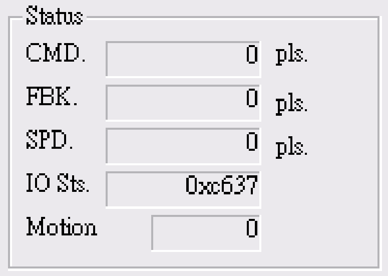
Press the ← or → key (as shown in figure 3.4.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_PV_Start_Move(gESCCardNo, gNodeID, gSlotID, Velocity,
Tacc, Tdec);
/* Set the parameters of speed mode (the acceleration and deceleration time) and enable
speed control. */
// The servo motor will run in forward direction if the rpm value is greater than 0, and will
run in reverse direction when the rpm value is smaller than 0.
```

Press the **STOP** key (as shown in figure 3.4.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
```

(6) Status display



Status

CMD. 0 pls.

FBK. 0 pls.

SPD. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.4.2.5

Command values of the motion:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID, gSlotID,
&Cmd);
// Get the command value (CMD. field).
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID, &Pos);
// Get the feedback value of the command (FBK. field).
```

Motion status:

```
RetCode = _ECAT_Slave_Motion_Get_Current_Speed(gESCCardNo,gNodeID, gSlotID,
&Spd);
// Get the current moving speed (SPD. field).
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID, gSlotID,
&Status);
// Get the current status (IO Sts. field).
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone);
// Get the current status of the motor. (Motion field)
```

(7) Reset the feedback value and clear the alarm

Press the **RESET** key (as shown in figure 3.4.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
// Clear feedback first (Value showed in servo drive panel will be set to 0).
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo,gNodeID,gSlotID,0);
// Then clear the command.
```

Press the **RALM** key (as shown in figure 3.4.2.4) to execute the alarm clearing command:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
// Clear the alarm of slave station.
```

(8) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.5 Motion control in PP mode

3.5.1 Function list

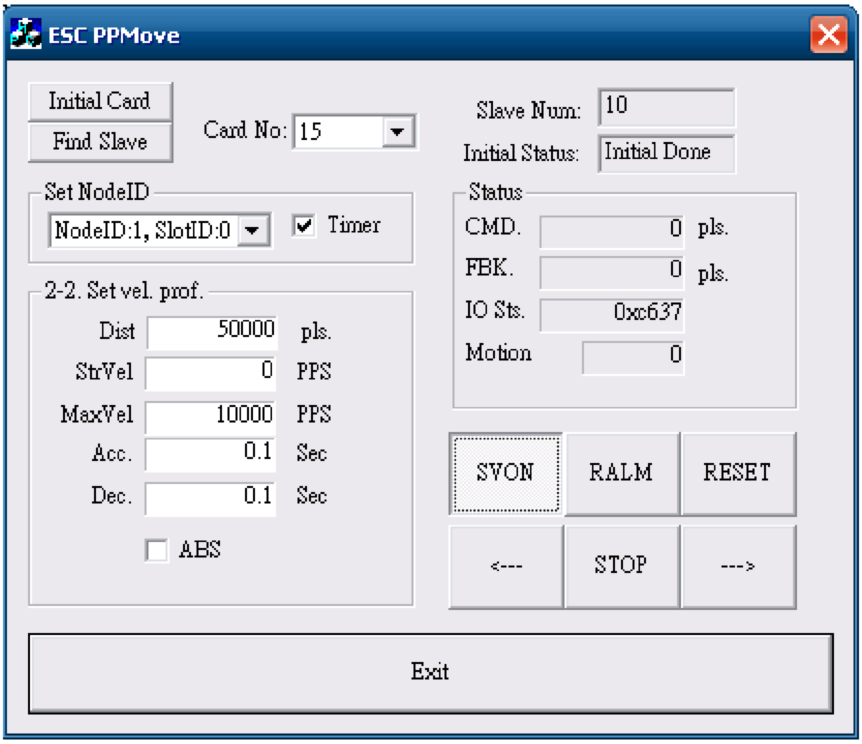
Function name
_ECAT_Slave_PP_Start_Move
_ECAT_Slave_Motion_Sd_Stop

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.5.2 Application examples

Program interface



ESC PPMove

Card No: 15

Slave Num: 10

Initial Status: Initial Done

Set NodeID

NodeID:1, SlotID:0 ☑ Timer

2-2. Set vel. prof.

Dist 50000 pls.

StrVel 0 PPS

MaxVel 10000 PPS

Acc. 0.1 Sec

Dec. 0.1 Sec

□ ABS

Status

CMD. 0 pls.

FBK. 0 pls.

IO Sts. 0xc637

Motion 0

Figure 3.5.2.1

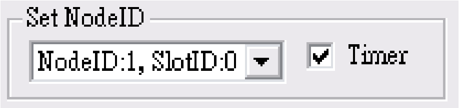
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.5.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.5.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID and Slot ID for the servo drive and enable motion status display



Set NodeID

NodeID:1, SlotID:0 ▾ ☑ Timer

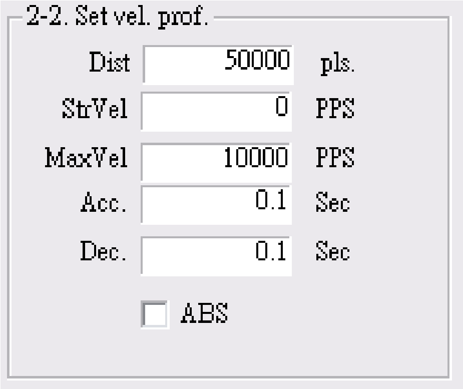
Figure 3.5.2.2

Select Node ID and Slot ID and check the Timer box to display the motion status.

Set NodeID: Specify the node ID to be executed. The parameters “AxisNo.” and “SlotNo” in the API function.

Timer: Check the Timer box to display the current motion status.

(3) Enter the parameter for motion control



2-2. Set vel. prof.

Dist	50000	pls.
StrVel	0	PPS
MaxVel	10000	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ ABS

Figure 3.5.2.3

Dist.: Input the moving distance. The parameter “Dist” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

*The servo drive accelerates to the initial speed set in StrVel at its max. speed. Then, it accelerates to the constant speed set in MaxVel with the acceleration time set in Acc.

MaxVel.: Input the constant speed. The parameter “MaxVel” in the API function.

Acc.: Input the duration from initial speed to constant speed. The parameter “Tacc” in the API function.

Dec.: Input the duration to decelerate from the constant speed to 0. The parameter “Tdec” in the API function.

ABS.: Check this box to have the motion conducted according to the absolute coordinates set in Dist.

(4) Set the servo motor to ON/OFF state (servo on/servo off)

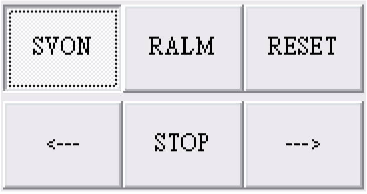


Figure 3.5.2.4

Press the **SVON** key (as shown in figure 3.5.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID, ON_OFF);
// ON_OFF:
// 0: Servo OFF
// 1: Servo ON
```

(5) Start to control the motion speed

Press the ← or → key (as shown in figure 3.5.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_PP_Start_Move(gESCCardNo, gNodeID, gSlotID, Dist, StrVel,
MaxVel, Tacc, Tdec, gbIsABS);
// gbIsABS
// 0: relative movement
// 1: absolute movement
```

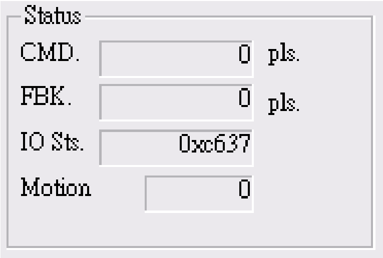
(6) Stop the motion

Press the **STOP** key (as shown in figure 3.5.2.4) to enable emergency stop:

```
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
```

In this example, the movement is stopped urgently by setting the deceleration time to 0.

(7) Status display



Status
CMD. 0 pls.
FBK. 0 pls.
IO Sts. 0xc637
Motion 0

Figure 3.5.2.5

Command values of the motion:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID, gSlotID,
&Cmd);
// Get the command value (CMD. field).
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID, &Pos);
// Get the feedback value of the command (FBK. field).
```

Motion status:

```
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID, gSlotID,
&Status);
// Get the current status (IO Sts. field).
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone);
// Get the current status of the motor (Motion field).
```

(8) Reset the feedback value and clear the alarm

Press the **RESET** key (as shown in figure 3.5.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
// Clear feedback first (Value showed in servo drive panel will be set to 0).
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo, gNodeID, gSlotID, 0);
// Then, clear the command.
```

Press the **RALM** key (as shown in figure 3.5.2.4) to execute the alarm clearing command:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
// Clear the alarm of slave station
```

(9) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.6 Motion control in CSP mode

3.6.1 Function list

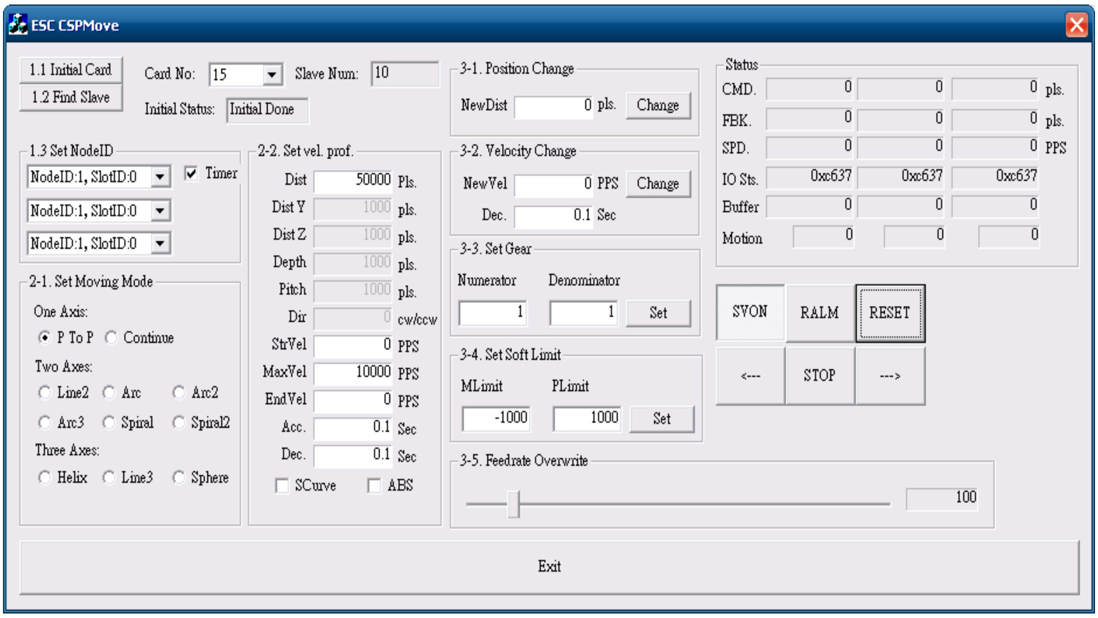
Function name
_ECAT_Slave_Motion_Set_Svon
_ECAT_Slave_CSP_Start_Move
_ECAT_Slave_CSP_Start_V_Move
_ECAT_Slave_CSP_Start_Multiaxes_Move
_ECAT_Slave_CSP_Start_Arc_Move
_ECAT_Slave_CSP_Start_Arc2_Move
_ECAT_Slave_CSP_Start_Arc3_Move
_ECAT_Slave_CSP_Start_Spiral_Move
_ECAT_Slave_CSP_Start_Spiral2_Move
_ECAT_Slave_CSP_Start_Heli_Move
_ECAT_Slave_CSP_Start_Sphere_Move
_ECAT_Slave_Motion_Sd_Stop
_ECAT_Slave_Motion_Set_Position
_ECAT_Slave_Motion_Set_Command
_ECAT_Slave_Motion_Ralm
_ECAT_Slave_Motion_Get_Command
_ECAT_Slave_Motion_Get_Position
_ECAT_Slave_Motion_Get_Current_Speed
_ECAT_Slave_Motion_Get_StatusWord
_ECAT_Slave_Motion_Get_Mdone
_ECAT_Master_Check_Initial_Done

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.6.2 Application examples

Program interface



Card No: 15 Slave Num: 10
Initial Status: Initial Done

NodeID:1, SlotID:0 ☑ Timer
NodeID:1, SlotID:0
NodeID:1, SlotID:0

One Axis: ● P To P ○ Continue
Two Axes: ○ Line2 ○ Arc ○ Arc2 ○ Arc3 ○ Spiral ○ Spiral2
Three Axes: ○ Helix ○ Line3 ○ Sphere

Dist	50000	Pls.
Dist Y	1000	pls.
Dist Z	1000	pls.
Depth	1000	pls.
Pitch	1000	pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

NewDist 0 pls.
NewVel 0 PPS
Dec. 0.1 Sec
Numerator 1 Denominator 1
MLimit -1000 PLimit 1000
100

CMD.	0	0	0	pls.
FBK.	0	0	0	pls.
SPD.	0	0	0	PPS
IO Sts.	0x637	0x637	0x637	
Buffer	0	0	0	
Motion	0	0	0	

Figure 3.6.2.1

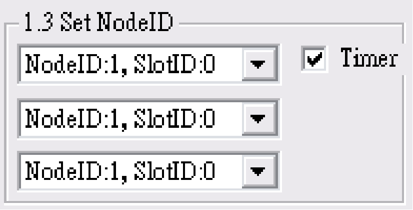
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.6.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.6.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Select Node ID and Slot ID for the servo drive and enable motion status display



NodeID:1, SlotID:0 ☑ Timer
NodeID:1, SlotID:0
NodeID:1, SlotID:0

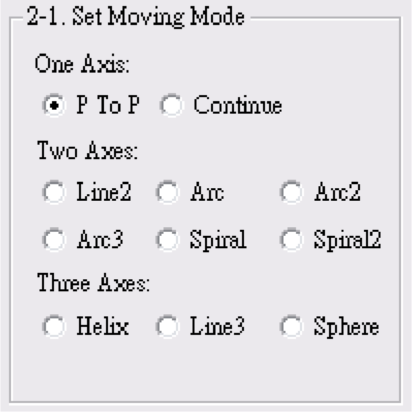
Figure 3.6.2.2

Select Node ID and Slot ID and check the Timer box to display the motion status.

Set NodeID: Specify the Node ID to be executed. The parameters “AxisNo.” and “SlotNo” in the API function.

Timer: Check the Timer box to display the current motion status.

(3) Select the moving mode



2-1. Set Moving Mode

One Axis:

● P To P ○ Continue

Two Axes:

○ Line2 ○ Arc ○ Arc2

○ Arc3 ○ Spiral ○ Spiral2

Three Axes:

○ Helix ○ Line3 ○ Sphere

Figure 3.6.2.2

Single-axis motion control:

P To P: Point to point movement

Continue: Linear movement

Two-axis motion control:

Line2: Linear interpolation control

Arc: Type 1 arc interpolation control (with the known circle center and angle)

Arc2: Type 2 arc interpolation control (with the known end point and angle)

Arc3: Type 3 arc interpolation control (with the known circle center and end point)

Spiral: Type 1 spiral interpolation control (with the known circle center and angle)

Spiral2: Type 2 spiral interpolation control (with the known end point and circle number)

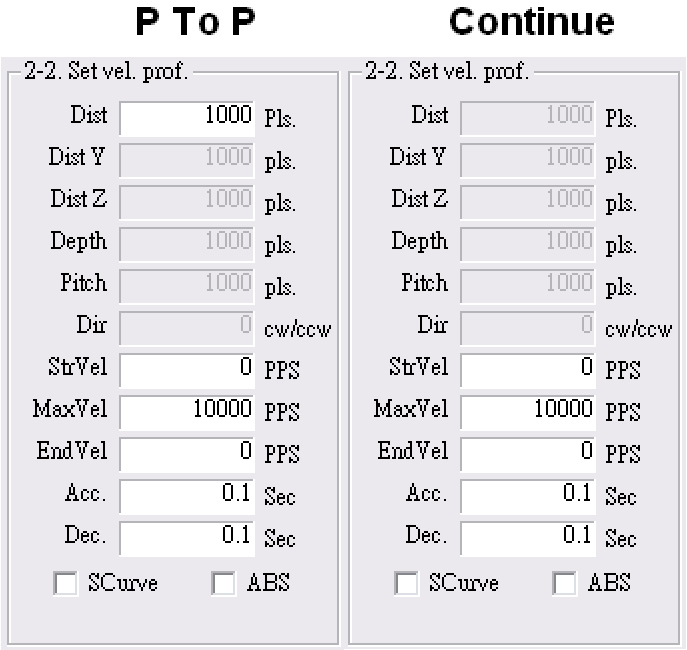
Three-axis motion control:

Helix: Helical interpolation control

Line3: Linear interpolation control

Sphere: Three-axis sphere motion control

(4) The parameter of single-axis motion control



2-2. Set vel. prof.

	P To P		Continue	
Dist	1000	Pls.	1000	Pls.
Dist Y	1000	pls.	1000	pls.
Dist Z	1000	pls.	1000	pls.
Depth	1000	pls.	1000	pls.
Pitch	1000	pls.	1000	pls.
Dir	0	cw/ccw	0	cw/ccw
StrVel	0	PPS	0	PPS
MaxVel	10000	PPS	10000	PPS
EndVel	0	PPS	0	PPS
Acc.	0.1	Sec	0.1	Sec
Dec.	0.1	Sec	0.1	Sec
	□ SCurve	□ ABS	□ SCurve	□ ABS

Figure 3.6.2.3

Dist.: Input the moving distance. The parameter “Dist” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-Curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

(5) Set the servo motor to ON/OFF state (servo on/servo off)

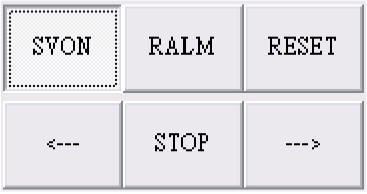


Figure 3.6.2.4

Press the **SVON** key (as shown in figure 3.6.2.4) to execute the following program:

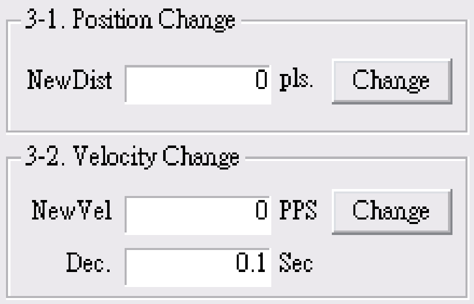
```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i], ON_OFF);
// ON_OFF:
// 0: Servo OFF
//1: Servo ON
```

(6) Select P To P and start the point to point motion control

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Move(gESCCardNo, gNodeID[0], gSlotID[0], Dist[0], StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(7) Change the position or speed in P To P mode



3-1. Position Change

NewDist 0 pls.

3-2. Velocity Change

NewVel 0 PPS

Dec. 0.1 Sec

Figure 3.6.2.5

Press the ← or → key (as shown in figure 3.6.2.4) to select P To P motion control. To replace the current position with a new position, press the **Change** key in Position Change section (as shown in figure 3.6.2.5) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_TargetPos_Change(gESCCardNo, gNodeID[0], gSlotID[0], NewPos);
// Replace the current position with a new position.
```

To replace the current speed with a new speed, press the **Change** key in Velocity Change section (as shown in figure 3.6.2.5) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Velocity_Change(gESCCardNo, gNodeID[0], gSlotID[0], NewSpd, NewTdec);
// Replace the current speed with a new speed.
```

(8) Set the Gear or software limit in P To P mode

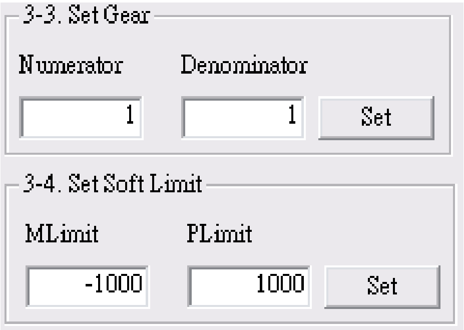


Figure 3.6.2.6

Press the ← or → key (as shown in figure 3.6.2.4) to select P To P motion control. To set the Gear values, press the **Set** key in the Set Gear section (as shown in figure 3.6.2.6) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Set_Gear(gESCCardNo, gNodeID[0], gSlotID[0],
Numerator, Denominator, Enable);
// Set new gear values.
```

To set the software limit, press the **Set** key in the Set Soft Limit section (as shown in figure 3.6.2.6) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Set_Softlimit(gESCCardNo, gNodeID[0], gSlotID[0],
MLimit, PLimit, Enable);
// Set software limit values.
```

(9) Set the value for Feedrate Overwrite in P To P mode.

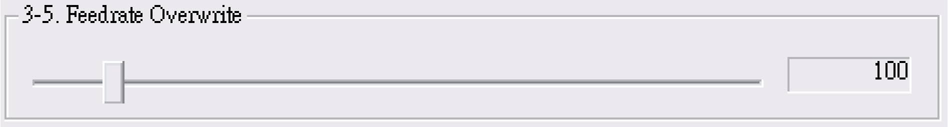


Figure 3.6.2.7

Press the ← or → key (as shown in figure 3.6.2.4) to execute P To P motion control. To set the value of Feedrate Overwrite, drag the scrollbar (as shown in figure 3.6.2.7) to execute the following program:

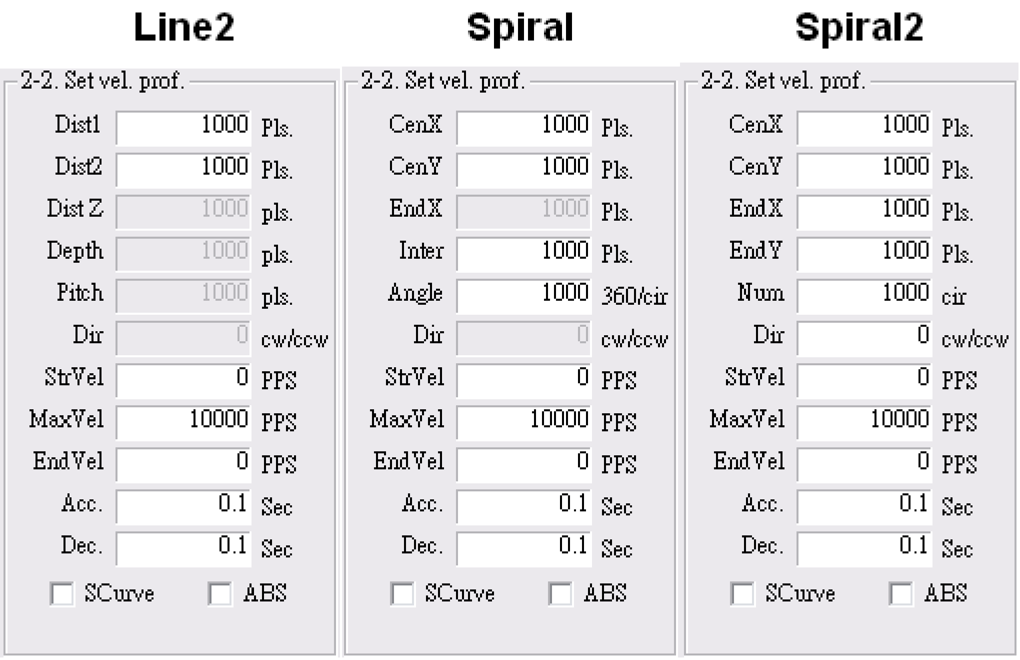
```
RetCode = _ECAT_Slave_CSP_Feedrate_Overwrite(gESCCardNo, gNodeID[0],
gSlotID[0], 2, NewSpd, 0.1);
// Mode=2; Users can change the speed and the speed (vector) of all motion commands
whether the command is being executed.
// Speed ratio
```

(10) Select Continue for motion control with constant speed

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_V_Move(gESCCardNo, gNodeID[0], gSlotID[0], 0,
StrVel, MaxVel, Tacc, gbIsSCurve);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
```

(11) Two-axis motion mode (Line2, Spiral, Spiral2) and its settings



Line2

Dist1	1000	Pls.
Dist2	1000	Pls.
Dist Z	1000	pls.
Depth	1000	pls.
Pitch	1000	pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec
□ SCurve	□ ABS	

Spiral

CenX	1000	Pls.
CenY	1000	Pls.
EndX	1000	Pls.
Inter	1000	Pls.
Angle	1000	360/cir
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec
□ SCurve	□ ABS	

Spiral2

CenX	1000	Pls.
CenY	1000	Pls.
EndX	1000	Pls.
EndY	1000	Pls.
Num	1000	cir
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec
□ SCurve	□ ABS	

Figure 3.6.2.8

Line2 parameter settings: Two-axis linear interpolation

Dist1: Input the moving distance of axis X. The parameter “DistX” in the API function.

Dist2: Input the moving distance of axis Y. The parameter “DistY” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-Curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

Spiral parameter settings: Type 1 of spiral interpolation (with the known circle center and angle)

CenX: Input the X-coordinate of the circle center. The parameter “CenterPoint” in the API function.

CenY: Input the Y-coordinate of the circle center. The parameter “CenterPoint” in the API function.

Inter: Input the relative distance between the known spiral pitches. The parameter “Spiral_Interval” in the API function.

Angle: Input the angle of the spiral movement. The parameter “Angle” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

Spiral2 parameter settings: Type 2 of spiral interpolation (with the know end point and circle number)

CenX: Input the X-coordinate of the circle center. The parameter “CenterPoint” in the API function.

CenY: Input the Y-coordinate of the circle center. The parameter “CenterPoint” in the API function.

EndX: Input the target position of X-coordinate. The parameter “EndPoint” in the API function.

EndY: Input the target position of Y-coordinate. The parameter “EndPoint” in the API function.

Num: Input the circle number of the spiral movement. The parameter “CycleNum” in the API function.

Dir: Input the moving direction (0: Clockwise, 1: Anticlockwise). The parameter “Dir” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter "TPhase1" in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter "TPhase2" in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

(12) Set the servo motor to ON/OFF state (servo on/servo off)

Press the **SVON** key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i],
ON_OFF);
// ON_OFF:
// 0: Servo OFF
// 1: Servo ON
```

(13) Select Line2 for two-axis linear motion control

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Multiaxes_Move(gESCCardNo, 2, gNodeID,
gSlotID, Dist, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(14) Select Spiral for two-axis arc motion control (center point and angle)

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

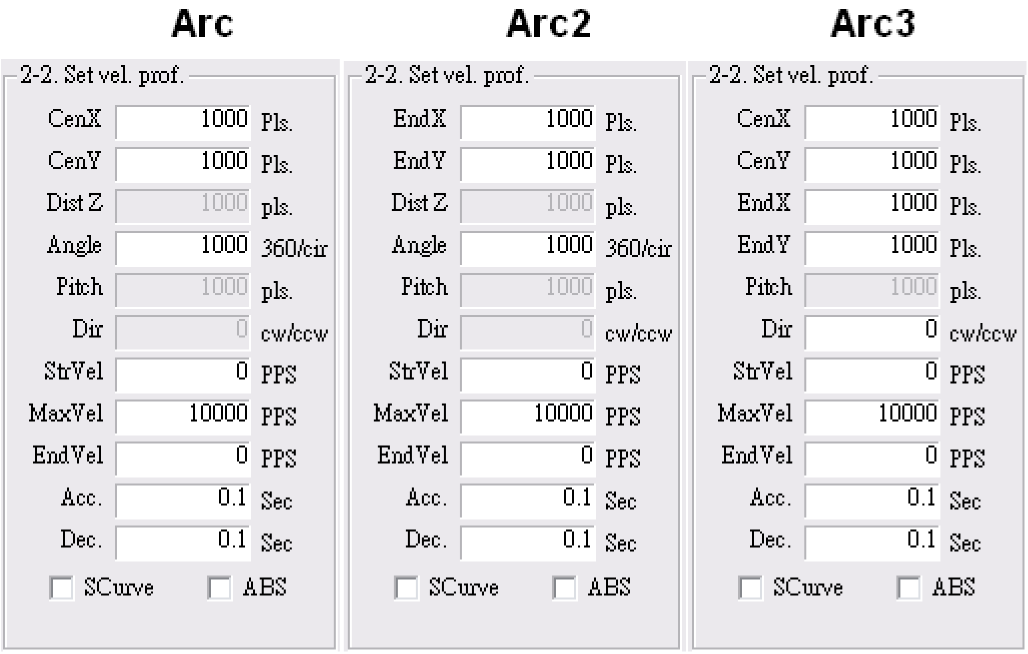
```
RetCode = _ECAT_Slave_CSP_Start_Spiral_Move(gESCCardNo,gNodeID, gSlotID,
CenPoint, Spiral_Interval, Angle, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve,
gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(15) Select Spiral2 for two-axis arc motion control (end point and circles).

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Spiral2_Move(gESCCardNo, gNodeID, gSlotID,
CenPoint, EndPoint, CycleNum, Dir, StrVel, MaxVel, EndVel, Tacc, Tdec, gblsSCurve,
gblsABS);
// gblsSCurve
// 1: T-Curve
// 2: S-Curve
// gblsABS
// 0: Relative movement
// 1: Absolute movement
```

(16) Two-axis motion mode (Arc, Arc2, Arc3) and the settings



Arc

2-2. Set vel. prof.

CenX	1000	Pls.
CenY	1000	Pls.
Dist Z	1000	pls.
Angle	1000	360/cir
Pitch	1000	pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Arc2

2-2. Set vel. prof.

EndX	1000	Pls.
EndY	1000	Pls.
Dist Z	1000	pls.
Angle	1000	360/cir
Pitch	1000	pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Arc3

2-2. Set vel. prof.

CenX	1000	Pls.
CenY	1000	Pls.
EndX	1000	Pls.
EndY	1000	Pls.
Pitch	1000	pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Figure 3.6.2.9

Arc parameter settings: Type 1 of arc interpolation (with the known arc’s circle center and angle)

CenX: Input the X-coordinate of the circle center. The parameter “CenterPoint” in the API function.

CenY: Input the Y-coordinate of the circle center. The parameter “CenterPoint” in the API function.

Angle: Set the arc angle. The parameter “Angle” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter

"TPhase1" in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter "TPhase2" in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

Arc2 parameter settings: Type 2 of arc interpolation (with the known arc's end point and angle)

EndX: Input the target position of X-coordinate. The parameter "EndPoint" in the API function.

EndY: Input the target position of Y-coordinate. The parameter "EndPoint" in the API function.

Angle: Set the arc angle. The parameter "Angle" in the API function.

StrVel.: Input the initial speed. The parameter "StrVel" in the API function.

ConstVel: Input the constant speed. The parameter "ConstVel" in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter "EndVel" in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter "TPhase1" in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter "TPhase2" in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

Arc3 parameter settings: Type 3 of arc interpolation (with the known arc's circle center and end point)

CenX: Input the X-coordinate of the circle center. The parameter "CenterPoint" in the API function.

CenY: Input the Y-coordinate of the circle center. The parameter "CenterPoint" in the API function.

EndX: Input the target position of X-coordinate. The parameter "EndPoint" in the API function.

EndY: Input the target position of Y-coordinate. The parameter "EndPoint" in the API function.

Dir: Input the moving direction (0: Clockwise, 1: Anticlockwise). The parameter "Dir" in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

(17) Set the servo motor to ON/OFF state (servo on/servo off)

Press the **SVON** key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i],
ON_OFF);
// ON_OFF
// 0: Servo OFF
// 1: Servo ON
```

(18) Select Arc for two-axis arc motion (circle center and angle)

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Arc_Move(gESCCardNo, gNodeID, gSlotID,
CenPoint, Angle, StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(19) Select Spiral for two-axis arc motion (end point and angle)

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

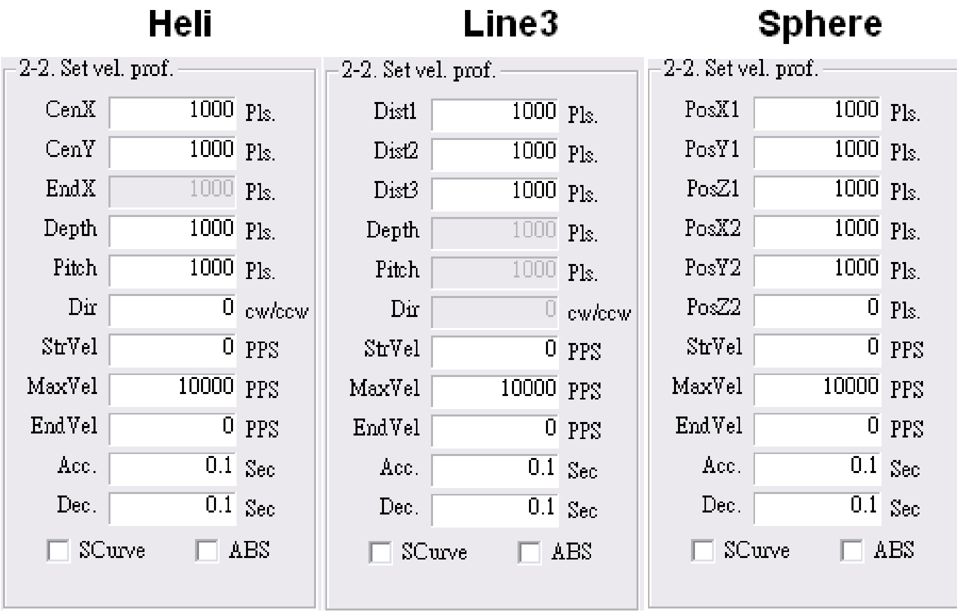
```
RetCode = _ECAT_Slave_CSP_Start_Arc2_Move(gESCCardNo, gNodeID,
gSlotID, EndPoint, Angle, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve,
gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(20) Select Spiral2 for two-axis arc motion (circle center and end point)

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Arc3_Move(gESCCardNo, gNodeID, gSlotID,
CenPoint,EndPoint,Dir,StrVel, ConstVel,EndVel,Tacc,Tdec,gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

(21) Three-axis motion mode (Heli, Line3, Sphere) and the settings



Heli

2-2. Set vel. prof.

CenX	1000	Pls.
CenY	1000	Pls.
EndX	1000	Pls.
Depth	1000	Pls.
Pitch	1000	Pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Line3

2-2. Set vel. prof.

Dist1	1000	Pls.
Dist2	1000	Pls.
Dist3	1000	Pls.
Depth	1000	Pls.
Pitch	1000	Pls.
Dir	0	cw/ccw
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Sphere

2-2. Set vel. prof.

PosX1	1000	Pls.
PosY1	1000	Pls.
PosZ1	1000	Pls.
PosX2	1000	Pls.
PosY2	1000	Pls.
PosZ2	0	Pls.
StrVel	0	PPS
MaxVel	10000	PPS
EndVel	0	PPS
Acc.	0.1	Sec
Dec.	0.1	Sec

□ SCurve □ ABS

Figure 3.6.2.10

Heli parameter settings: Three-axis helical interpolation.

CenX: Input the X-coordinate of the circle center. The parameter “CenterPoint” in the API function.

CenY: Input the Y-coordinate of the circle center. The parameter “CenterPoint” in the API function.

Depth: Input the depth of the specified axis (the overall height of Z-axis). The parameter “Depth” in the API function.

Pitch: Specify the pitch of the helix. The parameter “Pitch” in the API function.

Dir: Input the moving direction (0: Clockwise, 1: Anticlockwise). The parameter “Dir” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position.The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates set in Dist.

Line3 parameter settings: Linear interpolation.

Dist1: The moving distance of X-axis. The parameter “DistArray” in the API function.

Dist2: The moving distance of Y-axis. The parameter “DistArray” in the API function.

Dist3: The moving distance of Z-axis. The parameter “DistArray” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates in Dist.

Sphere parameter settings: Three-axis sphere motion (with given three points).

PosX1: The point to be passed through on X-axis (between starting and end point). The parameter “Target1Point” in the API function.

PosY1: The point to be passed through on Y-axis (between starting and end point). The parameter “Target1Point” in the API function.

PosZ1: The point to be passed through on Z-axis (between starting and end point). The parameter “Target1Point” in the API function.

PosX2: The target coordinate of X-axis. The parameter “Target2Point” in the API function.

PosY2: The target coordinate of Y-axis. The parameter “Target2Point” in the API function.

PosZ2: The target coordinate of Z-axis. The parameter “Target2Point” in the API function.

StrVel.: Input the initial speed. The parameter “StrVel” in the API function.

ConstVel: Input the constant speed. The parameter “ConstVel” in the API function.

EndVel: Input the end speed when it reaches the target position. The parameter “EndVel” in the API function.

TPhase1: Input the duration from initial speed to constant speed. The parameter “TPhase1” in the API function.

TPhase2: Input the duration from constant speed to end speed. The parameter “TPhase2” in the API function.

S-Curve: Check this box to use S-curve for the speed curve. Otherwise, it will use T-Curve.

ABS.: Check this box to have the motion conducted according to absolute coordinates in Dist.

(22) Set the servo motor to ON/OFF state (servo on/servo off)

Press the **SVON** key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i],
ON_OFF);
// ON_OFF
// 0: Servo OFF
// 1: Servo ON
```

(23) Select Heli for three-axis helical motion

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Heli_Move(gESCCardNo, gNodeID, gSlotID,
CenPoint,Depth,Pitch,Dir,StrVel,ConstVel,EndVel,Tacc,Tdec,gbIsSCurve,gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement
```

See the figure below:

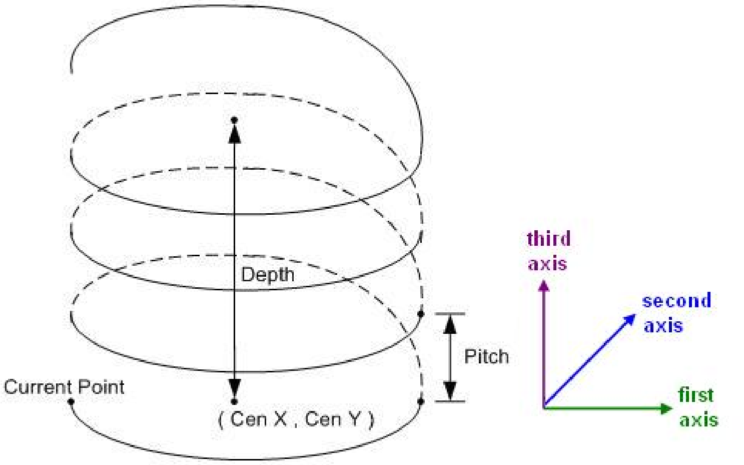


Figure 3.6.2.11

(24) Select Line3 for three-axis linear motion

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

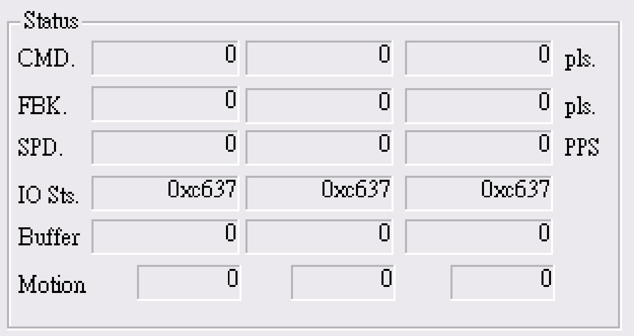
```
RetCode = _ECAT_Slave_CSP_Start_Multiaxes_Move(gESCCardNo, 3, gNodeID,
gSlotID, Dist, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement.
```

(25) Select Sphere for three-axis sphere motion

Press the ← or → key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_CSP_Start_Sphere_Move(gESCCardNo, gNodeID, gSlotID,
Dist, Dist2, StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
// gbIsSCurve
// 1: T-Curve
// 2: S-Curve
// gbIsABS
// 0: Relative movement
// 1: Absolute movement.
```

(26) Status display



Status

CMD.	0	0	0	pls.
FBK.	0	0	0	pls.
SPD.	0	0	0	PPS
IO Sts.	0xc637	0xc637	0xc637	
Buffer	0	0	0	
Motion	0	0	0	

Figure 3.6.2.12

Command values of the motion:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID, gSlotID,
&Cmd);
// Get the command value (CMD. field).
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID, &Pos);
// Get the feedback value of the command (FBK. field).
```

Motion status:

```
RetCode = _ECAT_Slave_Motion_Get_Current_Speed(gESCCardNo,gNodeID, gSlotID,
&Spd);
// Get the moving speed (SPD. field).
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID, gSlotID,
&Status);
// Get the current status (IO Sts. field).
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone);
// Get the current status of the motor (Motion field).
RetCode = _ECAT_Slave_Motion_Get_Buffer_Length(gESCCardNo, gNodeID, gSlotID,
&BufLen);
// Get the current buffer status (Buffer field).
```

(27) Reset the feedback position and clear the alarm

Press the **RESET** key (as shown in figure 3.6.2.4) to execute the following program:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
// Clear feedback first (Value in servo drive panel will be set to 0).
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo, gNodeID, gSlotID, 0);
// Then, clear the command.
```

Press the **RALM** key (as shown in figure 3.6.2.4) to execute the alarm clearing command:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
// Clear the alarm of slave station.
```

(28) Stop the motion

Press the **STOP** key (see figure 3.6.2.4) to decelerate to stop:

```
RetCode = _ECAT_Slave_Motion_Sd_Stop(gESCCardNo, gNodeID[0], gSlotID[0], Tdec);
```

In this example, the motion decelerates to stop, which is to stop the motion gradually according to the set deceleration time.

(29) Exit program

Press the **Exit** key to exit and close the program.

Execute “_ECAT_Master_Reset” and “_ECAT_Master_Close” to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.7 Digital input module

3.7.1 Function list

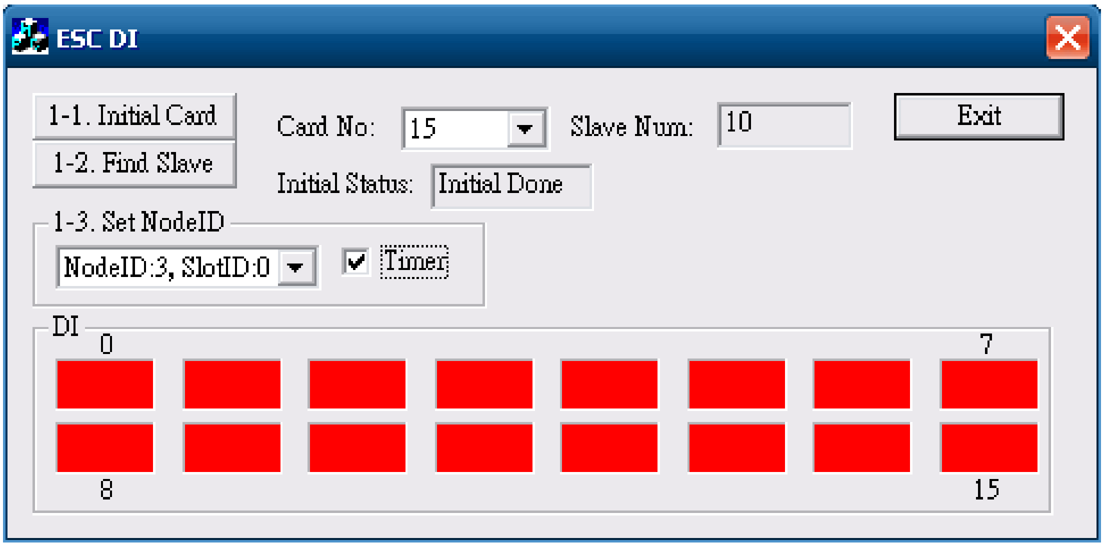
Function name
_ECAT_Slave_DIO_Get_Input_Value

■ **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.7.2 Application examples

Program interface



Card No: 15

Slave Num: 10

Initial Status: Initial Done

NodeID:3, SlotID:0

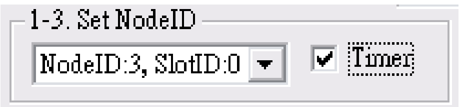
☑ Timer

Figure 3.7.2.1

(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.7.2.1) to start initializing the interface card. Press the **Find Slave** key (see figure 3.7.2.1) to start searching the connecting modules. For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID and Slot ID for the module and enable contact status display



NodeID:3, SlotID:0

☑ Timer

Figure 3.7.2.2

Select Node ID and Slot ID and check the Timer box to display the contact status.

Set NodeID: Specify the Node ID to be executed.The parameter “NodeID” and “SlotID” in the API function.

Timer: Check the Timer box to display the current contact status.

(3) Digital input (Slave DI)

To obtain the data sent from the digital input module, users have to use R1-EC-60X2 module and execute the program below:

```
RetCode = _ECAT_Slave_DIO_Get_Input_Value(gESCCardNo, gNodeID, gSlotID,
&gValue);
```

As shown in figure 3.7.2.3, no signal input is displayed in R1-EC-60X2 module.

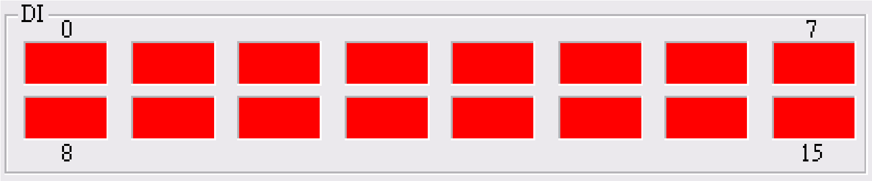


Figure 3.7.2.3

(4) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.8 Digital output module

3.8.1 Function list

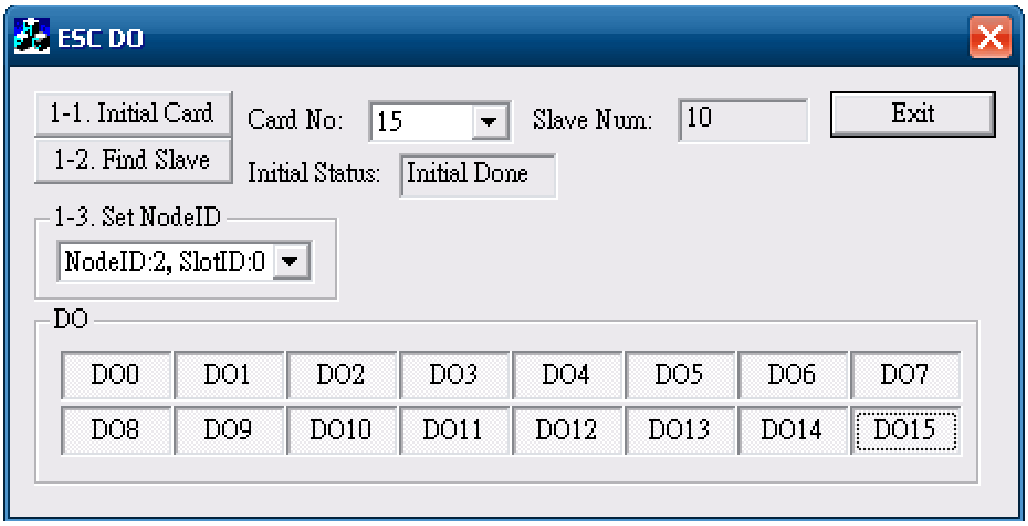
Function name
_ECAT_Slave_DIO_Set_Output_Value
_ECAT_Slave_DIO_Get_Output_Value

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.8.2 Application examples

Program interface



ESC DO

Card No: 15

Slave Num: 10

Initial Status: Initial Done

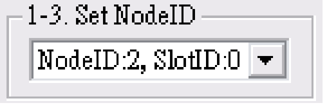
1-3. Set NodeID: NodeID:2, SlotID:0

Figure 3.8.3.1

(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.8.3.1) to start initializing the interface card. Press the **Find Slave** key (see figure 3.8.3.1) to start searching the connecting modules. For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID and Slot ID for the module and enble contact status display



1-3. Set NodeID: NodeID:2, SlotID:0

Figure 3.8.3.2

Select Node ID and Slot ID and check the Timer box to display the contact status.

Set NodeID: Specify the Node ID to be executed. The parameters “NodeID” and SlotID” in the API function.

(3) Digital output

To output data via the digital output module, users have to use R1-EC-70X2 module and execute the program below:

```
RetCode = ECAT_Slave_DIO_Set_Output_Value(gESCCardNo,gNodeID,gSlotID,
gValue);
```

The status of the digital output module can be obtained through the following program:

```
RetCode = _ECAT_Slave_DIO_Get_Output_Value(gESCCardNo,gNodeID,gSlotID,
&gValue);
```

As shown in the below figure, DO0 ~ DO15 are the output signals of Y00 ~ Y15 of R1-EC-70X2 module Port 0.

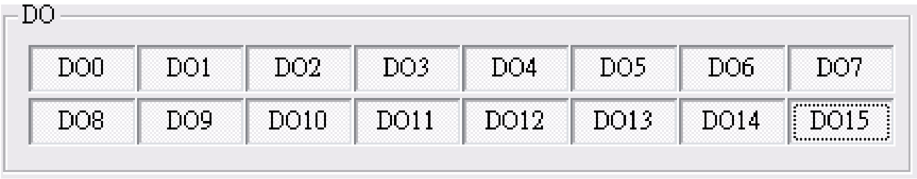


Figure 3.8.3.3

(4) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.9 Analog input module

3.9.1 Function list

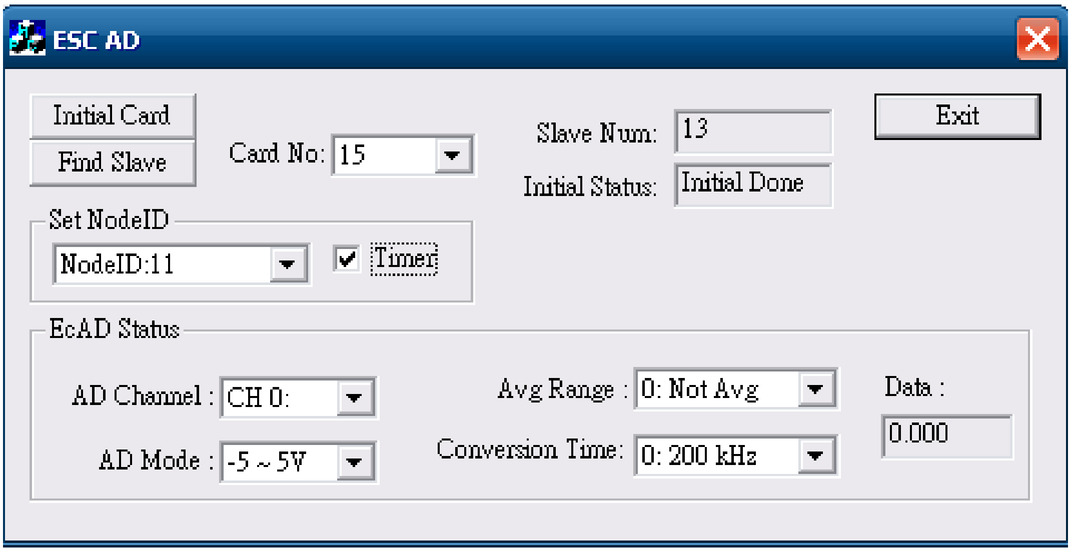
Function name
_ECAT_Slave_AIO_Set_Input_RangeMode
_ECAT_Slave_R1_EC8124_Set_Input_AverageMode
_ECAT_Slave_AIO_Set_Input_ConvstFreq_Mode
_ECAT_Slave_AIO_Get_Input_Value

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.9.2 Application examples

Program interface



ESC AD

Card No: 15

Slave Num: 13

Initial Status: Initial Done

Set NodeID

NodeID:11

☑ Timer

EcAD Status

AD Channel : CH 0:

AD Mode : -5 ~ 5V

Avg Range : 0: Not Avg

Conversion Time: 0: 200 kHz

Data : 0.000

Figure 3.9.2.1

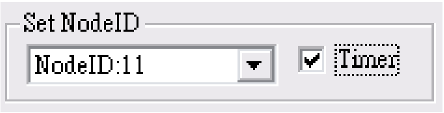
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.9.2.1) to start initializing the interface card.

Press the **Find Slave** key (see figure 3.9.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Set Node ID for the module and enable contact status display



Set NodeID

NodeID:11

☑ Timer

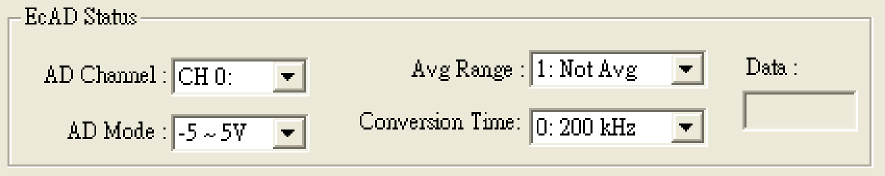
Figure 3.9.2.2

Select Node ID and check the Timer box to display the contact status.

Set NodeID: Specify the Node ID to be executed. The parameters “AxisNo” and “SlotNo” in the API function.

Timer: Check the Timer box to display the current contact status.

(3) Select AD Channel, AD Mode, Avg Range, and Conversion Time.



EcAD Status

AD Channel : CH 0:

AD Mode : -5 ~ 5V

Avg Range : 1: Not Avg

Conversion Time: 0: 200 kHz

Data :

Figure 3.9.2.3

AD Channel: Select the AD channel (CH 0 ~ 3). The parameter “SlotNo” in the API function.

AD Mode: Select the AD range. The parameter “RangeMode” in the API function.

Avg Range: Select the sample rate for the wave display. The parameter “AvgMode” in the API function.

Conversion Time: Select the conversion time. The parameter “Mode” in the API function.

(4) When selecting **AD Channel** and **AD Mode** (as shown in figure 3.9.2.3), the following program is executed:

```
RetCode = _ECAT_Slave_AIO_Set_Input_RangeMode(gESCCardNo,gNodeID, gSlotID,
Mode);
// SlotID is the channel of analog input
```

When selecting **Avg Range** (as shown in figure 3.9.2.3), the following program is executed:

```
RetCode = _ECAT_Slave_R1_EC8124_Set_Input_AverageMode(gESCCardNo,
gNodeID, gSlotID, AvgMode);
```

When selecting **Conversion Time** (as shown in figure 3.9.2.3), the following program is executed:

```
RetCode = _ECAT_Slave_AIO_Set_Input_ConvstFreq_Mode(gESCCardNo, gNodeID,
gSlotID, Mode);
```

To display **Data** field (as shown in figure 3.9.2.3), execute the following program:

```
RetCode = _ECAT_Slave_AIO_Get_Input_Value(gESCCardNo,gNodeID,gSlotID,
&Value);
```

(5) Exit program

Press the **Exit** key to exit and close the program.

Execute “_ECAT_Master_Reset” and “_ECAT_Master_Close” to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.10 Analog output module

3.10.1 Function list

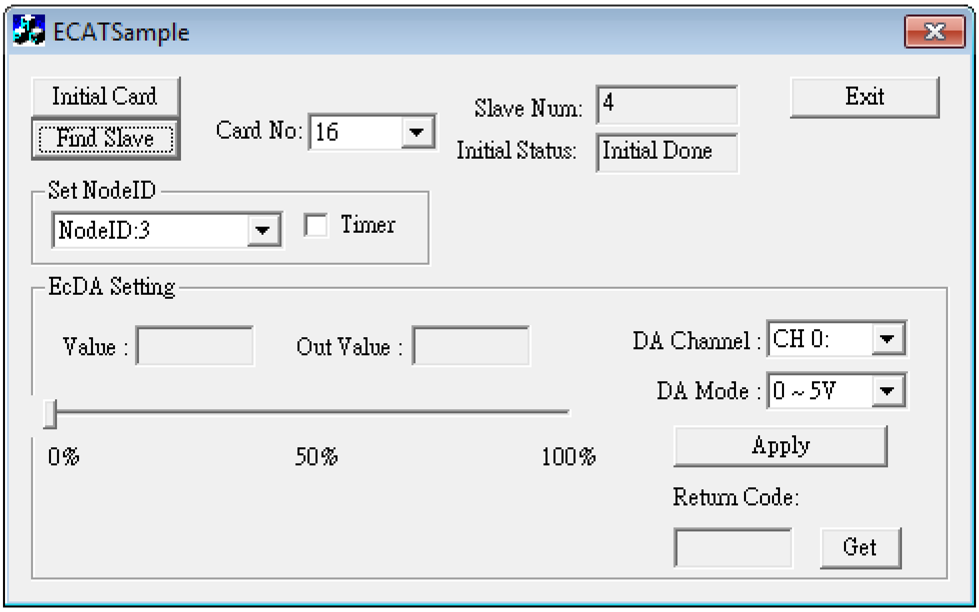
Function name
_ECAT_Slave_AIO_Set_Output_RangeMode
_ECAT_Slave_AIO_Set_Output_OverRange_Enable
_ECAT_Slave_R1_EC9144_Get_Output_ReturnCode
_ECAT_Slave_AIO_Set_Output_Value
_ECAT_Slave_AIO_Get_Output_Value

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	Y	Y

3.10.2 Application examples

Program interface



Card No: 16

Slave Num: 4

Initial Status: Initial Done

NodeID:3

Value :

Out Value :

DA Channel : CH 0:

DA Mode : 0 ~ 5V

Return Code:

Figure 3.10.2.1

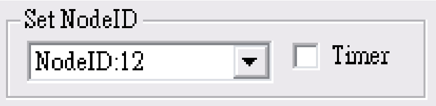
(1) Activate and initialize interface card

Press the **Initial Card** key (as shown in figure 3.10.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.10.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see "Activate interface card" and "Initialize interface card" in section 3.1.2.

(2) Set Node ID for the module and enable contact status display



Set NodeID

NodeID:12

□ Timer

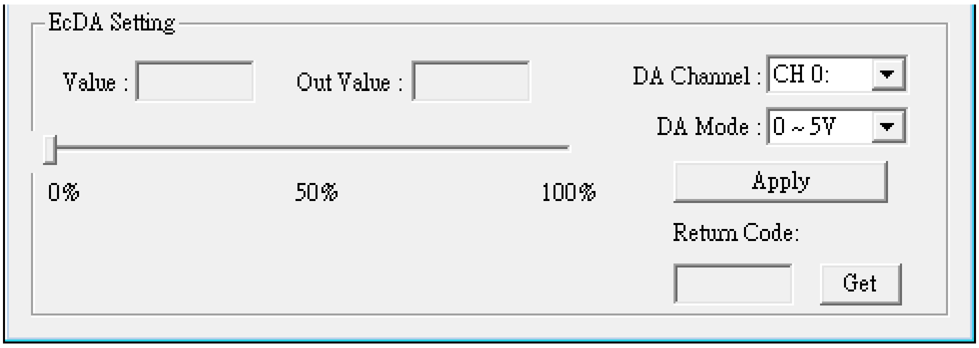
Figure 3.10.2.2

Select Node ID and check the Timer box to display the contact status.

Set NodeID: Specify the Node ID to be executed. The parameters "AxisNo" and "SlotNo" in the API function.

Timer: Check the Timer box to display the current contact status.

(3) Select DA Channel and DA mode:



EcDA Setting

Value :

Out Value :

DA Channel : CH 0:

DA Mode : 0 ~ 5V

Return Code:

Figure 3.10.2.3

DA Channel: Select the No. of DA Channel (0 ~ 3). The parameter "SlotNo" in the API function.

DA Mode: Select DA range. The parameter "Mode" in the API function.

Value: Displays the command value of the analog output set by the scrollbar. The parameter "Value" in the API function.

OutValue: Displays the actual value of analog output. The parameter "Value" in the API function.

Apply: Press the **Apply** key and the set voltage in Value will be converted to the actual output voltage.

Over Range: When this box is checked, the output voltage value will increase by 10%.

Return Code: Displays the DA status.

(4) When selecting **DA Channel** and **DA Mode** (as shown in figure 3.10.2.3), the following program is executed:

```
RetCode = _ECAT_Slave_AIO_Set_Output_RangeMode(gESCCardNo, gNodeID,
gSlotID, Mode);
/* Set DA output range */
```

When the **Apply** key (as shown in 3.10.2.3) is pressed, the following program will be executed:

```
RetCode = _ECAT_Slave_AIO_Set_Output_Value(gESCCardNo, gNodeID, gSlotID,
Value);
/* Set DA output value */
```

If the **Get** button in the Return Code section (as shown in 3.10.2.3) is pressed, the following program will be executed:

```
RetCode = _ECAT_Slave_R1_EC9144_Get_Output_ReturnCode(gESCCardNo,
gNodeID, gSlotID, &RtCode);
/* Get DA status. */
```

To acquire the value of analog output module and display in **Out Value** field (as shown in 3.10.2.3), execute the program below.

```
RetCode = _ECAT_Slave_AIO_Set_Output_Value(gESCCardNo, gNodeID, gSlotID,
Value);
/* Acquire the value of the analog output module. */
```

(5) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

3.11 EtherCAT motion card – high-speed pulse compare function

3.11.1 Function list

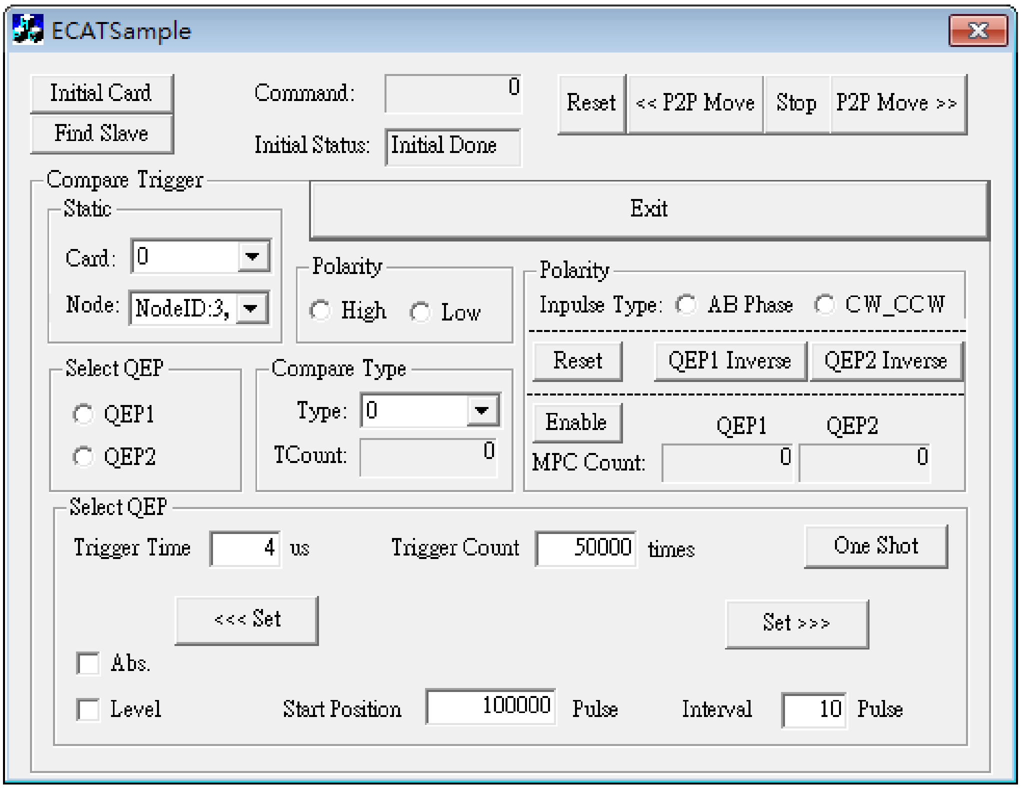
Function name
_ECAT_Compare_Set_Channel_Position
_ECAT_Compare_Get_Channel_Position
_ECAT_Compare_Set_Ipulser_Mode
_ECAT_Compare_Set_Channel_Direction
_ECAT_Compare_Set_Channel_Trigger_Time
_ECAT_Compare_Set_Channel_One_Shot
_ECAT_Compare_Set_Channel_Source
_ECAT_Compare_Set_Channel_Enable
_ECAT_Compare_Channel0_Position
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO
_ECAT_Compare_Set_Channel1_Output_Enable
_ECAT_Compare_Set_Channel1_Output_Mode
_ECAT_Compare_Get_Channel1_IO_Status
_ECAT_Compare_Set_Channel1_GPIO_Out
_ECAT_Compare_Set_Channel1_Position_Table
_ECAT_Compare_Get_Channel1_Positioin_Table_Level
_ECAT_Compare_Get_Channel1_Position_Table_Count
_ECAT_Compare_Set_Channel_Polarity
_ECAT_Compare_Reuse_Channel1_Postion_Table
_ECAT_Compare_Reuse_Channel1_Position_Table_Level

- **Properties**

Hardware	EtherCAT RTX (PAC)	EtherCAT motion card
Supported	N	Y

3.11.2 Application examples

Program interface



Command: 0

Initial Status: Initial Done

Card: 0

Node: NodeID:3,

Type: 0

TCount: 0

MPC Count: 0 0

Trigger Time 4 us

Trigger Count 50000 times

Start Position 100000 Pulse

Interval 10 Pulse

Figure 3.11.2.1

(1) Activate and initialize interface card

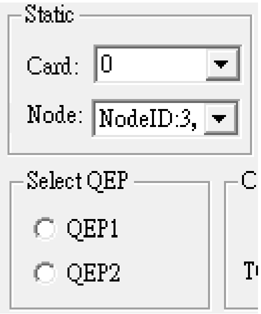
*Make sure the PCI-L221-B1 interface card has been installed and it has to work with Delta’s pulse module for pulse comparing in this example.

Press the **Initial Card** key (as shown in figure 3.11.2.1) to start initializing the interface card.

Press the **Find Slave** key (as shown in figure 3.11.2.1) to start searching the connecting modules.

For more information about the interface card initialization, please see “Activate interface card” and “Initialize interface card” in section 3.1.2.

(2) Select the Card No., Node ID, and QEP



Card: 0

Node: NodeID:3,

Figure 3.11.2.2

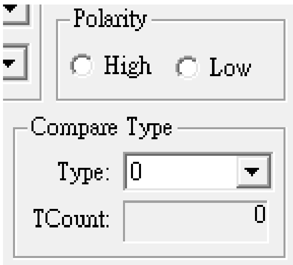
Card: Select the EtherCAT PCI motion card No. to be used.

Node: Select the Node ID; in the example, this axis will generate pulse for comparison.

QEP1: Select channel 1 for pulse input. (It should correspond to the physical wiring of QA1 and QB1.)

QEP2: Select channel 2 for pulse input. (It should correspond to the physical wiring of QA2 and QB2.)

(3) Select Polarity and Compare Type



Type: 0

TCount: 0

Figure 3.11.2.3

High: Select this option to carry out the following program: 1: High (high-level trigger)

```
rt = _ECAT_Compare_Set_Channel_Polarity (gu16_CardNo, 1);
```

Low: Select this option to carry out the following program: 0: Low (low-level trigger)

```
rt = _ECAT_Compare_Set_Channel_Polarity (gu16_CardNo, 0);
```

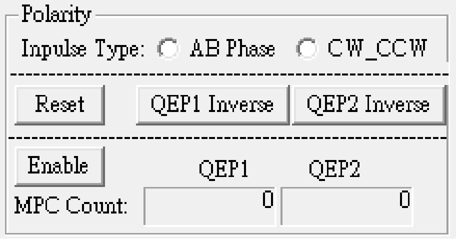
Type: Select the differential signal channel for the triggered signal.

```
rt = _ECAT_Compare_Set_Channel_Source (gu16_CardNo, u16_Channel, gu16_Qep);
// u16_Channel:
// 0 = Channel 0 (CMP_1) for outputting the differential signal. (Compare the pulse at a fixed pulse interval.)
// 1 = Channel 1 (CMP_2) for outputting the differential signal. (Compare the pulse at user-defined pulse intervals.)
// gu16_Qep
// 0 = When setting QEP1, this parameter is 0, which means the compared pulse is from the first QA and QB.
// 1 = When setting QEP2, this parameter is 1, which means the compared pulse is from the second QA and QB.
```

TCount: Number of the compared times.

```
rt = _ECAT_Compare_Get_Channel1_Position_Table_Count (gu16_CardNo,
&u32_Count);
/* Obtain the number of compared times. */
```

(4) MPC parameter settings



MPC Count: 0 0

Figure 3.11.2.4

Enable: Enable/disable compare function. The following program is executed:

```
rt = _ECAT_Compare_Set_Channel_Enable (gu16_CardNo, u16_Channel, 1);
// 1: Enable
// 0: Disable
```

Inpulse Type: Select between AB Phase or CW_CCW mode. The following program is executed:

```
rt = _ECAT_Compare_Set_Ipulser_Mode (gu16_CardNo, mode);
// 0: AB Phase
// 1: CW_CCW
```

Reset: Press this key to clear the pulse count of QEP1 and QEP2, and the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel_Position (gu16_CardNo, u16_Channel, 0);
// 0 represents the pulse number. It means to set the accumulated pulse position to 0.
```

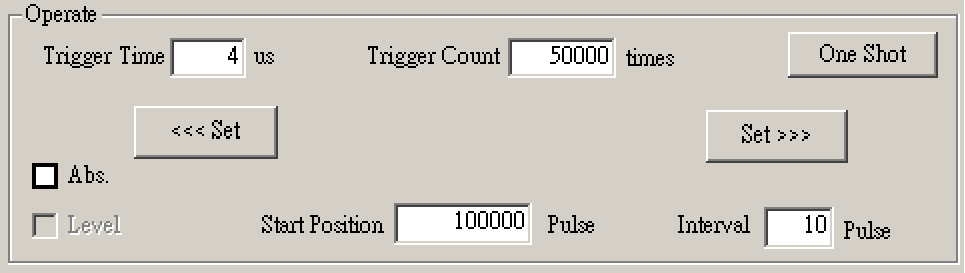
QEP1 Inverse: Pulse incremental way of QAQB 1. Press this key and the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel_Direction (gu16_CardNo, 0, dir);
// 0: Signal source QA1, QB1
// dir:
// 0: Incremented pulse count
// 1: Decremented pulse count
```

QEP2 Inverse: Pulse incremental way of QAQB 2. Press this key and the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel_Direction (gu16_CardNo, 1, dir);
// 1: Signal source QA 2, QB 2
// dir:
// 0 = Incremented pulse count
// 1 = Decremented pulse count
```

(5) Settings for pulse comparison



Operate

Trigger Time 4 us

Trigger Count 50000 times

One Shot

<<< Set

Set >>>

□ Abs.

□ Level

Start Position 100000 Pulse

Interval 10 Pulse

Figure 3.11.2.5

Trigger Time: Set the lasting time for triggering the signal.

Trigger count: Set the number of times for signal to be triggered.

Start Position: Set the starting position for signal to be compared.

Interval: Set the interval for signal to be compared. For example, if the value is set to 10, signal will be triggered every 10 pulses. And the lasting time is determined by **Trigger Time**.

ABS.: Check this box to compare the pulse based on the absolute coordinates.

Level: Check this box and the following program will be executed according to channel 1 (CMP_2), which is used to output differential signal:

```
rt = _ECAT_Compare_Set_Channel1_Output_Mode (gu16_CardNo, gbLevelValue);
// gbLevelValue: Channel 2 output mode of triggering signal.
// 0: Normal type (Users can define the pulse position).
// 1: User-defined type (Users can define the pulse position and determine the triggering
signal is low or high).
```

One Shot: Press this key, trigger the signal once and the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel_Trigger_Time (gu16_CardNo, u16_Channel,
time_us);
// time_us = The lasting time of this triggered signal; unit: us
rt = _ECAT_Compare_Set_Channel_One_Shot (gu16_CardNo, u16_Channel);
// Actual triggered signal outputted
```

Set: Select from **>>>** and **<<<** (direction) and the following program will be executed:

```
rt = _ECAT_Compare_Get_Channel_Position (gu16_CardNo, gu16_Qep, &i32_Pos )
// gu16_Qep: Pulse source for comparision
// &i32_Pos: Returns current incremented pulse (current position)
```

Select **Type** 0 for applying channel 0 (CMP_1) as the differential signal output channel to output the triggered signal. (Compare the pulse at a fixed pulse interval)

```
rt = _ECAT_Compare_Channel0_Position (gu16_CardNo, i32_StartPoint, 0,
u32_CompareCount);
// i32_StartPoint: The starting position for pulse comparison
// u32_CompareCount: Pulse count for comparison
```

Select **Type** 1 for applying channel 1 (CMP_2) as the differential signal output channel to output the triggered signal. (Compare the pulse at user-defined pulse intervals. In this example, the result of simulated pulse comparison at a fixed pulse interval is similar to Type 0. Users can define different intervals according to the demand.)

Disable the differential signal output function of channel 2 first:

```
rt = _ECAT_Compare_Set_Channel1_Output_Enable (gu16_CardNo, 0);
// 0: Disable
// 1: Enable
```

If you do not check the **Level** box, the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel1_Position_Table (gu16_CardNo,
&pi32_PointTable[0], u32_CompareCount);
// &pi32_PointTable: Data array. It is used for storing the compared pulse.
// u32_CompareCount: Pulse count comparison, which should be identical to the size of
data array.
```

Pulse comparison:

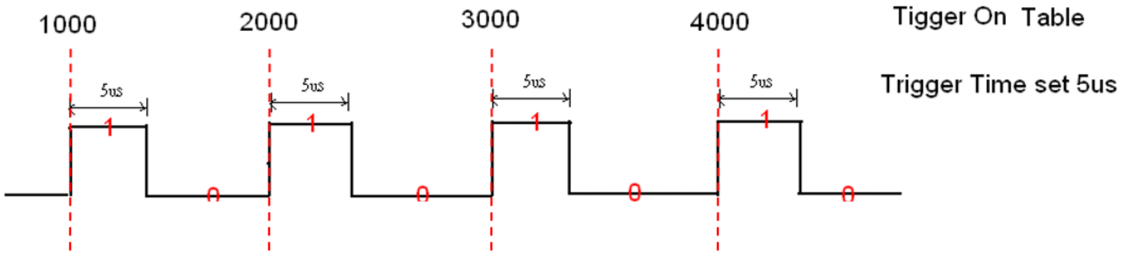


Figure 3.11.2.6 Compare the pulse at a fixed pulse interval

If you check the **Level** box, the following program will be executed:

```
rt = _ECAT_Compare_Set_Channel1_Position_Table_Level (gu16_CardNo,
&pi32_PointTable[0], &pu32_LevelTable[0], u32_CompareCount);
// &pi32_PointTable: Data array. It is used for storing the compared pulse.
// &pu32_LevelTable: Data array. It is used for storing the triggered level.
// u32_CompareCount: Cout of pulse comparison, which should be identical to the size of
data array.
```

Pulse comparison:

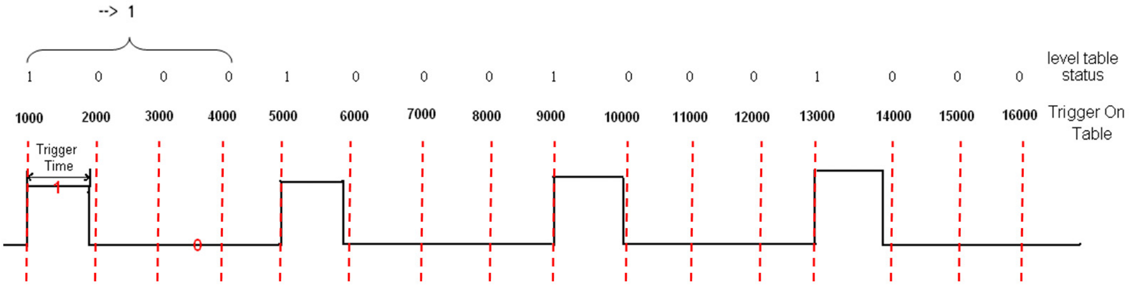


Figure 3.11.2.7 Compare the pulse at user-defined pulse intervals

Enable the differential signal output function of channel 2:

```
rt = _ECAT_Compare_Set_Channel1_Output_Enable (gu16_CardNo, 1);
// 0: Disable
// 1: Enable
```

(6) Command display and testing operations:

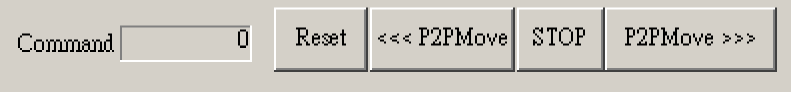


Figure 3.11.2.8

Reset: Press this key to reset the command.

P2PMove: Press this key to move forward or backward. And the generated pulse will be compared by the motion card.

STOP: Press this key to stop the motion.

Command: Display the motion's current position.

(7) Exit program

Press the **Exit** key to exit and close the program.

Execute "_ECAT_Master_Reset" and "_ECAT_Master_Close" to exit the function. Detailed description about these two API is presented in section 3.1.2 Exit program.

(This page is intentionally left blank.)

API List of Dynamic-Link Library

4

This chapter lists all the APIs, data type and setting range of Delta EtherCAT Dynamic-link library.

4.1 Data type and value range

The “TYPE_DEF.H” file located in the “inc\VC\” folder (installation directory) defines the general data type. See the following table. The data type, name, and rage are defined as follows.

Data Type	Description	Range
U8	8-bit ASCII character	0 ~ 255
I16	16-bit signed integer	-32768 ~ 32767
U16	16-bit unsigned integer	0 ~ 65535
I32	32-bit signed long integer	-2147483648 ~ 2147483647
U32	32-bit unsigned long integer	0 ~ 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 ~ 3.402823E38
F64	64-bit double-precision floating point	-1.797683134862315E308 ~ 1.797683134862315E309
Boolean	Boolean	TRUE, FALSE

4.2 API list and descriptions

EtherCAT Master Configuration	
_ECAT_Master_Set_CycleTime	Set the cycle time of the EtherCAT master communication. *Set before initialization.
_ECAT_Master_Get_CycleTime	Acquire the cycle time of the EtherCAT master communication.
_ECAT_Master_NodeID_Alias_Enable	Determine whether to enable user-defined station. *Set before initialization.
_ECAT_Master_Get_SerialNo	Get the serial No. of PAC or the motion card.
_ECAT_Master_Get_DLL_SeqID	Acquire the sequence ID of the current DLL.
_ECAT_Autoconfig_Open_File	Read and apply the configuration file of the communication topology and DC data for system initialization. *Set before initialization.
_ECAT_Autoconfig_Save_File	Save the current communication topology and DC data to the configuration file.
_ECAT_Autoconfig_Set_Slave_DCTime	Set the DC time of each node.
_EACT_Autoconfig_Clear_ConfigFile	Clear the currently imported EtherCAT master configuration.
_ECAT_Autoconfig_Set_NodeID_Alias	Set user-defined station alias of each node. *Set after initialization.
_ECAT_Autoconfig_Get_NodeID_Alias	Acquire the user-defined station alias of each node. *Set after initialization.
_ECAT_Autoconfig_Save_NodeID_Alias	Save the user-defined station alias to the module memory block.

EtherCAT Master Initialization	
_ECAT_Master_Open	Check the number of motion cards and EtherCAT kernels, as well as creating memory block.
_ECAT_Master_Initial	Initialize EtherCAT communication and switch the slave to OP mode
_ECAT_Master_Reset	Reset the EtherCAT master's status and switch the slave to initial mode.
_ECAT_Master_Close	Disable all functions of EtherCAT master and kernels and release the memory
_ECAT_Master_Get_CardSeq	Acquire motion card No.
_ECAT_Master_Get_SlaveNum	Acquire slave quantity on the communication bus of the specified EtherCAT master
_ECAT_Master_Get_Slave_Info	Acquire EtherCAT slave information
_ECAT_Master_Get_DC_Status	Acquire the motion card's DC status, time and time offset
_ECAT_Master_Get_Connect_Status	Acquire EtherCAT master's connection status
_ECAT_Master_Get_Api_BufferLength	Acquire the command amount of each slave that has not been completed
_ECAT_Master_Get_Cycle_SpendTime	Acquire the time spent on Tx and Rx every cycle and the maximum consuming time in the log
_ECAT_Master_Check_Initial_Done	Check whether the DLL initialization has been completed.
_ECAT_Master_Get_Initial_ErrorCode	Acquire the error code when error occurs
_ECAT_Master_Check_Working_Counter	Acquire the current connection status of EtherCAT communication
_ECAT_Master_Get_Return_Code_Message	Acquire the corresponding message of each return code

EtherCAT CoE Standard Communication	
_ECAT_Slave_SDO_Send_Message	Issue SDO command (CANopen) to the slave
_ECAT_Slave_SDO_Read_Message	Acquire the current SDO data (CANopen) of the slave
_ECAT_Slave_SDO_Quick_Send_Message	Issue SDO command (CANopen) to the slave without waiting for the response
_ECAT_Slave_SDO_Quick_Read_Message	Issue SDO read command (CANopen) to the slave without waiting for the response
_ECAT_Slave_SDO_Read_Response	Read the returned data from the slave.
_ECAT_Slave_SDO_Wait_All_Done	Wait multiple slaves to complete all the SDO commands.
_ECAT_Slave_SDO_Get_ErrorCode	Acquire the error code of ERR_ECAT_SDO_Return that returned during the execution of SDO Send_Message or Read_Message. Please refer to CANopen protocol or the definition of each device for error code.
_ECAT_Slave_SDO_Check_Done	Check if the specified slave has completed all the SDO commands
_ECAT_Slave_PDO_Get_OD_Data	Read the data of an OD index in the PDO mapping
_ECAT_Slave_PDO_Set_OD_Data	Send the data of an OD index in the PDO mapping

EtherCAT CoE Standard Communication	
_ECAT_Slave_PDO_Get_Information	Acquire the basic information of each slave device PDO.
_ECAT_Slave_PDO_Get_Detail_Mapping	Acquire the details of PDO mapping in the slave device
_ECAT_Slave_PDO_Get_Rx_Data	Acquire all slave Rx data of the PDO mapping
_ECAT_Slave_PDO_Get_Tx_Data	Acquire all slave Tx data of the PDO mapping
_ECAT_Slave_PDO_Set_Tx_Detail_Data	Configure all slave Tx data of the PDO mapping

General Operation of Motion Axis	
_ECAT_Slave_Motion_Set_Svon	Set the servo to On/Off state.
_ECAT_Slave_Motion_Ralm	Reset the alarm of the axis. Before applying this command, please clear the alarm first. Otherwise, the alarm might occur again.
_ECAT_Slave_Motion_Sd_Stop	Set the deceleration time for motor to decelerate to stop
_ECAT_Slave_Motion_Emg_Stop	This is for emergency stop of the axis. The motor will stop with its maximum deceleration
_ECAT_Slave_Motion_Set_Alm_Reaction	Set the action when alarm occurs
_ECAT_Slave_Motion_Set_Position	Specify current feedback position of the axis
_ECAT_Slave_Motion_Set_Command	Set the motion command data of the axis
_ECAT_Slave_Motion_Set_MoveMode	Set the motion mode of the axis
_ECAT_Slave_Motion_Get_MoveMode	Acquire the information of current motion mode
_ECAT_Slave_Motion_Get_ControlWord	Acquire the current control word of the axis
_ECAT_Slave_Motion_Get_StatusWord	Acquire the current status word of the axis.
_ECAT_Slave_Motion_Get_Mdone	Acquire the current status of motion done
_ECAT_Slave_Motion_Get_Position	Acquire the current position of the axis.
_ECAT_Slave_Motion_Get_Command	Acquire the current command information
_ECAT_Slave_Motion_Get_Target_Command	Acquire the target command data of the axis
_ECAT_Slave_Motion_Get_Actual_Position	Acquire the actual position command of the axis
_ECAT_Slave_Motion_Get_Actual_Command	Acquire the current command data. The data will vary with to the applied motion mode.
_ECAT_Slave_Motion_Get_Current_Speed	Acquire the current speed of the axis
_ECAT_Slave_Motion_Get_Torque	Acquire the feedback torque from the motor
_ECAT_Slave_Motion_Get_Buffer_Length	Acquiring the quantity of the commands that have not been carried out
_ECAT_Slave_Motion_Set_TouchProbe_Config	Set the mode of the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Set_TouchProbe_QuickStart	Enable the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Set_TouchProbe_QuickDone	Execute the first Touch Probe function (Touch Probe 1) again
_ECAT_Slave_Motion_Set_TouchProbe_Disable	Disable the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Get_TouchProbe_Status	Acquire the current status of the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Get_TouchProbe_Position	Acquire the current position of first Touch Probe function (Touch Probe 1)

Cyclic Synchronous Position Mode (CSP)	
_ECAT_Slave_CSP_Start_Move	Execute linear interpolation of single axis
_ECAT_Slave_CSP_Start_V_Move	Execute the single-axis motion with constant speed
_ECAT_Slave_CSP_Start_Arc_Move	Execute two-axis arc motion, moving from current position and the specified circle center to form the specified arc’s angle
_ECAT_Slave_CSP_Start_Arc2_Move	Execute two-axis arc motion, moving from current position and the specified circle center to form the specified arc’s angle
_ECAT_Slave_CSP_Start_Arc3_Move	Execute two-axis arc motion, moving from the current position and specified circle center to the specified end point
_ECAT_Slave_CSP_Start_Spiral_Move	Execute two-axis spiral motion, moving from current position and the specified circle center to form the specified angle
_ECAT_Slave_CSP_Start_Spiral2_Move	Execute two-axis spiral motion, moving from current position and the specified circle center to the end point with the specified cycle number.
_ECAT_Slave_CSP_Start_Sphere_Move	Execute three-axis sphere motion and moving from current position and the known circle center to the target position with three-dimensional vector
_ECAT_Slave_CSP_Start_Heli_Move	Set three-axis helical motion, moving from current position and the known circle center to the specified height in Z-axis direction
_ECAT_Slave_CSP_Start_Multiaxes_Move	Execute multi-axis linear motion
_ECAT_Slave_CSP_Start_Msbrline_Move	Execute multi-axis point to point motion with smooth speed
_ECAT_Slave_CSP_Set_Gear	Set the E-gear ratio
_ECAT_Slave_CSP_Set_Softlimit	Set the software limit
_ECAT_Slave_CSP_TargetPos_Change	Set a new target position
_ECAT_Slave_CSP_Velocity_Change	Set a new target speed
_ECAT_Slave_CSP_Feedrate_Overwrite	For the advanced setting of speed change for single axis
_ECAT_Slave_CSP_Speed_Continue_Enable	Enable or disable the continuous speed function
_ECAT_Slave_CSP_Speed_Continue_Set_Mode	Set the continuous speed mode
_ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio	Set the percentage of for starting blending speed of two commands.
_ECAT_Slave_CSP_Scurve_Rate	Set the ratio of S-curve and T-curve during acceleration and deceleration
_ECAT_Slave_CSP_Liner_Speed_Master	Set the speed (vector) of advanced interpolation function
_ECAT_Slave_CSP_Mask_Axis	When multi-axis command is being executed, this API can be used to stop the specified axes without influencing others
_ECAT_Slave_CSP_Sync_Config	Set the function of synchronous motion of multiple axes
_ECAT_Slave_CSP_Sync_Move	Enable the function of synchronous motion of multiple axes
_ECAT_Slave_CSP_Start_Mabrline_Move	Set to smooth the operation of point-to-point motion of multiple axes

Cyclic Synchronous Position Mode (CSP)	
_ECAT_Slave_CSP_Start_2Segment_Move	Set the single-axis linear motion by specifying two distances and speed
_ECAT_Slave_CSP_Start_PVT_Move	Set the single-axis motion to move to multiple points at fixed time
_ECAT_Slave_CSP_Start_PVTComplete_Move	Specify the initial speed and end speed of the single-axis motion, moving through multiple points at fixed time.
_ECAT_Slave_CSP_Virtual_Set_Enable	Enable function of virtual position
_ECAT_Slave_CSP_Virtual_Set_Command	Set the virtual position and replacing the current position with the specified position
_ECAT_Slave_CSP_Get_SoftLimit_Status	Acquire the status of software limit
_ECAT_Slave_CSP_Pitch_Set_Interval	Set the interval of the pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Mode	Set the mode of pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Org	Set the start position of pitch error compensation.
_ECAT_Slave_CSP_Pitch_Set_Rel_Table	Set the relative position of each interval for pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Abs_Table	Set the absolute position of each interval for pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Enable	Enable function of pitch error compensation.

Cyclic Synchronous Velocity Mode (CSV)	
_ECAT_Slave_CSV_Start_Move	Execute single-axis motion with the setting speed
_ECAT_Slave_CSV_Multi_Start_Move	Execute multi-axes synchronous motion with the setting speed

Cyclic Synchronous Torque Mode (CST)	
_ECAT_Slave_CST_Start_Move	Execute single-axis motion with the setting torque
_ECAT_Slave_CST_Multi_Start_Move	Execute multi-axis synchronous motion with the setting torque

Homing	
_ECAT_Slave_Home_Config	Set the homing mode
_ECAT_Slave_Home_Move	Execute homing
_ECAT_Slave_Home_Status	Acquire the current homing status

Profile Position Mode (PP)	
_ECAT_Slave_PP_Start_Move	Execute single-axis linear motion in PP mode
_ECAT_Slave_PP_Advance_Config	Advanced setting of PP mode

Profile Velocity Mode (PV)	
_ECAT_Slave_PV_Start_Move	Execute the single-axis motion with constant speed in PV mode
_ECAT_Slave_PV_Advance_Config	Advanced setting of PV mode

Invertor Motion Control	
_ECAT_Slave_VL_Start_Move	Inverter single-axis motion control with constant speed. (Delta inverter only)

Profile Torque Mode (PT)	
_ECAT_Slave_PT_Start_Move	Execute the single-axis motion with constant torque in PT mode
_ECAT_Slave_PT_Advance_Config	Advanced setting of PT mode

Group Motion Control	
_ECAT_Slave_User_Motion_Control_Set_Enable_ Mode	Set the group status. *Please note that before enabling the group, users should apply Set_Motion_Control_Type to specify the axis for one group and use _ECAT_Slave_User_Motion_Control_ Svon and _ECAT_Slave_User_Motion_Control_Get_Alm to confirm the status of each axis.
_ECAT_Slave_User_Motion_Control_Get_Enable_ Mode	Acquire the status in the current group.
_ECAT_Slave_User_Motion_Control_Set_Type	Set the motion mode in the specified group.
_ECAT_Slave_User_Motion_Control_Set_Data	Set the data of each axis in the specified group.
_ECAT_Slave_User_Motion_Control_Clear_Data	Clear the data of each axis in the specified group.
_ECAT_Slave_User_Motion_Control_Get_DataCnt	Read the data number that have not been processed in the specified group.
_ECAT_Slave_User_Motion_Control_ Ralm	Reset the alarm of all axes in the specified group.
_ECAT_Slave_User_Motion_Control_ Svon	Enable/disable all axes in the specified group.
_ECAT_Slave_User_Motion_Control_Get_Alm	Acquire the current alarm status in the specified group.

Operation of DI/DO module	
_ECAT_Slave_DIO_Get_Input_Value	Acquire the DI status
_ECAT_Slave_DIO_Get_Output_Value	Acquire the DO status
_ECAT_Slave_DIO_Set_Output_Value	Set the DO status
_ECAT_Slave_DIO_Get_Single_Input_Value	Acquire the input value of the specified channel
_ECAT_Slave_DIO_Get_Single_Output_Value	Acquire the output value of the specified channel
_ECAT_Slave_DIO_Set_Single_Output_Value	Set the output value of the specified channel

Operation of DI/DO module	
_ECAT_Slave_DIO_Set_Output_Error_Mode	Enable/Disable the retentive function of each channel on remote DO module when EtherCAT communication is disconnected
_ECAT_Slave_DIO_Set_Output_Error_Value	Set the output status of each channel on remote DO module when EtherCAT communication is disconnected and the retentive function is enabled

Operation of AI/AO Module	
_ECAT_Slave_AIO_Get_Input_Value	Acquire analog input value
_ECAT_Slave_AIO_Set_Output_Value	Set analog output value
_ECAT_Slave_AIO_Get_Output_Value	Acquire analog output value

Operation of Pulse Module (R1-EC5621D0 series)	
_ECAT_Slave_R1_EC5621_Set_Output_Mode	Set the mode of pulse output.
_ECAT_Slave_R1_EC5621_Set_Input_Mode	Set the mode of pulse input.
_ECAT_Slave_R1_EC5621_Set_ORG_Inverse	Set the contact type (NC/NO) of the origin switch (ORG).
_ECAT_Slave_R1_EC5621_Set_QZ_Inverse	Set the contact type (NC/NO) of encoder’s Z pulse (QZ).
_ECAT_Slave_R1_EC5621_Set_Home_SpMode	Apply the special mode when homing.
_ECAT_Slave_R1_EC5621_Set_MEL_Inverse	Set the contact type (NC/NO) of the negative limit switch (MEL).
_ECAT_Slave_R1_EC5621_Set_PEL_Inverse	Set the contact type (NC/NO) of the positive limit switch (PEL).
_ECAT_Slave_R1_EC5621_Set_Svon_Inverse	Set the contact type (NC/NO) of the servo enable switch (Svon).
_ECAT_Slave_R1_EC5621_Set_Home_Slow_Down	It sets the deceleration time after the motor reaches the origin
_ECAT_Slave_R1_EC5621_Get_IO_Status	Acquire the status of all I/O points
_ECAT_Slave_R1_EC5621_Get_Single_IO_Status	Acquire the status of single I/O point.

Operation of Pulse Module (R1-ECx62xD0 series)	
_ECAT_Slave_R1_ECx62x_Set_Output_Mode	Set the type of pulse output
_ECAT_Slave_R1_ECx62x_Set_Input_Mode	Set the type of pulse input
_ECAT_Slave_R1_ECx62x_Set_ORG_Inverse	Set the contact type (NC/NO) of the origin switch (ORG)
_ECAT_Slave_R1_ECx62x_Set_QZ_Inverse	Set the contact type (NC/NO) of encoder’s Z pulse signal (QZ)
_ECAT_Slave_R1_ECx62x_Set_Home_SpMode	Apply the special mode when homing
_ECAT_Slave_R1_ECx62x_Set_MEL_Inverse	Set the contact type (NC/NO) of the negative limit switch (MEL)
_ECAT_Slave_R1_ECx62x_Set_PEL_Inverse	Set the contact type (NC/NO) of the positive limit switch (PEL)

Operation of Pulse Module (R1-ECx62xD0 series)	
_ECAT_Slave_R1_ECx62x_Set_Svon_Inverse	Set the contact type (NC/NO) of the servo enable switch (Svon)
_ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down	It sets the deceleration time after the motor reaches the Home switch
_ECAT_Slave_R1_ECx62x_Get_IO_Status	Acquire the status of all I/O points
_ECAT_Slave_R1_ECx62x_Get_Single_IO_Status	Acquire the status of single I/O point

Operation of delta servo drive	
_ECAT_Slave_DeltaServo_Write_Parameter	Write servo parameter values to Delta servo drives
_ECAT_Slave_DeltaServo_Read_Parameter	Read servo parameter values from Delta servo drives
_ECAT_Slave_DeltaServo_Read_Parameter_Info	Read servo parameter attributes from Delta servo drives
_ECAT_Slave_DeltaServo_Set_Velocity_Limit	Set Delta servo motor's max. speed
_ECAT_Slave_DeltaServo_Set_Compare_Enable	Write the pulse compare parameter, which is identical to Delta servo parameter P5-59
_ECAT_Slave_DeltaServo_Get_Compare_Enable	Read the pulse compare parameter that is written to the servo drive, which is identical to Delta servo parameter P5-59
_ECAT_Slave_DeltaServo_Set_Compare_Config	Write the data array number and values of the pulse compare function to Delta servo drives

Analog Input Settings (R1-EC8124D0 series)	
_ECAT_Slave_R1_EC8124_Set_Input_RangeMode	Set the sampling range of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode	Set the sampling rate of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_Enable	Enable/Disable the analog input sampling function of Delta analog input module
_ECAT_Slave_R1_EC8124_Get_Input_RangeMode	Acquire the sampling range of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_AverageMode	Set the average times for the analog input filter of Delta analog input module.

Analog Output Settings (R1-EC9144D0 series)	
_ECAT_Slave_R1_EC9144_Set_Output_RangeMode	Set the output range of Delta analog output module
_ECAT_Slave_R1_EC9144_Set_Output _Enable	Enable/Disable the analog output of Delta module
_ECAT_Slave_R1_EC9144_Get_Output_ReturnCode	Acquire the operation status of Delta analog output module

Auto Recording Function for Motion Axis	
_ECAT_Slave_Record_Data_Set_Type	Set the recording data type of specified axis.
_ECAT_Slave_Record_Data_Set_Enable	Enable/Disable the recording function of specified axis
_ECAT_Slave_Record_Data_Get_Cnt	Acquire the data entry number of specified axis
_ECAT_Slave_Record_Data_ReadData	Acquire the recorded data of specified axis
_ECAT_Slave_Record_Clear_Data	Delete the saved record of specified axis
_ECAT_Slave_Record_Multi_Set_Enable	Enable/Disable the recoding function of specified multiple axes
_ECAT_Slave_Record_Multi_Clear_Data	Delete the saved record of specified multiple axes

Operation of Local Digital I/O	
_ECAT_GPIO_Set_Output	Control the output status of the GPIO on the motion card
_ECAT_GPIO_Get_Output	Read the output status of the GPIO on the motion card
_ECAT_GPIO_Get_Input	Read the input status of the GPIO on the motion card

High-Speed Pulse Compare Function	
_ECAT_Compare_Set_Channel_Position	Overwrite a position value for the specified channel
_ECAT_Compare_Get_Channel_Position	Acquire the current position value of the specified channel
_ECAT_Compare_Set_Ipulser_Mode	Set the mode of pulse input for the specified channel
_ECAT_Compare_Set_Channel_Direction	Set the pulse direction of the specified channel
_ECAT_Compare_Set_Channel_Trigger_Time	Set the trigger retaining time for the specified channel
_ECAT_Compare_Set_Channel_One_Shot	Force the trigger manually once for the specified channel
_ECAT_Compare_Set_Channel_Source	Set the compare source for the specified channel.
_ECAT_Compare_Set_Channel_Enable	Enable/disable the compare function for the specified channel
_ECAT_Compare_Channel0_Position	Set the parameters for triggering the signal at a fixed pulse interval of channel 0
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO	Set the parameters for triggering the signal at a fixed pulse interval of channel 0, which is enabled / disabled by GPIO
_ECAT_Compare_Set_Channel1_Output_Enable	Enable/Disable the trigger function of channel 1 (user-defined pulse intervals)
_ECAT_Compare_Set_Channel1_Output_Mode	Set the output mode of channel 1
_ECAT_Compare_Get_Channel1_IO_Status	Acquire the operation status of channel 1
_ECAT_Compare_Set_Channel1_GPIO_Out	Set the output status of the PIN15 on CN2 of GPIO

High-Speed Pulse Compare Function	
_ECAT_Compare_Set_Channel1_Position_Table	Set the pulse data of channel 1 (user-defined pulse intervals)
_ECAT_Compare_Set_Channel1_Position_Table_Level	Set the pulse data of channel 1 and its user-defined active level for triggering signals
_ECAT_Compare_Get_Channel1_Position_Table_C	Acquire the current trigger counts of channel 1
_ECAT_Compare_Set_Channel_Polarity	Set the trigger level of the compare function
_ECAT_Compare_Reuse_Channel1_Position_Table	Re-execute the compare function of channel 1 once
_ECAT_Compare_Reuse_Channel1_Position_Table_Level	Re-execute the compare function of channel 1 once, which the trigger level is user-defined

Dynamic Link Library Information	
_ECAT_Master_Get_DLL_Path	Acquire the directory of the EtherCat_DLL.dll file
_ECAT_Master_Get_DLL_Version	Acquire the version information of the EtherCat_DLL.dll file
_ECAT_Master_Get_DLL_Path_Single	Acquire the directory of the ECAT_RTX_DLL.dll or PCI_L221.dll file
_ECAT_Master_Get_DLL_Version_Single	Acquire the version information of the ECAT_RTX_DLL.dll or PCI_L221.dll file

Software Protection	
_ECAT_Security_Check_Verifykey	Check the verification key
_ECAT_Security_Get_Check_Verifykey_State	Check the verification status of the verification key
_ECAT_Security_Write_Verifykey	Write the verification key into the verification IC
_ECAT_Security_Get_Write_Verifykey_State	Obtain the status and result of writing in the verification key
_ECAT_Security_Check_UserPassword	Check the user password
_ECAT_Security_Get_Check_UserPassword_State	Acquire the status of verifying the user password
_ECAT_Security_Write_UserPassword	Write in the user password into the verification IC
_ECAT_Security_Get_Write_UserPassword_State	Acquire the status and result of writing in the user password

Operating MRAM in PAC	
_ECAT_Master_MRAM_Write_Word_Data	Write the U16 data (Word) to the specified address of MRAM in PAC
_ECAT_Master_MRAM_Read_Word_Data	Read the U16 data (Word) from the specified address of MRAM in PAC
_ECAT_Master_MRAM_Write_DWord_Data	Write the U32 data (DWord) into the specified address of MRAM in PAC
_ECAT_Master_MRAM_Read_DWord_Data	Read the U32 data (DWord) from the specified address of MRAM in PAC

Retentive Digital Output Function (R1-EC70E2D0 series)	
_ECAT_Slave_R1_EC70E2_Set_Output_Enable	Enable/Disable the digital output of the module

Retentive Digital Output Function (R1-EC70X2D0 series)	
_ECAT_Slave_R1_EC70X2_Set_Output_Enable	Enable/Disable digital output of the module

MPG operation (R1-EC5614D0 series)	
_ECAT_Slave_R1_EC5614_Set_MJ_Config	Set the parameters of MPG function
_ECAT_Slave_R1_EC5614_Set_MJ_Enable	Enable/Disable the MPG function
_ECAT_Slave_R1_EC5614_Get_IO_Status	Acquire the I/O contact status of the MPG module
_ECAT_Slave_R1_EC5614_Get_MPG_Counter	Acquire the value of the MPG counter

EtherCAT Master Configuration

5

This chapter provides introduction on how to use APIs for EtherCAT master before initialization. APIs mentioned here are for advanced users. If no special requirement is defined, EtherCAT master is set in default.

API list of EtherCAT master configuration

Function name	Description
_ECAT_Master_Set_CycleTime	Set the cycle time of the EtherCAT master communication. *Set before initialization.
_ECAT_Master_Get_CycleTime	Acquire the cycle time of the EtherCAT master communication.
_ECAT_Master_NodeID_Alias_Enable	Determine whether to enable user-defined station. *Set before initialization.
_ECAT_Master_Get_SerialNo	Get the serial No. of PAC or the motion card.
_ECAT_Master_Get_DLL_SeqID	Acquire the sequence ID of the current DLL.
_ECAT_Autoconfig_Open_File	Read and apply the configuration file of the communication topology and DC data for system initialization. *Set before initialization.
_ECAT_Autoconfig_Save_File	Save the current communication topology and DC data to the configuration file.
_ECAT_Autoconfig_Set_Slave_DCTime	Set the DC time of each node.
_EACT_Autoconfig_Clear_ConfigFile	Clear the currently imported EtherCAT master configuration.
_ECAT_Autoconfig_Set_NodeID_Alias	Set user-defined station alias of each node. *Set after initialization.
_ECAT_Autoconfig_Get_NodeID_Alias	Acquire the user-defined station alias of each node. *Set after initialization.
_ECAT_Autoconfig_Save_NodeID_Alias	Save the user-defined station alias to the module memory block.

5.1 _ECAT_Master_Set_CycleTime

■ Syntax

U16 PASCAL _ECAT_Master_Set_CycleTime (U16 CardNo, U16 Mode)

■ Purpose

This is for setting the cycle time of the EtherCAT master communication. *Set before initialization.

Note: This API can only be executed after the EtherCAT Master has been started ("_ECAT_Master_Open" in section 6.1) and before EtherCAT communication is initialized ("_ECAT_Master_Initial" in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Mode	U16	Option	Communication cycle time (us) 0: 2000 us 1: 1000 us 2: 500 us 3: 250 us 4: 125 us

■ Example

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
Status = _ECAT_Master_Get_CardSeq (0, &CardNo);
// Execute the API after enabling the motion card and before initialization.
U16 Mode = 3;
Status = _ECAT_Master_Set_CycleTime (CardNo, Mode);

Status = _ECAT_Master_Initial(CardNo);
}
```

5.2 _ECAT_Master_Get_CycleTime

■ Syntax

U16 PASCAL _ECAT_Master_Get_CycleTime (U16 CardNo, U16 *CycleTime)

■ Purpose

This is for acquiring the cycle time of the EtherCAT master communication.

Note: This function can be used only after the EtherCAT Master is enabled by "_ECAT_Master_Open" (refer to section 6.1).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
CycleTime	U16*	Time (us)	Communication cycle time (unit: us); The Master currently only supports cycle times of 2000, 1000, 500, 250, 125. For the slave cycle time, please refer to the description of each Slave.

■ Example

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;
U16 CycleTime =0;
Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
Status = _ECAT_Master_Get_CardSeq (0, &CardNo);
// Acquire the Master's CycleTime setting information.
Status = _ECAT_Master_Get_CycleTime (CardNo, &CycleTime);
}
```

5.3 _ECAT_Master_NodeID_Alias_Enable

■ Syntax

U16 PASCAL _ECAT_Master_NodeID_Alias_Enable (U16 CardNo, U16 Enable)

■ Purpose

This is for determining whether to enable user-defined station.

*Set before initialization. If there is any EtherCAT slave that has not been assigned with an alias, error message (0x1004) will occur.

Note:

1. Please make sure no repeated slave station alias is on the bus.
2. If you are using any of the Delta R1-EC series modules, please refer to Chapter 5.10 _ECAT_Autoconfig_Set_NodeID_Alias to set the station number of all R1-EC series modules on the EtherCAT bus. Once the setting is complete, refer to section 5.12 _ECAT_Autoconfig_Save_NodeID_Alias to save this setting to the module’s memory. Then, re-initialize EtherCAT bus.
3. If there is a repeated station alias or one of the station aliases is not set, an API error message (0x1004) will occur.
4. This API can only be executed after the EtherCAT Master has been started (“_ECAT_Master_Open” in section 6.1) and before EtherCAT communication is initialized (“_ECAT_Master_Initial” in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Enable	U16	Option	0: Disable user-defined station alias. 1: Enable user-defined station alias.

■ Example

```
U16 Status;
U16 CardNo=0, Enable =1;
U16 MapNodeID;
U16 Cardnum=0;
Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
Status = _ECAT_Master_Get_CardSeq (0, &CardNo);
// Disable EtherCAT communication.
Status = _ECAT_Master_Reset(CardNo);
// Enable user-defined station alias.
Status = _ECAT_Master_NodeID_Alias_Enable (CardNo, Enable);
// Eanble the EtherCAT communication again.
Status = _ECAT_Master_Initial(CardNo);
}
```

5.4 _ECAT_Get_SerialNo

- Syntax

U16 PASCAL _ECAT_Get_SerialNo (U16 CardNo, U32* SerialNo)

- **Purpose**

This is for acquiring the serial No. of the PAC or motion card.

Note: This API can only be executed after the EtherCAT Master has been started by API "_ECAT_Master_Open" (section 6.1).

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
SerialNo	U32	Number	Serial No.

- **Example**

```
U16 Status;
U16 CardNo=0;
U32 SerialNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

Status = _ECAT_Get_SerialNo (CardNo, &SerialNo);
}
```

5.5 _ECAT_Master_Get_DLL_SeqID

■ Syntax

U16 PASCAL _ECAT_Master_Get_DLL_SeqID (U16 CardNo, U16 *SeqID)

■ Purpose

This is for acquiring the sequence ID of the current dynamic link library (DLL).

Note: This API can only be executed after the EtherCAT Master has been started by _ECAT_Master_Open (section 6.1).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
SeqID	U16*	Value	Sequence ID of the current DLL.

■ Example

```
U16 Status;
U16 CardNo = 0;
U16 SeqID = 0;
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

        Status = _ECAT_Master_Get_DLL_SeqID (CardNo, &SeqID);
}
```

5.6 _ECAT_Autoconfig_Open_File

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Open_File (U16 CardNo, I8 *FilePath)

■ Purpose

This is for reading and applying the configuration file of communication topology and DC data.

EtherCAT Master will refer to the saved communication topology and DC data during connection. If the actual communication structure does not match the DC data, EtherCAT Master will return an error code. With this API, you can avoid EtherCAT Master from issuing the wrong command when the communication topology is changed accidently.

Note: This API can only be executed after the EtherCAT Master has been started
(“_ECAT_Master_Open” in section 6.1) and before EtherCAT communication is initialized
(“_ECAT_Master_Initial” in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
FilePath	I8*	String	The directory of the configuration file

■ Example

```
U16 Status;
U16 CardNo=0;
I8 FilePath[255];
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);


        strcpy(FilePath, "C:\\EtherCAT_Information.dat");
       Status = _ECAT_Autoconfig_Open_File (CardNo, FilePath);


       Status = _ECAT_Master_Initial(CardNo);
}
```

5.7 _ECAT_Autoconfig_Save_File

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Save_File (U16 CardNo, I8 *FilePath)

■ Purpose

This is for saving the communication topology and DC data to the configuration file.

Before initializing the master (_ECAT_Master_InitialEtherCAT in section 6.2), you can use "_ECAT_Autoconfig_Open_File" (section 5.6) to import this configuration file so that the EtherCAT Master will able to check if the actual topology complies with the configuration and return an error code.

Note: This API can only be executed after the EtherCAT Master has been started by API "_ECAT_Master_Open" (section 6.1).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
FilePath	I8*	String	Save the communication topology and DC data to the configuration file

■ Example

```
U16 Status;
U16 CardNo=0;
I8 FilePath[255];
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

        strcpy(FilePath, "C:\\EtherCAT_Information.dat");
        Status = _ECAT_Autoconfig_Save_File (CardNo, FilePath);
}
```

5.8 _ECAT_Autoconfig_Set_Slave_DCTime

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Set_Slave_DCTime (U16 CardNo, U16 NodeID, U16 Mode)

■ Purpose

This is for setting the DC time of each node, which default is 1000 us.

Note: This API can only be executed after the EtherCAT Master has been started ("_ECAT_Master_Open" in section 6.1) and before EtherCAT communication is initialized ("_ECAT_Master_Initial" in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
Mode	U16	Option	Set DC time for each node. 0: 2000 us 1: 1000 us 2: 500 us 3: 250 us 4: 125 us

■ Example

```
U16 Status;
U16 CardNo=0,NodeID=1;
U16 Mode=1; // 1ms
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

        Status = _ECAT_Autoconfig_Set_Slave_DCTime (CardNo, NodeID, Mode);


       Status = _ECAT_Master_Initial(CardNo);
}
```

5.9 _EACT_Autoconfig_Clear_ConfigFile

■ Syntax

U16 PASCAL _EACT_Autoconfig_Clear_ConfigFile (U16 CardNo)

■ Purpose

This is for clearing the configuration file loaded via _ECAT_Autoconfig_Open_File. It is mainly used when a wrong file has been opened or the changes are not compatible with the previous settings.

Note: This API can only be executed after the EtherCAT Master has been started
("_ECAT_Master_Open" in section 6.1) and before EtherCAT communication is initialized
("_ECAT_Master_Initial" in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

Status = _EACT_Autoconfig_Clear_ConfigFile (CardNo);

Status = _ECAT_Master_Initial(CardNo);
}
```

5.10 _ECAT_Autoconfig_Set_NodeID_Alias

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Set_NodeID_Alias (U16 CardNo, U16 NodeID, U16 MapNodeID)

■ Purpose

This is for configuring user-defined station alias of each node. *Set after initialization.

Note:

1. This API is only applicable to Delta R1-EC series remote modules.
2. After executing this API, users still need to validate the new station alias. To validate it, please set the station alias of each node first and by using "_ECAT_Autoconfig_Save_NodeID_Alias" (refer to section 5.12). Meanwhile, the station alias information can be acquired by "_ECAT_Autoconfig_Get_NodeID_Alias" (refer to section 5.11)
3. This API can only be executed after the EtherCAT Master has been started ("_ECAT_Master_Open" in section 6.1) and before setting the user-defined station alias of each node ("_ECAT_Autoconfig_Save_NodeID_Alias" in section 5.12) and ("_ECAT_Master_NodeID_Alias_Enable" in section 5.3) also before ("_ECAT_Master_Initial" in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
MapNodeID	U16	Number	Specify Node ID

■ Example

```
U16 Status;
U16 CardNo=0,NodeID=1;
U16 MapNodeID = 2;
U16 Cardnum=0;
Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);
        // Reset EtherCAT communication
        Status = _ECAT_Master_Reset(CardNo);
        // Specified first node of slave device as station 2
        Status = _ECAT_Autoconfig_Set_NodeID_Alias (CardNo, NodeID, MapNodeID);
        // Save user-defined alias to all of the connected device, and then EtherCAT
communication will stop automatically.
        Status = _ECAT_Autoconfig_Save_NodeID_Alias (CardNo);
        // …………waiting for reboot all of the device manually…………
        bool module_reboot = false;
        While(!module_reboot)
        {
        }
        // Enabled the communication mode of user-defined alias
        Status = _ECAT_Master_NodeID_Alias_Enable (CardNo, Enable);
        // initialize EtherCAT communication
        Status = _ECAT_Master_Initial(CardNo);
}
```

5.11 _ECAT_Autoconfig_Get_NodeID_Alias

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Get_NodeID_Alias (U16 CardNo, U16 RealNodeID, U16 *MapNodeID)

■ Purpose

This is for acquiring the user-defined station alias of each node.

*Set after initialization.

Note: This API can only be executed after the EtherCAT Master has been started ("_ECAT_Master_Open" in section 6.1). To use user-defined station alias of each node ("_ECAT_Autoconfig_Set_NodeID_Alias" in section 5.10), you will need to apply API "_ECAT_Autoconfig_Save_NodeID_Alias" (in section 5.12) to save specified alias to each slave device.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
RealNodeID	U16	Number	Node ID
MapNodeID	U16*	Number	Specify Node ID

■ Example

```
U16 Status;
U16 CardNo=16,NodeID=1;
U16 MapNodeID;U16 Cardnum=0;
Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
	Status = _ECAT_Master_Get_CardSeq (0, &CardNo);

	Status = _ECAT_Autoconfig_Get_NodeID_Alias (CardNo, NodeID, &MapNodeID);
}
```

5.12 _ECAT_Autoconfig_Save_NodeID_Alias

■ Syntax

U16 PASCAL _ECAT_Autoconfig_Save_NodeID_Alias (U16 CardNo)

■ Purpose

This is for saving the user-defined station alias to the module memory block. After executing this API, EtherCAT communication will be disconnected automatically and reboot of all slave devices is required.

Note:
1. This API is only applicable to Delta R1-EC series remote modules.
2. Please use API "_ECAT_Autoconfig_Set_NodeID_Alias" (section 5.10), to configure all of the user-defined alias and save them to the devices. After then, users can get the new configuration by API "_ECAT_Autoconfig_Get_NodeID_Alias" (section 5.11).
3. This API can only be executed after the EtherCAT Master has been started ("_ECAT_Master_Open" in section 6.1) and before EtherCAT communication is initialized ("_ECAT_Master_Initial" in section 6.2).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo=0,NodeID=1;
U16 MapNodeID = 2;
U16 Cardnum=0;
Status = _ECAT_Master_Open (&Cardnum);
if (Cardnum>0)
{
        Status = _ECAT_Master_Get_CardSeq (0, &CardNo);
        // Reset EtherCAT communication
        Status = _ECAT_Master_Reset(CardNo);
        // Specified first node of slave device as station 2
        Status = _ECAT_Autoconfig_Set_NodeID_Alias (CardNo, NodeID, MapNodeID);
        // Save user-defined alias to all of the connected device, and then EtherCAT
communication will stop automatically.
        Status = _ECAT_Autoconfig_Save_NodeID_Alias (CardNo);
        // …………waiting for reboot all of the device manually…………
        bool module_reboot = false;
        While(!module_reboot)
        {
        }
```

```
    // Enabled the communication mode of user-defined alias
    Status = _ECAT_Master_NodeID_Alias_Enable (CardNo, Enable);
    // initialize EtherCAT communication
    Status = _ECAT_Master_Initial(CardNo);
}
```

Master Initialization 6

This chapter provides detailed introduction on how to use the API for initializaing the EtherCAT master. Users have to execute the API mentioned here before applying the function of motion control and remote module in other chapters.

API list of master initialization

Function name	Description
_ECAT_Master_Open	Check the number of motion cards and EtherCAT kernels, as well as creating memory block
_ECAT_Master_Initial	Initialize EtherCAT communication and switch the slave to OP mode
_ECAT_Master_Reset	Reset the EtherCAT master’s status and switch the slave to initial mode
_ECAT_Master_Close	Disable all functions of EtherCAT master and kernels and release the memory
_ECAT_Master_Get_CardSeq	Acquire motion card No.
_ECAT_Master_Get_SlaveNum	Acquire slave quantity on the communication bus of the specified EtherCAT master
_ECAT_Master_Get_Slave_Info	Acquire EtherCAT slave information
_ECAT_Master_Get_DC_Status	Acquire the motion card's DC status, time and time offset
_ECAT_Master_Get_Connect_Status	Acquire EtherCAT master’s connection status
_ECAT_Master_Get_Api_BufferLength	Acquire the command amount of each slave that has not been completed
_ECAT_Master_Get_Cycle_SpendTime	Acquire the time spent on Tx and Rx every cycle and the maximum consuming time in the log
_ECAT_Master_Check_Initial_Done	Check whether the DLL initialization has been completed
_ECAT_Master_Get_Initial_ErrorCode	Acquire the error code when error occurs
_ECAT_Master_Check_Working_Counter	Acquire the current connection status of EtherCAT communication
_ECAT_Master_Get_Return_Code_Message	Acquire the corresponding message of each return code

6.1 _ECAT_Master_Open

■ Syntax

U16 PASCAL _ECAT_Master_Open(U16 *Cardnum)

■ Purpose

This is for checking the number of motion cards and EtherCAT kernels, as well as creating memory block.

Note:

1. This is the most essential API for controlling Delta EtherCAT master. Please execute this API before starting using other functions.
2. If the acquired quantity is 0, it means the environment for executing the program does not support EtherCAT communication.
3. To avoid blue screen death, DO NOT use this API in the thread when using C# to develop RTX environment.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Quantity of the motion control cards or EtherCAT kernels

■ Example

```
U16 Status;
U16 Cardnum=0;

Status = _ECAT_Master_Open(&Cardnum);
```

6.2 _ECAT_Master_Initial

■ Syntax

U16 PASCAL _ECAT_Master_Initial(U16 CardNo)

■ Purpose

This is for initialiaing EtherCAT communication and switching the slave to OP mode.

Note: After using this API to do the initialization, please apply API “_ECAT_Master_Check_Initial_Done” (section 6.12) to check the status and wait for it to return “0”. Then, you can start using other functions.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo = 0;
U16 Cardnum = 0;
U16 InitDone = 100;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

    while (InitDone != 0)
{
        Status = _ECAT_Master_Check_Initial_Done(CardNo, &InitDone);
    if (InitDone == 99)
    {
            // Error
        Status = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
        break;
    }
}
}
```

6.3 _ECAT_Master_Reset

■ Syntax

U16 PASCAL _ECAT_Master_Reset(U16 CardNo)

■ Purpose

This is for resetting the EtherCAT master’s status and switching the slave to initial mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

Status = _ECAT_Master_Reset(CardNo);
}
```

6.4 _ECAT_Master_Close

■ **Syntax**

U16 PASCAL _ECAT_Master_Close()

■ **Purpose**

This is for disabling all functions of EtherCAT master and kernels and releasing the memory.

■ **Example**

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

Status = _ECAT_Master_Reset(CardNo);
}

Status = _ECAT_Master_Close();
```

6.5 _ECAT_Master_Get_CardSeq

■ **Syntax**

U16 PASCAL _ECAT_Master_Get_CardSeq (U16 CardSeq, U16 * CardNo)

■ **Purpose**

This is for acquiring motion card No.

After acquiring the quantity of EtherCAT master, you can get the number of EtherCAT master in sequence by the master sequence ID, which starts from 0. And the card No. of EtherCAT master is the number on the knob exactly. If you are using RTX version of Delta PAC, the master's No. is always 16.

■ **Parameter**

Name	Data type	Property	Description
CardSeq	U16	Number	Sequence ID of the motion card.
CardNo	U16*	Number	Card No.

■ **Example**

```
U16 Status;
U16 CardNo=0;
U16 Cardnum=0;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

Status = _ECAT_Master_Reset(CardNo);
}

Status = _ECAT_Master_Close();
```

6.6 _ECAT_Master_Get_SlaveNum

Syntax

U16 PASCAL _ECAT_Master_Get_SlaveNum(U16 CardNo, U16 *Slavenum)

Purpose

This is for acquiring slave quantity on the communication bus of the specified EtherCAT master.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Slavenum	U16*	Quantity	EtherCAT slave quantity that connected to the specified master.

Example

```
U16 Status;
U16 CardNo=16, Slavenum=0;

Status = _ECAT_Master_Get_SlaveNum(CardNo, &Slavenum);
```

6.7 _ECAT_Master_Get_Slave_Info

■ Syntax

U16 PASCAL _ECAT_Master_Get_Slave_Info(U16 CardNo, U16 SeqID, U16 *NodeID, U32 *VenderID, U32 *ProductCode, U32 *RevisionNo, U32 *DCTime)

■ Purpose

This is for acquiring EtherCAT slave information.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
SeqID	U16	Number	Module's physical sequence ID.
NodeID	U16*	Number	The module's corresponding node ID.
VenderID	U32*	Number	Vendor ID.
ProductCode	U32*	Number	Product code.
RevisionNo	U32*	Number	Version No.
DCTime	U32*	Time	DC time of the module.

■ Example

```
U16 Status;
U16 CardNo=16, SeqID =2, NodeID =2;
U32 VenderID, ProductCode, RevisionNo, DCTime;

Status = _ECAT_Master_Get_Slave_Info(CardNo, SeqID, NodeID , &VenderID, &ProductCode,
&RevisionNo, & DCTime);
```

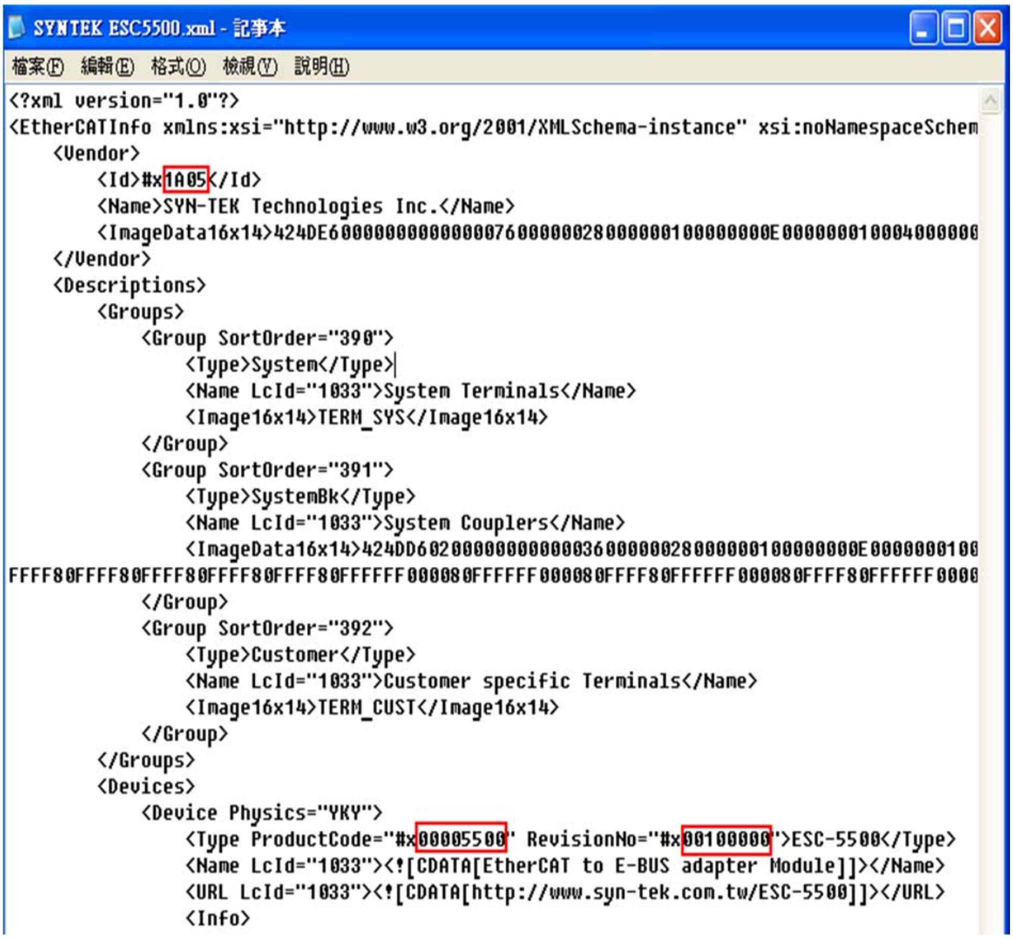
■ Description

EtherCAT master has certain restriction on the supported modules. If your slave module (not Delta products) is not supported by the master station, you should firstly obtain this slave module's Vender ID, product code and revision No. Then, contact Delta and we will help you to solve the issue. To obtain the aforementioned information, please open the XML document of the slave module via text editor. (See Figure 6.7.1)

VenderID: 1A05 or 1DD

ProductCode: 00005500

RevisionNo: 00100000


SYNTEK ESC5500.xml - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
<?xml version="1.0"?>
<EtherCATInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchem
    <Vendor>
        <Id>#x1A05</Id>
        <Name>SYN-TEK Technologies Inc.</Name>
        <ImageData16x14>424DE6000000000000007600000028000000100000000E000000010004000000
    </Vendor>
    <Descriptions>
        <Groups>
            <Group SortOrder="390">
                <Type>System</Type>
                <Name LcId="1033">System Terminals</Name>
                <Image16x14>TERM_SYS</Image16x14>
            </Group>
            <Group SortOrder="391">
                <Type>SystemBk</Type>
                <Name LcId="1033">System Couplers</Name>
                <ImageData16x14>424DD6020000000000003600000028000000100000000E0000000100
FFFF80FFFF80FFFF80FFFF80FFFF80FFFFFF000080FFFFFF000080FFFF80FFFFFF000080FFFF80FFFFFF0000
            </Group>
            <Group SortOrder="392">
                <Type>Customer</Type>
                <Name LcId="1033">Customer specific Terminals</Name>
                <Image16x14>TERM_CUST</Image16x14>
            </Group>
        </Groups>
        <Devices>
            <Device Physics="YKY">
                <Type ProductCode="#x00005500" RevisionNo="#x00100000">ESC-5500</Type>
                <Name LcId="1033"><![CDATA[EtherCAT to E-BUS adapter Module]]></Name>
                <URL LcId="1033"><![CDATA[http://www.syn-tek.com.tw/ESC-5500]]></URL>
                <Info>
```

Figure 6.7.1 Contents of XML document

6.8 _ECAT_Master_Get_DC_Status

■ Syntax

U16 PASCAL _ECAT_Master_Get_DC_Status (U16 CardNo, U32 *State, I32 *Time, I32 *OffsetTime)

■ Purpose

This is for acquiring the motion card's DC status, time and time offset.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
State	U32*	Status	DC current status. 0: Synchronizing 1: Initial synchronization completed
Time	I32*	Time (us)	The current synchronized DC time (it is about half of the cycle time when system is stabilized)
OffsetTime	I32*	Time (us)	The offset time of DC time clock

■ Example

```
U16 Status;
U16 CardNo=16;
U32 Status;
I32 Time, OffsetTime;

Status = _ECAT_Master_Get_DC_Status(CardNo, &State, &Time, &OffsetTime);
```

6.9 _ECAT_Master_Get_Connect_Status

■ Syntax

```
U16 PASCAL _ECAT_Master_Get_Connect_Status(U16 CardNo, U16 * MasterStatus)
```

■ Purpose

This is for acquiring EtherCAT master’s connection status.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
MasterStatus	U16*	Status	Master’s current status 1: Init mode 2: Pre-OP mode 4: Safe-OP mode 8: OP mode

■ Example

```
U16 Status;
U16 CardNo=16;
U16 MasterStatus=0;

Status = _ECAT_Master_Get_Connect_Status(CardNo, &MasterStatus);
```

6.10 _ECAT_Master_Get_Api_BufferLength

- **Syntax**

U16 PASCAL _ECAT_Master_Get_Api_BufferLength(U16 CardNo, U16 SlaveNo, U16 *BuffLength)

- **Purpose**

This is for acquiring the command amount of each slave that has not been completed.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
SlaveNo	U16	Number	Node ID
BuffLength	U16*	Value	The accumulated Buffer length of API instructions.

- **Example**

```
U16 Status;
U16 CardNo=16, SlaveNo=1, BuffLength;

Status = _ECAT_Master_Get_Api_BufferLength(CardNo, SlaveNo , &BuffLength);
```

6.11 _ECAT_Master_Get_Cycle_SpendTime

■ Syntax

U16 PASCAL _ECAT_Master_Get_Cycle_SpendTime (U16 CardNo, F64 *Tx_Time, F64 *Tx_MaxTime, F64 *Rx_Time, F64 *Rx_MaxTime)

■ Purpose

This is for acquiring the time spent on Tx and Rx every cycle and the maximum consuming time in the log.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Tx_Time	F64*	Time (us)	Acquire the time spent on Tx (us)
Tx_MaxTime	F64*	Time (us)	Acquire the maximum consumed time of Tx in the log
Rx_Time	F64*	Time (us)	Acquire the time spent on Rx (us)
Rx_MaxTime	F64*	Time (us)	Acquire the maximum consumed time of Rx in the log

■ Example

```
U16 Status;
U16 CardNo=16;
F64 Tx_Time, Tx_MaxTime, Rx_Time, Rx_MaxTime;

Status = _ECAT_Master_Get_Cycle_SpendTime(CardNo, &Tx_Time, &Tx_MaxTime, &Rx_Time,
&Rx_MaxTime);
```

6.12 _ECAT_Master_Check_Initial_Done

■ Syntax

U16 PASCAL _ECAT_Master_Check_Initial_Done(U16 CardNo, U16 *InitDone)

■ Purpose

This is for checking whether the DLL initialization has been completed.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
InitDone	U16*	Status	0: Completed 1: Initializing 99: Error

■ Example

```
U16 Status;
U16 CardNo = 0;
U16 Cardnum = 0;
U16 InitDone = 100;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

        while (InitDone != 0)
{
                Status = _ECAT_Master_Check_Initial_Done(CardNo, &InitDone);
        if (InitDone == 99)
        {
                        // Error
                Status = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
                break;
        }
}
}
```

6.13 _ECAT_Master_Get_Initial_ErrorCode

■ Syntax

U16 PASCAL _ECAT_Master_Get_Initial_ErrorCode(U16 CardNo)

■ Purpose

This is for acquiring the error code when error occurs (“_ECAT_Master_Check_Initial_Done” returns 99). Refer to Chapter 34 for more information about error code description.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo = 0;
U16 Cardnum = 0;
U16 InitDone = 100;

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);

Status = _ECAT_Master_Initial(CardNo);

    while (InitDone != 0)
{
        Status = _ECAT_Master_Check_Initial_Done(CardNo, &InitDone);
    if (InitDone == 99)
    {
            // Error
        Status = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
        break;
    }
}
}
```

6.14 _ECAT_Master_Check_Working_Counter

■ Syntax

U16 PASCAL _ECAT_Master_Check_Working_Counter(U16 CardNo, U16 *Abnormal_Flag, U16 *Working_Slave_Cnt)

■ Purpose

This is for acquiring the current connection status of EtherCAT communication.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number (No.)	Card No.
Abnormal_Flag	U16*	Status	0: Normal 1: Error
Working_Slave_Cnt	U16*	Quantity	The current EtherCAT slave quantity detected during communication. In normal condition, the detected number is identical to that obtained via _ECAT_Master_Get_SlaveNum. When error occurs, this number can be used to locate the error regarding physical wiring.

■ Example

```
U16 Status;
U16 CardNo=16;
U16 Abnormal_Flag, Working_Slave_Cnt;

Status = _ECAT_Master_Check_Working_Counter(CardNo, &Abnormal_Flag,
&Working_Slave_Cnt);
```

6.15 _ECAT_Master_Get_Return_Code_Message

■ **Syntax**

U16 PASCAL _ECAT_Master_Get_Return_Code_Message(U16 ReturnCode, I8 *Message);

■ **Purpose**

This is for acquiring the corresponding message of each return code.

■ **Parameter**

Name	Data type	Property	Description
ReturnCode	U16	Number	Return code
Message	I8*	String	Corresponding message for each return code

■ **Example**

```
U16 Status, Rt;
U16 CardNo = 0;
U16 Cardnum = 0;
U16 InitDone = 100;
I8 Message[500]= {0};

Status = _ECAT_Master_Open(&Cardnum);
for (U16 CardSeq = 0; CardSeq < Cardnum; CardSeq ++)
{
Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);
    Status = _ECAT_Master_Initial(CardNo);
    while (InitDone != 0)
{
        Status = _ECAT_Master_Check_Initial_Done(CardNo, &InitDone);
    if (InitDone == 99)
    {
            // Error
        Status = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
        Rt = _ECAT_Master_Get_Return_Code_Message(Status, Message);
        break;
    }
}
}
```

EtherCAT CoE Standard Communication

7

This chapter introduces the use of API for CoE (CANopen over EtherCAT) standard communication. EtherCAT protocol allows issuing SDO or PDO command to the Slave directly via CoE standard communication.

An OD (Object Dictionary) represents different parameters for the slaves, such as motor's current position. And each OD consists of one index and one sub-index, which signify the communication address in hexadecimal format and property respectively.

PDO (Process Data Object) is the communication method defined by CANopen. It is the cyclic communication between the Master and all Slaves. Before the communication (initialization) starts, the Master will define one PDO mapping table, which consists of several ODs. Then, the PDO mapping table will be sent to each Slave according to the set cycle on a regular basis. In the same cycle, the Slave executes the command or sends the status to the Master in accordance with the PDO mapping table.

SDO (Service Data Objects) is also the communication method defined by CANopen. Different from PDO, the communication time is determined by users. The Slave gives response only when the Master sends a request. That is to say, SDO cannot issue the command frequently. However, users can read/write any OD that is not in PDO mapping table via SDO.

API list of EtherCAT CoE standard communication

Function name	Description
_ECAT_Slave_SDO_Send_Message	Issue SDO command (CANopen) to the slave
_ECAT_Slave_SDO_Read_Message	Acquire the current SDO data (CANopen) of the slave
_ECAT_Slave_SDO_Quick_Send_Message	Issue SDO command (CANopen) to the slave without waiting for the response
_ECAT_Slave_SDO_Quick_Read_Message	Issue SDO read command (CANopen) to the slave without waiting for the response
_ECAT_Slave_SDO_Read_Response	Read the returned data from the slave.
_ECAT_Slave_SDO_Wait_All_Done	Wait multiple slaves to complete all the SDO commands.
_ECAT_Slave_SDO_Get_ErrorCode	Acquire the error code of ERR_ECAT_SDO_Return that returned during the execution of SDO Send_Message or Read_Message. Please refer to CANopen protocol or the definition of each device for error code.
_ECAT_Slave_SDO_Check_Done	Check if the specified slave has completed all the SDO commands
_ECAT_Slave_PDO_Get_OD_Data	Read the data of an OD index in the PDO mapping
_ECAT_Slave_PDO_Set_OD_Data	Send the data of an OD index in the PDO mapping
_ECAT_Slave_PDO_Get_Information	Acquire the basic information of each slave device PDO.
_ECAT_Slave_PDO_Get_Detail_Mapping	Acquire the details of PDO mapping in the slave device
_ECAT_Slave_PDO_Get_Rx_Data	Acquire all slave Rx data of the PDO mapping
_ECAT_Slave_PDO_Get_Tx_Data	Acquire all slave Tx data of the PDO mapping
_ECAT_Slave_PDO_Set_Tx_Detail_Data	Configure all slave Tx data of the PDO mapping

7.1 _ECAT_Slave_SDO_Send_Message

- **Syntax**

U16 PASCAL _ECAT_Slave_SDO_Send_Message(U16 CardNo, U16 NodeID,
U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

- **Purpose**

This is for issuing SDO command (CANopen) to the slave.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Index	U16	Index	The index of CANopen object dictionary
SubIndex	U16	Subindex	The subindex of CANopen object dictionary
DataSize	U16	byte	Data size of the sent message. Unit: byte
Data	U8*	data	Data of the sent message

- **Example**

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4]={0};

Status = _ECAT_Slave_SDO_Send_Message(CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize, Data);
```

7.2 _ECAT_Slave_SDO_Read_Message

■ Syntax

U16 PASCAL _ECAT_Slave_SDO_Read_Message(U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

■ Purpose

This is for acquiring the current SDO data (CANopen) of the slave.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Index	U16	Index	The index of CANopen object dictionary
SubIndex	U16	Subindex	The subindex of CANopen object dictionary
DataSize	U16	byte	Data size of the received message. Unit: byte
Data	U8*	data	Data of the received message

■ Example

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x1000, SubIndex=0, DataSize=4;
U8 Data[4] = {0};

Status =_ECAT_Slave_SDO_Read_Message(CardNo, NodeID, SlotNo, Index, SubIndex,
DataSize, &Data[0]);
```

7.3 _ECAT_Slave_SDO_Quick_Send_Message

■ Syntax

U16 PASCAL _ECAT_Slave_SDO_Quick_Send_Message(U16 CardNo, U16 NodeID,
U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

■ Purpose

This is for issuing SDO command (CANopen) to the slave without waiting for the response.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number (No.)	Card No.
NodeID	U16	Number (No.)	Node ID
SlotNo	U16	Number (No.)	Slot ID
Index	U16	Index	The index of CANopen object dictionary
SubIndex	U16	Subindex	The subindex of CANopen object dictionary
DataSize	U16	byte	Data size of the received message. Unit: Byte
Data	U8*	data	Data of the received message

■ Example

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4] = {0};

Status = _ECAT_Slave_SDO_Quick_Send_Message (CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize, &Data[0]);
```

7.4 _ECAT_Slave_SDO_Quick_Read_Message

■ Syntax

U16 PASCAL _ECAT_Slave_SDO_Quick_Read_Message(U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize)

■ Purpose

This is for issuing SDO read command (CANopen) to the slave without waiting for the response.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Index	U16	Index	The index of CANopen object dictionary
SubIndex	U16	Subindex	The subindex of CANopen object dictionary
DataSize	U16	byte	Data size of the received message. Unit: Byte.

■ Example

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;

Status = _ECAT_Slave_SDO_Quick_Read_Message (CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize);
```

7.5 _ECAT_Slave_SDO_Read_Response

■ Syntax

U16 PASCAL _ECAT_Slave_SDO_Read_Response (U16 CardNo, U16 NodeID,
U16 SlotNo, U16* Done, U8* Data, U32* ErrorCode)

■ Purpose

This is for reading the returned data from the slave.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Done	U16*	Status	0: Completed 1: In execution 2: Error
Data	U8	Data	Data of the received message
ErrorCode	U32*	Number	Error code

■ Example

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4] = {0};
U32 ErrorCode;

Status = _ECAT_Slave_SDO_Read_Response (CardNo, NodeID, SlotNo,
&Done, Data, &ErrorCode);
```

7.6 _ECAT_Slave_SDO_Wait_All_Done

- Syntax

U16 PASCAL _ECAT_Slave_SDO_Wait_All_Done(U16 CardNo, U16 AxisNum,
U16* NodeID, U16* SlotNo)

- Purpose

This is for waiting multiple slaves to complete all SDO commands.

- Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	quantity	The quantity of the engaged axis
NodeID	U16*	Number	Node ID
SlotNo	U16*	Number	Slot ID

- **Example**

```
U16 Status;
U16 AxisNum = 2;
U16 CardNo=16, NodeID[2] = {0,1} ,SlotNo[2] = {0,0};

Status = _ECAT_Slave_SDO_Wait_All_Done(CardNo, AxisNum ,NodeID, SlotNo);
```

7.7 _ECAT_Slave_SDO_Get_ErrorCode

Syntax

U16 PASCAL _ECAT_Slave_SDO_Get_ErrorCode(U16 CardNo, U16 NodeID, U16 SlotNo, U32* ErrorCode)

Purpose

Aquire the error code of ERR_ECAT_SDO_Return that returned during the execution of SDO Send_Message or Read_Message. Please refer to CANopen protocol or the definition of each device for error code.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
ErrorCode	U32*	Number	Error code

Example

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U32 ErrorCode;

Status = _ECAT_Slave_SDO_Get_ErrorCode(CardNo, NodeID, SlotNo, &ErrorCode);
```

List of error code

Code	Description
0x0503 0000	Toggle bit not alternated.
0x0504 0000	SDO protocol timed out
0x0504 0001	Client/server command specifier not valid or unknown.
0x0504 0002	Invalid block size
0x0504 0003	Invalid sequence numbe
0x0504 0004	CRC error
0x0504 0005	Out of memory
0x0601 0000	Unsupported access to an object
0x0601 0001	Attempt to read a write-only object
0x0601 0002	Attempt to write to a read-only object
0x0602 0000	Object not listed in object directory
0x0604 0041	Object cannot be mapped to PDO
0x0604 0042	Number and length of objects to be transferred longer than PDO length.
0x0604 0043	General parameter incompatibility
0x0604 0047	General internal device incompatibility
0x0606 0000	Access denied because of hardware error
0x0607 0010	Data type does not match, unsuitable PDO/SDO parameter length
0x0607 0012	Data type does not match, PDO/SDO parameter length exceeded
0x0607 0013	Data type does not match, PDO/SDO parameter length not long enough
0x0609 0011	Subindex does not exist
0x0609 0030	Parameter value range exceeded
0x0609 0031	Value of parameter written too high
0x0609 0032	Value of parameter written too low
0x0609 0036	Maximum value is less than minimum value
0x0800 0000	General error
0x0800 0020	Data cannot be transferred/saved to the application
0x0800 0021	Data cannot be transferred/saved to the application due to local control.
0x0800 0022	Data cannot be transferred/saved to the application due to current device status
0x0800 0023	Dynamic generation of object directory error or no object directory available

7.8 _ECAT_Slave_SDO_Check_Done

■ Syntax

U16 PASCAL _ECAT_Slave_SDO_Check_Done (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Done)

■ Purpose

This is for checking if the specified slave has completed all the SDO commands.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Done	U16*	Status	Definition 0: Completed 1: Processing

■ Example

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, Done;

Status = _ECAT_Slave_SDO_Check_Done (CardNo, NodeID, SlotNo, &Done);
```

7.9 _ECAT_Slave_PDO_Get_OD_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Get_OD_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 ODIndex, U16 ODSubIndex, U16 ByteSize, U8 *Data)

■ Purpose

This is for reading the data of an OD index in the PDO mapping. Before initializing the master, the OD code should be defined in PDO mapping table by EcNavi in advance.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOType	U16	Type	OD Format to be read 0: Rx 1: Tx
ODIndex	U16	Index	OD index of the data
ODSubIndex	U16	Subindex	OD subindex of the data
ByteSize	U16	byte	The size of the data space
Data	U8*	Data	Acquire the data contents of the specified OD index

■ Example

```
U16 Status;
U8 Data = 0;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODIndex = 0x1810, ODSubIndex = 0x01, ByteSize = 0x43;

Status = _ECAT_Slave_PDO_Get_OD_Data (CardNo, NodeID, SlotNo, IOType, ODIndex,
ODSubIndex, ByteSize, &Data);
```

7.10 _ECAT_Slave_PDO_Set_OD_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Set_OD_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U16 ODIndex, U16 ODSubIndex, U8 *Data)

■ Purpose

This is for sending the data of an OD index in the PDO mapping. Before initializing the master, the OD code should be defined in PDO mapping table by EcNavi in advance.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
ODIndex	U16	Index	OD index of the data
ODSubIndex	U16	Subindex	OD subindex of the data
Data	U8*	data	The data contents of the specified OD index

■ Example

```
U16 Status;
U8 Data = 0;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODIndex = 0x1780, ODSubIndex = 0x01

Status = _ECAT_Slave_PDO_Set_OD_Data (CardNo, NodeID, SlotNo, ODIndex, ODSubIndex,
&Data);
```

7.11 _ECAT_Slave_PDO_Get_Information

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Get_Information (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 *ODCnt, U16 *StartIndex)

■ Purpose

This is for acquiring the basic information of each slave device PDO.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOType	U16	Type	OD format to be read 0: Master Rx 1: Master Tx
ODCnt	U16*	Quantity	OD number of this IO type in the slave.
StartIndex	U16*	Index	The starting index of the slave device.

■ Example

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODCnt, StartIndex;

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType,
&ODCnt, &StartIndex);
```

7.12 _ECAT_Slave_PDO_Get_Detail_Mapping

- Syntax

U16 PASCAL _ECAT_Slave_PDO_Get_Detail_Mapping (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 ODSeqID, U16 *ODIndex, U16 *ODSubIndex, U16 *ODByteSize, U16 *ODStartIndex)

- Purpose

This is for acquiring the details of PDO mapping in the slave device.

- Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOType	U16	Type	OD format to be read 0: Master Rx 1: Master Tx
ODSeqID	U16	Number	OD sequence ID
ODIndex	U16*	Index	OD index of the data
ODSubIndex	U16*	Subindex	OD subindex of the data
ByteSize	U16*	byte	The size of the data space
ODStartIndex	U16*	Index	The starting index of the OD

- **Example**

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0, ODSeqID = 0, ODCnt , StartIndex;
U16 ODIndex[8]={0}, ODSubIndex[8]={0}, ODBitSize[8]={0}, ODStartIndex[8]={0};

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType, &ODCnt,
&StartIndex);

  for (ODSeqID = 0; ODSeqID < ODCnt; ODSeqID++)
  {
      Status = _ECAT_Slave_PDO_Get_Detail_Mapping(CardNo, NodeID, SlotNo,
      IOType, ODSeqID, &ODIndex[ODSeqID], &ODSubIndex[ODSeqID],
      &ODBitSize[ODSeqID],&ODStartIndex[ODSeqID]);
  }
```

7.13 _ECAT_Slave_PDO_Get_Rx_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Get_Rx_Data(U16 CardNo, BYTE *Data)

■ Purpose

This is for acquiring all slave Rx data of the PDO mapping.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Data	BYTE	data	All the salve Rx data. The max. data size is 1536 (0x600).

■ Example

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};

Status = _ECAT_Slave_PDO_Get_Rx_Data(CardNo, &Data);
```

7.14 _ECAT_Slave_PDO_Get_Tx_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Get_Tx_Data(U16 CardNo, BYTE *Data)

■ Purpose

This is for acquiring all slave Tx data of the PDO mapping.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Data	BYTE	data	All the slave Tx data. The max. data size is 1536 (0x600).

■ Example

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};

Status = _ECAT_Slave_PDO_Get_Tx_Data(CardNo, &Data);
```

7.15 _ECAT_Slave_PDO_Set_Tx_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Set_Tx_Data(U16 CardNo, BYTE *Data)

■ Purpose

This is for configuring all slave Tx data of the PDO mapping.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number (No.)	Card No.
Data	BYTE	data	All the slave Tx data. The max. data size is 1536 (0x600).

■ Example

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};
Status = _ECAT_Slave_PDO_Get_Tx_Data(CardNo, &Data);

// Directly edit the data in TxData
Data[0x001] = 0x01;

Status = _ECAT_Slave_PDO_Set_Tx_Data(CardNo, &Data);
```

7.16 _ECAT_Slave_PDO_Set_Tx_Detail_Data

■ Syntax

U16 PASCAL _ECAT_Slave_PDO_Set_Tx_Detail_Data(U16 CardNo, U16 NodeID, U16 SlotNo, U16 ODStartIndex, U16 ByteSize, U8 *Data)

■ Purpose

This is for configuring all the slave Tx data of the PDO mapping.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
StartIndex	U16	index	The starting index of the slave.
ByteSize	U16	byte	Tx data size to be transmitted
Data	U8*	Data	Tx data to be transmitted

■ Example

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 1;
U16 ODCnt, StartIndex = 0x60, ByteSize = 4;
U8 Data[4]={0, 1, 0, 1};

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType,
&ODCnt, &StartIndex);

Status = _ECAT_Slave_PDO_Set_Tx_Detail_Data(CardNo, NodeID, SlotNo, StartIndex,
ByteSize, &Data);
```

(This page is intentionally left blank.)

General Operation of Motion Axis

8

This chapter presents the APIs for general operation of motion axis, which can be used to acquire system status or set the motion parameters/commands of motion axis. And Touch Probe setting is also included.

API list of general operation of motion axis

Function name	Description
_ECAT_Slave_Motion_Set_Svon	Set the servo to On/Off state.
_ECAT_Slave_Motion_Ralm	Reset the alarm of the axis. Before applying this command, please clear the alarm first. Otherwise, the alarm might occur again.
_ECAT_Slave_Motion_Sd_Stop	Set the deceleration time for motor to decelerate to stop
_ECAT_Slave_Motion_Emg_Stop	This is for emergency stop of the axis. The motor will stop with its maximum deceleration
_ECAT_Slave_Motion_Set_Alm_Reaction	Set the action when alarm occurs
_ECAT_Slave_Motion_Set_Position	Specify current feedback position of the axis
_ECAT_Slave_Motion_Set_Command	Set the motion command data of the axis
_ECAT_Slave_Motion_Set_MoveMode	Set the motion mode of the axis
_ECAT_Slave_Motion_Get_MoveMode	Acquire the information of current motion mode
_ECAT_Slave_Motion_Get_ControlWord	Acquire the current control word of the axis
_ECAT_Slave_Motion_Get_StatusWord	Acquire the current status word of the axis.
_ECAT_Slave_Motion_Get_Mdone	Acquire the current status of motion done
_ECAT_Slave_Motion_Get_Position	Acquire the current position of the axis.
_ECAT_Slave_Motion_Get_Command	Acquire the current command information
_ECAT_Slave_Motion_Get_Target_Command	Acquire the target command data of the axis
_ECAT_Slave_Motion_Get_Actual_Position	Acquire the actual position command of the axis
_ECAT_Slave_Motion_Get_Actual_Command	Acquire the current command data. The data will vary with to the applied motion mode.
_ECAT_Slave_Motion_Get_Current_Speed	Acquire the current speed of the axis
_ECAT_Slave_Motion_Get_Torque	Acquire the feedback torque from the motor
_ECAT_Slave_Motion_Get_Buffer_Length	Acquiring the quantity of the commands that have not been carried out
_ECAT_Slave_Motion_Set_TouchProbe_Config	Set the mode of the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Set_TouchProbe_QuickStart	Enable the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Set_TouchProbe_QuickDone	Execute the first Touch Probe function (Touch Probe 1) again
_ECAT_Slave_Motion_Set_TouchProbe_Disable	Disable the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Get_TouchProbe_Status	Acquire the current status of the first Touch Probe function (Touch Probe 1)
_ECAT_Slave_Motion_Get_TouchProbe_Position	Acquire the current position of first Touch Probe function (Touch Probe 1)

8.1 _ECAT_Slave_Motion_Set_Svon

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_Svon(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the servo to On/Off state.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Servo Off 1: Servo On

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Enable=1;

Status = _ECAT_Slave_Motion_Set_Svon(CardNo, AxisNo, SlotNo, Enable);
```

8.2 _ECAT_Slave_Motion_Ralm

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Ralm(U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for resetting the alarm of the axis. Before applying this command, please clear the alarm first. Otherwise, the alarm might occur again.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Ralm(CardNo, AxisNo, SlotNo);
```

8.3 _ECAT_Slave_Motion_Sd_Stop

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Sd_Stop(U16 CardNo, U16 AxisNo, U16 SlotNo, F64 Tdec)

■ Purpose

This is for setting the deceleration time for motor to decelerate to stop.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Tdec	F64	Time	The specified deceleration time. CSP, CSV, and CST mode are in the unit of second. HOME, PP, PV and PT mode are using the unit of the drive inc/s^2. inc represents the unit for the slave setting. Please refer to the user manual of the applied slave. (OD: 0x6083 Sub 0).

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
F64 Tdec=0.1;

Status = _ECAT_Slave_Motion_Sd_Stop(CardNo, AxisNo, SlotNo, Tdec);
```

8.4 _ECAT_Slave_Motion_Emg_Stop

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Emg_Stop(U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for emergency stop of the axis. The motor will stop with its maximum deceleration.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Emg_Stop(CardNo, AxisNo, SlotNo);
```

8.5 _ECAT_Slave_Motion_Set_Alm_Reaction

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_Alm_Reaction (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Fault_Type, U16 Waring_Type);

■ Purpose

This is for setting the action when alarm occurs.

Note: It is also applicable to group function (Please refer to Chapter 17).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Fault_Type	U16	Option	Set the action when error (Fault) occurs. 0: It will not stop or interrupt the new command automatically. 1: Stop the current action when the rising-edge signal is triggered. It will not interrupt the new command automatically. 2: Remain idle status until the error is cleared.
Waring_Type	U16	Option	Set the action when warning occurs. 0: It will not stop nor interrupt new command automatically. 1: Stop the current action when the rising edge triggered. It will not interrupt the new command automatically. 2: Remain at idle unless the warning is cleared.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 Fault_Type = 2, Waring_Type = 1;

Status = _ECAT_Slave_Motion_Set_Alm_Reaction (CardNo, AxisNo, SlotNo, Fault_Type,
Waring_Type);
```

8.6 _ECAT_Slave_Motion_Set_Position

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_Position(U16 CardNo, U16 AxisNo, U16 SlotNo,I32 NewPosition)

■ Purpose

This is for specifying current feedback position of the axis. This will change the position data set in the servo drive or pulse module, which might also alter the machine’s coordinates. Please pay extra attention when using this API.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
NewPosition	I32	Value	Specify current feedback position

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 NewPosition=2500000;

Status = _ECAT_Slave_Motion_Set_Position(CardNo, AxisNo, SlotNo, NewPosition);
```

8.7 _ECAT_Slave_Motion_Set_Command

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_Command(U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewCommand)

■ Purpose

This is for setting the motion command data of the axis. The unit (property) of the data will vary with the applying motion mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
NewCommand	I32	Value	Acquire the current command information: In CSP mode, the data is the current position. In CSV mode, the data is the current speed. In CST mode, the data is the permillage of current torque.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 NewCommand=3000000;

Status = _ECAT_Slave_Motion_Set_Command(CardNo, AxisNo, SlotNo, NewCommand);
```

8.8 _ECAT_Slave_Motion_Set_MoveMode

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_MoveMode(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 MoveMode)

■ Purpose

This is for setting the motion mode of the axis.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
MoveMode	U16	Mode	Motion mode of the axis 0: Idle mode 1: Profile Position (PP) mode 2. Velocity mode 3: Profile Velocity (PV) mode 4: Profile Torque (PT) mode 6: Home mode 7: Interpolated Position (IP) 8: Cyclic Synchronous Position (CSP) mode 9: Cyclic Synchronous Velocity (CSV) mode 10: Cyclic Synchronous Torque (CST) mode

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, MoveMode=1;

Status = _ECAT_Slave_Motion_Set_MoveMode(CardNo, AxisNo, SlotNo, MoveMode);
```

8.9 _ECAT_Slave_Motion_Get_MoveMode

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_MoveMode (U16 CardNo , U16 AxisNo , U16 SlotNo, U8 *Mode)

■ Purpose

This is for acquiring the information of current motion mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U8*	Mode	1: Profile Position (PP) mode 2: Velocity mode 3: Profile Velocity (PV) mode 4: Profile Torque (PT) mode 6: Home mode 7: Interpolated Position (IP) mode 8: Cyclic Synchronous Position (CSP) mode 9: Cyclic Synchronous Velocity (CSV) mode 10: Cyclic Synchronous Torque (CST) mode

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U8 Mode;

Status = _ECAT_Slave_Motion_Get_MoveMode (CardNo, AxisNo, SlotNo, &Mode);
```

8.10 _ECAT_Slave_Motion_Get_ControlWord

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_ControlWord (U16 CardNo , U16 AxisNo,
U16 SlotNo, U16 *ControlWord)

■ Purpose

This is for acquiring the current control word of the axis.

■ Parameter

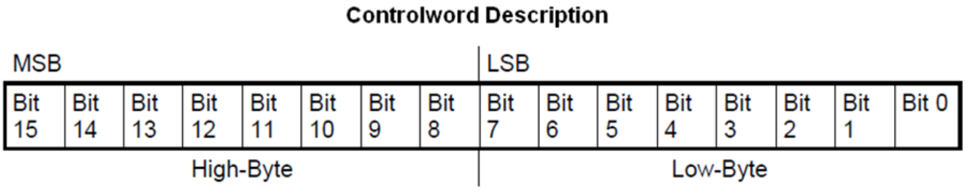
Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
ControlWord	U16*	Data	Control word of the axis (Please refer to the figure below for the definition. For the reserved items, see the description of each slave.)

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 ControlWord ;

Status = _ECAT_Slave_Motion_Get_ControlWord (CardNo, AxisNo, SlotNo, &ControlWord);
```

■ Description

Definition of 6040H: CANopen communication



Controlword Description

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

Figure 8.2.1 Corresponding bits of control word

Definitions of the control word bits

Bit	Name
0	Switch On
1	Enable Voltage (Servo on)
2	Quick Stop
3	Enable Operation (Motor enabled)

Bit	Name
4	Operation Mode Specific (Operation mode)
5	Operation Mode Specific (Operation mode)
6	Operation Mode Specific (Operation mode)
7	Fault reset (Clear servo alarm)
8	Halt
9 ~15	N/A

Operation Mode Specific represents by bit 4 ~ 6 are as follows. See the table below:

Bit	Profile Position (PP) mode	Homing mode	Position Interpolation mode	Profile Velocity (PV) mode	Profile Torque (PT) mode
4	New Position Command (Rising-edge triggered)	Start homing (Rising-edge triggered)	N/A	N/A	N/A
5	Activate Immediately	N/A	N/A	N/A	N/A
6	0: Absolute motion 1: Relative motion	N/A	N/A	N/A	N/A

Table 8.2.3 Definition of Operation Mode Specific

8.11 _ECAT_Slave_Motion_Get_StatusWord

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_StatusWord (U16 CardNo ,U16 AxisNo ,U16 SlotNo , U16 * StatusWord)

■ Purpose

This is for acquiring the current status word of the axis.

■ Parameter

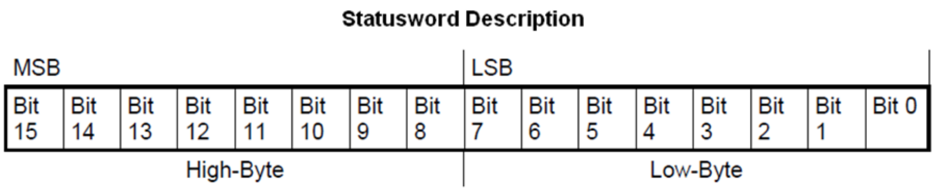
Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
StatusWord	U16*	Data	Status word of the axis (Please refer to the figure below for the definition of Control Word. For the reserved items, see the description of each Slave.)

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 StatusWord;

Status = _ECAT_Slave_Motion_Get_StatusWord (CardNo, AxisNo, SlotNo, &StatusWord);
```

■ Description

Definition of 6041H: CANopen communication


Statusword Description

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

Figure 8.3.1 Corresponding bits of status word

Definitions of the status word bits

Bit	Name
0	Ready to Switch On
1	Switched On
2	Operation Enabled (Motor enabled)

Bit	Name
3	Fault (Servo error occurs; motor disabled)
4	Voltage Enabled (Servo on)
5	Quick Stop
6	Switch On Disabled
7	Warning
8	N/A
9	Remote
10	Target Reached (Target-reached signal)
11	Internal Limit Active (internal software limit; not supported)
12	Operation Mode Specific (Operation mode)
13	Operation Mode Specific (Operation mode)
14	N/A
15	N/A

Operation Mode Specific represents by bit 12 ~ 13 are as follows. See the table below:

Bit	Profile Position (PP) mode	Homing mode	Position Interpolation mode	Profile Velocity (PV) mode	Profile Torque (PT) mode
12	New position command available	Executing homing	Executing position interpolation	Velocity is 0	N/A
13	Position following overrange	Homing error occurs	N/A	N/A	N/A

Table 8.3.3 Definition of Operation Mode Specific

8.12 _ECAT_Slave_Motion_Get_Mdone

Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Mdone(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Mdone)

Purpose

This is for acquiring the current status of motion done.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mdone	U16*	Status	In Cyclic Synchronous (CS) mode 0: Idle state 1: Accelerating 2: Constant speed (CSP) / Target speed reached (CSV) / Target torque reached (CST) 3: Decelerating 5: MailBox processing In Profile series mode 0: Idle state 1: Motion in progress

Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Mdone;

Status = _ECAT_Slave_Motion_Get_Mdone (CardNo, AxisNo, SlotNo, &Mdone);
```

8.13 _ECAT_Slave_Motion_Get_Position

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Position(U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *Position)

■ Purpose

This is for acquiring the current position of the axis.

When applying the function of enabling virtual position (_ECAT_Slave_CSP_Virtual_Set_Enable) in section 9.29, users can acquire the master's virtual position with this API. The virtual position here signifies the master's position before compensation, which is the machine's actual target position desired by users.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Position	I32*	Value	Get the current position of the axis.

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 Position=0;

Status = _ECAT_Slave_Motion_Get_Position (CardNo, AxisNo, SlotNo, &Position);
```

8.14 _ECAT_Slave_Motion_Get_Command

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *Command)

■ Purpose

This is for acquiring the current command information. The data unit (property) will vary with the applying motion mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Command	I32*	Data	Acquire the current command information: In CSP mode, the data is the current position. In CSV mode, the data is the current speed. In CST mode, the data is the permillage of current torque.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Command=0;

Status = _ECAT_Slave_Motion_Get_Command (CardNo, AxisNo, SlotNo, &Command);
```

8.15 _ECAT_Slave_Motion_Get_Target_Command

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Target_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *TargetPosition)

■ Purpose

This is for acquiring the target command data of the axis.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
TargetCommand	I32*	Value	The command differs from the applied motion mode. CSP, PP mode: Target position CSV, PV mode: Target speed CST, PT mode: Target torque Homing mode: Returned value will be 0.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 TargetCommand = 0;

Status = _ECAT_Slave_Motion_Get_Target_Command (CardNo, AxisNo, SlotNo,
&TargetCommand);
```

8.16 _ECAT_Slave_Motion_Get_Actual_Position

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Actual_Position(U16 CardNo, U16 AxisNo,
U16 SlotNo, I32* ActualPositon)

■ Purpose

This is for acquiring the actual position command of the axis.

When applying the function of enabling virtual position (_ECAT_Slave_CSP_Virtual_Set_Enable) in section 9.29, users should use this API to get motor's actual feedback position.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Axis No.
SlotNo	U16	Number	Node ID
ActualPosition	I32*	Value	Get actual feedback position of the motor.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 ActualPosition;

Status = _ECAT_Slave_Motion_Get_Actual_Position(CardNo, AxisNo, SlotNo,
&ActualPosition);
```

8.17 _ECAT_Slave_Motion_Get_Actual_Command

■ Syntax

```
U16 PASCAL _ECAT_Slave_Motion_Get_Actual_Command(U16 CardNo, U16 AxisNo,
U16 SlotNo,I32* ActualCommand)
```

■ Purpose

This is for acquiring the current command data. The data will vary with to the applied motion mode.

When applying the function of enabling virtual position (_ECAT_Slave_CSP_Virtual_Set_Enable) in section 9.29, users should use this API to get the information of current command.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Axis No.
SlotNo	U16	Number	Node ID
ActualCommand	I32*	Value	Acquire the current command information: In CSP mode, the data is the current position. In CSV mode, the data is the current speed. In CST mode, the data is the permillage (‰) of current torque.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 ActualCommand;

Status = _ECAT_Slave_Motion_ Get_Actual_Command (CardNo, AxisNo, SlotNo,
&ActualCommand);
```

8.18 _ECAT_Slave_Motion_Get_Current_Speed

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Current_Speed(U16 CardNo, U16 AxisNo,
U16 SlotNo, I32 *Speed)

■ Purpose

This is for acquiring the current speed of the axis.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Speed	I32*	Value	Current speed of the axis.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Speed;

Status = _ECAT_Slave_Motion_Get_Current_Speed (CardNo, AxisNo, SlotNo, &Speed);
```

8.19 _ECAT_Slave_Motion_Get_Torque

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Torque (U16 CardNo, U16 AxisNo, U16 SlotNo, I16 *Torque)

■ Purpose

This is for acquiring the feedback torque from the motor.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Torque	I16*	permillage	Get the feedback torque from the motor （Unit: permillage ‰ ）.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 Torque = 0;

Status = _ECAT_Slave_Motion_Get_Torque (CardNo, AxisNo, SlotNo, &Torque);
```

8.20 _ECAT_Slave_Motion_Get_Buffer_Length

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_Buffer_Length (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 *BufferLength)

■ Purpose

This is for acquiring the quantity of the commands that have not been carried out.

EtherCAT master provides a buffer which can store 20 motion commands. When the current command has not been completely processed and the next motion command is received, the next command will be put in the buffer temporarily. It will start to be executed after the current command is finished. When the buffer stores more than 20 commands, the new coming ones will be ignored. This API can be used to check the buffer status.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BufferLength	U16*	Quantity	Get the quantity of the command that has not been carried out.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, BufferLength = 0;

Status = _ECAT_Slave_Motion_Get_Buffer_Length (CardNo, AxisNo, SlotNo,
& BufferLength);
```

8.21 _ECAT_Slave_Motion_Set_TouchProbe_Config

Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_Config(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 TriggerMode, U16 Signal_Source)

Purpose

This is for setting the mode of the first Touch Probe function.

Through the servo drive or pulse module which provides Touch Probe function, users can acquire the pulse (position) when the high-speed digital (DI) signal is triggered.
Apart from setting the first Touch Probe function, this API can be used to enable the first Touch Probe function simultaneously. (Set CANopen OD code-60B8 bit 0 to 1)

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
TriggerMode	U16	Option	This parameter is to define how the Touch Probe function is triggered: Please refer to the definition of Slave CANopen 60B8 bit1. 1. Please refer to **Description** below for Delta’s product.
Signal_Source	U16	Option	This parameter is to define the source trigger signal: 1. Please refer to the definition of Slave CANopen 60B8 bit2. 2. Please refer to **Description** below for Delta’s product.

Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 TriggerMode =1; // Continuous recording; It will record the pulse (position) once the signal is triggered.
Signal_Source=1; // Motor’s Z pulse is regarded as the trigger signal of Touch Probe 1

Status = _ECAT_Slave_Motion_Set_TouchProbe_Config(CardNo, AxisNo, SlotNo,
TriggerMode, Signal_Source);
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);
```

- **Description**

CANopen defines two Touch Probe functions in OD-60B8. However, the applying method varies with the applied servo drive. Please check the user manual in advance.

The following table illustrates the Touch Probe function and settings which defined in OD-60B8 by Delta ASDA A2-E and EtherCAT remote pulse module. Please note that this API does not support the second touch probe function (Touch Probe 2.)

Bit	Value	Description
0	0	Disable Touch Probe 1
	1	Enable Touch Probe 1
1	0	Record the pulse (position) when signal is triggered for the first time
	1	Continuous recording; as long as the signals is triggered, the pulse (position) will be recorded
2	0	Regard the digital input as the trigger signal of Touch Probe 1
	1	Regard the motor’s Z pulse as the trigger signal of Touch Probe 1
3	0	Reserved
4	0	Stop capturing the pulse (position) when the trigger signal for Touch Probe 1 is rising-edge triggered.
	1	Start capturing the pulse (position) when the trigger signal for Touch Probe 1 is rising-edge triggered.
5	0	Stop capturing the pulse (position) when the trigger signal for Touch Probe 1 is falling-edge triggered.
	1	Start capturing the pulse (position) when the trigger signal for Touch Probe 1 is falling-edge triggered.
6 ~ 7	0	Reserved
8	0	Disable touch probe 2 (This API does not support Touch Probe function 2.)
	1	Enable touch probe 2 (This API does not support Touch Probe function 2.)
9	0	Record the pulse (position) when signal is triggered for the first time (This API does not support Touch Probe function 2.)
	1	Continuous recording; as long as the signals is triggered, the pulse (position) will be recorded (This API does not support Touch Probe function 2.)
10	0	Regard the digital input as the trigger signal of Touch Probe 2 (This API does not support Touch Probe function 2.)
	1	Regard the motor’s Z pulse as the trigger signal of Touch Probe 2 (This API does not support Touch Probe function 2.)
11	0	Reserved

Bit	Value	Description
		(This API does not support Touch Probe function 2.)
12	0	Stop capturing the pulse (position) when the trigger signal for Touch Probe 2 is rising-edge triggered. (This API does not support Touch Probe function 2.)
	1	Start capturing the pulse (position) when the trigger signal for Touch Probe 2 is rising-edge triggered. (This API does not support Touch Probe function 2.)
13	0	Stop capturing the pulse (position) when the trigger signal for Touch Probe 2 is falling-edge triggered. (This API does not support Touch Probe function 2.)
	1	Start capturing the pulse (position) when the trigger signal for Touch Probe 2 is falling-edge triggered. (This API does not support Touch Probe function 2.)
14 ~ 15	0	Reserved (This API does not support Touch Probe function 2.)

8.22 _ECAT_Slave_Motion_Set_TouchProbe_QuickStart

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for enabling the first Touch Probe function (Touch Probe 1). Use this API to set OD code – 60B8 bit 4 to 1.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ Example

```
U16 Status, TouchProbe_Status, CardNo=16,AxisNo=1,SlotNo=0;
I32 LatchPosition = 0;
U16 TriggerMode =0;    // It records the pulse (position) when the signal is triggered for the first
time.
U16 Signal_Source=1; // Motor’s Z pulse signal is regarded as the trigger signal of the first Touch
Probe function.
Status = _ECAT_Slave_Motion_Set_TouchProbe_Config(CardNo, AxisNo, SlotNo,
TriggerMode, Signal_Source);
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);
while (1)
{
        Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
        if (TouchProbe_Status & 0x2)
        {
                Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
                break;
        }
}
// Users have to enable the Touch Probe function again to record the pulse.
Status = _ECAT_Slave_Motion_Set_TouchProbe_Disable(CardNo, AxisNo, SlotNo);
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);
```

```
while (1)
{
        Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
        if (TouchProbe_Status & 0x2)
        {
                Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
                break;
        }
}
```

- **Description**

Please refer to the following diagram (figure 8.22.1) when activating Touch Probe function.

1. When 60B8 bit 1 is set to 0, Touch Probe function will be triggered for a single time.
2. Since this API function is not enabled, when it is triggered for the 0th time, 60B9 bit 1 is not triggered and the value of 60BA (the recorded pulse position) is invalid.
3. When it is triggered for the 1st time (60B8 bit 4 is on), 60B9 bit 1 is triggered and the captured pulse position will be stored in OD code – 60BA.
4. When it is triggered for the second time and also this API is applied (60B8 bit 4 is on), the recorded pulse position is still invalid. This is because setting 60B8 bit 1 to 0 means the signal will be triggered for once only.
   - If 60B8 bit 1 is set to 1, then the new pulse position will be recorded.
   - To record this pulse position, please refer to Trigger for the 3rd time (see the figure below). Users should use the API "_ECAT_Slave_Motion_Set_TouchProbe_Disable" (section 8.24) to disable the Touch Probe function and restart it.
   - Apart from the method mentioned above, you can API "_ECAT_Slave_Motion_Set_TouchProbe_QuickDone" (section 8.23) to restart Touch Probe function.

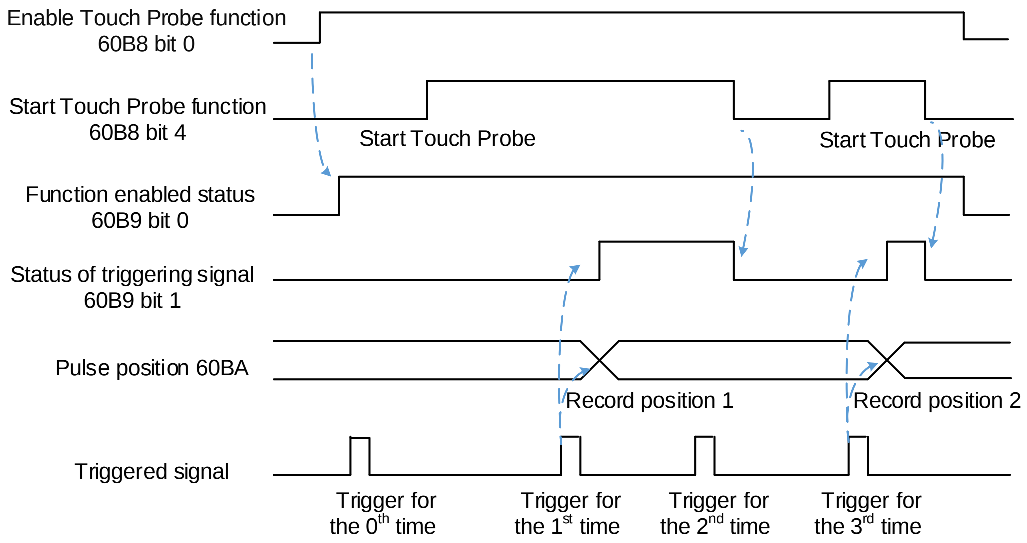


Figure 8.22.1 Touch Probe – Trigger the signal for once (60B8 bit 1 = 0)

8.23 _ECAT_Slave_Motion_Set_TouchProbe_QuickDone

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_QuickDone(U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for executing the 1st Touch Probe function again.

When OD code – 60B8 bit 1 is set to 0, this function will be executed once. To execute this function repeatedly, users have to use _ECAT_Slave_Motion_Set_TouchProbe_Disable (section 8.24) to disable the function first. Then, use _ECAT_Slave_Motion_Set_TouchProbe_QuickStart (section 8.22) to enable it again. For more user-friendly way, you can directly re-activate the first Touch Probe function through this API.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ **Example**

```
U16 Status, TouchProbe_Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 LatchPosition = 0;
U16 TriggerMode =0;    // It records the pulse (position) when the signal is triggered for the first
time.
U16 Signal_Source=1; // Motor’s Z pulse is regarded as trigger signal of the first Touch Probe
function.

Status = _ECAT_Slave_Motion_Set_TouchProbe_Config(CardNo, AxisNo, SlotNo,
TriggerMode, Signal_Source);
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);

while (1)
{
    Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
    if (TouchProbe_Status & 0x2)
    {
        Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
        break;
    }
}
// This API simplifies the step of re-activating the Touch Probe function (TriggerMode=0).
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickDone(CardNo, AxisNo, SlotNo);
while (1)
{
    Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
    if (TouchProbe_Status & 0x2)
    {
        Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
        break;
    }
}
```

8.24 _ECAT_Slave_Motion_Set_TouchProbe_Disable

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_Disable(U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for disabling the first Touch Probe function (Touch Probe 1).

Use this API to set OD code – 60B8 bit 0 to 0 to disable the first Touch Probe function.

Note: When TriggerMode in API (_ECAT_Slave_Motion_Set_TouchProbe_Config) in section 8.21 is set to 0, users have to apply this API (disable Touch Probe 1) before using the function (_ECAT_Slave_Motion_Set_TouchProbe_QuickStart) in section 8.22.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Set_TouchProbe_Disable(CardNo, AxisNo, SlotNo);
```

8.25 _ECAT_Slave_Motion_Get_TouchProbe_Status

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_TouchProbe_Status(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Status)

■ Purpose

This is for acquiring the current status of the first Touch Probe function (Touch Probe 1).
This API can be used to read the value of OD code – 60B9.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
State	U16*	Value	This API can acquire the current status of the first Touch Probe function and read the value of OD code – 60B9. 1. Please refer to the definition of Slave CANopen 60B9. 2. Please refer to **Description** below for Delta's product.

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 TouchProbe_Status;

Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
```

■ Description

CANopen defines the status of Touch Probe function in OD codes-60B9. However, the applying method varies with the applied servo drive. Please check the user manual in advance.
The following table illustrates status of Touch Probe function defined by Delta ASDA A2-E and EtherCAT remote pulse module. (This API does not support the second Touch Probe function.)

CANopen-OD 60B9:

Bit	Value	Description
0	0	Touch probe 1 is disabled
	1	Touch probe 1 is enabled
1	0	The signal of Touch Probe 1 is not rising-edge triggered
	1	The signal of Touch Probe 1 has been rising-edge triggered
2	0	The signal of Touch Probe 1 is not falling-edge triggered
	1	The signal of Touch Probe 1 has been falling-edge triggered
3 ~ 5	0	Reserved
6	0	Regard digital input signal as the trigger signal for Touch Probe 1
	1	Regard motor’s Z pulse as the trigger signal for Touch Probe 1
7	0, 1	Update the captured value of the Touch Probe 1
8	0	Touch probe 2 is disabled (Remote pulse module does not support this function)
	1	Touch probe 2 is enabled (Remote pulse module does not support this function)
9	0	The signal of Touch Probe 2 is not rising-edge triggered (Remote pulse module does not support this function)
	1	The signal of Touch Probe 2 has been rising-edge triggered (Remote pulse module does not support this function)
10	0	The signal of Touch Probe 2 is not falling-edge triggered (Remote pulse module does not support this function)
	1	The signal of Touch Probe 2 has been falling-edge triggered (Remote pulse module does not support this function)
11 ~ 13	0	Reserved (Remote pulse module does not support this function)
14	0	Regard digital input signal as the trigger signal for Touch Probe 2 (Remote pulse module does not support this function)
	1	Regard motor’s Z pulse as the trigger signal for Touch Probe 2 (Remote pulse module does not support this function)
15	0, 1	Update the captured value of the Touch Probe 2 (Remote pulse module does not support this function)

8.26 _ECAT_Slave_Motion_Get_TouchProbe_Position

■ Syntax

U16 PASCAL _ECAT_Slave_Motion_Get_TouchProbe_Position(U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *LatchPosition)

■ Purpose

This is for acquiring the current position of first Touch Probe function (Touch Probe 1).

EtherCAT master reads the captured position from OD code - 60BA. It is suggested to save the captured position to PDO mapping table via EcNavi for quick access.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
LatchPosition	I32*	Value	The captured position; When it captures the signal again, this value will be updated. Please remember to save the captured position of each time. Please refer to the returned value of Slave CANopen 60BA bit2.

■ Example

```
U16 Status, TouchProbe_Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 LatchPosition = 0;
U16 TriggerMode =1;    // Continuous recording; It will record the pulse (position) once the signal
is triggered.
U16 Signal_Source=1; // Motor’s Z pulse signal is regarded as the trigger signal of the first Touch
Probe.

Status = _ECAT_Slave_Motion_Set_TouchProbe_Config(CardNo, AxisNo, SlotNo,
TriggerMode, Signal_Source);
Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);

while (1)
{
        Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo, &
TouchProbe_Status);
        if (TouchProbe_Status & 0x2)
        {
                Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
        }
}
```

Cyclic Synchronous Position Mode (CSP) 9

This chapter explains the APIs used in CSP mode. Different from PP mode, EtherCAT master issues one position command in each communication cycle. Thus, the motion path is controlled by EtherCAT master. CSP mode proveds single-axis motion, multi-axis interpolation and synchronous motion and advanced motion compensation.

CSP (Cyclic Synchronous Position) mode uses PDO communication to issue position commands. EtherCAT master will calculate the position commands for the next communication cycle after analyzing the moving distance, speed and acceleration of the API. Then, it sends the new command to all motion axes in each communication cycle to achieve single-axis motion or multi-axis interpolation.

API list of cyclic synchronous position mode (CSP)

Function name	Description
_ECAT_Slave_CSP_Start_Move	Execute linear interpolation of single axis
_ECAT_Slave_CSP_Start_V_Move	Execute the single-axis motion with constant speed
_ECAT_Slave_CSP_Start_Arc_Move	Execute two-axis arc motion, moving from current position and the specified circle center to form the specified arc's angle
_ECAT_Slave_CSP_Start_Arc2_Move	Execute two-axis arc motion, moving from current position and the specified circle center to form the specified arc's angle
_ECAT_Slave_CSP_Start_Arc3_Move	Execute two-axis arc motion, moving from the current position and specified circle center to the specified end point
_ECAT_Slave_CSP_Start_Spiral_Move	Execute two-axis spiral motion, moving from current position and the specified circle center to form the specified angle
_ECAT_Slave_CSP_Start_Spiral2_Move	Execute two-axis spiral motion, moving from current position and the specified circle center to the end point with the specified cycle number.
_ECAT_Slave_CSP_Start_Sphere_Move	Execute three-axis sphere motion and moving from current position and the known circle center to the target position with three-dimensional vector
_ECAT_Slave_CSP_Start_Heli_Move	Set three-axis helical motion, moving from current position and the known circle center to the specified height in Z-axis direction
_ECAT_Slave_CSP_Start_Multiaxes_Move	Execute multi-axis linear motion
_ECAT_Slave_CSP_Start_Msbrline_Move	Execute multi-axis point to point motion with smooth speed
_ECAT_Slave_CSP_Set_Gear	Set the E-gear ratio
_ECAT_Slave_CSP_Set_Softlimit	Set the software limit
_ECAT_Slave_CSP_TargetPos_Change	Set a new target position
_ECAT_Slave_CSP_Velocity_Change	Set a new target speed
_ECAT_Slave_CSP_Feedrate_Overwrite	For the advanced setting of speed change for single axis
_ECAT_Slave_CSP_Speed_Continue_Enable	Enable or disable the continuous speed function
_ECAT_Slave_CSP_Speed_Continue_Set_Mode	Set the continuous speed mode
_ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio	Set the percentage of for starting blending speed of two commands.
_ECAT_Slave_CSP_Scurve_Rate	Set the ratio of S-curve and T-curve during acceleration and deceleration
_ECAT_Slave_CSP_Liner_Speed_Master	Set the speed (vector) of advanced interpolation function.

Function name	Description
_ECAT_Slave_CSP_Mask_Axis	When multi-axis command is being executed, this API can be used to stop the specified axes without influencing others.
_ECAT_Slave_CSP_Sync_Config	Set the function of synchronous motion of multiple axes
_ECAT_Slave_CSP_Sync_Move	Enable the function of synchronous motion of multiple axes
_ECAT_Slave_CSP_Start_Mabrline_Move	Set to smooth the operation of point-to-point motion of multiple axes
_ECAT_Slave_CSP_Start_2Segment_Move	Set the single-axis linear motion by specifying two distances and speed
_ECAT_Slave_CSP_Start_PVT_Move	Set the single-axis motion to move to multiple points at fixed time
_ECAT_Slave_CSP_Start_PVTComplete_Move	Specify the initial speed and end speed of the single-axis motion, moving through multiple points at fixed time.
_ECAT_Slave_CSP_Virtual_Set_Enable	Enable function of virtual position
_ECAT_Slave_CSP_Virtual_Set_Command	Set the virtual position and replacing the current position with the specified position
_ECAT_Slave_CSP_Get_SoftLimit_Status	Acquire the status of software limit
_ECAT_Slave_CSP_Pitch_Set_Interval	Set the interval of the pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Mode	Set the mode of pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Org	Set the start position of pitch error compensation.
_ECAT_Slave_CSP_Pitch_Set_Rel_Table	Set the relative position of each interval for pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Abs_Table	Set the absolute position of each interval for pitch error compensation
_ECAT_Slave_CSP_Pitch_Set_Enable	Enable function of pitch error compensation.

9.1 _ECAT_Slave_CSP_Start_Move

- **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Start_Move(U16 CardNo, U16 AxisNo, U16 SlotNo, I32 Dist, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

- **Purpose**

This is for executing linear interpolation of single axis.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Dist	I32	Pulse	The specified moving distance
StrVel	I32	Pulse / second	The initial speed of the motion
ConstVel	I32	Pulse / second	The constant speed of the motion
EndVel	I32	Pulse / second	The end speed of the motion
TPhase1	F64	Second	Duration to change from initial speed to constant speed
TPhase2	F64	Second	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 Dist=12000000, Strvel=0, ConstVel =2000000, EndVel=0;
F64 TPhase1=0.1, TPhase2=0.1;
U16 Scurve=0, Abs_Rel=1;

Status = _ECAT_Slave_CSP_Start_Move (CardNo, AxisNo, SlotNo, Dist, Strvel, ConstVel,
EndVel, Tacc, Tdec, Scurve, Abs_Rel);
```

- **Description**

Linear interpolation of CSP command (acc/deceleration)

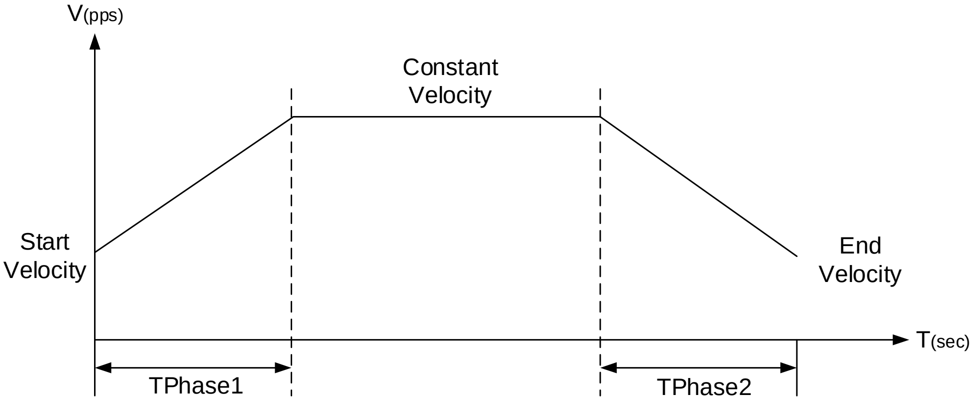


Figure 9.1.1 Perform lineter interpolation by referring to relative coordinates (T-curve)

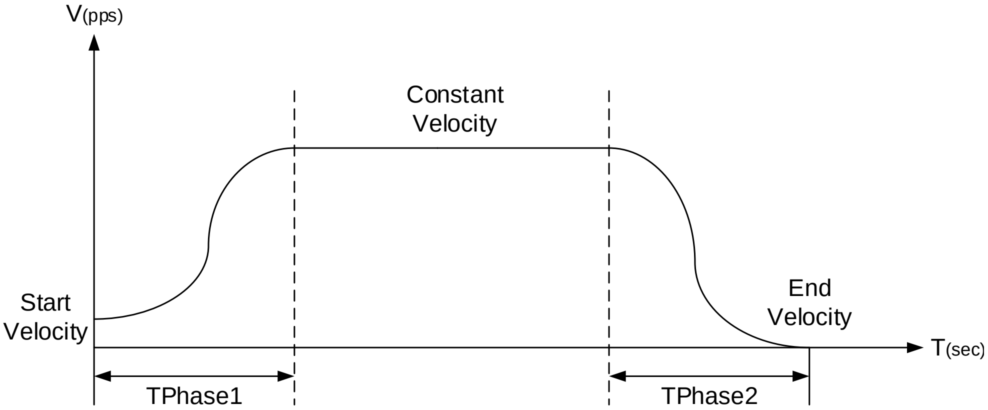


Figure 9.1.2 Perform linear interpolation by referring to relative coordinates (S-curve)

9.2 _ECAT_Slave_CSP_Start_V_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_V_Move (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 Dir, I32 Strvel, I32 MaxVel, F64 Tacc, U16 Scurve)

■ Purpose

This is for executing the single-axis motion with constant speed.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Dir	U16	Option	Moving direction 0: Forward 1: Reverse
Strvel	I32	Pulse / second	Parameter of initial speed
MaxVel	I32	Pulse / second	Parameter of the constant speed
Tacc	F64	Time (second)	Acceleration time
Scurve	U16	Option	0: T-curve(Default) 2: S-curve

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Dir=1, Strvel=0, MaxVel=2000000;
F64 Tacc =0.1;
U16 Scurve=0, Abs_Rel=1;

Status = _ECAT_Slave_CSP_Start_V_Move (CardNo, AxisNo, SlotNo, Dir, Strvel,
MaxVel, Tacc, Scurve, Abs_Rel);
```

9.3 _ECAT_Slave_CSP_Start_Arc_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Arc_Move (U16 CardNo, U16 *AxisNo,
U16 *SlotNo,I32 *CenterPoint, F64 Angle , I32 Strvel, I32 ConstVel, I32 EndVel,
F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for executing two-axis arc motion, moving from current position and the specified circle center to form the specified arc’s angle.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
CenterPoint	I32*	Array of pulse for each axis	Circle center
Angle	F64	Angle (°)	Set the angle of an arc.
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Description**

The specified angle of the arc can be over 360 degrees.

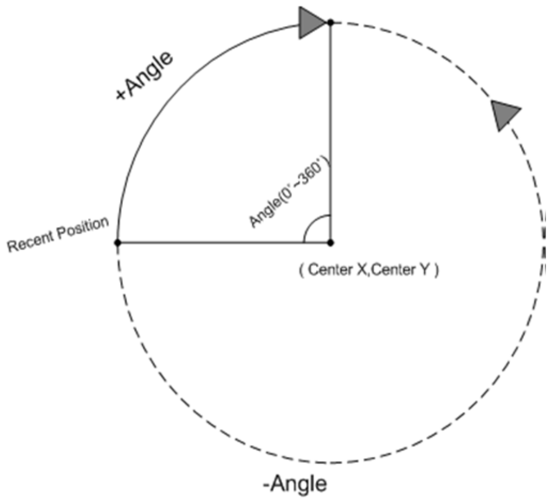


Figure 9.3.1 The axis forms a specified angle from the current position and the given circle center

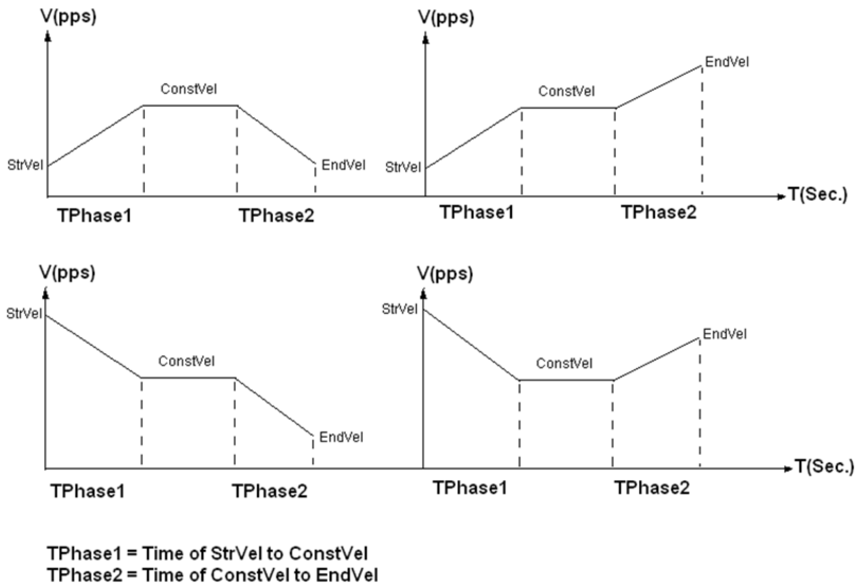


Figure 9.3.2 Description of TPhase1 and TPhase2 (acc/deceleration)

- **Example**

```
U16 Status;
U16 CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0};
I32 CenterPoint[2] ={50000,50000};
F64 Angle=180, TPhase1=0.2, TPhase2=0.1;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
U16 Scurve =0, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Arc_Move (CardNo, AxisNoArray, SlotID, CenterPoint,
Angle,Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.4 _ECAT_Slave_CSP_Start_Arc2_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Arc2_Move(U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *EndPoint, F64 Angle, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for executing two-axis arc motion, regarding the speicify angle from the current position as the included angle and moving to the specified end point.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
EndPoint	I32*	Array of pulse for each axis	End point
Angle	F64	Angle (°)	Set the angle of an arc. When the angle > 0, it means the arc motion will be carried out in clockwise direction. When the angle < 0, it means the arc motion will be carried out in counterclockwise direction.
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status;
U16 CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0};
I32 EndPoint [2]= {100000,100000};
F64 Angle=180, TPhase1=0.2, TPhase2=0.1;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
U16 Scurve =0, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Arc2_Move(CardNo, AxisNoArray, SlotID, EndPoint, Angle,
Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

- **Descirption**

The specified angle cannot be exactly 360 degrees (or its multiple).

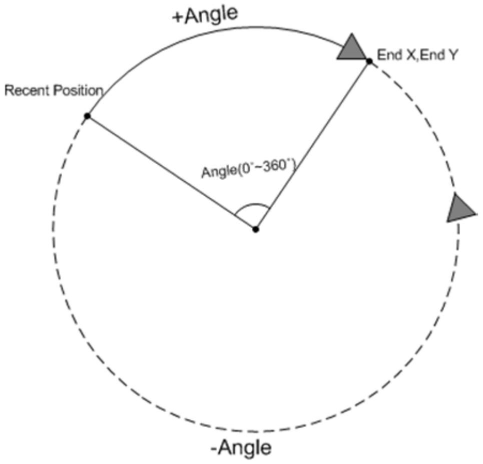


Figure 9.4.1 Regard the specified angle from current position as the inlucded angle and moves to the specified end point

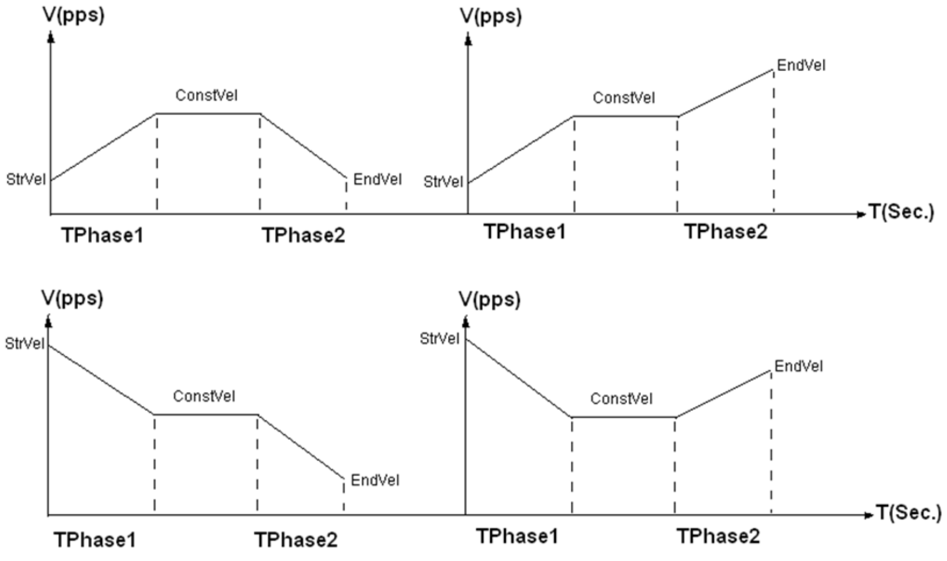


Figure 9.4.2 Desectipion of TPhase1 and TPhase2 (acc/deceleration)

9.5 _ECAT_Slave_CSP_Start_Arc3_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Arc3_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *CenterPoint, I32 *EndPoint, U16 Dir, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for executing two-axis arc motion, moving from the current position and specified circle center to the specified end point.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
CenterPoint	I32*	Array of pulse for each axis	Circle center
EndPoint	I32*	Array of pulse for each axis	End point
Dir	U16	Option	Moving direction of the arc motion 0: Clockwise; 1: counterclockwise
StrVel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

Example

```
U16 Status, CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0}, Dir=1, Scurve =0, Abs_Rel =0;
I32 CenterPoint[2] = {50000,50000}, EndPoint[2] ={10000,100000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Arc3_Move(CardNo, AxisNoArray, SlotID, CenterPoint,
EndPoint, Angle, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

Description

This command can be executed only when the arc is less than 360 degrees.

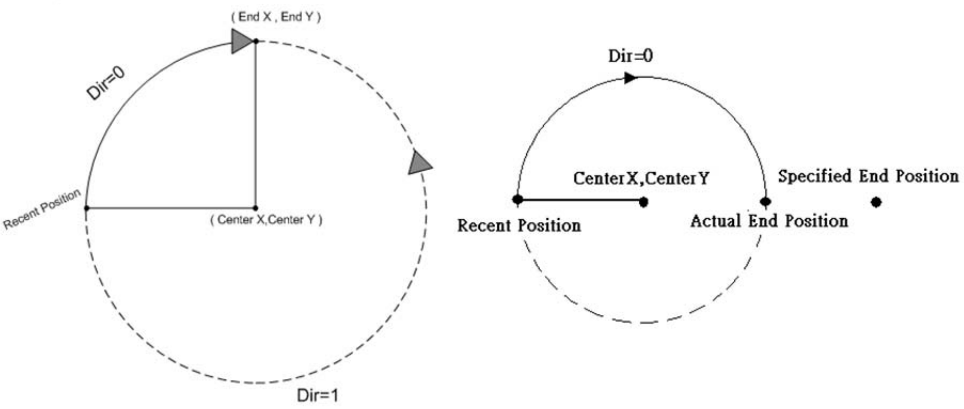


Figure 9.5.1 Move from current position and specified circle center to the specified end point

Note:

1. In the left figure, the axis reaches the specified position.
2. In the right figure, the axis cannot reach the specified position. It is because the distance between its current position and specified position (arc radius) is different from the distance between the specified end point and specified circle center.

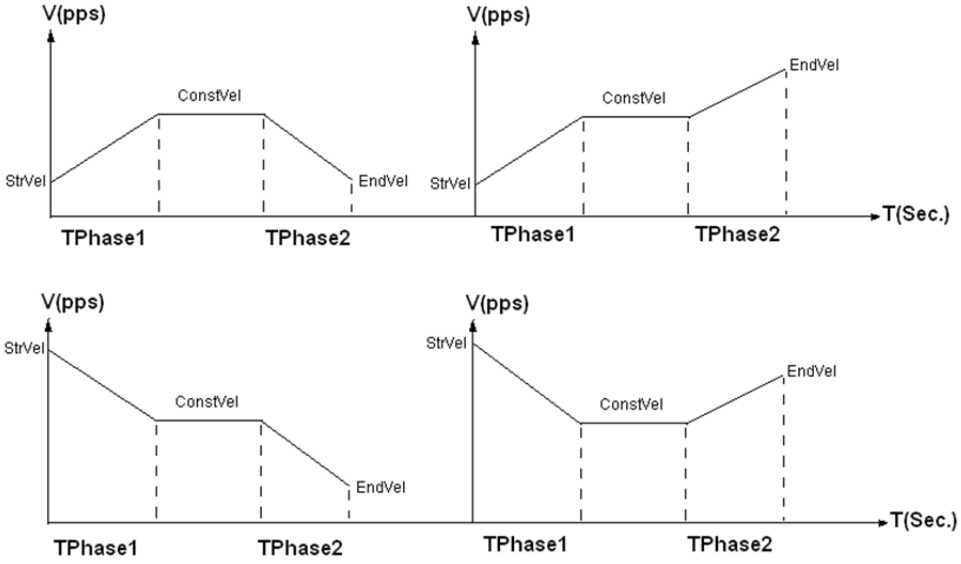


Figure 9.3.2 Description of TPhase1 and TPhase2 (acc/deceleration)

9.6 _ECAT_Slave_CSP_Start_Spiral_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Spiral_Move(U16 CardNo, U16 *AxisNo,
U16 *SlotNo, I32 *CenterPoint, I32 Spiral_Interval, F64 Angle, I32 StrVel, I32 ConstVel,
I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for executing two-axis spiral motion, moving from current position and the specified circle center to form the specified angle.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
CenterPoint	I32*	Array of pulse for each axis	Circle center
Spiral_Interval	I32*	Array of pulse for each axis	Sprial pitch When the value > 0, it means the spiral rotates in outward direction. When the value < 0, it means the spiral rotates in inward direction.
Angle	F64	Angle	Rotation angle of the spiral motion (360 degrees for 1 cycle)
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

■

■ **Example**

```
U16 Status;
U16 CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0}, Scurve =0, Abs_Rel =0;
I32 CenterPoint[2] ={50000,50000}, Spiral_Interval = 5000;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 Angel, TPhase1 = 0.2, TPhase2 = 0.1;

Status = _ECAT_Slave_CSP_Start_Spiral_Move(CardNo, AxisNo, SlotID, CenterPoint,
Spiral_Interval, Angle , StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.7 _ECAT_Slave_CSP_Start_Spiral2_Move

■ **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Start_Spiral2_Move(U16 CardNo, U16 *AxisNo,
U16 *SlotNo, I32 *CenterPoint, I32 EndPoint, U16 Dir, U16 CycleNum, I32 StrVel,
I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **Purpose**

This is for executing two-axis spiral motion, moving from current position and the specified circle center to the end point with the specified cycle number.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
CenterPoint	I32*	Array of pulse for each axis	Circle center
EndPoint	I32*	Array of pulse for each axis	End point
Dir	U16	Option	Moving direction of spiral motion 0: Clockwise 1: Counterclockwise
CycleNum	U16	Number	Cycle number of spiral motion
Strvel	I32	Pulse / second	Parameter of the motion initial speed

Name	Data type	Property	Description
ConstVel	I32	Pulse r / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	The time it takes to change from initial speed to constant speed
TPhase2	F64	Time (second)	This time it takes to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status, CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0},Dir=0, Scurve =0, Abs_Rel =0,
U16 CycleNum=2;
I32 CenterPoint[2] ={50000,50000}, EndPoint [2] ={60000,100000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
Status = _ECAT_Slave_CSP_Start_Spiral2_Move(CardNo, AxisNo, SlotID, CenterPoint,
EndPoint, Dir, CycleNum, StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.8 _ECAT_Slave_CSP_Start_Sphere_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Sphere_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo, I32 * Target1Point, I32 Target2Point, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for executing three-axis sphere motion, moving from current position and the known circle center to the target position with three-dimensional vector.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node AxisNo Array[2] stores the third node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
Target1Point	I32*	Array of pulse for each axis	Target1Point[0] stores any point on X-axis in the arc Target1Point[1] stores any point on Y-axis in the arc Target1Point[2] stores any point on Z-axis in the arc
Target2Point	I32*	Array of pulse for each axis	Target2Point[0] stores the end point of X-axis Target2Point[1] stores the end point of Y-axis Target2Point[2] stores the end point of Z-axis
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	The time is spend from initial speed to constant speed
TPhase2	F64	Time (second)	This time is spend from constant speed to end speed
Scurve	U16	Option	0: T-curve (Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status, CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0}, Scurve =0, Abs_Rel =0;
I32 Target1Point [2] ={25000,50000,20000}, Target2Point [2] ={95000,110000,60000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
Status = _ECAT_Slave_CSP_Start_Sphere_Move (CardNo, AxisNo, SlotID, Target1Point,
Target2Point, StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.9 _ECAT_Slave_CSP_Start_Heli_Move

- **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Start_Heli_Move (U16 CardNo, U16 *AxisNo,
U16 *SlotNo,I32 *CenterPoint, I32 Depth, I32 Pitch, U16 Dir, I32 Strvel, I32 ConstVel,
I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

- **Purpose**

This is for executing three-axis helical motion, moving from current position and the known circle center to the specified height in Z-axis direction.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node AxisNo Array[2] stores the third node
SlotNo	U16*	Array for each slot	Array for each axis (slot ID)
CenterPoint	I32*	Array of pulse for each axis	Circle center
Depth	I32	Pulse	The depth of the specified axis (the overall height of Z-axis; this value can be negative)
Pitch	I32	Pulse	Specify the pitch of the helix
Dir	U16	Option	Helical oving direction 0: Clockwise; 1: counterclockwise
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed

Name	Data type	Property	Description
TPhase1	F64	Time (second)	The time is spend from initial speed to constant speed
TPhase2	F64	Time (second)	This time is spend from constant speed to end speed
Scurve	U16	Option	0: T-curve (Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

■ Example

```
U16 Status, CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0}, Dir=1, Scurve =0, Abs_Rel =0;
I32 CenterPoint[2]= {50000,50000}, Depth =10000, Pitch = 20000;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Heli_Move (CardNo, AxisNoArray, SlotID, CenterPoint ,
Depth, Pitch, Dir, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

■ Descirption

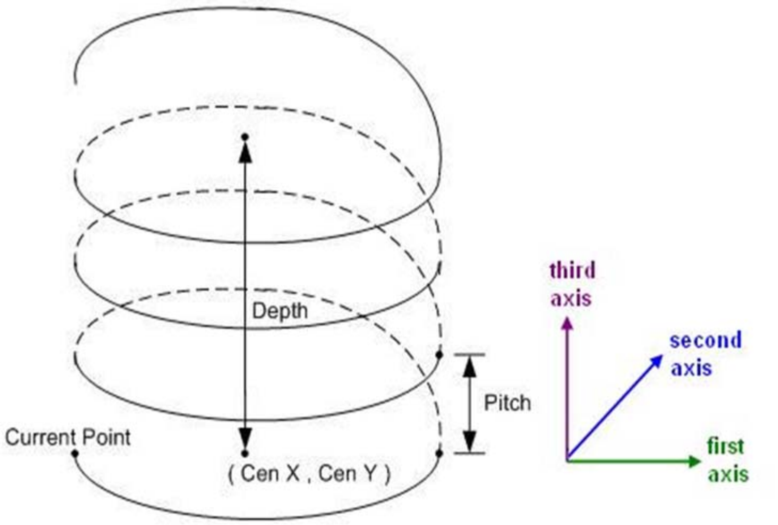


Figure 9.9.1 Moving from current position and the known circle center to the specified height in Z-axis direction

9.10 _ECAT_Slave_CSP_Start_Multiaxes_Move

Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Multiaxes_Move (U16 CardNo, U16 AxisNum,
U16 *AxisArray, U16 *SlotArray, I32 *DistArray, I32 Strvel, I32 ConstVel, I32 EndVel,
F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

Purpose

This is for executing multi-axis linear motion.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNum	U16	Quantity	The engaged axis number. Please refer to Secion 2.1 for the maximun value.
AxisArray	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotArray	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
DistArrary	I32*	Array of pulse for each axis	Array of the moving distance for each axis; the array number should equal to the axis number
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve(Default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status, CardNo=0, AxisNum =2, AxisArray [2]={1,2}, SlotArray [2]={0, 0};
U16 Scurve =0, Abs_Rel =0;
I32 DistArrary [2]= {100000,200000}, StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Multiaxes_Move (CardNo, AxisNum, AxisArray,
SlotArray, DistArrary, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.11 _ECAT_Slave_CSP_Start_Msbrline_Move

- **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Start_Msbrline_Move (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, U16 ArcNodeBit, I32 *Target1Point , I32 *Target2Point, U16 Mode, I32 Parameter, F64 ArcAngle1, F64 ArcAngle2, F64 SpeedRatio, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

- **Purpose**

This is for executing multi-axis point to point motion with smooth speed.

When connecting two paths with smooth acceleration and deceleration, motion path might not be identical to the target position. This API can be applied to set the position when connecting two paths and the position of the actual path.

Note: If the position error is set too small, it might cause machine vibration.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNum	U16	Quantity	The engaged axis number (max. number is 8)
AxisArray	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotArray	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
ArcNodeBit	U16	Value	If one of the paths is arc, please complete the setting below: As shown in figure 9.11.1, M1 and M2 is motion path running from current position, TargetPoint1 to

Name	Data type	Property	Description
			TargetPoint2. M1 M2 16 Bits 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 2 1 4 3 2 1 Node ID Assuming that M1 is the path of arc motion executed by axis 1 and 2, and M2 is the parth executed by axis 3 and 4, then the parameter value should be 0x030C. If M1 is the path of linear motion and M2 is the path of arc motion executed by axis 3 and 4, the parameter value will be 0x000C.
TargetPoint1	I32*	Array of pulse for each axis	Array of the first target position; the array number should equal to the axis number If M1 is the path of linear motion, please input the end point. If M1 is the path of arc motion, please input the arc's circle center. For other axes, please input the value which is identical to the start position.
TargetPoint2	I32*	Array of pulse for each axis	Array of the secondt target position; the array number should equal to the axis number If M2 is the path of linear motion, please input the end point. If M1 is the path of arc motion, please input the arc's circle center. For other axes, please input the value which is identical to the first target position.
Mode	U16	Option	0: M1 and M2 are the paths of linear motion 1: M1 is the path of linear motion and M2 is the path of arc motion 2: M1 is the path of arc motion and M2 is the path of linear motion 3: M1 and M2 are the paths of arc motion
Parameter	I32	Value	Position error of the moving distance ■ Position error is defined as the shortest distance between the actual motion path and target position. ■ When this value is set to 0, it means it will not decelerate when passing the first target position and machine will vibrate. ■ When this value is set too large, the system will

Name	Data type	Property	Description
			adjust the value to the valid range. Please refer to _ECAT_Slave_CSP_Speed_Continue_Set_Combine_ Ratio (section 9.19) for the description when the ratio is set to 100.
ArcAngle1	F64	Angle (°)	If M1 is the path of linear motion, this value will be ignored. If M1 is the path of arc motion, this value is the arc angle.
ArcAngle2	F64	Angle (°)	If M2 is the path of linear motion, this value will be ignored. If M2 is the path of arc motion, this value is the arc angle.
SpeedRatio	F64	Value	Speed ratio of M1 and M2, which is $\frac{\text{Speed of M2}}{\text{Speed of M1}}$.
Strvel	I32	Pulse / second	Parameter of the motion initial speed
ConstVel	I32	Pulse / second	Parameter of the motion constant speed
EndVel	I32	Pulse / second	Parameter of the motion end speed
TPhase1	F64	Time (second)	Duration to change from initial speed to constant speed
TPhase2	F64	Time (second)	Duration to change from constant speed to end speed
Scurve	U16	Option	0: T-curve (Default) 2: S-curve

■ Description

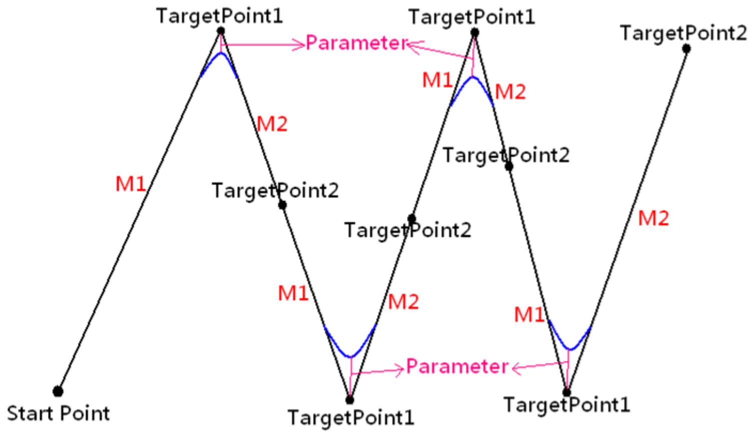


Figure 9.11.1 In this figure, the path is consisted of four _ECAT_Slave_CSP_Start_Msbrline_Move commands and Mode is set to 0

- **Example**

```
U16 Status;
U16 CardNo=0, AxisNum =2, AxisArray [2]={1,2}, SlotArray [2]={0, 0}, ArcNodeBit = 0;
U16 Mode =1, Scurve =0, Abs_Rel =0;
I32 TargetPoint1 [2]= {100000,200000}, TargetPoint2 [2]= {100000,200000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 Parameter = 2 , ArcAngle1 = 0, ArcAngle2 = 0, SpeedRatio = 1;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Msbrline_Move (CardNo, AxisNum, AxisArray,
SlotArray, ArcNodeBit, TargetPoint1, TargetPoint2, Mode, Parameter, ArcAngle1,
ArcAngle2, SpeedRatio, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.12 _ECAT_Slave_CSP_Set_Gear

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Set_Gear(U16 CardNo, U16 AxisNo, U16 SlotNo, I16 Nummerator, I16 Denominator, I16 Enable)

■ Purpose

This is for setting the E-gear ratio.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Nummerator	F64	Value	Set the numerator of E-gear ratio
Denominator	F64	Value	Set the denominator of E-gear ratio
Enable	I16	Option	0: Disable the function of E-gear 1: Enable the function of E-gear

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16Enable=1;
F64 Nummerator=1.0, Denominator=2.0;

Status = _ECAT_Slave_CSP_Set_Gear(CardNo, AxisNo, SlotNo, Nummerator,
Denominator, Enable);
```

9.13 _ECAT_Slave_CSP_Set_Softlimit

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Set_Softlimit(U16 CardNo, U16 AxisNo, U16 SlotNo, I32 PosiLimit, I32 NegaLimit, U16 Mode)

■ Purpose

This is for setting the software limit.

Note:

1. Function of software limit is only available in CSP mode.
2. Users can acquire the status of software limit of each axis via the API (_ECAT_Slave_CSP_Get_SoftLimit_Status) in section 9.31.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
PosiLimit	I32	Value	Set the value of software positive limit
NegaLimit	I32	Value	Set the value of software negative limit
Mode	U16	Option	1: Motor stops as soon as it reaches the limit 2: Motor decelerates to stop after reaching the limit (The deceleration time is 0.01 second).

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 PosiLimit =1000000, NegaLimit =-1000000, Mode =1;

Status = _ECAT_Slave_CSP_Set_Softlimit (CardNo, AxisNo, SlotNo, PosiLimit,
NegaLimit, Mode);
```

9.14 _ECAT_Slave_CSP_TargetPos_Change

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_TargetPos_Change (U16 CardNo, U16 AxisNo,
U16 SlotNo, I32 NewTargetCmd)

■ Purpose

This is for replacing the target position.

Note: This function is disabled during acceleration.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
NewTargetCmd	I32	Pulse	New target position

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 NewTargetCmd =2000000;

Status=_ECAT_Slave_CSP_TargetPos_Change (CardNo, AxisNo, SlotNo, NewTargetCmd);
```

9.15 _ECAT_Slave_CSP_Velocity_Change

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Velocity_Change (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewTargetSpd, F64 Tsec)

■ Purpose

This is for setting the new target speed of single axis.

Note:

1. This function is available when the current command running with constant speed.
2. This function can be carried out in mode 0 of _ECAT_Slave_CSP_Feedrate_Overwrite.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
NewTargetSpd	I32	Pulse / second	The new target speed
Tsec	F64	Second	The specified acceleration/deceleration time for speed changing.

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 NewTargetSpd =1000000;
F64 Tsec=0.1;

Status = _ECAT_Slave_CSP_Velocity_Change (CardNo, AxisNo, SlotNo,
NewTargetSpd,Tsec);
```

■ Description

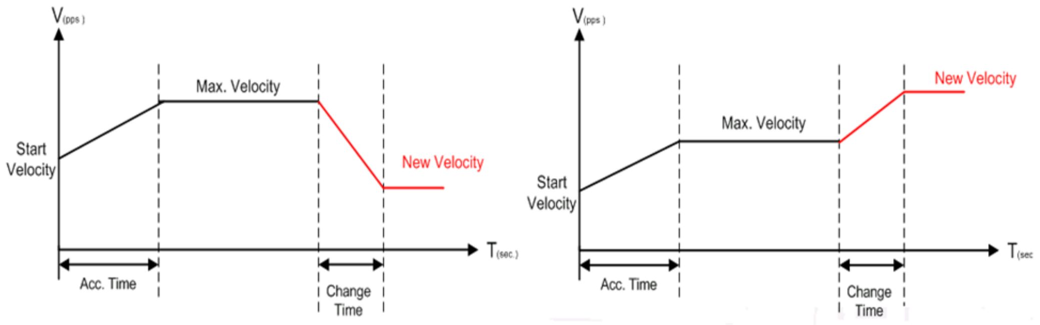


Figure 9.15.1 This function is identical to mode 0 of _ECAT_Slave_CSP_Feedrate_Overwrite

9.16 _ECAT_Slave_CSP_Feedrate_Overwrite

Syntax

U16 PASCAL _ECAT_Slave_CSP_Feedrate_Overwrite (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 Mode, I32 NewSpeed, F64 Tsec)

Purpose

This is used for the advanced setting of speed change for single axis. Multiple modes are provided.

Note: To change to the default speed after it is in mode 1 or mode 2, users have to set it to mode 0 again to and set NewSpeed to avoid sudden inintended acceleration.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: Change the vector velocity when the current motion is at constant speed. 1: The current speed and the speed in the later commands will be changed once the new speed is set. 2: The current speed and the speed in the later commands will be changed by the speed ratio once the new speed is set. The range is from 0% ~ 1000%.
NewSpeed	I32	Pulse / second	When Mode is set to 0 and 1, input the speed to be replaced. When Mode is set to 2, input the speed rator to be replaced, range from 0% ~ 1000.
Tsec	F64	Second	The specified acceleration/deceleration time for speed changing.

Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=0;
I32 NewSpeed =3000;
F64 Tsec=0.1;

Status = _ECAT_Slave_CSP_Feedrate_Overwrite (CardNo, AxisNo, SlotNo, Mode,
NewSpeed, Tsec);
```

- **Description**

Speed change function in mode 0 is identical to _ECAT_Slave_CSP_Velocity_Change. However, the change is valid only when the motion is at constant speed.

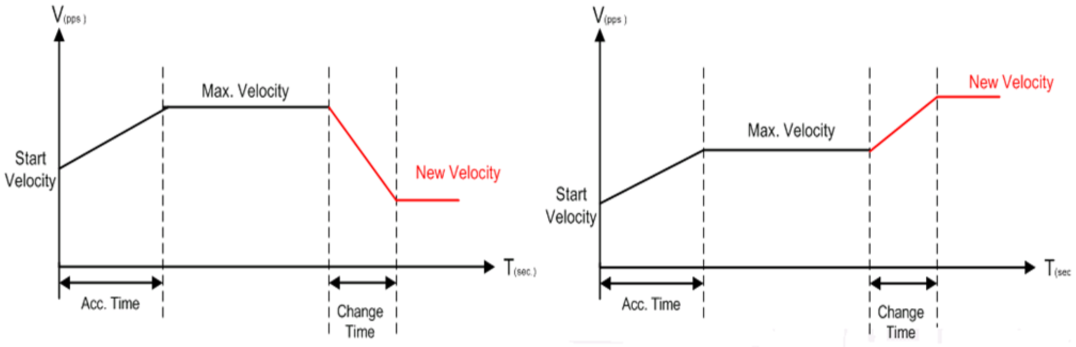


Figure 9.16.1 Set Mode to 0 to change the constant speed

In Mode 1, speed can be changed in any condition. And the speed of all CSP motion commands will also be changed once the NewSpeed command is issued.

Note: The multi-axis motion speed will be changed only when the first axis in the array is identical to the axis set in this API.

Example:

AxisNo = 1; NewSpeed = 100;

_ECAT_Slave_CSP_Feedrate_Overwrite (0, AxisNo, 0, 1, NewSpeed, 1);

AxisNoArray = {2, 1}; SlotID = {0, 0}; ConstVel = 100000;

_ECAT_Slave_CSP_Start_Arc_Move (0, AxisNoArray, SlotID, CenterPoint, 180, 0, ConstVel, 0, 1, 1, 1, 1);

The first axis of AxisNoArray is axis 2, which is different from the AxisNo. Thus, the constant speed of the arc motion will not be changed.

AxisNoArray = {1, 2}; SlotID = {0, 0}; ConstVel = 100000;

_ECAT_Slave_CSP_Start_Arc_Move (0, AxisNoArray, SlotID, CenterPoint, 180, 0, ConstVel, 0, 1, 1, 1, 1);

The first axis of AxisNoArray is axis 1, which is identical to AxisNo. Thus, the constant speed of the arc motion will be replaced with the value set in NewSpeed.

Note: if the speed interpolation of multi axes has been redefined by _ECAT_Slave_CSP_Liner_Speed_Master, the NewSpeed will change the redefined speed.

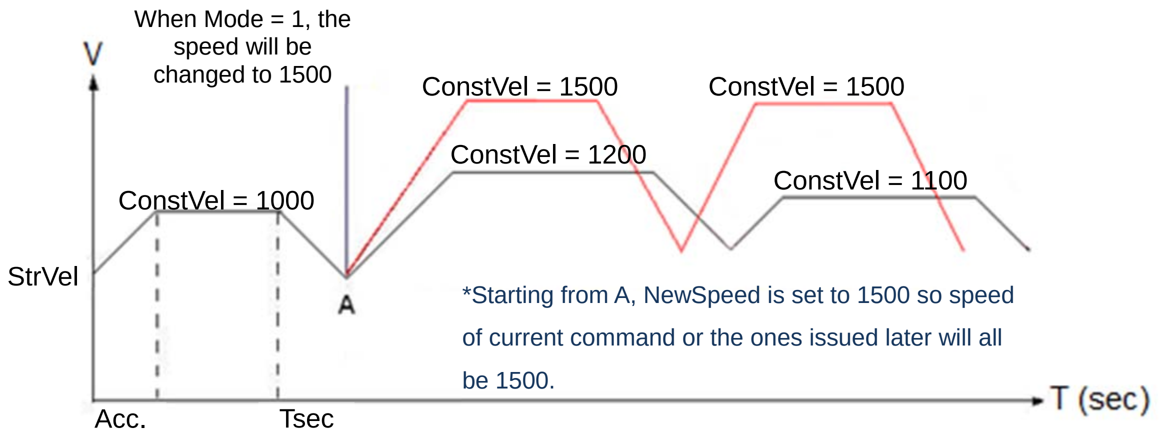


Figure 9.16.2 Set Mode to 1 to set the new constant speed for all commands

Function of speed change in Mode 2 is similar to Mode 1. However, the speed will be changed by speed ratio in Mode 2. The speed can be changed in any condition. And the speed of all CSP motion commands will also be changed once the NewSpeed command is issued.

Note: The multi-axis motion speed will be changed only when the first axis in the array is identical to the axis set in this API.

Example:

AxisNo = 1; NewSpeed = 10;

_ECAT_Slave_CSP_Feedrate_Overwrite (0, AxisNo, 0, 0, NewSpeed, 1);

AxisNoArray = {0, 1}; SlotID = {0, 0}; ConstVel = 100000;

_ECAT_Slave_CSP_Start_Arc_Move (0, AxisNoArray, SlotID, CenterPoint, 180, 0, ConstVel, 0, 1, 1, 1, 1);

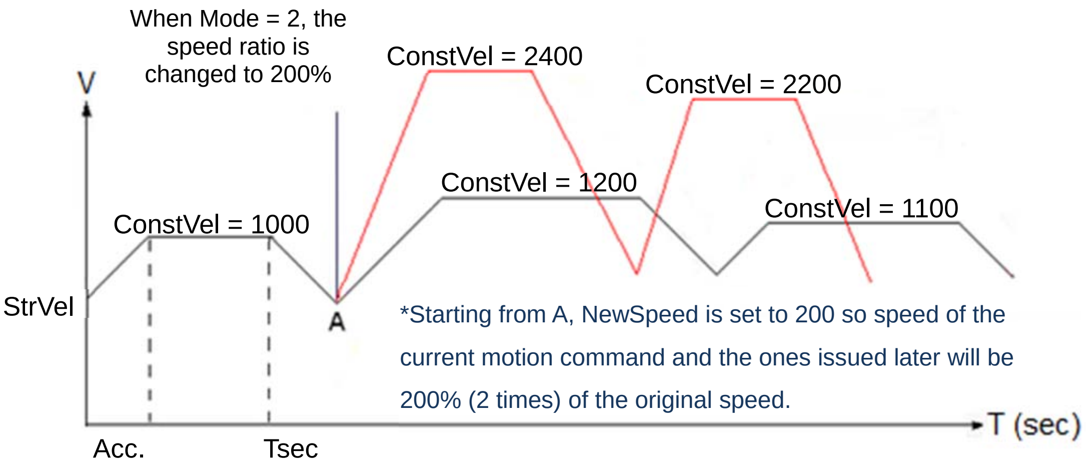
The first axis of AxisNoArray is axis 2, which is different from the AxisNo. Thus, the constant speed of the arc motion will not be changed.

AxisNoArray = {1, 2}; SlotID = {0, 0}; ConstVel = 100000;

_ECAT_Slave_CSP_Start_Arc_Move (0, AxisNoArray, SlotID, CenterPoint, 180, 0, ConstVel, 0, 1, 1, 1, 1);

The first axis of AxisNoArray is axis 1, which is identical to AxisNo. Thus, the constant speed of the arc motion will be replaced with the value set in NewSpeed.

Note: If the speed interpolation of multiple axes has been redefined by _ECAT_Slave_CSP_Liner_Speed_Master, the NewSpeed will change the redefined speed ratio.



*Starting from A, NewSpeed is set to 200 so speed of the current motion command and the ones issued later will be 200% (2 times) of the original speed.

Figure 9.16.3 Set Mode to 2 to change the speed ratio of constant speed for all commands

9.17 _ECAT_Slave_CSP_Speed_Continue_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for enabling or disabling the continuous speed function.

When this function is enabled, EtherCAT master will adjust the speed for all CSP motion commands of the specified axis. That is to say, the acceleration and deceleration of continuous commands will be adjusted by the ratio set in _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio. This can be used to prevent machine vibration.

Note: Like _ECAT_Slave_CSP_Feedrate_Overwrite, the multi-axis motion speed will be adjusted only when the first axis in the array is identical to the axis set in this API.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Disable continuous speed function 1: Enable continuous speed function

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Enable=1;

Status = _ECAT_Slave_CSP_Speed_Continue_Enable (CardNo, AxisNo, SlotNo, Enable);
```

9.18 _ECAT_Slave_CSP_Speed_Continue_Set_Mode

Syntax

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Set_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

Purpose

This is for setting the continuous speed mode that brings smooth operation.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: Keep the the acceleration/deceleration unchanged 1: Keep the time of acceleration/ deceleration unchanged 2: Keep the constant speed unchanged

Example

```
U16Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;

Status = _ECAT_Slave_CSP_Speed_Continue_Set_Mode (CardNo, AxisNo, SlotNo, Mode);
```

Description

When the user issues an invalid command, EtherCAT master will modify the command according to the set Mode.

For example: The moving distance is set to 1000 mm, constant speed is 20000 mm/s and the acceleration and deceleration time are both 0.1 second. If the motor needs to accelerate from 0 mm/s to 20000 mm/s in 0.1 second and then decelerates to 0 mm/s within 0.1 second, it requires setting its moving distance to 4000 mm at least. Therefore, setting the moving distance to 1000 mm is unreasonable. In this case, EtherCAT master provides three methods:

When Mode is set to 0:

If the moving distance and acceleration slope remain unchanged, users have to modify the constant speed and acceleration/deceleration time.

When Mode is set to 0, it will decrease the constant speed and acceleration /deceleration time in accordance with the proportion. And the command can be completed with the set acceleration. See figure 9.18.1. Black line shows the programmed path and the red line shows the actual path (1000 mm).

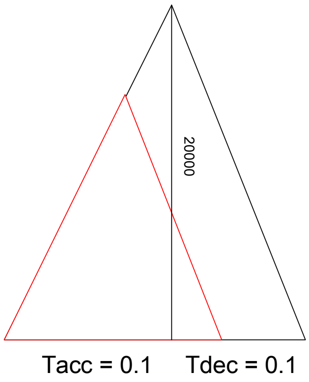


Figure 9.18.1

When Mode is set to 1:

If the moving distance and acceleration time remain unchanged, users have to modify the constant speed and acceleration/deceleration slope. When Mode is set to 1, the acceleration and deceleration time will remain 0.1 second, but the constant speed will be decreased. See figure 9.18.2. See figure 9.18.2. Black line shows the programmed path and the red line shows the actual path (1000 mm).

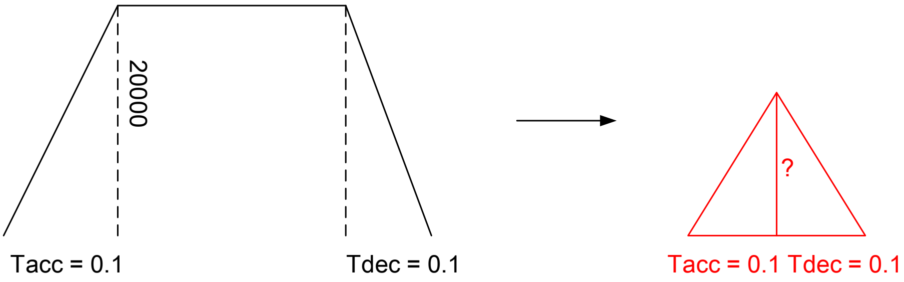


Figure 9.18.2

When Mode is set to 2:

If the moving distance and constant speed remain unchanged, users have to modify the acceleration/deceleration time and slope. When Mode is set to 2, the constant speed will remain, but the acceleration/deceleration time will be altered according to the actual application. See figure 9.18.3. Black line shows the programmed path and the red line shows the actual path (1000 mm).

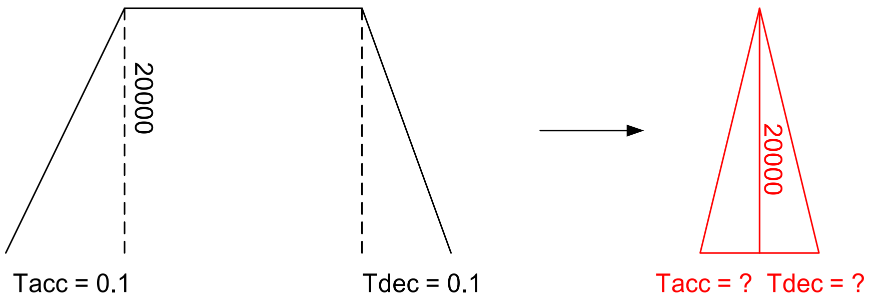


Figure 9.18.3

9.19 _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio (U16 CardNo , U16 AxisNo , U16 SlotNo , U16 Ratio)

■ Purpose

This is for setting the percentage of for starting blending speed of two commands.

■ Parameter

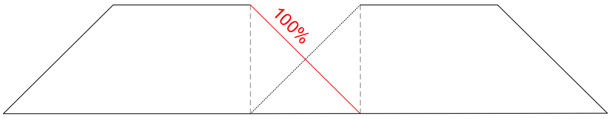
Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Ratio	U16	Percentage	Velocity blended percentage (Range: 0 ~ 100)

■ Example

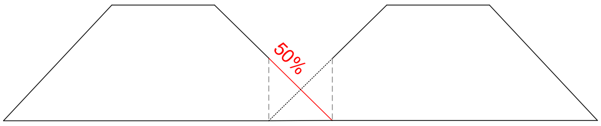
```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Ratio=100;

Status = _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio (CardNo, AxisNo,
SlotNo, Ratio);
```

■ Description

Set Ratio to 100:



Set Ratio to 50:



9.20 _ECAT_Slave_CSP_Scurve_Rate

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Scurve_Rate (U16 CardNo , U16 AxisNo ,
U16 SlotNo , U16 Ratio)

■ Purpose

This is for setting the ratio of T-curve and S-curve during acceleration and deceleration.

Note:

1. Once this function is enabled, it will change the S-curve ratio of CSP motion command. However, the setting of linear acceleration (T-curve) remains the same.
2. Like the function of _ECAT_Slave_CSP_Feedrate_Overwrite, multi-axis motion with continuous speed can be achieved only when the first axis in the array is identical to the axis set in this API.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Ratio	U16	Percentage	Ratio of linear acceleration (T-curve): Range: 0 ~ 100 Please refer to the following description for more information.

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Ratio =100;

Status = _ECAT_Slave_CSP_Scurve_Rate (CardNo, AxisNo, SlotNo, Ratio);
```

■ **Description**

Please refer to figure 9.20.1. When setting Ratio to 60, it means the T-curve ratio is 60%.The front part (20%) of the accerleration uses S-curve. The middle part (60%) applies T-curve for acceleration. And the rest (20 %) applies S-curve again. The configuration of deceleration is identical to the acceleration.

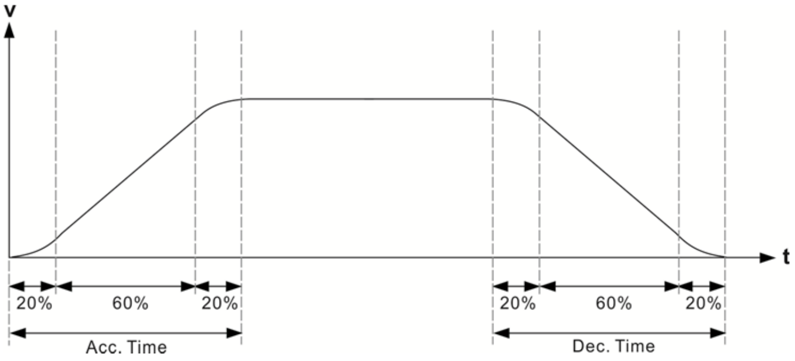


Figure 9.20.1 Setting of scurve_rate

9.21 _ECAT_Slave_CSP_Liner_Speed_Master

■ **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Liner_Speed_Master (U16 CardNo , U16 AxisNo , U16 SlotNo , U16 Mode)

■ **Purpose**

This is for setting the speed (vector) of advanced interpolation function.

Note: When enabling this function, the speed (vector) of all CSP commands will be changed.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID; it is valid only when Mode is set to 2
SlotNo	U16	Number	Slot ID; it is valid only when Mode is set to 2
Mode	U16	Option	It defines the speed of all CSP command. 0: Speed (vector) (Default) 1: Speed (vector) for the longest moving distance 2: Speed (vector) for the specified axis

- **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;

Status = _ECAT_Slave_CSP_Liner_Speed_Master (CardNo, AxisNo, SlotNo, Mode);
```

- **Description**

Mode parameter setting:

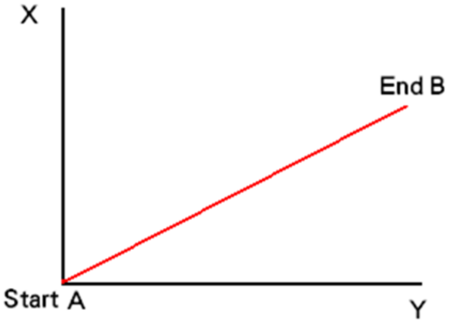


Figure 9.21.1 When Mode = 0, the command will change the resultant velocity of X and Y.

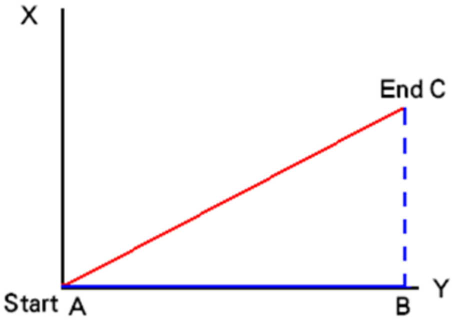


Figure 9.21.2

When Mode = 1, it changes the speed of the axis that is farther from the origin. In this figure, it changes the speed (vector) of Y-axis. And the system will automatically calculate the speed of X-axis.

When Mode = 2, it changes the speed of the specified axis (Y-axis) and automatically calculated the speed of another one (X-axis).

Note: When selecting Mode 2, the multi-axis motion speed will be changed only when the first axis in the array is identical to the axis set in this API, which is the same as the API (_ECAT_Slave_CSP_Feedrate_Overwrite) in section 9.16.

9.22 _ECAT_Slave_CSP_Mask_Axis

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Mask_Axis(U16 CardNo , U16 AxisNo , U16 SlotNo , U16 Mode)

■ Purpose

When multi-axis motion command is being executed, this API can be used to stop the specified axes without influencing others.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	Quantity	Axis No. to be stopped
AxisArray	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotArray	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to AxisNum

■ Example

```
U16 Status;
U16 CardNo=0, MoveAxisNum =3, MoveAxisArray[3]={1,2, 3}, MoveSlotArray[3]={0, 0, 0};
U16 StopAxisNum = 2, StopAxisArray[2]={2, 3}, StopSlotArray[2]={0, 0};
I32 DistArrary[3]= {100000,200000, 300000}, StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
U16 Scurve =0, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Multiaxes_Move(CardNo, MoveAxisNum,
MoveAxisArray, MoveSlotArray, DistArrary, Strvel, ConstVel, EndVel, TPhase1, TPhase2,
Scurve, Abs_Rel);
Status = _ECAT_Slave_CSP_Mask_Axis(CardNo, StopAxisNum, StopAxisArray,
StopSlotArray)
```

9.23 _ECAT_Slave_CSP_Sync_Config

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Sync_Config(U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, U16 Enable)

■ Purpose

This is for setting the function of synchronous motion of multiple axes. The function allows the EtherCAT master issue up to 20 CSP motion commands at the same time.

Note:

1. When some commands are issued to the the same axis, they will be executed according to the issuing sequence.
2. Before starting the synchronous motion control, users should complete the setting of this API to enable the function and issue the CSP motion commands one by one. Then, apply the API "_ECAT_Slave_CSP_Sync_Move" to start executing all CSP commands.

■ Example

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	Quantity	Axis number to be enabled
AxisArray	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotArray	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
Enable	U16	Option	Function of synchronous motion of multiple axes 0: Disable 1: Enable

■ Example

```
U16 Status;
U16 CardNo = 16, AxisNum = 2, AxisArray[2] = {0, 1}, SlotArray[2] = {0, 0};
U16 Enable = 1;

Status = _ECAT_Slave_CSP_Sync_Config (CardNo, AxisNum, AxisArray, SlotArray,
Enable);
```

9.24 _ECAT_Slave_CSP_Sync_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Sync_Move (U16 CardNo)

■ Purpose

This is for enabling the function of synchronous motion of multiple axes. The function allows the EtherCAT master issue up to 20 CSP motion commands at the same time.

Note:

1. When some commands are issued to the the same axis, they will be executed by following the issuing sequence.
2. Users should complete the setting of API "_ECAT_Slave_CSP_Sync_Config" to enable the function and issue the CSP motion commands one by one. Then, apply the API " _ECAT_Slave_CSP_Sync_Move" to start executing all CSP commands.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 Status;
U16 CardNo=16;

Status = _ECAT_Slave_CSP_Sync_Move (CardNo);
```

9.25 _ECAT_Slave_CSP_Start_Mabrline_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_Mabrline_Move (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, I32 *Target1Point, I32 *Target2Point, I32 StrVel, I32 First_ConstVel, I32 Second_ConstVel, I32 EndVel, F64 Tacc_Step1, F64 Tacc_Step2,U16 Abs_Rel)

■ Purpose

This is for setting to smooth the operation of point-to-point motion of multiple axes.

Note:

1. This API function is similar to the API "_ECAT_Slave_CSP_Start_Msbrline_Move" (section 9.11). The difference between them is that this API will automatically calculate the corner speed based on the set accerlation and deceleration.
2. The corner speed of this API is identical to the API "_ECAT_Slave_CSP_Speed_Continue_Enable" (section 9.17). However, it takes effect only in the path that is being executed by this API.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	Quantity	Axis number to be used for synchronous motion control
AxisArray	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotArray	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
Target1Point	I32*	Array of pulse for each axis	First position; the array number should equal to the axis number Target1Point[0] stores the first point of the first axis. Target1Point[1] stores the first point of the second axis ….

Name	Data type	Property	Description
Target2Point	I32*	Array of pulse for each axis	End point; the array number should equal to the axis number Target2Point[0] stores the first point of the first axis. Target2Point[1] stores the first point of the second axis ….
StrVel	I32	Pulse / second	Parameter of the motion initial speed
First_ConstVel	I32	Pulse / second	The constant speed from the start position to Target1Point
Second_ConstVel	I32	Pulse / second	The constant speed from the Target1Point to Target2Point
EndVel	I32	Pulse / second	Parameter of the motion end speed
Tacc_Step1	F64	Time (second)	The time it spent to change from First_ConstVel to 0
Tacc_Step2	F64	Time (second)	The time it spent to change from Second_ConstVel to 0
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

- **Example**

```
U16 Status;
U16 CardNo = 16, AxisNum = 2, AxisArray[2] = {0, 1}, SlotArray[2] = {0, 0};
I32 Target1Point[2] = {20000, 40000};
I32 Target2Point[2] = {40000, 80000};
I32 StrVel = 0, First_ConstVel = 100000, Second_ConstVel = 200000, EndVel = 0;
F64 Tacc_Step1 = 0.1, TaccStep2 = 0.1;
U16 Abs_Rel = 1;

Status = _ECAT_Slave_CSP_Start_Mabrline_Move (CardNo, AxisNum, AxisArray,
SlotArray, Target1Point, Target2Point, StrVel, First_ConstVel, Second_ConstVel, EndVel,
Tacc_Step1, Tacc_Step2, Abs_Rel);
```

9.26 _ECAT_Slave_CSP_Start_2Segment_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_2Segment_Move(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 SegMode, I32 Dist, I32 Dist2, I32 StrVel, I32 MaxVel, I32 MaxVel2, I32 EndVel, F64 Tacc, F64 Tsec, F64 Tdec, U16 Scurve, U16 Abs_Rel)

■ Purpose

This is for setting the single-axis linear motion by specifying two distances and speed.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Axis ID
SlotNo	U16	Number	Slot ID
SegMode	U16	Option	Segment mode: 0: The motion axis completes the 1st distance first and then accelerates/decelerates to start the 2nd distance. 1: While the 1st distance is not completed, the motion axis accelerates/decelerates to start the 2nd distance.
Dist	I32	Pulse	The 1st moving distance
Dist2	I32	Pulse	The 2nd moving distance
StrVel	I32	Pulse / second	Initial speed for the 1st moving distance
MaxVel	I32	Pulse / second	The maximum speed for the 1st moving distance
MaxVel2	I32	Pulse / second	The maximum speed for the 2nd moving distance
EndVel	I32	Pulse / second	End speed for the 2nd moving distance
Tacc	F64	second	Acceleration time of the 1st moving distance
Tsec	F64	second	Acceleration/deceleration time when switching to travel the 2nd moving distance
Tdec	F64	second	Deceleration time of the 2nd moving distance
Scurve	U16	Option	1: T-curve (default) 2: S-curve
Abs_Rel	U16	Option	0: Relative movement (default) 1: Absolute movement

■ Example

```
U16 Status;
```

```
U16 CardNo = 16, AxisNo=0, SlotNo=0, SegMode=0, Scurve=0, Abs_Rel=0;
I32 Dist=0, Dist2=0, StrVel=0, MaxVel=0, MaxVel2=0, EndVel=0;
F64 Tacc=0, Tsec=0, Tdec=0;

Status = _ECAT_Slave_CSP_Start_2Segment_Move (CardNo, AxisNo, SlotNo, SegMode,
Dist, Dist2, StrVel, MaxVel, MaxVel2, EndVel, Tacc, Tsec, Tdec, Scurve, Abs_Rel);
```

- **Description**

When parameter SegMode is set to 0, the motion axis will start the 1st distance first and then accelerates/decelerates to start the 2nd distance.

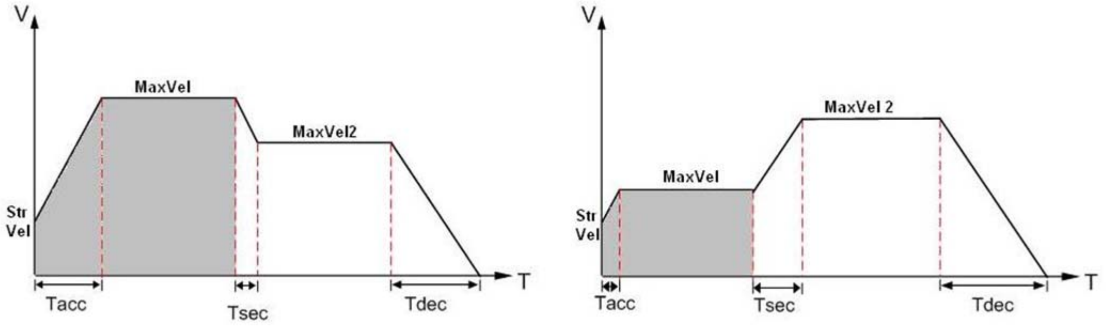


Figure 9.26.1 Time-Velocity chart of relative motion

(The gray section represents the first moving distance and the white section signifies the second moving distance)

When parameter SegMode is set to 1, the motion axis accelerates/decelerates to start the 2nd distance while the 1st moving distance is not completed yet.

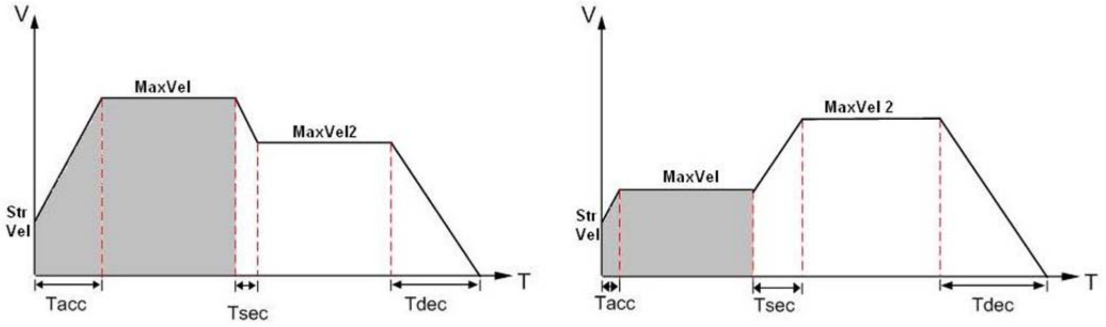


Figure 9.26.2 Time-Velocity chart of relative motion

(The gray section represents the first moving distance and the white section signifies the second moving distance)

Note: Please note that when SegMode is set to 0, the direction of the first and second moving distance must be the same. See the example of incorrect setting in figure 9.26.3

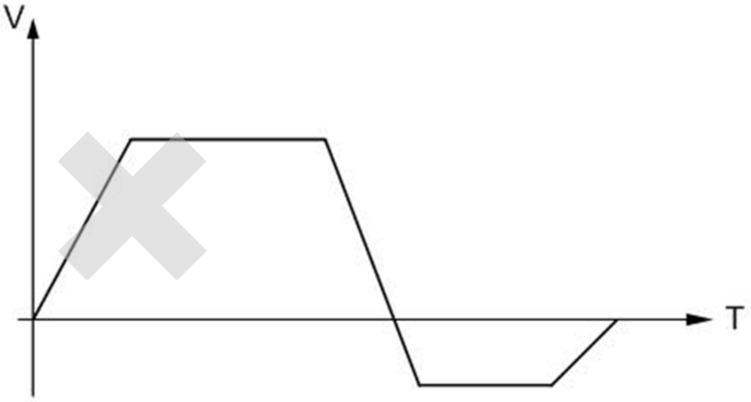


Figure 9.26.3 Incorrect setting – The direction of the first moving distance is not the same as the second moving distance.

9.27 _ECAT_Slave_CSP_Start_PVT_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Start_PVT_Move (U16 CardNo, U16 NodeID, U16 SlotID, I32 DataCnt, I32 *TargetPos, I32 *TargetTime, I32 *TargetVel)

■ Purpose

This is for setting single-axis motion to move to multiple target points at fixed time.

The axis will move to the target positions (Max.: 8000) within the set time. And EtherCAT master automatically calcaultes the acceleration and deceleration during operation.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotID	U16	Number	Slot ID
DataCnt	I32	Amount	Number of target positions (Max. 8000)
TargetPos	I32*	Array of pulse	Target position
TargetTime	I32*	Array of milisecond	Target time
TargetVel	I32*	Array of velocity	Target speed (Unit: pulse / second)

- **Example**

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotID=0;
I32 DataCnt=3, TargetPos[3]={0, 20000, 30000}, TargetTime[3]={0, 1000, 2000},
TargetVel[3]={0,11000, 0};
Status = _ECAT_Slave_CSP_Start_PVT_Move (CardNo, NodeID, SlotID, DataCnt,
TargetPos, TargetTime, TargetVel);
```

9.28 _ECAT_Slave_CSP_Start_PVTComplete_Move

■ **Syntax**

U16 PASCAL _ECAT_Slave_CSP_Start_PVTComplete_Move (U16 CardNo, U16 NodeID, U16 SlotID, I32 DataCnt, I32 *TargetPos, I32 *TargetTime, I32 StrVel, I32 EndVel);

■ **Purpose**

This is for specifying the initial speed and end speed of the single-axis motion, moving through multiple points at fixed time.

This API function is similar to _ECAT_Slave_CSP_Start_PVT_Move. Users can use this API to define the initial and end speed. And the axis will move to the target positions (Max.: 8000) within the set times. Its acceleration and deceleration are calculated by the EtherCAT master.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotID	U16	Number	Slot ID
DataCnt	I32	Amount	Number of target positions (Max.: 8000)
TargetPos	I32*	Array of pulse	Target position
TargetTime	I32*	Array of milisecond	Target time
StrVel	I32	Pulse / second	Initial speed
EndVel	I32	Pulse / second	End speed

- **Example**

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotID=0;
I32 DataCnt=4, TargetPos[4]={0, 20000, 30000, 40000}, TargetTime[4]={0, 4000, 10000, 15000},
StrVel=10000, EndVel=0;

Status = _ECAT_Slave_CSP_Start_PVTComplete_Move (CardNo, NodeID, SlotID, DataCnt,
TargetPos, TargetTime, StrVel, EndVel);
```

9.29 _ECAT_Slave_CSP_Virtual_Set_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Virtual_Set_Enable(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for enabling function of virtual position.

After enabling the function of virtual position, the specified position (acquired by _ECAT_Slave_Motion_Get_Position in section 8.5) will be changed to the vitual position from the motor's feedback position.

Since EtherCAT master will do the compensation for the specified axis, the motor's feedback position will be slightly different from the machine's actual position (virtual position) when applying the function of interval compensation and E-cam.

To acquire the motor's actual feedback position, please use the API (_ECAT_Slave_Motion_Get_Actual_Position) in section 8.13.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Axis ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	Enable the function of virtual position 0: Disable 1: Enable

■ Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Enable=1;

Status = _ECAT_Slave_CSP_Virtual_Set_Enable(CardNo, AxisNo, SlotNo, Enable);
```

9.30 _ECAT_Slave_CSP_Virtual_Set_Command

■ Syntax

```
U16 PASCAL _ECAT_Slave_CSP_Virtual_Set_Command(U16 CardNo, U16 AxisNo,
U16 SlotNo, I32 Command)
```

■ Purpose

This is for setting the virtual position and replacing the current position with the specified position.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Axis ID
SlotNo	U16	Number	Slot ID
Command	I32	Value	The specified position

■ Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Command=0;

Status = _ECAT_Slave_CSP_Virtual_Set_Command(CardNo, AxisNo, SlotNo, Command);
```

9.31 _ECAT_Slave_CSP_Get_SoftLimit_Status

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Get_SoftLimit_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Status)

■ Purpose

This is for acquiring the status of software limit.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Status	U16*	Value	Acquire the axis position when it reaches negative or positive limit. Bit 0: Negative limit Bit 1: Positive limit

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotNo=0, Status=0;

Status = _ECAT_Slave_CSP_Get_SoftLimit_Status (CardNo, NodeID, SlotNo, &Status);
```

9.32 _ECAT_Slave_CSP_Pitch_Set_Interval

Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Interval(U16 CardNo, U16 AxisNo,
U16 Slot No, I32 Interval)

Purpose

This is for setting the interval of the pitch error compensation.

EtherCAT master executes the position compensation with the set interval, so that the machine moves in the correct coordinates system.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Interval	I32	Value	It sets the pulse for compensation interval

Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Interval=0

Status = _ECAT_Slave_CSP_Pitch_Set_Interval(CardNo, AxisNo, SlotNo, Interval);
```

9.33 _ECAT_Slave_CSP_Pitch_Set_Mode

Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Mode(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

Purpose

This is for setting the mode of pitch error compensation.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Mode	Compensation mode: 0: Single-direction compensation 1: Dual-direction compensation

Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Mode=0;

Status = _ECAT_Slave_CSP_Pitch_Set_Mode(CardNo, AxisNo, SlotNo, Mode);
```

9.34 _ECAT_Slave_CSP_Pitch_Set_Org

Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Org(U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 Dir, I32 OrgPos)

Purpose

This is for setting the start position of pitch error compensation.

EtherCAT master starts to do the compensation when it reaches the start position.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Dir	U16	Option	Set the start position in positive/ negative direction 0: The start position in positive direction 1: The start position in negative direction
OrgPos	I32	Value	Start position of pitch error compensation

Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0;
I32 OrgPos=0;

Status = _ECAT_Slave_CSP_Pitch_Set_Org(CardNo, AxisNo, SlotNo, Dir, OrgPos);
```

9.35 _ECAT_Slave_CSP_Pitch_Set_Rel_Table

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Rel_Table(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32* Table, U16 Num)

■ Purpose

This is for setting the relative position of each interval for pitch error compensation.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Dir	U16	Option	It sets the compensation value in positive/negative direction. 0: Set the compensation value for positive direction 1: Set the compensation value for negative direction
Table	I32*	Array	Compensation array of each interval (relative position)
Num	U16	Value	Amount in the compensation array. Max. value: 10000

■ Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0;
I32 Table[3] = {0, 1, 2};

Status = _ECAT_Slave_CSP_Pitch_Set_Rel_Table (CardNo, AxisNo, SlotNo, Dir, &Table,
Num);
```

9.36 _ECAT_Slave_CSP_Pitch_Set_Abs_Table

Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Abs_Table(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32* Table, U16 Num)

Purpose

This is for setting the absolute position of each interval for pitch error compensation.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Dir	U16	Option	It sets the compensation value in positive/negative direction. 0: Set the compensation value for positive direction 1: Set the compensation value for negative direction
Table	I32*	Array	Compensation array of each interval (absolute position)
Num	U16	Value	Amount of compensation array. Max. value: 10000

Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0, Num=0;
I32 Table[3]={0, 1, 2};

Status = _ECAT_Slave_CSP_Pitch_Set_Abs_Table (CardNo, AxisNo, SlotNo, Dir, &Table,
Num);
```

9.37 _ECAT_Slave_CSP_Pitch_Set_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Enable(U16 CardNo, U16 AxisNo ,
U16 SlotNo, U16 Enable)

■ Purpose

This is for enabling the pitch error compensation.

EtherCAT master will carry out the compensation in accordance with the interval set by _ECAT_Slave_CSP_Pitch_Set_Interval (section 9.32) and the relative position set by _ECAT_Slave_CSP_Pitch_Set_Rel_Table (section 9.35) or the absolute position set by _ECAT_Slave_CSP_Pitch_Set_Abs_Table (section 9.36).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNum	U16	Option	Enable the function of pitch error compensation 0: Disable 1: Enable

■ Example

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Enable=1

Status = _ECAT_Slave_CSP_Pitch_Set_Enable(CardNo, AxisNo, SlotNo, Enable);
```

Cyclic Synchronous Velocity Mode (CSV) 10

This chapter introduces the application of APIs in CSV mode. Unlike PV mode, EtherCAT master issues one speed command every communication cycle in CSV mode. That is, the motion speed is controlled by EtherCAT master. And CSV mode also provides functions of single-axis and multi-axis motion control with the setting speed.

In CSV (Cyclic Synchronous Velocity) mode, it issues the speed command via PDO communication. EtherCAT master will calculate all speed commands for the next communication cycle after analyzing the speed and acceleration set in the API. Then, it sends the new command to all motion axes every communication cycle so that single axis or multiple axes can operate with the setting speed.

API list of CSV mode

Function name	Description
_ECAT_Slave_CSV_Start_Move	Execute single-axis motion with the setting speed
_ECAT_Slave_CSV_Multi_Start_Move	Execute multi-axes synchronous motion with the setting speed

10.1 _ECAT_Slave_CSV_Start_Move

- **Syntax**

U16 PASCAL _ECAT_Slave_CSV_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,I32 Target_Velocity, F64 Acceleration, U16 Curve_Mode, U16 Acc_Type)

- **Purpose**

This is for single-axis motion control with the setting speed.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Target_Velocity	I32	inc/s	Target speed inc signifies the unit set in the device. Please refer to the manual of the slave device for more details. (OD: 0x60FF Sub 0)
Acceleration	F64	second Pulse / s^2	Time to reach the target speed (inc/s^2) inc signifies the unit set in the device. Please refer to the manual of the slave device for more details. (OD: 0x6083 Sub 0)
Curve_Mode	U16	Option	1: T-curve (Default) 2: S-curve

Name	Data type	Property	Description
Acc_Type	U16	Option	Acceleration unit: 0: Second 1: inc/ s^2

■ **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Curve_Mode = 1, Acc_Type = 0;
I32 Target_Velocity =600000;
F64 Acceleration = 0.2;

Status = _ECAT_Slave_CSV_Start_Move (CardNo, AxisNo, SlotNo, Target_Velocity,
Acceleration, Curve_Mode, Acc_Type);
```

10.2 _ECAT_Slave_CSV_Multi_Start_Move

■ **Syntax**

U16 PASCAL _ECAT_Slave_CSV_Multi_Start_Move (U16 CardNo, U16 AxisNum,
U16 *AxisNo, U16 *SlotNo, I32 *Target_Velocity, F64 *Acceleration, U16 Curve_Mode,
U16 Acc_Type)

■ **Purpose**

This is for multi-axes synchronous motion control with the setting speed.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	Quantity	Quantity of the axes for synchronous motion control. Refer to section 2.1 for information about suggested maximum axis number.
AxisNo	U16*	Array	Array of node ID
SlotNo	U16*	Array	Array of slot ID
Target_Velocity	I32*	inc/sec	Array of target speed inc signifies the unit set in the device. Please refer to the manual of the slave device for more details. (OD: 0x60FF Sub 0)

Name	Data type	Property	Description
Acceleration	F64*	second Inc / s^2	Array of time to reach the target speed (inc/s^2) inc signifies the unit set in the device. . Please refer to the manual of the slave device for more details. (OD: 0x6083 Sub 0)
Curve_Mode	U16	Option	1: T-curve(Default) 2: S-curve
Acc_Type	U16	Option	Acceleration unit: 0: Second 1: inc / s^2

- **Example**

```
U16 Status;
U16 CardNo=16, AxisNum = 2;
U16 AxisNo[2]={1, 2}, SlotNo[2]={0, 0}, Curve_Mode = 1, Acc_Type = 0;
I32 Target_Velocity[2] ={600000, 600000};
F64 Acceleration[2] = {0.2, 0.2};

Status = _ECAT_Slave_CSV_Multi_Start_Move (CardNo, AxisNum, &AxisNo, &SlotNo,
&Target_Velocity, &Acceleration, Curve_Mode, Acc_Type);
```

Cyclic Synchronous Torque Mode (CST) 11

This chapter introduces the APIs in CST mode. Different from PT (Profile Torque) mode, EtherCAT master issues one torque command in each communication cycle in CST mode. That is, the torque of all axes is controlled by EtherCAT master. And CST mode also provides function of single-axis and multi-axis torque control.

In CST (Cyclic Synchronous Torque) mode, it issues the torque command via PDO communication. EtherCAT master will calculate all torque commands for the next communication cycle after analyzing the speed and acceleration set in the API. Then, it sends the new command to all motion axes in each communication cycle so that single axis or multiple axes can operate with the setting torque.

API list of CST mode

Function name	Description
_ECAT_Slave_CST_Start_Move	Execute single-axis motion with the setting torque
_ECAT_Slave_CST_Multi_Start_Move	Execute multi-axis synchronous motion with the setting torque

11.1 _ECAT_Slave_CST_Start_Move

- **Syntax**

U16 PASCAL _ECAT_Slave_CST_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,I16 Target_Torque, U32 Slope, U16 Curve_Mode)

- **Purpose**

This is for executing single-axis motion l with the setting torque.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Target_Torque	I16	Permillage	Target torque, the permillage of the maximum torque. Setting range: 1 ~ 1000.
Slope	U32	0.1%/s	Set the torque’s rising slope
Curve_Mode	U16	Option	1: T-curve (Default) 2: S-curve

- **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Curve_Mode = 1;
I16 Target_Torque = 2;
U32 Slope = 10;

Status = _ECAT_Slave_CST_Start_Move (CardNo, AxisNo, SlotNo, Target_Torque, Slope,
Curve_Mode);
```

11.2 _ECAT_Slave_CST_Multi_Start_Move

■ Syntax

U16 PASCAL _ECAT_Slave_CST_Multi_Start_Move (U16 CardNo, U16 AxisNum, U16 *AxisNo, U16 *SlotNo, I16 * Target_Torque, U32 *Slope, U16 Curve_Mode)

■ Purpose

This is for executing multi- axes synchronous motion with the setting torque.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNum	U16	Quantity	Quantity of the axes for synchronous motion control. Refer to section 2.1 for information about suggested maximum axis number.
AxisNo	U16*	Array for node ID	Array of node ID
SlotNo	U16*	Array for Slot ID	Array of slot ID
Target_Torque	I16*	Array of target torque	Array of the target torque, the permillage of the maximum torque. Setting range: 1~1000
Slope	U32*	0.1%/s	Array of the torque's rising slope
Curve_Mode	U16	Option	1: T-curve (Default) 2: S-curve

■ Example

```
U16 Status;
U16 CardNo=16, AxisNum = 2;
U16 AxisNo[2]={1, 2}, SlotNo[2]={0, 0}, Curve_Mode = 1, Acc_Type = 0;
I16 Target_Torque[2] ={2, 2};
U32 Slope[2] = {10, 10};

Status = _ECAT_Slave_CST_Multi_Start_Move (CardNo, AxisNum, &AxisNo, &SlotNo,
&Target_Velocity, &Slope, Curve_Mode);
```

(This page is intentionally left blank.)

Homing 12

This chapter elaborates the APIs for single-axis homing with 35 modes available. The motion axis will start homing according to the received command set by users.

API list of homing

Function name	Description
_ECAT_Slave_Home_Config	Set the homing mode
_ECAT_Slave_Home_Move	Execute homing
_ECAT_Slave_Home_Status	Acquire the current homing status

12.1 _ECAT_Slave_Home_Config

- **Syntax**

U16 PASCAL _ECAT_Slave_Home_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode,I32 Offset, U32 FirstVel, U32 SecondVel, U32 Acceleration)

- **Purpose**

This is for setting the homing mode. Executing this API will not start the homing procedure. To execute homing, use API "_ECAT_Slave_Home_Move" (section 12.2)

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Mode	Homing mode Please refer to the manual of the slave device for more details. (OD: 0x6098 Sub 0)
Offset	I32	inc	Homing offset inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607C Sub 0)
FirstVel	U32	inc/second	Speed for searching the ORG reference signal (The 1st speed) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6099 Sub 1)
SecondVel	U32	inc/second	Speed for searching the Z pulse (The 2nd speed) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD code: 0x6099 Sub 2)
Acceleration	U32	inc/s^2	Acceleration during homing (inc/s^2) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD code: 0x609A Sub 0)

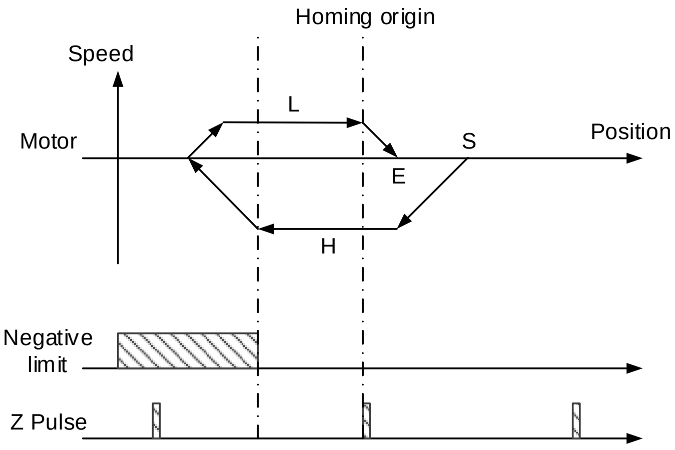
■ **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;
I32 Offset=200;
U32 FirstVel =600000, SecondVel =100000;
U32 Acceleration = 3;

/*Homing mode setting*/
Status = _ECAT_Slave_Home_Config (CardNo, AxisNo, SlotNo, Mode, Offset, FirstVel,
SecondVel, Acceleration);
```

■ **Description**

1. **Mode 1**

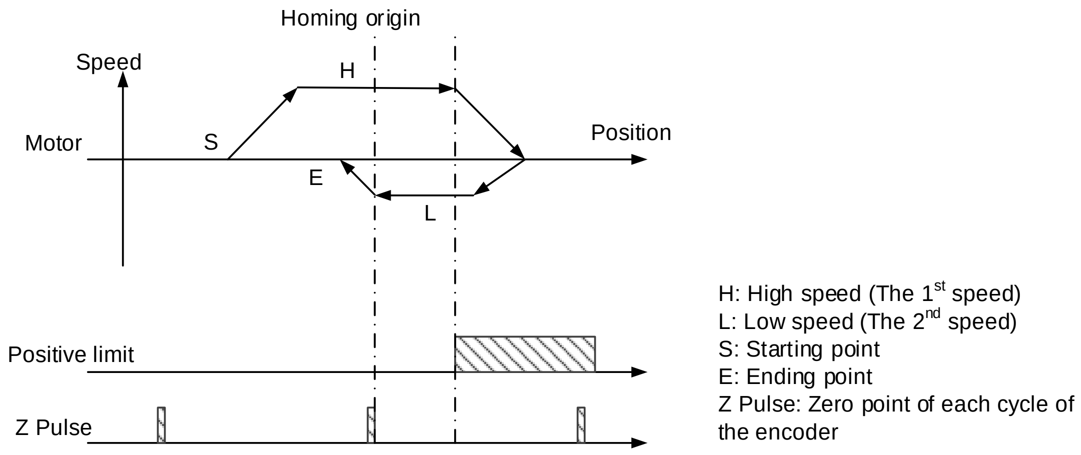
The motor runs in reverse direction at high speed until it reaches the negative limit. Once reaching the limit, it decelerates. Upon leaving the negative limit at low speed, it starts to look for the first Z pulse in forward direction and regards the first Z pulse as the new homing origin.



H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

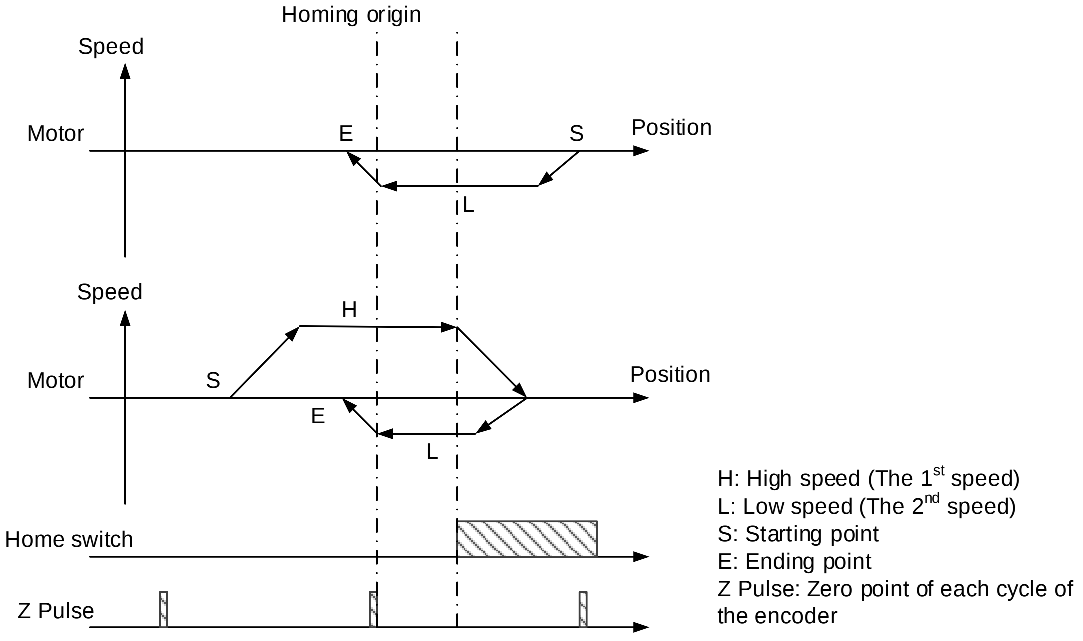
2. Mode 2

The motor runs in forward direction at high speed and starts to decelerate once it reaches the positive limit. When the motor leaves the positive limit at low speed, it starts to look for the first Z pulse in reverse direction and regards the first Z pulse as the new homing origin.


Homing origin
Speed
H
Motor
S
Position
E
L
Positive limit
Z Pulse
H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

3. Mode 3

- Home switch ON: The motor runs in reverse direction at low speed until it reaches the home switch. Once reaching the home switch, it decelerates to leave the switch with low speed and starts to look for the first Z pulse, regarding it as the new homing origin.
- Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, it decelerates to leave the home switch in reverse direction and starts to look for the first Z pulse with low speed and regards the first Z pulse as the new homing origin.

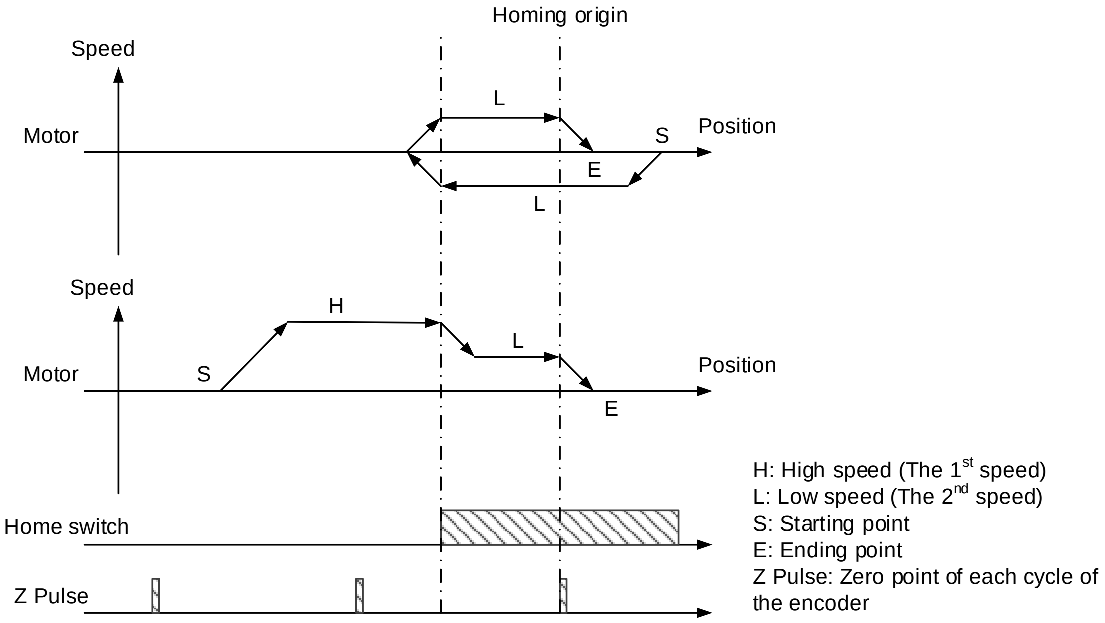


4. **Mode 4**

Mode 4 is similar to Mode 3. The only difference is the moving direction after the motor detects that the home switch has changed its state.

Home switch ON: The motor runs in reverse direction at low speed until it leaves the home switch. Then, the motor reaches the home switch again in forward direction. When the motor reaches the home switch again, it will regard the first Z pulse as the new homing origin.

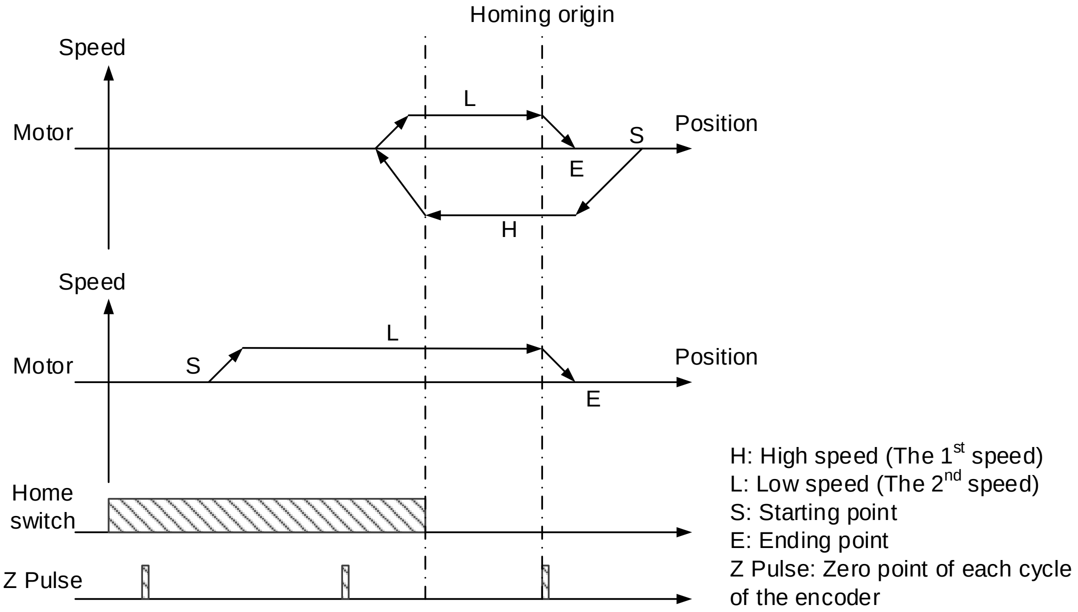
■ Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, the motor decelerates and runs at low speed and regards the first Z pulse it looked for as the new homing origin.



5. **Mode 5**

Mode 5 is similar to similar to mode 3 but with different initial moving directions.

- Home switch OFF: The motor runs in reverse direction at high speed until it reaches the home switch. Then, the motor decelerates in forward direction. When the motor leaves the switch at low speed and looks for the first Z pulse, it regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, it looks for the first Z pulse and regards the first Z pulse as the new homing origin.

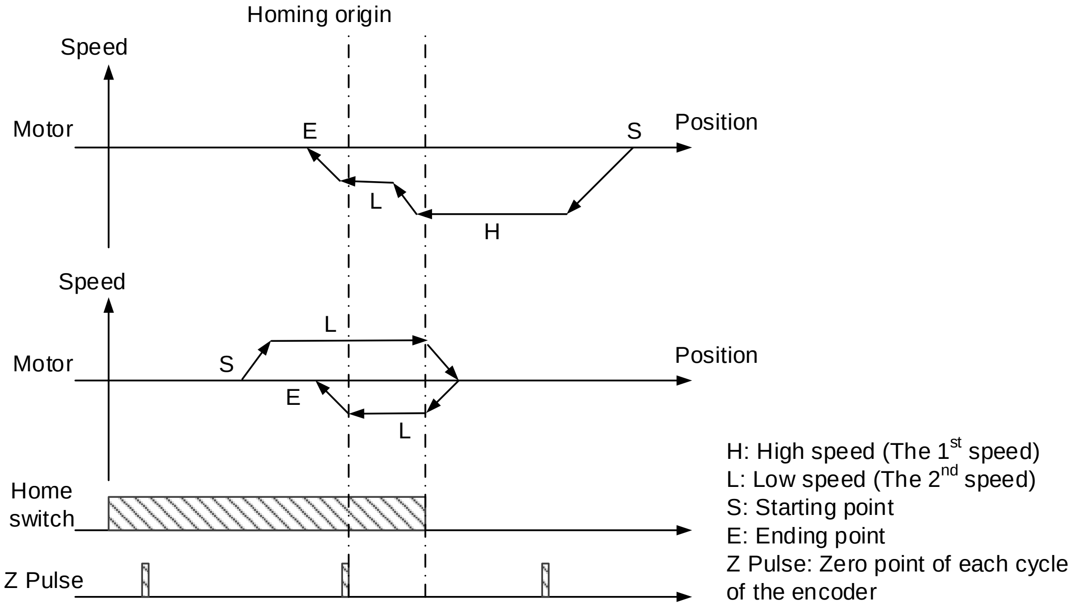


H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

6. Mode 6

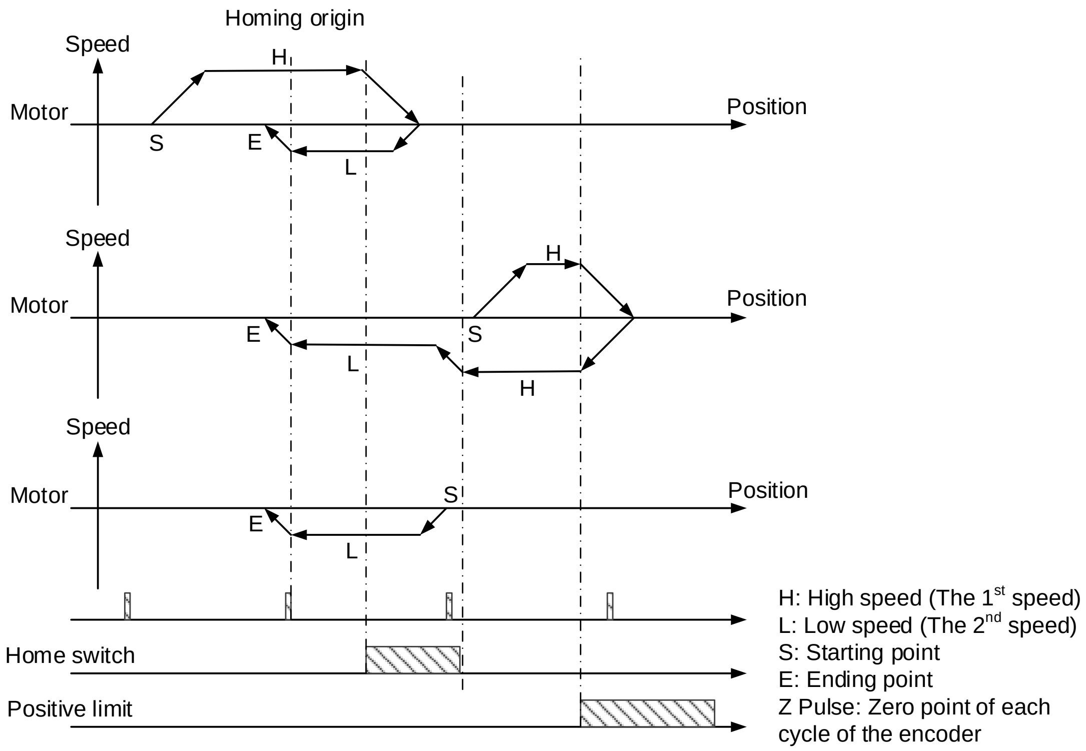
Mode 6 is similar to mode 4 but with different initial moving directions.

- Home switch OFF: The motor runs in reverse direction at high speed until it reaches the home switch. Then, the motor runs at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, the motor runs in reverse direction. When the motor reaches the home switch again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



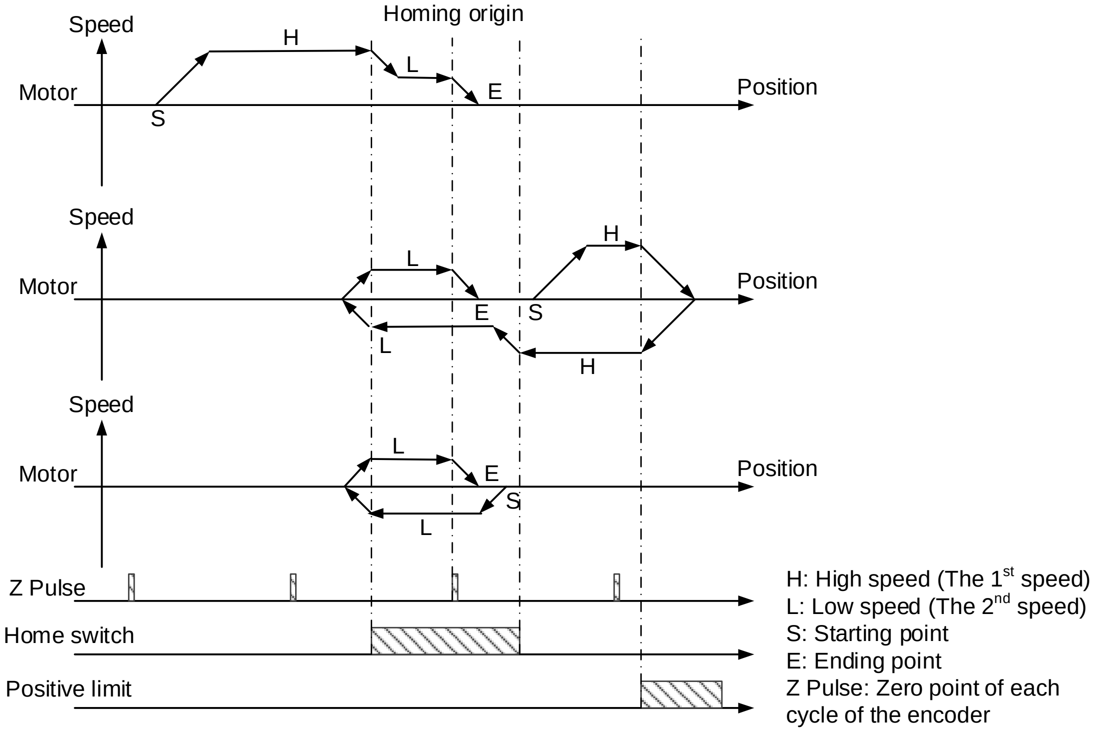
7. Mode 7

- Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, it decelerates in reverse direction. When the motor leaves the home switch at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in forward direction at high speed. When the motor triggers the positive limit before reaching the home switch, it runs in reverse direction until reaching the home switch. Then, the motor decelerates to low speed. When the motor leaves the home switch, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in reverse direction at low speed until it leaves the home switch. Then, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



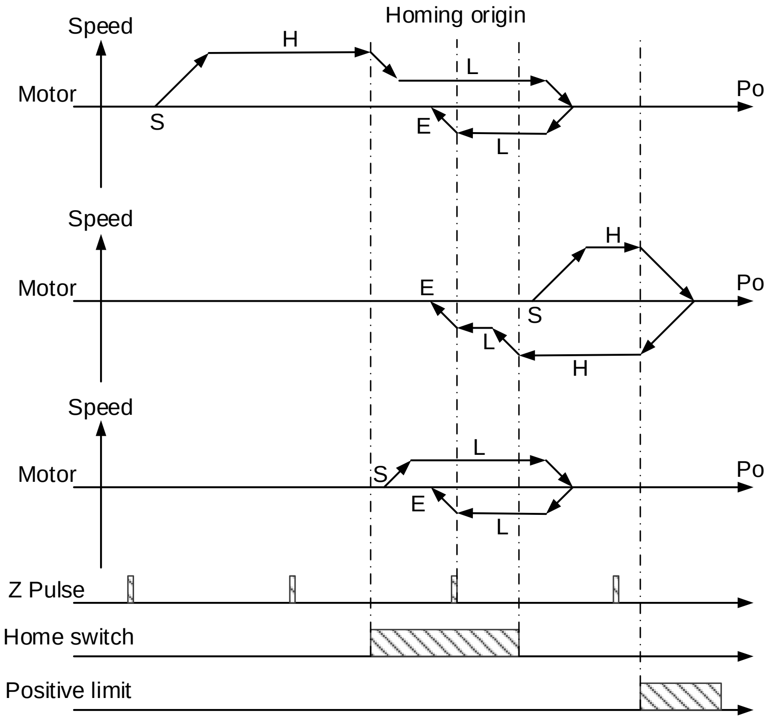
8. **Mode 8**

- Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, the motor runs at low speed, starting to look for the first Z pulse and regards it as the new homing origin.
- Home switch OFF: The motor runs in forward direction at high speed. When the motor triggers the positive limit before reaching the home switch, it runs in reverse direction until reaching the home switch. Then, the motor decelerates and leaves the home switch at low speed. Afterwards, the motor runs in forward direction. When the home switch is reached again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in reverse direction at low speed until it leaves the home switch. Then, it runs in forward direction. When the home switch is reached again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



9. Mode 9

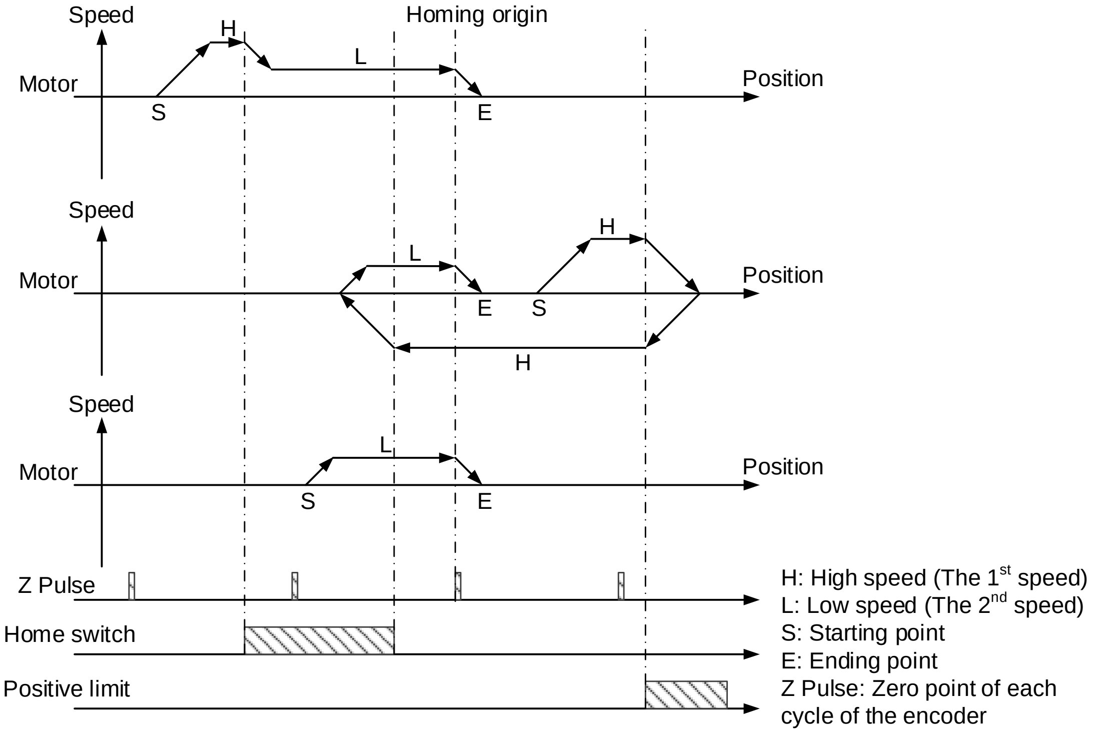
- Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, the motor decelerates and leaves the home switch at low speed. Afterwards, the motor runs in reverse direction. When the motor reaches the home switch again, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in forward direction at high speed. When the motor triggers the positive limit before reaching the home switch, it runs in reverse direction until reaching the home switch. After the motor runs at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, it runs in reverse direction. When the home switch is reached again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

10. Mode 10

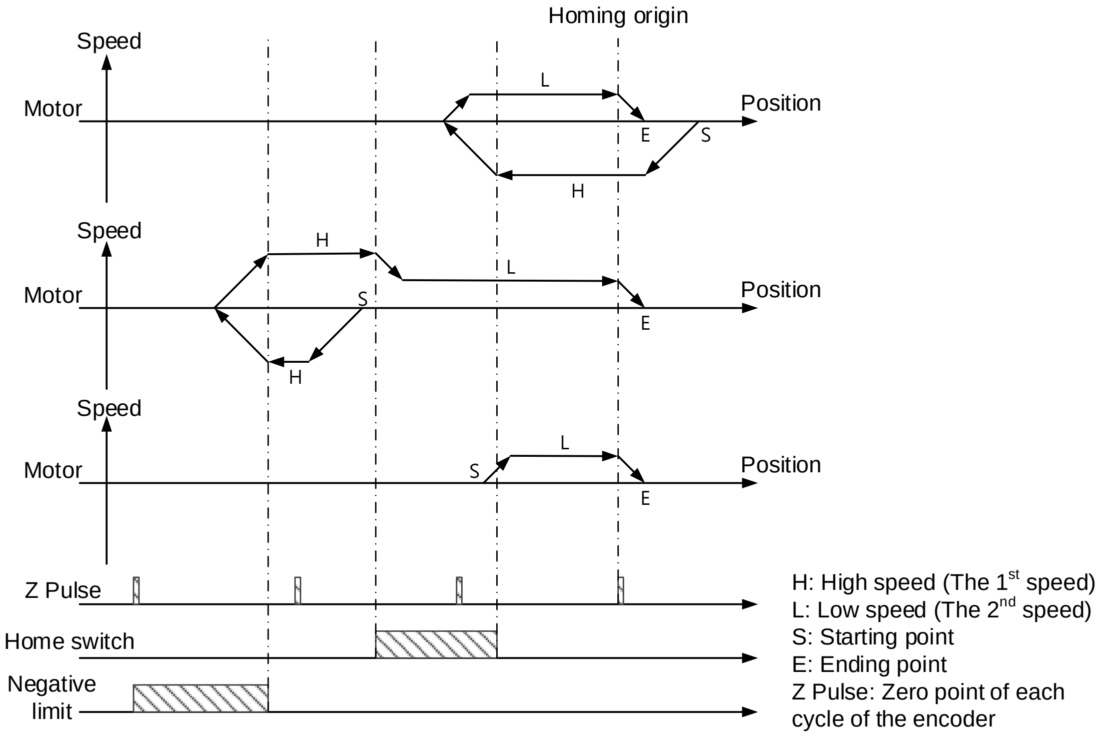
- Home switch OFF: The motor runs in forward direction at high speed until it reaches the home switch. Then, it runs at low speed. When the motor leaves the home switch, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in forward direction at high speed. When the motor triggers the positive limit before reaching the home switch, it runs in reverse direction until reaching the home switch. Then, the motor decelerates and runs in forward direction. When the motor leaves the home switch at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

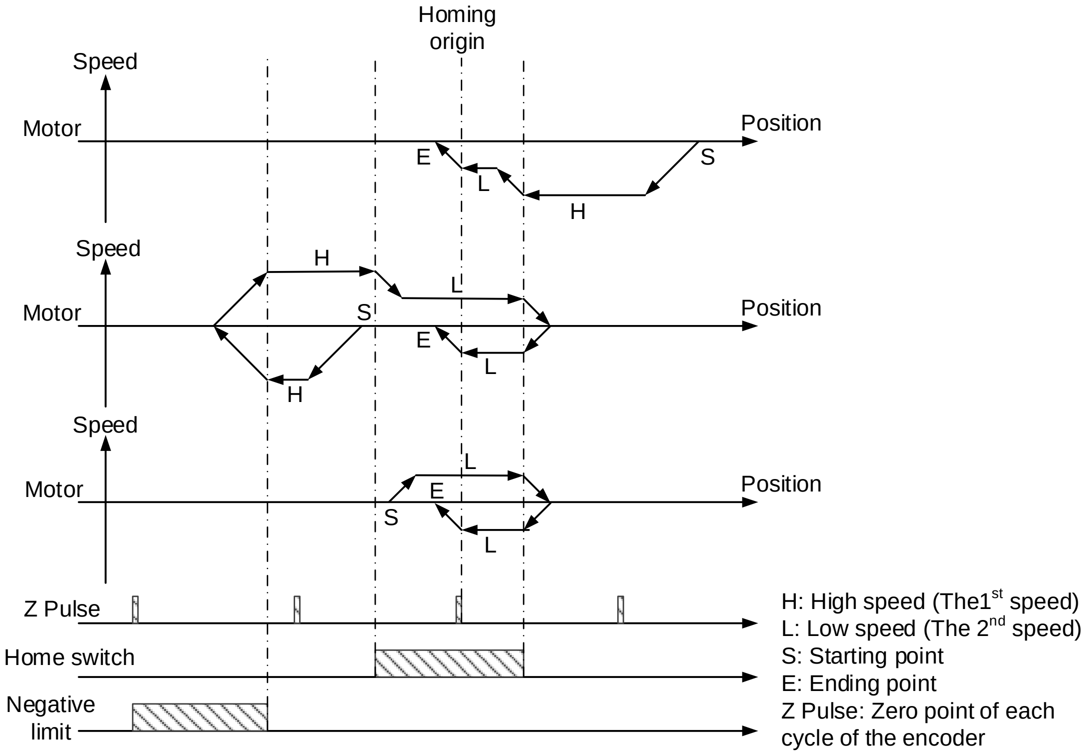
11. Mode 11

- Home switch OFF: The motor runs in reverse direction at high speed until it reaches the home switch. Then, the motor decelerates and then run in forward direction. When the motor leaves the switch at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in reverse direction at high speed. When the motor triggers the negative limit before reaching the home switch, it runs in forward direction until reaching the home switch. Then, the motor decelerates. When the motor leaves the home switch, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



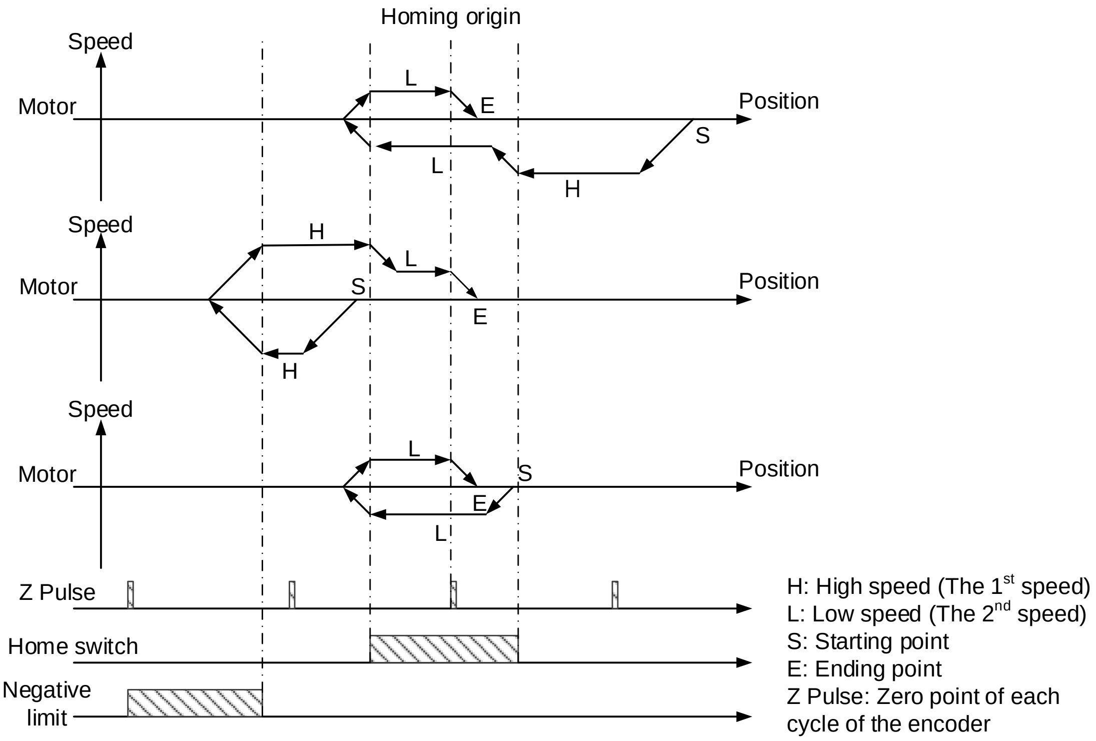
12. Mode 12

- Home switch OFF: The motor runs in reverse direction at high speed until it reaches the home switch. Then, the motor runs at low speed, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in reverse direction at high speed. When the motor triggers the negative limit before reaching the home switch, it runs in forward direction until reaching the home switch. Then, the motor decelerates and leaves the home switch at low speed. Afterwards, it runs in reverse direction. When the motor reaches the home switch again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in forward direction at low speed until it leaves the home switch. Then, the motor runs in reverse direction. When the home switch is reached again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



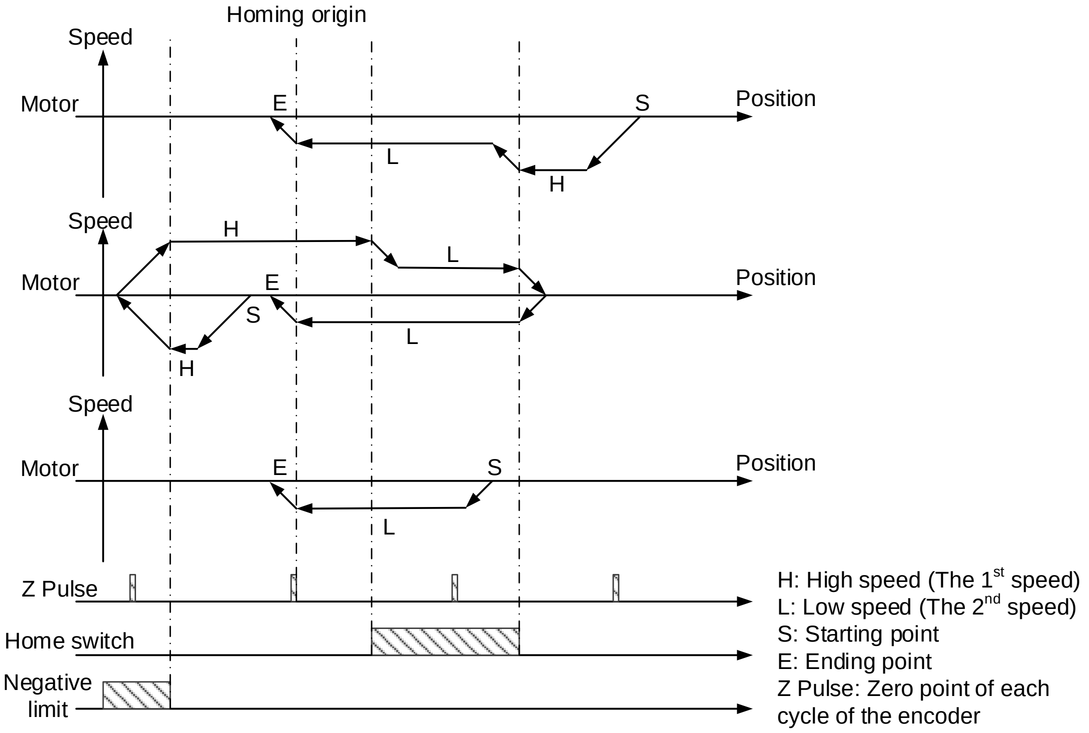
13. Mode 13

- Home switch OFF: The motor runs in reverse direction at high speed until it reaches the home switch. Then, the motor decelerates and leaves the switch at low speed. Afterwards, the motor runs in forward direction. When it reaches the home switch again, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in reverse direction at high speed. When the motor triggers the negative limit before reaching the home switch, it starts running in forward direction. Once reaching the home switch, the motor runs at low speed and starts to look for the first Z pulse and regards it as the new homing origin.
- Home switch ON: The motor runs in reverse direction at low speed until it leaves the home switch. Then, the motor runs in forward direction. When it reaching the home switch again, the motor starts to look for the first Z pulse and regard the first Z pulse as the new homing origin.



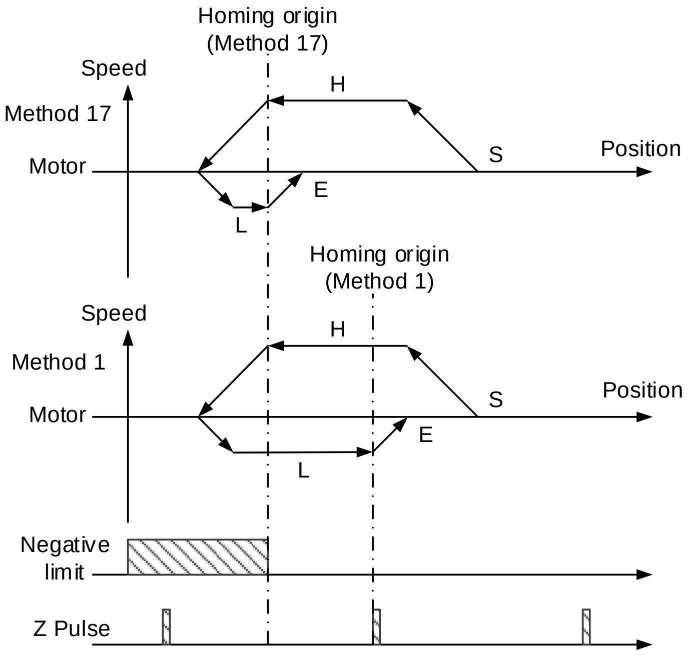
14. Mode 14

- Home switch OFF: The motor runs in reverse direction at high speed before it reaches the home switch. Then, the motor decelerates to low speed. When the motor leaves the home switch, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch OFF: The motor runs in reverse direction at high speed. When the motor triggers the negative limit before reaching the home switch, it runs in forward direction until reaching the home switch. Then, the motor decelerates and leaves the home switch at low speed. Afterwards, the motor runs in reverse direction and reaches the home switch again. When the motor leaves the home switch again, it starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.
- Home switch ON: The motor runs in reverse direction at low speed until it leaves the home switch. Then, the motor starts to look for the first Z pulse and regards the first Z pulse as the new homing origin.



15. Mode 17 ~ 30

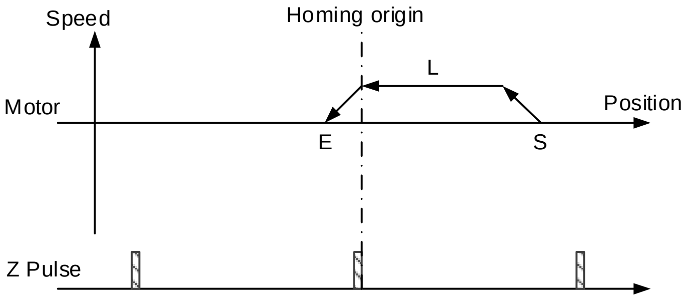
Mode 17 ~ 30 are similar to mode 1 ~ 14 with following differences: In mode 1 ~ 14, after receiving signals of the limits or home switch, the motor looks for Z pulse and regards the Z pulse as the new homing origin, whereas in mode 17 ~ 30, the motor regards the signals as the new homing origin. Please refer to the figure below for the differences between mode 1 and mode 17.



H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

16. Mode 33

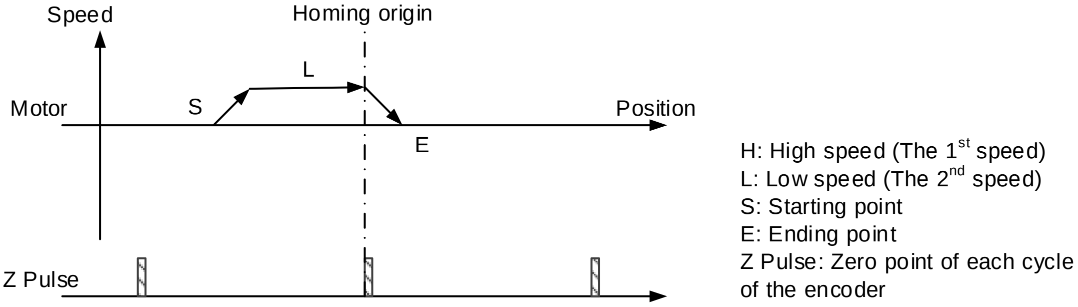
The motor runs in reverse direction looking for the first Z pulse and regards the it as the new homing origin.



H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

17. **Mode 34**

The motor runs in forward direction looking for the first Z pulse and regards it as the new homing origin.


Speed
Homing origin
L
Motor
S
Position
E
Z Pulse
H: High speed (The 1st speed)
L: Low speed (The 2nd speed)
S: Starting point
E: Ending point
Z Pulse: Zero point of each cycle of the encoder

18. **Mode 35**

The motor regards the current position as the new homing origin.

12.2 _ECAT_Slave_Home_Move

■ Syntax

U16 PASCAL _ECAT_Slave_Home_Move (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ Purpose

This is for executing homing. The specified motion axis will start homing according to the setting mode (section 12.1).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

■ Example

```
U16 Status;
U16 CardNo=1, AxisNo=1, SlotNo=0;

/*Start homing*/
Status = _ECAT_Slave_Home_Move (CardNo, AxisNo, SlotNo);
```

12.3 _ECAT_Slave_Home_Status

■ Syntax

U16 PASCAL _ECAT_Slave_Home_Status (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Status)

■ Purpose

This is for acquiring the current homing status.

Note: This API can only be used in homing mode. If it is used in other motion modes, the following returned code will prompt out: ERR_ECAT_MODE_NOT_SUPPORT (4612)

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Status	U16*	Status	Status: 0: Motion not started or homing completed. 1: Homing in progress 2: Homing terminated while the procedure is not completed 3: Error occurs during homing

■ Example

```
U16 Status;
U16 CardNo=1, AxisNo=1, SlotNo=0;

/*Acquire the homing status of the specified axis*/
Status = _ECAT_Slave_Home_Status (CardNo, AxisNo, SlotNo, &Status);
```

Profile Position Mode (PP) 13

This chapter explains the APIs for single-axis motion control in PP mode. In PP mode, all relevant commands will be issued to the motion axis at a time. Then, the motion axis will automatically complete those commands. Its motion will not be interfered by EtherCAT master unless it is a stop command.

Commands will be issued via SDO communication in PP (Profile Position) mode. In this mode, EtherCAT master sends the position, speed and acceleration related parameters to the motion axis. When all commands are issued, the motion axis starts working and will be controlled by the servo drive and pulse module. It can achieve real-time control. However, it does not support multi-axis interpolation.

API list of PP mode

Function name	Description
_ECAT_Slave_PP_Start_Move	Execute single-axis linear motion in PP mode
_ECAT_Slave_PP_Advance_Config	Advanced setting of PP mode

13.1 _ECAT_Slave_CST_Start_Move

■ Syntax

U16 PASCAL _ECAT_Slave_PP_Start_Move(U16 CardNo, U16 AxisNo, U16 SlotNo,
I32 TargetPos, U32 ConstVel, U32 Acceleration, U32 Deceleration, U16 Abs_Rel)

■ Purpose

This is for executing single-axis linear motion in PP mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
TargetPos	I32	inc	The specified moving distance. inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607A Sub 0)
ConstVel	U32	Pulse per second (pps)	Constant speed of the motion (inc/s) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6081 Sub 0)
Acceleration	U32	Pulse / s^2	Acceleration (inc/s^2) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6083 Sub 0)
Deceleration	U32	Pulse / s^2	Deceleration (inc/s^2) inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6084 Sub 0)
Abs_Rel	U16	Option	0: Relative movement (Default) 1: Absolute movement

■ Example

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, Abs_Rel=1;
I32 Dist=5000000;
U32 MaxVel=2000000;
U32 TAcc = 100, Tdec = 100; // A2E: the time (ms) it takes to accelerate to 3000 rpm
```

```
Status = _ECAT_Slave_PP_Start_Move(CardNo, AxisNo, SlotNo, Dist, MaxVel, TAcc, TDec,
Abs_Rel);
```

13.2 _ECAT_Slave_PP_Advance_Config

■ **Syntax**

U16 PASCAL _ECAT_Slave_PP_Advance_Config(U16 CardNo, U16 AxisNo, U16 SlotNo,U16 SetBit, I32 End_Vel, I32 Min_Range_Limit, I32 Max_Range_Limit, I32 Min_Soft_Limit, I32 Max_Soft_Limit)

■ **Purpose**

This is for the advanced setting of PP mode.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
SetBit	U16	Option	Enable the following parameters via the bit values: Bit 0 → End_Vel Bit 1 → Min_Range_Limit Bit 2 → Max_Range_Limit Bit 3 → Min_Soft_Limit Bit 4 → Max_Soft_Limit
End_Vel	I32	Inc / Sec	End velocity. inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607A Sub 0)
Min_Range_Limit	I32	Inc	Set the minimum range for relative movement. inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607B Sub 1)
Max_Range_Limit	I32	Inc	Set the maximum range for relative movement. inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607B Sub 2)
Min_Soft_Limit	I32	Inc	Set the minimum range for absolute movement. inc signifies the unit set in the slave device. Please

Name	Data type	Property	Description
			refer to the manual of the slave device for more details. (OD: 0x607D Sub 1)
Max_Soft_Limit	I32	Inc	Set the maximum range for absolute movement inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x607D Sub 2)

- **Example**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, SetBit=0x07;
//Set End_Vel, Min_Range_Limit&Max_Range_Limit
I32 End_Vel =50, Min_Range_Limit =1, Max_Range_Limit =200;
I32 Min_Soft_Limit= 2, Max_Soft_Limit=180;

Status = _ECAT_Slave_PP_Advance_Config(CardNo, AxisNo, SlotNo, SetBit, End_Vel,
Min_Range_Limit, Max_Range_Limit, Min_Soft_Limit, Max_Soft_Limit);
```

(This page is intentionally left blank.)

Profile Velocity Mode (PV) 14

This chapter introduces the API used in PV mode. Different from CSV mode, all relevant commands will be issued to the motion axis at once in PV mode. Then, the motion axis will automatically complete the motion and not be interfered by EtherCAT master during the process (except the stop command). Thus, it only issues the command for single-axis motion.

Commands will be issued via SDO communication in PV (Profile Velocity) mode. EtherCAT master sends the speed and acceleration parameters to the motion axis. When all commands are issued, the motion axis starts working and will be controlled by the servo drive and pulse module.

API list of PV mode

Function name	Description
_ECAT_Slave_PV_Start_Move	Execute the single-axis motion with constant speed in PV mode
_ECAT_Slave_PV_Advance_Config	Advanced setting of PV mode

14.1 _ECAT_Slave_PV_Start_Move

■ Syntax

```
U16 PASCAL _ECAT_Slave_PV_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,
I32 TargetVel, U32 Acceleration, U32 Deceleration)
```

■ Purpose

This is for executing the single-axis motion with constant speed in PV mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
TargetVel	I32	inc/s	Target speed inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x60FF Sub 0)
Acceleration	U32	inc/s^2	Acceleration speed inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6083 Sub 0)
Deceleration	U32	inc/s^2	Deceleration speed inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x6084 Sub 0)

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 TargetVel=300;
F64 Acceleration=5, Deceleration=5;

Status=_ECAT_Slave_PV_Start_Move (CardNo, AxisNo, SlotNo, TargetVel, Acceleration,
Deceleration);
```

14.2 _ECAT_Slave_PV_Advance_Config

■ Syntax

U16 PASCAL _ECAT_Slave_PV_Advance_Config (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 SetBit, U16 Max_Torque, U16 Velocity_Window,
U16 Velocity_Window_Time, U16 Velocity_Threshold, U16 Velocity_Threshold_Time)

■ Purpose

This is for the advanced setting of PV mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
SetBit	U16	Option	Enable the following parameters via the bit values: Bit 0 → Max_Torque Bit 1 → Velocity_Window Bit 2 → Velocity_Window_Time Bit 3 → Velocity_Threshold Bit 4 → Velocity_Threshold_Time
Max_Torque,	U16	Permillage	The torque output, which setting range is 1 ~ 1000. (OD: 0x6072 Sub 0)
Velocity_Window	U16	Inc/s	Specify the range for state of “target speed reached”. inc signifies the unit set in the slave device. Please refer to the manual of the slave device for more details. (OD: 0x606D Sub 0)
Velocity_Window_Time	U16	Millisecond (ms)	Set the duration for state of “target speed-reached”. (OD: 0x606E Sub 0) Bit 10-Target_Reached of Status_Word will be enabled when the difference between the motion speed and the set speed is smaller than the value specified by Velocity_Window, and such difference has lasted longer than the time set by Velocity_Window_Time.
Velocity_Threshold	U16	Inc/s	Specify the speed range (Address: 0x606F Sub 0)

Name	Data type	Property	Description
Velocity_Threshold_Time	U16	Millisecond (ms)	Set the duration of non-zero speed (OD: 0x6070 Sub 0) Bit 12 of " _ECAT_Slave_Motion_Get_StatusWord" will be disabled when the motion speed reaches the value specified by Velocity_Threshold, and this speed has lasted longer than the time set by Velocity_Threshold_Time.

- **Example**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, SetBit=0x03; //Set Max_Torque, Velocity_Window;
U16 Max_Torque=200, Velocity_Window=20, Velocity_Window_Time=3;
U16 Velocity_Threshold=10, Velocity_Threshold_Time=3;

Status=_ECAT_Slave_PV_Advance_Config (CardNo, AxisNo, SlotNo, SetBit,
Max_Torque, Velocity_Window, Velocity_Window_Time, Velocity_Threshold,
Velocity_Threshold_Time);
```

(This page is intentionally left blank.)

Inverter Motion Control 15

This chapter presents the API for inverter motion control. Without encoder, the inverter cannot complete position feedback control. Thus, it only provides the information about single-axis constant speed control.

API list of inverter motion control

Function name	Description
_ECAT_Slave_VL_Start_Move	Inverter single-axis motion control with constant speed. (Only applicable to Delta inverter)

15.1 _ECAT_Slave_VL_Start_Move

■ Syntax

U16 PASCAL _ECAT_Slave_VL_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 TargetVel, U32 Acceleration, U32 Deceleration)

■ Purpose

This is for executing Delta inverter single-axis motion control with constant speed.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
TargetVel	I32	inc/s	Target speed (inc/sec) inc signifies the unit set in the slave. Please refer to the user manual of the applied slave device. (OD : 0x6042 Sub 0)
Acceleration	U32	inc/s^2	Acceleration inc signifies the unit set in the slave. Please refer to the user manual of the applied slave device. (OD: 0x604F Sub 0)
Deceleration	U32	inc/s^2	Deceleration inc signifies the unit set in the slave. Please refer to the user manual of the applied slave device. (OD: 0x6050 Sub 0)

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 TargetVel=300;
F64 Acceleration=50, Deceleration=50;

Status=_ECAT_Slave_VL_Start_Move (CardNo, AxisNo, SlotNo, TargetVel, Acceleration,
Deceleration);
```

Profile Torque Mode (PT) 16

This chapter introduces the APIs used for single-axis motion control in PT mode. Unlike CST mode, all relevant commands will be issued to the motion axis at once in PT mode. Then, the motion axis will automatically complete the motion and not be interfered by EtherCAT master during the process (except the stop command). Thus, it only issues the command for single-axis motion.

Commands will be issued via SDO communication in PT (Profile Torque) mode. EtherCAT master sends the acceleration parameters to the motion axis. When all commands are issued, the motion axis starts working and will be controlled by the servo drive and pulse module.

API list of PT mode

Function name	Description
_ECAT_Slave_PT_Start_Move	Execute the single-axis motion with constant torque in PT mode.
_ECAT_Slave_PT_Advance_Config	Advanced setting of PT mode

16.1 _ECAT_Slave_PT_Start_Move

■ Syntax

U16 PASCAL _ECAT_Slave_PT_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,
I16 Target_Torque, U32 Slope, I16 Torque_Profile)

■ Purpose

This is for executing the single-axis motion with constant torque in PT mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Target_Torque	I16	Permillage	0.1% of the maximum rated torque, which setting range is 1 ~ 1000.
Slope	U32	0.1% / s	The torque’s rising slope; 0.1% per second.

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I16 Target_Torque=2;
U32 Slope=10;

Status=_ECAT_Slave_PT_Start_Move (CardNo, AxisNo, SlotNo, Target_Torque, Slope);
```

16.2 _ECAT_Slave_PT_Advance_Config

■ Syntax

U16 PASCAL _ECAT_Slave_PT_Advance_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 SetBit, U16 Max_Current, I16 Torque_Profile)

■ Purpose

This is for the advanced setting of PT mode.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
SetBit	U16	Option	Enable the following parameters via bit setting: Bit 0 → Max_Current Bit 1 → Torque_Profile
Max_Current	U16	Permillage	0.1% of the maximum rated torque, which setting range is 1 ~ 1000.
Torque_Profile	I16	Option	0: Linear change 1: Sine wave change

■ Example

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Max_Current=200, SetBit=0, Torque_Profile=0;

Status=_ECAT_Slave_PT_Advance_Config (CardNo, AxisNo, SlotNo, Max_Current,
Torque_Profile);
```

Group Motion Control 17

This chapter introduces the API for setting the specified groups. The grouped motion axes use two methods to carry out motion commands. In the first method, EtherCAT master completes the command of the group one after another to avoid motion commands being executed simultaneously. Another method is to fill in the DDA table, which is for the profile of the path. Users have to fill in the position required in each communication cycle for the motion axis. It can perform the user designed interpolation function for three axes or above. However, users have to calculate the interpolation for all axes during acceleration and deceleration.

List of Group Motion Control API

Function name	Description
_ECAT_Slave_User_Motion_Control_Set_Enable_Mode	Set the group status. *Please note that before enabling the group, users should apply Set_Motion_Control_Type to specify the axis for one group and use _ECAT_Slave_User_Motion_Control_ Svon and _ECAT_Slave_User_Motion_Control_Get_Alm to confirm the status of each axis.
_ECAT_Slave_User_Motion_Control_Get_Enable_Mode	Acquire the status in the current group.
_ECAT_Slave_User_Motion_Control_Set_Type	Set the motion mode in the specified group.
_ECAT_Slave_User_Motion_Control_Set_Data	Set the data of each axis in the specified group.
_ECAT_Slave_User_Motion_Control_Clear_Data	Clear the data of each axis in the specified group.
_ECAT_Slave_User_Motion_Control_Get_DataCnt	Read the data number that have not been processed in the specified group.
_ECAT_Slave_User_Motion_Control_ Ralm	Reset the alarm of all axes in the specified group.
_ECAT_Slave_User_Motion_Control_ Svon	Enable/disable all axes in the specified group.
_ECAT_Slave_User_Motion_Control_Get_Alm	Acquire the current alarm status in the specified group.

17.1 _ECAT_Slave_User_Motion_Control_Set _Enable_Mode

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Set_Enable_mode (U16 CardNo, U16 GroupNo, U16 Mode)

■ Purpose

This is for setting the group status.

Note:

1. Before using this API, please enable all the axes in the group.
2. Please note that before enabling the group, users should apply _ECAT_Slave_User_Motion_Control_Set_Type to specify the axes to be grouped and motion of the group (see section 17.3). Then, use _ECAT_Slave_User_Motion_Control_ Svon (see section 17.8) to enable the motor of each axis. Finally, this API(_ECAT_Slave_User_Motion_Control_Set _Enable_Mode) can be used. Please refer to the example below.
3. Suggested steps: Set mode to 2 (Mode=2) to switch to the pause state. Then, use API "_ECAT_Slave_User_Motion_Control_Set_Data" (see section 17.4) to input 100 data beforehand so that the commands can be issued in time and avoid vibration of the machine.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
Mode	U16	Option	Status: 0: Disable 1: Enable 2: Pause

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo =1, AxisNum=2, AxisNoArray[2] ={1,2}, SlotNoArray[2] = {0,0},
Type=0, Mode =2, ON_OFF=1, Counter;
I32 DataArray[2]={12,33};
Status =_ECAT_Slave_User_Motion_Control_Set_Type (CardNo, GroupNo ,AxisNum ,
AxisNoArray, SlotNoArray, Type);

// The axes have to be enabled first so that the user can carry on using other relevant functions.
Status = _ECAT_Slave_User_Motion_Control_Svon(CardNo, GroupNo, ON_OFF);
// Enable the mode and set it to pause.
Mode =2;
Status =_ECAT_Slave_User_Motion_Control_Set_Enable_Mode (CardNo, GroupNo, Mode);
// Input 100 data in advance. If the communication cycle of the EtherCAT master is 1 ms, it
means these 100 data requires 100 ms to process.
```

```
for (int I = 0; i<100 ; i++)
{
    Status = _ECAT_Slave_User_Motion_Control_Set_Data (CardNo, GroupNo, DataArray);
}
// Motion start
Mode =1;
Status =_ECAT_Slave_User_Motion_Control_Set_Enable_Mode (CardNo, GroupNo, Mode);
// Carry on issuing rest of the motion commands.
while (1)
{
    Status = _ECAT_Slave_User_Motion_Control_Get_DataCnt (CardNo, GroupNo,
&Counter);
    if (Counter<100)
    {
        // If you are using RTX version EtherCAT of Delta PAC, users can check up to 800
data.
        Status = _ECAT_Slave_User_Motion_Control_Set_Data (CardNo, GroupNo,
DataArray);
    }
    else if (Counter==0)
    {
        Mode =0; // There is no command so function is disabled; exit the loop.
        Status =_ECAT_Slave_User_Motion_Control_Set_Enable_Mode (CardNo, GroupNo,
Mode);
        break;
    }
}
```

17.2 _ECAT_Slave_User_Motion_Control_Get_Enable_Mode

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Get_Enable_mode (U16 CardNo, U16 GroupNo, U16* Mode)

■ Purpose

This is for acquiring the status of the current group.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
Mode	U16*	Option	Status: 0: Disable 1: Enable 2: Pause

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;
U16 Mode;

Status=_ECAT_Slave_User_Motion_Control_Get_Enable_Mode (CardNo, GroupNo, &Mode);
```

17.3 _ECAT_Slave_User_Motion_Control_Set_Type

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Set_Type (U16 CardNo, U16 GroupNo, U16 AxisNum, U16 *AxisNo, U16 *SlotNo, U16 Type)

■ Purpose

This is for setting the motion mode of the group.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
AxisNum	U16	Value	Axis number in one group
AxisNo	U16*	Array for each axis	Array for each axis (node ID); the array number should equal to the axis number AxisNo Array[0] stores the first node AxisNo Array[1] stores the second node ….
SlotNo	U16*	Array for each slot	Array for each axis (slot ID); the array number should equal to the axis number
Type	U16	Option	Description of each mode: 0: Mode for general motion command; When applying this mode, users have to apply the motion commands from other chapters. And the control axis of the specified group will complete the command in sequence. 1: User-defined path (CSP mode); When applying this mode, users have to defined the path for each axis in each EtherCAT communication cycle. 2: User-defined path (CSV mode); When applying this mode, users have to specify the speed for each axis in each EtherCAT communication cycle.

Example

```
U16 Status;
U16 CardNo=16, GroupNo=1, AxisNum=2, AxisNoArray[2]={1,2}, SlotNoArray[2]={0,0};
U16 Type=0;
U16 Mode=1;

Status=_ECAT_Slave_User_Motion_Control_Set_Type (CardNo, GroupNo, AxisNum,
AxisNoArray, SlotNoArray, Type);
```

Description

EtherCAT master provides 3 modes of group motion. See the description below:

When the mode is set to 0, users can use the functions described in section 17.4 ~ 17.9 to issue the position command to be executed by the axis of the group in each communication cycle. By doing so, the user-defined interpolation can be done. Please note that users have to set the acceleration/deceleration and logic of interpolation because position commands in each cycle are user-defined.
Apply API “ECAT_Slave_User_Motion_Control_Set_Data” (refer to section 17.4) and then use “_ECAT_Slave_User_Motion_Control_Set_Enable_Mode” (section 17.1) to enable this function so that the speed of filling the table content can keep up with the communication cycle time. In this case, motion with continuous speed can be carried out.

When mode is set to 1 or 2, EtherCAT master will execute the CSP or CSV commands issued by users in sequence. If the current command of any one of the grouped axes is not completed and a new command is issued, the master will complete the current one first and then execute the new one.
For example, if a motion command of the group’s 2nd axis is not completed and the EtherCAT master receives interpolation commands from the 1st and 3rd axis, the master will firstly execute the motion command of the 2nd axis and then the 1st axis and the 3rd axis.

17.4 _ECAT_Slave_User_Motion_Control_Set_Data

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Set_Data (U16 CardNo, U16 GroupNo, I32 *Data)

■ Purpose

This is for inputting the absolute position data of each axis of each communication cycle in the specified group when mode is set to 0 in API "_ECAT_Slave_User_Motion_Control_Set_Type" (section 17.3).

Note: The maximum data for PAC is 800 and 100 for motion control card.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
Data	I32*	Data array for each axis	Data array for each axis; the array number should equal to the axis number. DataArray[0] stores the data of the first axis DataArray[1] stores the data of the second axis …

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;
I32 DataArray[2]={12,33};

Status=_ECAT_Slave_User_Motion_Control_Set_Data (CardNo, GroupNo, DataArray);
```

17.5 _ECAT_Slave_User_Motion_Control_Clear_Data

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Clear_Data (U16 CardNo, U16 GroupNo)

■ Purpose

This is for clearing the data of each axis in the group that is specified by API " _ECAT_Slave_User_Motion_Control_Set_Data" (section 17.4) when the mode is set to 0 in API "_ECAT_Slave_User_Motion_Control_Set_Type " (section 17.3).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;

Status=_ECAT_Slave_User_Motion_Control_Clear_Data (CardNo, GroupNo);
```

17.6 _ECAT_Slave_User_Motion_Control_Get_DataCnt

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Get_DataCnt (U16 CardNo, U16 GroupNo, U16* Counter)

■ Purpose

This is for reading the data number that has not been processed in the specified group when mode is set to 0 in API “_ECAT_Slave_User_Motion_Control_Set_Type”.

Note: The maximum data for PAC is 800 and 100 for motion control card.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
Counter	U16*	Value	Data number that has not been processed

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;
U16 Counter;

Status=_ECAT_Slave_User_Motion_Control_Get_DataCnt (CardNo, GroupNo, &Counter);
```

17.7 _ECAT_Slave_User_Motion_Control_Ralm

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Ralm (U16 CardNo, U16 GroupNo)

■ Purpose

This is for resetting the alarm of all axes in the specified group.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;

Status=_ECAT_Slave_User_Motion_Control_Ralm (CardNo, GroupNo);
```

17.8 _ECAT_Slave_User_Motion_Control_Svon

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Svon (U16 CardNo, U16 GroupNo, U16 ON_OFF)

■ Purpose

This is for enabling/disabling all axes in the group. Except the API "_ECAT_Slave_User_Motion_Control_Set_Type" (see section 17.3), all axes of the group have to be enabled before use. Users can use either "_ECAT_Slave_Motion_Set_Svon" or "_ECAT_Slave_User_Motion_Control_Svon" to enable axes in batch.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
ON_OFF	U16	Option	0: Disable the axes 1: Enable the axes

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;
U16 ON_OFF=1;

Status=_ECAT_Slave_User_Motion_Control_Svon (CardNo, GroupNo, ON_OFF);
```

17.9 _ECAT_Slave_User_Motion_Control_Get_Alm

■ Syntax

U16 PASCAL_ECAT_Slave_User_Motion_Control_Get_Alm (U16 CardNo, U16 GroupNo, U16 *Alm)

■ Purpose

This is for acquiring the current alarm status of the specified group.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
GroupNo	U16	Number	Group No.
Alm	U16*	Status	Alarm description 0: No alarm occurs 1: Alarm occurs in one of the axes in the group

■ Example

```
U16 Status;
U16 CardNo=16, GroupNo=1;
U16 Alm;

Status=_ECAT_Slave_User_Motion_Control_Get_Alm (CardNo, GroupNo, &Alm);
```

(This page is intentionally left blank.)

Operation of DI/DO Module

18

This chapter introduces DI/DO API, including acquiring the state of DI/DO module and signal, DO settings and its output value when error occurs, etc.

The APIs included in this chapter are applicable to all EtherCAT digital input/output modules. Please note that Delta DI/DO modules require to be enabled before being used; please refer to Chapter 32 for more information. For the retentive function for the digital output when an error occurs, refer to Chapter 31. And see Chapter 26 for setting local digital input and digital output of GPIO on motion card.

API list of DI/DO module

Function name	Description
_ECAT_Slave_DIO_Get_Input_Value	Acquire the DI status
_ECAT_Slave_DIO_Get_Output_Value	Acquire the DO status
_ECAT_Slave_DIO_Set_Output_Value	Set the DO status
_ECAT_Slave_DIO_Get_Single_Input_Value	Acquire the input value of the specified channel
_ECAT_Slave_DIO_Get_Single_Output_Value	Acquire the output value of the specified channel
_ECAT_Slave_DIO_Set_Single_Output_Value	Set the output value of the specified channel
_ECAT_Slave_DIO_Set_Output_Error_Mode	Enable/Disable the retentive function of each channel on remote DO module when EtherCAT communication is disconnected
_ECAT_Slave_DIO_Set_Output_Error_Value	Set the output status of each channel on remote DO module when EtherCAT communication is disconnected and the retentive function is enabled

18.1 _ECAT_Slave_DIO_Get_Input_Value

Syntax

U16 PASCAL_ECAT_Slave_DIO_Get_Input_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

Purpose

This is for acquiring the DI status of the DI module. To acquire the status of X15, X14, …, X1, X0 (from left to right), users can convert the value to binary format.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16*	Value	Data received by the DI module

Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0;
U16 Value;

Status=_ECAT_Slave_DIO_Get_Input_Value (CardNo, NodeID, SlotNo, &Value)
```

18.2 _ECAT_Slave_DIO_Get_Output_Value

Syntax

U16 PASCAL_ECAT_Slave_DIO_Get_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

Purpose

This is for acquiring the DO status of the DO module. To acquire the status of Y15, Y14, …, Y1, Y0 (from left to right), users can convert the value to binary format.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16*	Value	Acquire the value output by the digital output remote module.

Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0;
U16 Value;

Status=_ECAT_Slave_DIO_Get_Output_Value (CardNo, NodeID, SlotNo, &Value);
```

18.3 _ECAT_Slave_DIO_Set_Output_Value

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Set_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Value)

■ Purpose

This is for setting the output status of the DO module. To output the status of Y15, Y14, …, Y1, Y0 (from left to right), users can convert the bit status to decimal format.

Note: To use this API with Delta's remote DO module R1-EC70E2D0 and R1-EC70F2D0, please firstly execute the API "_ECAT_Slave_R1_EC70X2_Set_Output_Enable" (see section 32.1) to enable module's output function.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16	Value	The value output by the digital output remote module.

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0;
U16 Value=0xFFFF;

Status=_ECAT_Slave_DIO_Set_Output_Value (CardNo, NodeID, SlotNo, Value);
```

18.4 _ECAT_Slave_DIO_Get_Single_Input_Value

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Get_Single_Input_Value (U16 CardNo, U16 NodeID,
U16 SlotNo, U16 BitNum, U16 *Value)

■ Purpose

This is for acquiring the input value of the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNum	U16	Number	Channel number of digital input
Value	U16*	Value	The input value of the specified channel on DI module

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0, BitNum=1;
U16 Value;

Status=_ECAT_Slave_DIO_Get_Single_Input_Value (CardNo, NodeID, SlotNo, BitNum,
&Value);
```

18.5 _ECAT_Slave_DIO_Get_Single_Output_Value

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Get_Single_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNum, U16 *Value)

■ Purpose

This is for acquiring the value output by the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNum	U16	Number	Channel number of digital output
Value	U16*	Value	The output value of the specified channel on DO module

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0, BitNum=1;
U16 Value;

Status=_ECAT_Slave_DIO_Get_Single_Output_Value (CardNo, NodeID, SlotNo, BitNum,
&Value);
```

18.6 _ECAT_Slave_DIO_Set_Single_Output_Value

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Set_Single_Ouput_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNum, U16 Value)

■ Purpose

This is for setting the output value of the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNum	U16	Number	Channel number of digital output
Value	U16	Value	The output value of the single channel on DO module

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0, BitNum=1;
U16 Value=1;

Status=_ECAT_Slave_DIO_Set_Single_Output_Value (CardNo, NodeID, SlotNo, BitNum,
Value);
```

18.7 _ECAT_Slave_DIO_Set_Output_Error_Mode

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Set_Output_Error_Mode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitMode)

■ Purpose

This is for enabling/disabling the retentive function of each output channel on remote DO module when EtherCAT communication is disconnected.

Note:

1. Please use _ECAT_Slave_DIO_Set_Output_Error_Value (see section 18.8) to set the output value when EtherCAT communication is disconnected.
2. Retentive function is only supported by R1-EC70E2D0 and R1-EC70F2D0.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitMode	U16	Value	Bit 0 ~ 15 represent the output channels Y0 ~ Y15 respectively. 0: Disable 1: Enable

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0;
// Enable the retentive function for the first 8 channels.
U16 BitMode=0x0F;

Status=_ECAT_Slave_DIO_Set_Output_Error_Mode (CardNo, NodeID, SlotNo, BitMode);
```

18.8 _ECAT_Slave_DIO_Set_Output_Error_Value

■ Syntax

U16 PASCAL_ECAT_Slave_DIO_Set_Output_Error_Value (U16 CardNo, U16 NodeID, U16SlotNo, U16 Value)

■ Purpose

This is for setting the retentive status of each channel on remote DO module when EtherCAT communication is disconnected.

Note: Please use _ECAT_Slave_DIO_Set_Output_Error_Mode (see section 18.7) to enable the retentive function when EtherCAT communication is disconnected.

■ Parameter

Name	Data type	Unit	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16	Value	Bit0 ~ 15 represent the output channels Y0~Y15 respectively. 0: Output channel is off 1: Output channel is on

■ Example

```
U16 Status;
U16 CardNo=16, NodeID=1, SlotNo=0;
// Change the function of the first 8 channels to retentive function when communication is
disconnected.
U16 Value=0x0F;

Status=_ECAT_Slave_DIO_Set_Output_Error_Value (CardNo, NodeID, SlotNo, Value);
```

Operation of AI/AO Module

19

This chapter introduces the APIs for obtaining the input/output value and setting value of the AI/AO module.

APIs included in this chapter are applicable to all EtherCAT analog input/output modules. Please note that Delta analog modules require to be enabled before being used. Please refer to chapter 23 for more information. For its function settings, refer to chapter 24.

API list of AI/AO module

Function name	Description
_ECAT_Slave_AIO_Get_Input_Value	Acquire analog input value
_ECAT_Slave_AIO_Set_Output_Value	Set analog output value
_ECAT_Slave_AIO_Get_Output_Value	Acquire analog output value

19.1 _ECAT_Slave_AIO_Get_Input_Value

■ Syntax

U16 PASCAL_ECAT_Slave_AIO_Get_Input_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

■ Purpose

This is for acquiring analog input value.

Note: Delta analog input module only allows users to measure voltage signal.
To measure the input current, you need to modify the wriing of the analog input module first. (Please refer to the user manual of Delta analog input model regarding the wiring for current measurement.) After finishing the wiring, users can convert the measurement into current by using the circuit with 250Ω- resistor.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16*	Value	Acquire the data from AI remote module. The value is 0 ~ 65535.

■ Example

```
U16 Status=0;
U16 CardNo=16, NodeID=1, SlotNo=0;
U16 Value;

Status=_ECAT_Slave_AIO_Get_Input_Value (CardNo, NodeID, SlotNo, &Value);
```

19.2 _ECAT_Slave_AIO_Set_Output_Value

■ Syntax

U16 PASCAL_ECAT_Slave_AIO_Set_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Value)

■ Purpose

This is for setting analog output value, which range is 0 ~ 65535.

The output value will be converted into value 0 ~ 65535 in accordance with the proportion. Then, this API will be controlling the analog output module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16	Value	The output value of the AO module, which ragne is 0 ~ 65535.

■ Example

```
U16 Status=0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Value=0x5ff;

Status=_ECAT_Slave_AIO_Set_Output_Value (CardNo, NodeID, SlotNo, Value);
```

19.3 _ECAT_Slave_AIO_Set_Output_Value

■ Syntax

U16 PASCAL_ECAT_Slave_AIO_Get_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16* Value)

■ Purpose

This is for acquiring analog output value.

The output value will be converted into value 0 ~ 65535 in accordance with the proportion. Then, this API will be controlling the analog output module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Value	U16*	Value	The output value of the AO module, which ragne is 0 ~ 65535.

■ Example

```
U16 Status=0;
U16 CardNo=16, NodeID=1, SlotNo=0;
U16 Value=0;

Status=_ECAT_Slave_AIO_Get_Output_Value (CardNo, NodeID, SlotNo, &Value);
```

(This page is intentionally left blank.)

Operation of Pulse Module (For R1-EC5621D0 Series) 20

This chapter provides the details about how to use the APIs for operating the pulse module (for R1-EC5621D0 series). These APIs can set mode of pulse input/output, contact type of the origin signal/Z pulse signal, determine whether to apply the special mode when homing.

If you are using Delta R1-X62XD0 series pulse module (multiple-axis), please refer to chapter 21 for the API usage.

API list of pulse module operation (for R1-EC5621D0 series)

API	Description
_ECAT_Slave_R1_EC5621_Set_Output_Mode	Set the mode of pulse output.
_ECAT_Slave_R1_EC5621_Set_Input_Mode	Set the mode of pulse input.
_ECAT_Slave_R1_EC5621_Set_ORG_Inverse	Set the contact type (NC/NO) of the origin switch (ORG).
_ECAT_Slave_R1_EC5621_Set_QZ_Inverse	Set the contact type (NC/NO) of encoder's Z pulse (QZ).
_ECAT_Slave_R1_EC5621_Set_Home_SpMode	Apply the special mode when homing.
_ECAT_Slave_R1_EC5621_Set_MEL_Inverse	Set the contact type (NC/NO) of the negative limit switch (MEL).
_ECAT_Slave_R1_EC5621_Set_PEL_Inverse	Set the contact type (NC/NO) of the positive limit switch (PEL).
_ECAT_Slave_R1_EC5621_Set_Svon_Inverse	Set the contact type (NC/NO) of the servo enable switch (Svon).
_ECAT_Slave_R1_EC5621_Set_Home_Slow_Down	It sets the deceleration time after the motor reaches the origin
_ECAT_Slave_R1_EC5621_Get_IO_Status	Acquire the status of all I/O points
_ECAT_Slave_R1_EC5621_Get_Single_IO_Status	Acquire the status of single I/O point.

20.1 _ECAT_Slave_R1_EC5621_Set_Output_Mode

Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Output_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

Purpose

This is for setting the mode of pulse output.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: A/B Phase 1: CW/CCW 2: PLS/DIR

Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_EC5621_Set_Output_Mode (CardNo, AxisNo, SlotNo, Mode);
```

20.2 _ECAT_Slave_R1_EC5621_Set_Input_Mode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Input_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

■ Purpose

This is for setting the mode of pulse input.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: A/B Phase 1: CW/CCW 2: PLS/DIR

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_EC5621_Set_Input_Mode (CardNo, AxisNo, SlotNo, Mode);
```

20.3 _ECAT_Slave_R1_EC5621_Set_ORG_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_ORG_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of the origin switch (ORG).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_ORG_Inverse (CardNo, AxisNo,SlotNo, Enable);
```

20.4 _ECAT_Slave_R1_EC5621_Set_QZ_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_QZ_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of encoder’s Z pulse (QZ).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_QZ_Inverse (CardNo, AxisNo,SlotNo, Enable);
```

20.5 _ECAT_Slave_R1_EC5621_Set_Home_SpMode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Home_SpMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ Purpose

This is for applying the special mode when homing. (For special applications only.) EtherCAT master will look for encoder’s Z pulse (QZ) at extremely low speed.)

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: Mode 0 (Normal) 1: Mode 1 (Special); EtherCAT master will look for encoder’s Z pulse (QZ) at extremely low speed.

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Mode=0;

Status= _ECAT_Slave_R1_EC5621_Set_Home_SpMode (CardNo, AxisNo,SlotNo, Mode);
```

20.6 _ECAT_Slave_R1_EC5621_Set_MEL_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_MEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of the negative limit switch (MEL).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_MEL_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

20.7 _ECAT_Slave_R1_EC5621_Set_PEL_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_PEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of the positive limit switch (PEL).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_PEL_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

20.8 _ECAT_Slave_R1_EC5621_Set_Svon_Inverse

- **Syntax**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Svon_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

- **Purpose**

This is for setting the contact type (NC/NO) of the servo enable switch (Svon).

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

- **Example**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_Svon_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

20.9 _ECAT_Slave_R1_EC5621_Set_Home _Slow_Down

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Home_Slow_Down (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable, U16 SlowDownTime, U16 WaitTime)

■ Purpose

It sets the deceleration time after the motor reaches the origin:

1. If the motor runs in reverse direction, it sets the deceleration of the 1st speed and the waiting time after the motor stops.
2. If the motor runs forward, only the setting of WaitTime is valid.

This API will be invalid when the positive / negative limit is regarded as the origin.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Axis number
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Disable 1: Enable
SlowDownTime	U16	Millisecond	The deceleration time when motor reaches the limit.
WaitTime	U16	Millisecond	The waiting time after the motor reaches the limit and then stops.

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0, Enable=1, SlowDownTime =1, WaitTime=1;

Status= _ECAT_Slave_R1_EC5621_Set_Home_Slow_Down (CardNo, AxisNo, SlotNo,
Enable, SlowDownTime, WaitTime);
```

20.10 _ECAT_Slave_R1_EC5621_Get_IO_Status

Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

Purpose

This is for obtaining the status of all I/O points.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOStatus	U16*	Numeric value	Status of all I/O points.

Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus;

Status= _ECAT_Slave_R1_EC5621_Get_IO_Status (CardNo, NodeID, SlotNo, &IOStatus);
```

20.11 _ECAT_Slave_R1_EC5621_Get_Single_IO_Status

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5621_Get_Single_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNo, U16 *IOStatus)

■ Purpose

This is for obtaining the status of single I/O point.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNo	U16	Number	I/O point No.
IOStatus	U16*	Numeric value	Status of single I/O point

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus, BitNo=1;

Status= _ECAT_Slave_R1_EC5621_Get_Single_IO_Status (CardNo, NodeID, SlotNo, BitNo,
&IOStatus);
```

(This page is intentionally left blank.)

Operation of Pulse Module (For R1-ECx62xD0 Series)

21

This chapter provides the detailed information about how to use the APIs for operating the pulse type module (R1-ECx62xD0 series). These APIs can set the type of pulse input/output, contact type of the origin signal/Z pulse signal, and determine whether to apply the special mode when homing.

API list of pulse module operation (for R1-ECx62xD0 series)

API	Description
_ECAT_Slave_R1_ECx62x_Set_Output_Mode	Set the type of pulse output
_ECAT_Slave_R1_ECx62x_Set_Input_Mode	Set the type of pulse input
_ECAT_Slave_R1_ECx62x_Set_ORG_Inverse	Set the contact type (NC/NO) of the origin switch (ORG)
_ECAT_Slave_R1_ECx62x_Set_QZ_Inverse	Set the contact type (NC/NO) of encoder’s Z pulse signal (QZ)
_ECAT_Slave_R1_ECx62x_Set_Home_SpMode	Apply the special mode when homing
_ECAT_Slave_R1_ECx62x_Set_MEL_Inverse	Set the contact type (NC/NO) of the negative limit switch (MEL)
_ECAT_Slave_R1_ECx62x_Set_PEL_Inverse	Set the contact type (NC/NO) of the positive limit switch (PEL)
_ECAT_Slave_R1_ECx62x_Set_Svon_Inverse	Set the contact type (NC/NO) of the servo enable switch (Svon)
_ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down	It sets the deceleration time after the motor reaches the Home switch
_ECAT_Slave_R1_ECx62x_Get_IO_Status	Acquire the status of all I/O points
_ECAT_Slave_R1_ECx62x_Get_Single_IO_Status	Acquire the status of single I/O point

21.1 _ECAT_Slave_R1_ECx62x_Set_Output_Mode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Output_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ Purpose

This is for setting the type of pulse output.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
Slot No	U16	Number	Slot ID
Mode	U16	Option	0: A/B Phase 1: CW/CCW 2: PLS/DIR

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Output_Mode (CardNo, AxisNo, SlotNo, Mode);
```

21.2 _ECAT_Slave_R1_ECx62x_Set_Input_Mode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Input_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

■ Purpose

This is for setting the type of pulse input.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: A/B Phase 1: CW/CCW 2: PLS/DIR

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Input_Mode (CardNo, AxisNo, SlotNo, Mode);
```

21.3 _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of the origin switch (ORG).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse (CardNo, AxisNo,SlotNo, Enable);
```

21.4 _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of encoder's Z pulse signal (QZ).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse (CardNo, AxisNo,SlotNo, Enable);
```

21.5 _ECAT_Slave_R1_ECx62x_Set_Home_SpMode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Home_SpMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ Purpose

This is for applying the special mode when homing. (For special applications only. EtherCAT master will look for encoder’s Z pulse (QZ) at extremely low speed.)

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Mode	U16	Option	0: Mode 0 (Normal) 1: Mode 1 (Special); EtherCAT master will look for encoder’s Z pulse (QZ) at extremely low speed.

■ Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Mode=0;

Status= _ECAT_Slave_R1_ECx62x_Set_Home_SpMode (CardNo, AxisNo,SlotNo, Mode);
```

21.6 _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ Purpose

This is for setting the contact type (NC/NO) of the negative limit switch (MEL).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

21.7 _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse

Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

Purpose

This is for setting the contact type (NC/NO) of the positive limit switch (PEL).

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

Example

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

21.8 _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse

Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

Purpose

This is for setting the contact type (NC/NO) of the servo enable switch (Svon).

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: High-potential trigger (NO) 1: Low-potential trigger (NC)

Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

21.9 _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable, U16 SlowDownTime, U16 WaitTime)

■ Purpose

It sets the deceleration time after the motor reaches the origin:

1. If the motor runs in reverse direction, it sets the deceleration of the 1st speed and the waiting time after the motor stops.
2. If the motor runs forward, only the setting of WaitTime is valid.

This API will be invalid when the positive / negative limit is regarded as the origin.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Disable 1: Enable
SlowDownTime	U16	Millisecond	The deceleration time when motor reaches the limit.
WaitTime	U16	Millisecond	The waiting time after the motor reaches the limit and then stops.

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 Enable=1, SlowDownTime = 1000, WaitTime= 1000;

Status= _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down (CardNo, NodeID, SlotNo, Enable,
SlowDownTime, WaitTime);
```

21.10 _ECAT_Slave_R1_ECx62x_Get_IO_Status

Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

Purpose

This is for acquiring all status of all I/O points.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOStatus	U16*	Numeric value	Status of all I/O points

Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus;

Status= _ECAT_Slave_R1_ECx62x_Get_IO_Status (CardNo, NodeID, SlotNo, &IOStatus);
```

21.11 _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status

■ Syntax

U16 PASCAL _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNo, U16 *IOStatus)

■ Purpose

This is for acquiring single status of single I/O point.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
BitNo	U16	Number	I/O point number
IOStatus	U16*	Numeric value	Status of single I/O point

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus, BitNo=1;

Status= _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status (CardNo, NodeID, SlotNo, BitNo,
&IOStatus);
```

(This page is intentionally left blank.)

Operation of Delta Servo System

22

This chapter will provide the information about how to use the APIs for operating Delta servo system. APIs for reading/writing servo parameters, setting servo maximum speed, reading/writing the compare parameters of the servo drive will be elaborated.

API list of operarting delta servo drive

API	Description
_ECAT_Slave_DeltaServo_Write_Parameter	Write servo parameter values to Delta servo drives
_ECAT_Slave_DeltaServo_Read_Parameter	Read servo parameter values from Delta servo drives
_ECAT_Slave_DeltaServo_Read_Parameter_Info	Read servo parameter attributes from Delta servo drives
_ECAT_Slave_DeltaServo_Set_Velocity_Limit	Set Delta servo motor’s max. speed
_ECAT_Slave_DeltaServo_Set_Compare_Enable	Write the pulse compare parameter, which is identical to Delta servo parameter P5-59
_ECAT_Slave_DeltaServo_Get_Compare_Enable	Read the pulse compare parameter that is written to the servo drive, which is identical to Delta servo parameter P5-59
_ECAT_Slave_DeltaServo_Set_Compare_Config	Write the data array number and values of the pulse compare function to Delta servo drives

22.1 _ECAT_Slave_DeltaServo_Write_Parameter

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Write_Parameter (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, I32 WriteData)

■ Purpose

This is for writing servo parameter values to Delta servo drives.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
Slot No	U16	Number	Slot ID
Page	U16	Value	Device (servo drive) parameter group number
Index	U16	Value	Index of the servo parameter group
WriteData	I32	Value	The data to be written to this group index.

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Page =3, Index =0; // P3-00
I32 WriteData = 1;

Status= _ECAT_Slave_DeltaServo_Write_Parameter (CardNo, AxisNo, SlotNo, Page, Index,
WriteData);
```

22.2 _ECAT_Slave_DeltaServo_Read_Parameter

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Read_Parameter (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, I32* ReadData)

■ Purpose

This is for reading servo parameter values from Delta servo drives.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Node ID
Slot No	U16	Number	Slot ID
Page	U16	Value	Device (servo drive) parameter group number
Index	U16	Value	Index of the servo parameter group
ReadData	I32*	Value	The data returned from the group index.

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0;
U16 Page = 2, Index = 12; // P2-12
I32 ReadData = 0;

Status= _ECAT_Slave_DeltaServo_Read_Parameter (CardNo, AxisNo, SlotNo, Page, Index,
&ReadData);
```

22.3 _ECAT_Slave_DeltaServo_Read_Parameter_Info

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Read_Parameter_Info (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, U16 *ParaType, U16 *DataSize, U16 *DataType)

■ Purpose

This is for reading attributes of the servo parameter from Delta servo drives.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Page	U16	Value	Device (servo drive) parameter group number
Index	U16	Value	Index of the servo parameter group
ParaType	U16*	Value	Parameter type: 0: This parameter is not available. 1: This parameter is read-only. 2: This parameter cannot be set when the servo is in “enabled” state. 3: Power off and restart the servo drive is required in order to validate this parameter setting. 4: This is a volatile parameter. 5: N/A
DataSize	U16*	Value	Data size of this parameter (Unit: Byte)
DataType	U16*	Value	Data type 1: This parameter is displayed in decimal form. 2: This parameter is displayed in hexadecimal form. 3: Display of this parameter is user-defined. Please refer to the servo drive user manual.

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Page =1;
U16 Index =0;
U16 ParaType = 0, DataSize=0, DataType = 0;

Status = _ECAT_Slave_DeltaServo_Read_Parameter_Info (CardNo, AxisNo, SlotNo, Page,
Index, &ParaType, &DataSize, &DataType);
```

22.4 _ECAT_Slave_DeltaServo_Set_Velocity_Limit

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Velocity_Limit (U16 CardNo, U16 AxisNo, U16 SlotNo, U32 LimitValue)

■ Purpose

This is for setting Delta servo motor’s max. speed.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
AxisNo	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
LimitValue	U32	RPM	Speed limit

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U32 LimitValue = 100;

Status=_ECAT_Slave_DeltaServo_Set_Velocity_Limit (CardNo, AxisNo, SlotNo, LimitValue);
```

22.5 _ECAT_Slave_DeltaServo_Set_Compare_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Compare_Enable(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable, U16 CompareSource, U16 SignalLength, U16 SignalPolarity)

■ Purpose

This is for writing the pulse compare parameter, which is identical to Delta servo parameter P5-59.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Axis number
SlotNo	U16	Number	Slot ID
Enable	U16	Option	Compare function of the servo drive 1: Enable 2: Disable
CompareSource	U16	Option	Source for the pulse compare function: 0: Capture Axes (not supported by A2-E servo drive). 1: AUX ENC (linear scale). (only supported by A2R-E servo drive) 2: External pulse command (not supported by A2-E servo drive). 3: Main ENC (Main encoder).
SignalLength	U16	Millisecond	Trigger duration of the output signal
SignalPolarity	U16	Option	Polarity of the output signal. 0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0, Enable=1, CompareSource=1, SignalLength=1,
SignalPolarity=1;

Status=_ECAT_Slave_DeltaServo_Set_Compare_Enable (CardNo, AxisNo, SlotNo, Enable,
CompareSource, SignalLength, SignalPolarity);
```

22.6 _ECAT_Slave_DeltaServo_Get_Compare_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Get_Compare_Enable(U16 CardNo, U16 AxisNo, U16 SlotNo, U16* Enable, U16* CompareSource, U16* SignalLength, U16* SignalPolarity)

■ Purpose

This is for reading the pulse compare parameter that is written in the servo drive, which is identical to Delta servo parameter P5-59.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Axis number
SlotNo	U16	Number	Slot ID
Enable	U32	Option	Compare function of the servo drive 1: Enable 2: Disable
CompareSource	U32	Option	Source for the pulse compare function: 0: Capture Axes (not supported by A2-E servo drive). 1: AUX ENC (linear scale). (only supported by A2R-E servo drive) 2: External pulse command (not supported by A2-E servo drive). 3: Feedback pulse of the servo drive.
SignalLength	U32	Millisecond	Trigger duration of the output signal
SignalPolarity	U32	Option	Polarity of the output signal. 0: High-potential trigger (NO) 1: Low-potential trigger (NC)

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0, Enable, CompareSource, SignalLength,
SignalPolarity;

Status=_ECAT_Slave_ DeltaServo_Get_Compare_Enable (CardNo, AxisNo, SlotNo,
&Enable, &CompareSource, &SignalLength, &SignalPolarity);
```

22.7 _ECAT_Slave_DeltaServo_Set_Compare_Config

■ Syntax

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Compare_Config(U16 CardNo, U16 AxisNo, U16 SlotNo, U16 CompareNum, I32* ComparePos)

■ Purpose

This is for writing the data array number and values of the pulse compare function to the Delta servo drive.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
AxisNo	U16	Number	Axis number
SlotNo	U16	Number	Slot ID
CompareNum	U16	Quantity	Quantity of the data array.
ComparePos	I32*	Array of value	Data array, which length has to be equal to or greater than value of CompareNum.

■ Example

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
I32 ComparePos;

Status=_ECAT_Slave_DeltaServo_Set_Compare_Config(CardNo, AxisNo, SlotNo,
CompareNum, ComparePos);
```

(This page is intentionally left blank.)

Analog Input Settings (For R1-EC8124D0)

23

This chapter introduces the APIs for analog input settings. Please note that these APIs are only applicable to Delta analog input modules.
Information about general operation of analog input module can be found in chapter 19.

API of analog input settings (for R1-EC8124D0 series)

API	Description
_ECAT_Slave_R1_EC8124_Set_Input_RangeMode	Set the sampling range of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode	Set the sampling rate of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_Enable	Enable/Disable the analog input sampling function of Delta analog input module
_ECAT_Slave_R1_EC8124_Get_Input_RangeMode	Acquire the sampling range of Delta analog input module
_ECAT_Slave_R1_EC8124_Set_Input_AverageMode	Set the average times for the analog input filter of Delta analog input module

23.1 _ECAT_Slave_R1_EC8124_Set_Input_RangeMode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_RangeMode (U16 CardNo,
U16 NodeID, U16 SlotNo, U16 RangeMode)

■ Purpose

This is for setting the sampling range of Delta analog input module.

Note: Delta analog input module only allows users to measure voltage signal.

To measure the input current, you need to modify the wriing of the analog input module first. (Please refer to the user manual of Delta analog input model regarding the wiring for current measurement.) After finishing the wiring, users can convert the measurement into current by using the circuit with 250Ω- resistor.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
RangeMode	U16	Option	Output range setting: 0: -5 V ~ 5 V (default) 1: -10 V ~ 10 V

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 RangeMode=1;

Status= _ECAT_Slave_R1_EC8124_Set_Input_RangeMode (CardNo, NodeID,SlotNo,
RangeMode);
```

23.2 _ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 RangeMode)

■ Purpose

This is for setting the sampling rate of Delta analog input module.

■ Parameter

<table>
<tr><th>Name</th><th>Data type</th><th>Property</th><th>Description</th></tr>
<tr><td>CardNo</td><td>U16</td><td>Number</td><td>Card number</td></tr>
<tr><td>NodeID</td><td>U16</td><td>Number</td><td>Node ID</td></tr>
<tr><td>SlotNo</td><td>U16</td><td>Number</td><td>Slot ID</td></tr>
<tr><td>RangeMode</td><td>U16</td><td>Option</td><td>Sampling rate setting
<table>
<tr><th>Option</th><th>Sampling rate (kHz)</th></tr>
<tr><td>0</td><td>200</td></tr>
<tr><td>1</td><td>100</td></tr>
<tr><td>2</td><td>50</td></tr>
<tr><td>3</td><td>25</td></tr>
<tr><td>4</td><td>12.5</td></tr>
<tr><td>5</td><td>6.25</td></tr>
<tr><td>6</td><td>3.125</td></tr>
</table></td></tr>
</table>

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Mode=2;

Status= _ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode (CardNo, NodeID,SlotNo,
Mode);
```

23.3 _ECAT_Slave_R1_EC8124_Set_Input_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ Purpose

This is for enabling/disabling the analog input sampling function of Delta analog input module..

This API has to be executed before measuring or acquiring the input value.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Disable 1: Enable

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC8124_Set_Input_Enable (CardNo, NodeID,SlotNo, Enable);
```

23.4 _ECAT_Slave_R1_EC8124_Get_Input_RangeMode

■ Syntax

```
U16 PASCAL _ECAT_Slave_R1_EC8124_Get_Input_RangeMode (U16 CardNo,
U16 NodeID, U16 SlotNo, U16* RangeMode)
```

■ Purpose

This is for aquiring the sampling range of Delta analog input module.

Note: Delta analog input module only allows users to measure voltage signal.

To measure the input current, you need to modify the wiring of the analog input module first. (Please refer to the user manual of Delta analog input model regarding the wiring for current measurement.) After finishing the wiring, users can convert the measurement into current by using the circuit with 250Ω- resistor.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
RangeMode	U16*	Option	Output range setting: 0: -5 V ~ 5 V (default) 1: -10 V ~ 10 V

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 RangeMode;

Status= _ECAT_Slave_R1_EC8124_Get_Input_RangeMode (CardNo, NodeID,SlotNo,
&RangeMode);
```

23.5 _ECAT_Slave_R1_EC8124_Set_Input_AverageMode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_AverageMode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Avg_Times)

■ Purpose

This is for setting the average times of the analog input signal filter of Delta analog input module. EtherCAT Master will average out the current value from 1st to127th data based on the setting count.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Avg_Times	U16	Value	Average times for the analog input, which range is 1 ~ 127.

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Avg_Times =5;

Status= _ECAT_Slave_R1_EC8124_Set_Input_AverageMode (CardNo, NodeID,SlotNo,
Avg_Times);
```

(This page is intentionally left blank.)

Analog Output Settings (For R1-EC9144D0) 24

This chapter introduces the APIs for analog output settings of Delta modules (R1-EC9144D0). Please note that these APIs are only applicable to Delta modules. General function for analog output operation of Delta modules can be found in chapter 19.

API list of analog output settings (for R1-EC9144D0 series)

API	Description
_ECAT_Slave_R1_EC9144_Set_Output _RangeMode	Set the output range of Delta analog output module
_ECAT_Slave_R1_EC9144_Set_Output _Enable	Enable/Disable the analog output of Delta module
_ECAT_Slave_R1_EC9144_Get_Output _ReturnCode	Acquire the operation status of Delta analog output module

24.1 _ECAT_Slave_R1_EC9144_Set_Output _RangeMode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC9144_Set_Output_RangeMode(U16 CardNo, U16 NodeID, U16 SlotNo, U16 RangeMode)

■ Purpose

This is for setting the output range of Delta analog output module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
RangeMode	U16	Option	Output range of analog output: 0: 0 ~ 5 V (Default) 1: 0 ~ 10 V 2: -5 V ~ 5 V 3: -10 V ~ 10 V 4: 4 ~ 20 mA 5: 0 ~ 20 mA 6: 0 ~ 24 mA

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 RangeMode=3;

Status= _ECAT_Slave_R1_EC9144_Set_Output_RangeMode (CardNo, NodeID,SlotNo,
RangeMode);
```

24.2 _ECAT_Slave_R1_EC9144_Set_Output_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC9144_Set_Output_Enable(U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ Purpose

This is for enabling/disabling the analog output of Delta module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Option	0: Disable 1: Enable

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC9144_Set_Output_Enable (CardNo, NodeID, SlotNo,
Enable);
```

24.3 _ECAT_Slave_R1_EC9144_Get_Output _ReturnCode

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC9144_Get_Output_ReturnCode (U16 CardNo,
U16 NodeID, U16 SlotNo, U16 *ReturnCode)

■ Purpose

This is for aquiring the operation status of Delta analog output module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
*ReturnCode	U16*	Return code	Please see ths list of _ECAT_Slave_AIO_Get_Output_ReturnCode in the table below.

■ Example

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 ReturnCode;
Status= _ECAT_Slave_R1_EC9144_Get_Output_ReturnCode (CardNo, NodeID, SlotNo,
&ReturnCode);
```

List of _ECAT_Slave_AIO_Get_Output_ReturnCode

Bit	Description
0	Iout_3 error: This bit is set if an error is detected on current output channel 3.
1	Iout_2 error: This bit is set if an error is detected on current output channel 2.
2	Iout_1 error: This bit is set if an error is detected on current output channel 1.
3	Iout_0 error: This bit is set if an error is detected on current output channel 0.
4	Vout_3 error: This bit is set if an error is detected on voltage output channel 3.
5	Vout_2 error: This bit is set if an error is detected on voltage output channel 2.
6	Vout_1 error: This bit is set if an error is detected on voltage output channel 1.
7	Vout_0 error: This bit is set if an error is detected on voltage output channel 0.
8	Overheat: This bit is set If temperature of the AD converter chip is over 150°C.
9	Ramp active: This bit is set while any one of the output channel is slewing.
10	PEC error: Denotes a PEC error on the last data-word received over the SPI interface.
11	PEC enabled: This is a read only bit. It allows the user to verify the status of the packet error checking feature.
12	DC-DC3: In current output mode, this bit is set on channel 3 if the dc-to-dc converter cannot maintain compliance (it may be reaching its maximum voltage. In voltage output mode, this bit is set if, on channel 3, the dc-to-dc converter is unable to regulate to 15 V as expected. When this bit is set, it does not result in the error pin going high.
13	DC-DC2: In current output mode, this bit is set on channel 2 if the dc-to-dc converter cannot maintain compliance (it may be reaching its maximum voltage.) In voltage output mode, this bit is set if, on channel 2, the dc-to-dc converter is unable to regulate to 15 V as expected. When this bit is set, it does not result in the error pin going high.
14	DC-DC1: In current output mode, this bit is set on channel 1 if the dc-to-dc converter cannot maintain compliance (it may be reaching its maximum voltage.). In voltage output mode, this bit is set if, on channel 1, the dc-to-dc converter is unable to regulate to 15 V as expected. When this bit is set, it does not result in the error pin going high.
15	DC-DC0: In current output mode, this bit is set on channel 0 if the dc-to-dc converter cannot maintain compliance (it may be reaching its maximum voltage.). In voltage output mode, this bit is set if, on channel 1, the dc-to-dc converter is unable to regulate to 15 V as expected. When this bit is set, it does not result in the error pin going high.

Auto Recording Function of Motion Axis 25

This chapter introduces the APIs of auto recording function for the motion axis. With this function, the EtherCAT system’s kernel will automatically save the relvevant data of the servo motion axis every communication cycle. Up to 800 data can be saved in kernel of PAC RTX version and up to 200 data in kernel of EtherCAT motion card version.

API list of auto recording function for motion axis

API	Description
_ECAT_Slave_Record_Data_Set_Type	Set the recording data type of specified axis
_ECAT_Slave_Record_Data_Set_Enable	Enable/Disable the recording function of specified axis
_ECAT_Slave_Record_Data_Get_Cnt	Acquire the data entry number of specified axis
_ECAT_Slave_Record_Data_ReadData	Acquire the recorded data of specified axis
_ECAT_Slave_Record_Clear_Data	Delete the saved record of specified axis
_ECAT_Slave_Record_Multi_Set_Enable	Enable/Disable the recoding function of specified multiple axes
_ECAT_Slave_Record_Multi_Clear_Data	Delete the saved record of specified multiple axes

25.1 _ECAT_Slave_Record_Set_Type

■ Syntax

```
U16 PASCAL _ECAT_Slave_Record_Set_Type (U16 CardNo, U16 NodeID,
U16 SlotNo, U16 MonitorIndex, U16 IOType, U16 Index, U16 SubIndex)
```

■ Purpose

Set the recording data type of specified axis.

EtherCAT master saves one data in each communication cycle. Each slave can save 8 different OD codes.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
MonitorIndex	U16	Number	The group number of OD data to be recorded (0 ~ 7). Each slot can record 8 OD data.
IOType	U16	Option	Set the mode of recorded OD code. 0: Input 1: Output
Index	U16	Index	OD to be recorded. (Find more information in CANopen documentation)
SubIndex	U16	SubIndex	Sub OD to be recorded. (Find more information in CANopen documentation)

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 MonitorIndex =0;
U16 IOType =0, Index=0x607A, SubIndex=1;

Status= _ECAT_Slave_Record_Set_Type (CardNo, NodeID, SlotNo, MonitorIndex,
IOType, Index, SubIndex);
```

25.2 _ECAT_Slave_Record_Set_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_Record_Set_Enable (U16 CardNo, U16 NodeID,
U16 SlotNo, U16 Enable)

■ Purpose

Enable/Disable the recording function of specified axis.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Enable	U16	Value	This value will refer to the bit setting and enable/disable the recording function for the 8 groups of OD data.

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 MonitorIndex =0;
U16 IOType =0, Index=0x607A, SubIndex=1;

Status= _ECAT_Slave_Record_Set_Type (CardNo, NodeID, SlotNo, MonitorIndex,
IOType, Index, SubIndex);

// Enable the recording function of group number 1 ~ 3 of Node ID 1. Disable the recordind
function of group number 4 ~ 8.
U16 Enable =0x07;
Status= _ECAT_Slave_Record_Set_Enable (CardNo, NodeID, SlotNo, Enable);
```

25.3 _ECAT_Slave_Record_Get_Cnt

■ Syntax

U16 PASCAL _ECAT_Slave_Record_Get_Cnt (U16 CardNo, U16 NodeID,
U16 SlotNo, U16 *Cnt)

■ Purpose

Acquire the data entry number of the specified axis.

EtherCAT master saves one data in each communication cycle. Each slave can save up to 8 groups of OD data.

Note:

1. It can save 800 data in PAC RTX version and 200 data in EtherCAT motion card. Users can use this API to check the buffer zone status. Then, use the API in 25.4 to access the data and save it to another space.
2. When accessing one data by the API (_ECAT_Slave_Record_Read_Data) in 25.4, the acquired data numbers will reduce 1 automatically.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Cnt	U16*	Value	Acquire current data count of the specified axis

■ Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 MonitorIndex =0;
U16 IOType =0, Index=0x607A, SubIndex=1;
U16 Enable =0x07;
U16 Cnt ;

Status= _ECAT_Slave_Record_Set_Type (CardNo, NodeID, SlotNo, MonitorIndex,
IOType, Index, SubIndex);
Status= _ECAT_Slave_Record_Set_Enable (CardNo, NodeID, SlotNo, Enable);

while (1)
{
    Status= _ECAT_Slave_Record_Get_Cnt (CardNo, NodeID, SlotNo, &Cnt);
}
```

25.4 _ECAT_Slave_Record_Read_Data

Syntax

U16 PASCAL _ECAT_Slave_Record_Read_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U32 *Data)

Purpose

Acquire the recording data of the specified axis. Users can withdraw the data (one by one) in the buffer and save it to the other files.

Note:

1. It can save 800 data in PAC RTX version and 200 data in EtherCAT motion card. Users can use ththe API in section 25.3 to check the buffer zone status.
2. When accessing one data by this API, the acquired data numbers will reduce 1 automatically.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Data	U32*	Data	Aquire the recording data of the specified axis. Please note that this API will return 8 data at a time so it requires sufficient space for storage (Data[8]). If _ECAT_Slave_Record_Data_Set_Type is not executed, this API will return 0.

Example

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 MonitorIndex =0;
U16 IOType =0, Index=0x607A, SubIndex=1;
U16 Enable =0x07;
U16 Cnt ;
U32 Data[8];

Status= _ECAT_Slave_Record_Set_Type (CardNo, NodeID, SlotNo, MonitorIndex,
IOType, Index, SubIndex);
Status= _ECAT_Slave_Record_Set_Enable (CardNo, NodeID, SlotNo, Enable);

while (1)
{
    Status= _ECAT_Slave_Record_Get_Cnt (CardNo, NodeID, SlotNo, &Cnt);
    if (Cnt>0)
    {
```

```
                Status= _ECAT_Slave_Record_Read_Data (CardNo, NodeID, SlotNo, Data);
        }
}
```

25.5 _ECAT_Slave_Record_Clear_Data

- **Syntax**

U16 PASCAL _ECAT_Slave_Record_Clear_Data (U16 CardNo, U16 NodeID, U16 SlotNo)

- **Purpose**

Delete the saved record of the specified axis.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID

- **Example**

```
U16 Status = 0;
U16 CardNo=16 , NodeID=1, SlotNo=0;

// Clear the saved data of the specified axis
Status= _ECAT_Slave_Record_Clear_Data (CardNo, NodeID, SlotNo);
```

25.6 _ECAT_Slave_Record_Multi_Set_Enable

Syntax

```
U16 PASCAL _ECAT_Slave_Record_Multi_Set_Enable (U16 CardNo, U16 NodeNum,
U16 *NodeIDArray, U16 *SlotIDArray, U16 Enable)
```

Purpose

Enable/Disable the recoding function of specified multiple axes.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeNum	U16	Quantilty	Quantity of nodes
NodeIDArray	U16*	Node array	Set the Node ID array for recording function of multiple axes, which quantity is indentical to AxisNum. NodeIDArray [0] is for specifying the 1st node to be used. NodeIDArray [1] is for specifying the 2nd node to be used and so on.
SlotIDArray	U16*	Slot array	Data array of slot ID, which quantity is identical to AxisNum.
Enable	U16	Value	This value will refer to the bit setting and determine whether to enable/disable the recording function for the 8 groups of OD data.

Example

```
U16 Status = 0;
U16 CardNo=16 , NodeNum = 2, NodeID[2]={0, 1}, SlotNo[2]={0, 0};
// Enable the recording function of group number 1 ~ 3 of Node ID 1. Disable the recordind
function of group number 4 ~ 8.
U16 Enable =0x07;

// Enable the recoding function of multiple axes.
Status= _ECAT_Slave_Record_Multi_Set_Enable (CardNo, NodeNum, NodeIDArray,
SlotIDArray, Enable);
```

25.7 _ECAT_Slave_Record_Multi_Clear_Data

Syntax

U16 PASCAL _ECAT_Slave_Record_Multi_Clear_Data (U16 CardNo, U16 NodeNum, U16 *NodeIDArray, U16 *SlotIDArray)

Purpose

Delete the saved record of specified multiple axes.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeNum	U16	Quantity	Quantity of nodes
NodeIDArray	U16*	Node array	Set the Node ID array for recording function of multiple axes, which quantity is indentical to AxisNum. NodeIDArray [0] is for specifying the 1st node to be used. NodeIDArray [1] is for specifying the 2nd node to be used and so on.
SlotIDArray	U16*	Slot array	Data array of slot ID, which quantity is identical to AxisNum.

Example

```
U16 Status = 0;
U16 CardNo=16 , NodeNum = 2, NodeIDArray[2] = {0, 1}, SlotIDArray[2] = {0, 0};
// Clear recording data of multiple axes.
Status= _ECAT_Slave_Record_Multi_Clear_Data (CardNo, NodeNum, NodeIDArray,
SlotIDArray);
```

(This page is intentionally left blank.)

Operaion of Local Digital I/O

26

This chaper introduces the APIs of the built-in local digital I/O, such as setting and reading the input and output status of GPIO (General-purpose input/output) on the motion card.

API List of operating local digital I/O

API	Description
_ECAT_GPIO_Set_Output	Control the output status of the GPIO on the motion card
_ECAT_GPIO_Get_Output	Read the output status of the GPIO on the motion card
_ECAT_GPIO_Get_Input	Read the input status of the GPIO on the motion card

26.1 _ECAT_GPIO_Set_Output

Syntax

U16 PASCAL _ECAT_GPIO_Set_Output (U16 CardNo, U16 Data)

Purpose

This is for controling the output status of the GPIO (General-purpose input/output) on the motion card. To set the output status of bit 15, bit 14, …, bit 0, users can convert the value to decimal format (from left to right).

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Data	U16	Value	The output value of GPIO

Example

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Data = 0xF;
Status= _ECAT_GPIO_Set_Output( CardNo, Data );
```

26.2 _ECAT_GPIO_Get_Output

■ Syntax

U16 PASCAL _ECAT_GPIO_Get_Output (U16 CardNo, U16* Data)

■ Purpose

This is for reading the output status of the GPIO on the motion card. To get the output status of bit 15, bit 14, …, bit 0, users can convert the value to binary format (from left to right).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Data	U16*	Value	The output status of GPIO

■ Example

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Status= _ECAT_GPIO_Get_Output( CardNo, &Data );
```

26.3 _ECAT_GPIO_Get_Input

■ Syntax

U16 PASCAL _ECAT_GPIO_Get_Input (U16 CardNo, U16* Data)

■ Purpose

This is for reading the input status of the GPIO (General-purpose input/output) on the motion card. To get the input status of bit 15, bit 14, …, bit 0, users can convert the value to binary format (from left to right).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Data	U16*	Value	The input status of GPIO

■ Example

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Status= _ECAT_GPIO_Get_Input( CardNo, &Data );
```

(This page is intentionally left blank.)

High-Speed Pulse Compare Function

27

This chapter introduces the APIs for high speed pulse compare of motion cards. With the pulse input interface, FPGA real-time compare function of the motion card can be carried out. It can compare the pulse number and simultaneously ouput the differential signal to control the camera.

All advanced type motion cards of Delta provide high speed pulse compare function. As soon as the pulse compare function is carried out, it sends a differential signal for triggering the camera to take photos.

The motion card has two high speed pulse input channels. There are two ways to trigger the camera function, externally or internally trigger (from the PC). Delta provides two sets of differential signal output points on one side of the motion card as well as the other side that is inserted in the PC. The channel of both internal and external output points will be triggered simultaneously when condition is fulfilled.

The high speed pulse compare function has two types, which is determined by two channels for outputting differential signals. The channel 0 of differential singal (PIN11 &12 of CN2 and CN9), hereafter simplified as channel 0, is for pulse comparing that is performed at fixed intervals. The channel 1 of differential singal (PIN13 &14 of CN2 and CN11), simplified as channel 1, is for pulse comparing that is performed at user-defined intervals and its trigger position is user-defined. The two sets of differential signals can share both channel 0 and 1.

API list of high speed pulse compare function

API	Description
_ECAT_Compare_Set_Channel_Position	Overwrite a position value for the specified channel
_ECAT_Compare_Get_Channel_Position	Acquire the current position value of the specified channel
_ECAT_Compare_Set_Ipulser_Mode	Set the mode of pulse input for the specified channel
_ECAT_Compare_Set_Channel_Direction	Set the pulse direction of the specified channel
_ECAT_Compare_Set_Channel_Trigger_Time	Set the trigger retaining time for the specified channel
_ECAT_Compare_Set_Channel_One_Shot	Force the trigger manually once for the specified channel
_ECAT_Compare_Set_Channel_Source	Set the compare source for the specified channel
_ECAT_Compare_Set_Channel_Enable	Enable/disable the compare function for the specified channel
_ECAT_Compare_Channel0_Position	Set the parameters for triggering the signal at a fixed pulse interval of channel 0
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO	Set the parameters for triggering the signal at a fixed pulse interval of channel 0, which is enabled / disabled by GPIO
_ECAT_Compare_Set_Channel1_Output_Enable	Enable/Disable the trigger function of channel 1 (user-defined pulse intervals)
_ECAT_Compare_Set_Channel1_Output_Mode	Set the output mode of channel 1
_ECAT_Compare_Get_Channel1_IO_Status	Acquire the operation status of channel 1
_ECAT_Compare_Set_Channel1_GPIO_Out	Set the output status of the PIN15 on CN2 of GPIO
_ECAT_Compare_Set_Channel1_Position_Table	Set the pulse data of channel 1 (user-defined pulse intervals).
_ECAT_Compare_Set_Channel1_Position_Table_Level	Set the pulse data of channel 1 and its user-defined active level for triggering signals
_ECAT_Compare_Get_Channel1_Position_Table_Count	Acquire the current trigger counts of channel 1
_ECAT_Compare_Set_Channel_Polarity	Set the trigger level of the compare function
_ECAT_Compare_Reuse_Channel1_Position_Table	Re-execute the compare function of channel 1 once
_ECAT_Compare_Reuse_Channel1_Position_Table	Re-execute the compare function of channel 1 once, which the trigger level is user-defined.

27.1 _ECAT_Compare_Set_Channel_Position

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel_Position(U16 CardNo, U16 CompareChannel, I32 Position)

■ Purpose

Overwrite a new position value (pulse) for the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_Channel	U16	Number	Channel No., which range is 0 ~ 1.
Position	I32	Pulse	The new position value to be set for the specified channel

■ Example

```
U16 CardNo = 0;
U16 Compare_Channel = 0;
I32 position = 0;

U16 status = _ECAT_Compare_Set_Channel_Position (CardNo, compare_channel, position);
```

27.2 _ECAT_Compare_Get_Channel_Position

■ Syntax

```
U16 PASCAL _ECAT_Compare_Get_Channel_Position (U16 CardNo,
U16 compare_Channel, I32 *position)
```

■ Purpose

Acquire the current position value (pulse) of the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_Channel	U16	Number	Channel No., which range is 0 ~ 1.
Position	I32*	Pulse	Acquire the current position value of the specified channel.

■ Example

```
U16 CardNo = 0;
U16 Compare_channel = 0;
I32 position;

U16 status = _ECAT_Compare_Get_Channel_Position (CardNo, compare_channel,
&position);
```

27.3 _ECAT_Compare_Set_Ipulser_Mode

■ **Syntax**

U16 PASCAL _ECAT _Compare_Set _Ipulser_Mode (U16 CardNo, U16 mode)

■ **Purpose**

Set the mode of pulse input for the specified channel. There are two modes available, AB phase or CW/CCW.

■ **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Mode	U16	Option	0: AB Phase 1: CW/CCW

■ **Example**

```
U16 CardNo = 0;
U16 mode = 0;  //AB Phase

U16 status = _ECAT _Compare_Set _Ipulser_Mode (CardNo, mode);
```

27.4 _ECAT_Compare_Set_Channel_Direction

■ Syntax

U16 PASCAL _ECAT _Compare_Set _Channel_Direction (U16 CardNo,
U16 compare_channel, U16 dir)

■ Purpose

Set the pulse direction of the specified channel. With this API, modifying the wiring will not be required if pulse direction has to be alternated.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_channel	U16	Number	Channel No., which range is 0 ~ 1.
Dir	U16	Option	0: Forward 1: Inverse

■ Example

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 dir = 1; //Inverse

U16 status = _ECAT_Compare_Set _Channel_Direction (CardNo, compare_channel, dir);
```

27.5 _ECAT_Compare_Set_Channel_Trigger_Time

■ Syntax

U16 PASCAL _ECAT _Compare_Set _Channel_Trigger_Time (U16 CardNo,
U16 compare_channel, U32 time_us)

■ Purpose

Set the trigger retainining time for the specified channel.

Note:

1. When carrying out compare function with channel 0, the minimum trigger time is set to 1 us. If 0 is input, it displays 0.8 us.
2. When carrying out compare function with channel 1, the minimum trigger time is 3 us. If the input value is less than 3, it displays 3 us.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_channel	U16	Number	Channel No., which range is 0 ~ 1.
Time_us	U32	Time	Input the retaining time for each trigger. (Unit: 0.001 us)

■ Example

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 time_us = 20; //20us

U16 status = _ECAT_Compare_Set _Channel_Trigger_Time(CardNo, compare_channel,
time_us);
```

27.6 _ECAT_Compare_Set_Channel_One_Shot

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel_One_Shot (U16 CardNo,
U16 compare_channel)

■ Purpose

Allow the trigger of the specified channel to generate one trigger signal.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_channel	U16	Number	Channel No., which range is 0 ~ 1.

■ Example

```
U16 CardNo = 0;
U16 compare_channel=0;

U16 status = _ECAT_Compare_Set _Channel_One_Shot (CardNo, compare_channel);
```

27.7 _ECAT_Compare_Set_Channel_Source

■ Syntax

U16 PASCAL _ECAT_Compare_Set _Channel_Source (U16 CardNo,
U16 compare_channel, U16 source)

■ Purpose

Set the compare source for the specified channel.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_channel	U16	Number	Channel No., which range is 0 ~ 1.
Source	U16	Option	0: QA1 and QB1 on CN2 connector 1: QA2 and QB2 on CN2 connector

■ Example

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 source = 0;

U16 status = _ECAT_Compare_Set _Channel_source (CardNo, compare_channel, source);
```

27.8 _ECAT_Compare_Set_Channel_Enable

■ Syntax

U16 PASCAL _ECAT_Compare_Set _Channel_Enable (U16 CardNo,

U16 compare_channel,U16 enable)

■ Purpose

Enable/disable the high speed compare function of the specified channel.

Note:

1. Channel 0 can only be enabled/disabled with this API.
2. Apart from this API, channel 1 needs to be enabled/disabled with API "_ECAT_Compare_Set_Channel1_Output_Enable". Otherwise, this channel will not be triggered even when the pulse is matched.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Compare_channel	U16	Number	Channel No., which range is 0 ~ 1.
enable	U16	Option	0: Disable the compare function. 1: Enable the compare function.

■ Example

```
U16 CardNo = 0, ;
U16 compare_channel = 0;
U16 enable = 1;//Enable the high-speed compare function.

U16 status = _ECAT_Compare_Set Channel_Enable (CardNo, compare_channel, enable);
```

27.9 _ECAT_Compare_Channel0_Position

■ Syntax

U16 PASCAL _ECAT_Compare_Channel0_Position(U16 CardNo, I32 Start, U16 Dir, U16 Interval, U32 TriggerCount);

■ Purpose

Set the parameters for triggering the signal at a fixed pulse interval of channel 0.

Note:

1. For the trigger retaining time, please refer to description of API "_ECAT_Compare_Set_Channel_Trigger_Time".
2. Output signal of channel 0 is triggered via PIN11 and PIN12 of CN2 and CN9 connector on motion card PCI-L221-B1. These two sets of points will both be triggered when pulses are matched.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Start	I32	Pulse	Start position of the compare.
Dir	U16	Option	Direction 0: forward 1: backward
Interval	U16	Pulse number	Pulse interval for carrying out the compare.
Trigger_cnt	U32	Amount	Triggering count of the compare function.

■ Example

```
U16 CardNo = 0;
I32 start = 100000;
U16 dir = 0;
U16 Interval = 10; //10 pulse
U32 trigger_cnt = 50000;

U16 status=_ECAT_Compare_Channel0_Position(CardNo, start, dir, interval, trigger_cnt);
/* The motion card starts comparing pulses from position 100000 and channel 0 will be triggered
once every 10 pulses with total trigger count 50000. */
/* That is, when it is at position 100010, 100020, 100030, …, 600000 (pulse), channel 0 will be
triggered once.*/
```

■ Description

The motion card starts comparing pulses from the start position and trigger a differential signal based on the set pulse interval (every time a given pulse number is reached). Meanwhile, the motion card carries on comparing the pulses for the next fixed interval to trigger the differential signal of channel 0. Then, it stops comparing pulses when the total trigger count is reached.

27.10 _ECAT_Compare_Set_Channel0_Trigger _By_GPIO

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel0_Trigger_By_GPIO (U16 CardNo, U16 dir, U16 interval, I32 trigger_cnt)

■ Purpose

Set the parameters for triggering the signal at a fixed pulse interval of channel 0, which function is triggered via PIN10 of CN2 connector (GPIO, general-purpose input/output)

Note:

1. Before GPIO is enabled, please make sure channel 0 is enabled by API "_ECAT_Compare_Set_Channel_Enable".
2. About the triggering retaining time, please refer to section 27.5 "_ECAT_Compare_Set_Channel_Trigger_Time".
3. Output signal of channel 0 is triggered via PIN11 and PIN12 of CN2 and CN9 connector on motion card PCI-L221-B1. These two sets of points will both be triggered when pulses are matched.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Dir	U16	Numeric value	0: forward 1: backward
Interval	U16	Option	Pulse number of each interval
Trigger _cnt	I32	Pulse number	Pulse interval for carrying out the compare.

■ Example

```
U16 CardNo = 0;
U16 Dir = 1;
U16 Interval = 10;
I32 trigger_cnt = 50000;

U16 status = _ECAT_Compare_Set_Channel0_Trigger_By_GPIO (CardNo, Dir, Interval,
trigger_cnt);
/* The motion card starts comparing pulses from position 100000 and channel 0 will be triggered
once every 10 pulses with total trigger count 50000. */
/* That is, when the pulse is 100010, 100020, 100030, …, 600000, channel 0 will be triggered
once.*/
```

■ Description

The comparing method is the same as that described in Chaper 27.9 "_ECAT_Compare_Channel0_Position". However, to enable this compare function, the GPIO (PIN 10 of CN2) has to be set to on.

27.11 _ECAT_Compare_Set_Channel1_Output_Enable

Syntax

U16 PASCAL _ECAT_Compare_Set_Channel1_Output_Enable (U16 CardNo, U16 on_off)

Purpose

Enable/Disable the compare function with user-defined interval of channel 1.

Note:

1. Before GPIO is enabled, please make sure channel 0 is enabled by API "_ECAT_Compare_Set_Channel_Enable".
2. About the triggering retaining time, please refer to section 27.5 "_ECAT_Compare_Set_Channel_Trigger_Time".
3. Output signal of channel 1 is triggered via PIN13 and PIN14 of CN2 and CN11 connector on motion card PCI-L221-B1. These two sets of points will both be triggered when pulses are matched.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
On_off	U16	Option	0: Off 1: On

Example

```
U16 CardNo = 0;
U16 on_off = 1;

U16 status = _ECAT_Compare_Set_Channel1_Output_Enable (CardNo, on_off);
```

Description

The motion card will start comparing the current pulse and the array of pulse position set by API "_ECAT_Compare_Set_Channel1_Position_Table", and trigger output signal of channel 1 if the pulses are matched.

27.12 _ECAT_Compare_Set_Channel1_Output_Mode

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel1_Output_Mode (U16 CardNo, U16 Mode)

■ Purpose

Set the output mode of channel 1. There are two modes available: Output at user-defined pulse intervals or output at user-defined intervals and with user-defined trigger level.

Note: Output signal of channel 1 is triggered via PIN13 and PIN14 of CN2 or CN11 connector on motion card PCI-L221-B1. These two sets of points will both be triggered when pulses are matched.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Mode	U16	Option	0: Output mode of user-defined pulse intervals 1: Output mode of user-defined intervals and trigger level

■ Example

```
U16 CardNo = 0;
U16 mode = 1;
U16 status = _ECAT_Compare_Set_Channel1_Output_Mode (CardNo, mode);
```

■ Description

The two output modes of channel 1 will be illustrated as follows.

1. Output mode of user-defined pulse interval. User can define the position for performing the compare function**.**

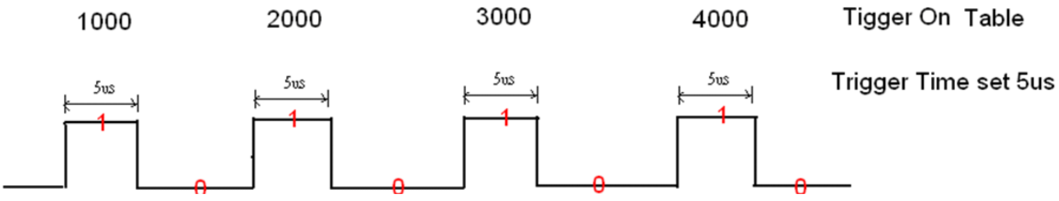


Figure 27.12.1 Output mode of user-defined pulse interval

Note:

1. The trigger-on position is set by “Pos_table” of the API “_ECAT_Compare_Set_Channel1_Position_Table”.
2. For the trigge retaining time, please refer to section 27.5 API “_ECAT_Compare_Set_Channel_Trigger_Time”.

2. Output mode of user-defined intervals and trigger level

In this mode, you can define the position (pulse) for the trigger, which is the same as mode 0.

Meanwhile, you can also set the trigger level (high / low). See the figure below.

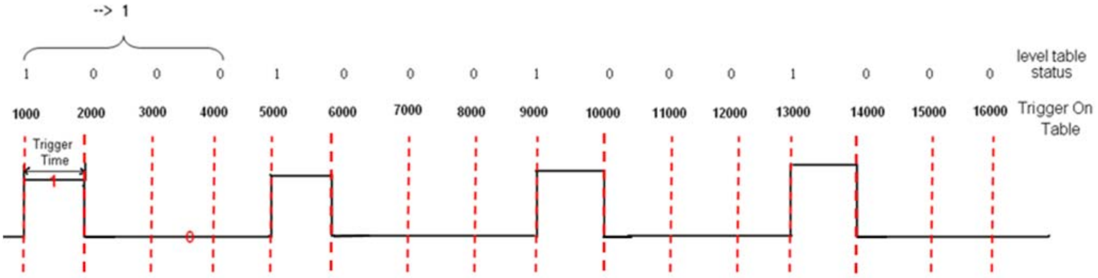


Figure 27.12.2 Output mode of user-defined intervals and trigger level

Note:

1. The trigger position is set by “level_table” of the API “_ECAT_Compare_Set_Channel1_Position_Table_Level”.
2. The trigger retaining time is not influenced by API setting of “_ECAT_Compare_Set_Channel_Trigger_Time”. Instead, the trigger time is the time for reaching the next position. (Ex: the time it takes from position 1000 to 2000.)

27.13 _ECAT_Compare_Get_Channel1_IO_Status

■ Syntax

U16 PASCAL _ECAT_Compare_Get_Channel1_IO_Status (U16 CardNo, U16* io_status)

■ Purpose

Read the operation status of channel 1.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Io_status	U16*	Numeric value	The operation status of channel 1. Bit0: Status of PIN 10 on CN2 (GPIO IN); Bit1: The power supply (hardware) status 1: 5V 0: 0V (IC error) Bit2: Whether the function of user-defined trigger level is used. (_ECAT_Compare_Set_Channel1_Position_Table_Level)

■ Example

```
U16 CardNo = 0;
U16 io_status;

U16 status = _ECAT_Compare_Get_Channel1_IO_Status (CardNo,& io_status);
```

27.14 _ECAT_Compare_Set_Channel1_GPIO_Out

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel1_GPIO_Out (U16 CardNo, U16 on_off)

■ Purpose

Set the output status of PIN15 on CN2 of General-purpose input/output (GPIO).

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
On_off	U16	Option	0: Off 1: On

■ Example

```
U16 CardNo = 0;
U16 on_off = 1;

U16 status = _ECAT_Compare_Set_Channel1_GPIO_Out (CardNo, on_off);
```

27.15 _ECAT_Compare_Set_Channel1_Position_Table

Syntax

U16 PASCAL _ECAT_Compare_Set_Channel1_Position_Table (U16 CardNo, I32* pos_table, U32 table_size)

Purpose

Set the pulse data of user-defined pulse intervals for channel 1. The max. entry of position data is 100000.

Note:

1. For the trigger retaining time, please refer to description of API "_ECAT_Compare_Set_Channel_Trigger_Time".
2. Output signal of channel 1 is triggered via PIN13 and PIN14 of CN2 or CN11 connector on motion card PCI-L221-B1. These two signals will both be triggered when pulses are matched.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Pos_table	I32*	Data array	The position for comparing pulses at user-defined intervals.
Table_size	U32	Number	Data size to be compared, which value has to be identical to the array size of Pos_table. The max. is 100000.

Example

```
U16 CardNo = 0;
I32 pos_table[4] = {1000,2000,3000,4000};
U32 table_size =4;

U16 status = _ECAT_Compare_Set_Channel1_Position_Table (CardNo, pos_table,
table_size);
```

- **Output mode of user-defined pulse interval. User can define the position for performing compare function.**

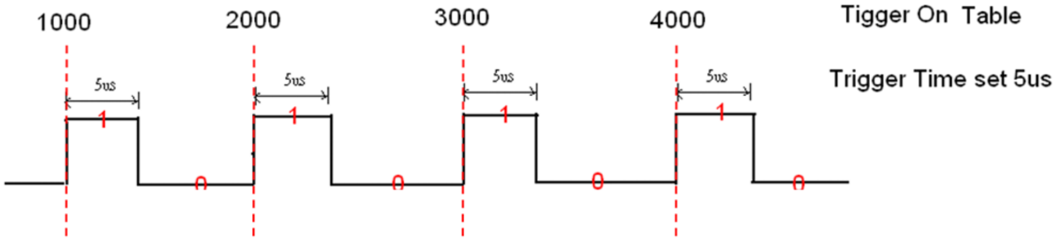


Figure 27.12.3 Output mode of user-defined pulse interval

Note:

1. The trigger-on position is set by “Pos_table” of the API “_ECAT_Compare_Set_Channel1_Position_Table”.
2. For the trigger retaining time, please refer to description of API “_ECAT_Compare_Set_Channel_Trigger_Time”.

27.16 _ECAT_Compare_Set_Channel1_Position_Table_Level

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel1_Position_Table_Level (U16 CardNo, I32* pos_table, U32* level_table, U32 table_size)

■ Purpose

Set the pulse data of channel 1 and its user-defined active level for triggering signals. The max. data entry is 100000.

Note:

1. For the trigger retaining time, please refer to description of API "_ECAT_Compare_Set_Channel_Trigger_Time".
2. Output signal of channel 1 is triggered via PIN13 and PIN14 of CN2 or CN11 connector on motion card PCI-L221-B1. These two signals will both be triggered when pulses are matched.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Pos_table	I32*	Data array	The position for comparing pulses at user-defined intervals.
Level_table	U32*	Data array	The active level for triggering signals; Translate the trigger state at the 32 positions into a 32-bit value.
Table_size	U32	Number	Data size to be compared, which value has to be identical to the array size of Pos_table. The max. is 100000.

■ Example

```
U16 CardNo = 0;
I32 pos_table[16] ={1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000,
12000, 13000, 14000, 15000, 16000};
U32 level_table[16] ={0x1111,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
// Hexadecimal value, 0x00001111 = Binary array, [0000 0000 0000 0000 0001 0001 0001 0001]
B
// Refer to the pulse array with the setting of pos_table, a signal will be triggered at 1000, 5000,
9000, and 13000 (pulse).

U32 table_size =16;
U16 status =_ECAT_Compare_Set_Channel1_Position_Table_Level (CardNo, pos_table,
level_table,table_size);
```

Description

As shown in the above example, level_table is a data array. It translates the trigger state at the 32 positions into a 32-bit value. Then, these values are input in the array. Find more information in the following section.

Output mode of user-defined intervals and trigger level

In this mode, users can define the position (pulse) for the trigger, which is the same as mode 0. Meanwhile, you can also set the trigger level (high / low). See the figure below.

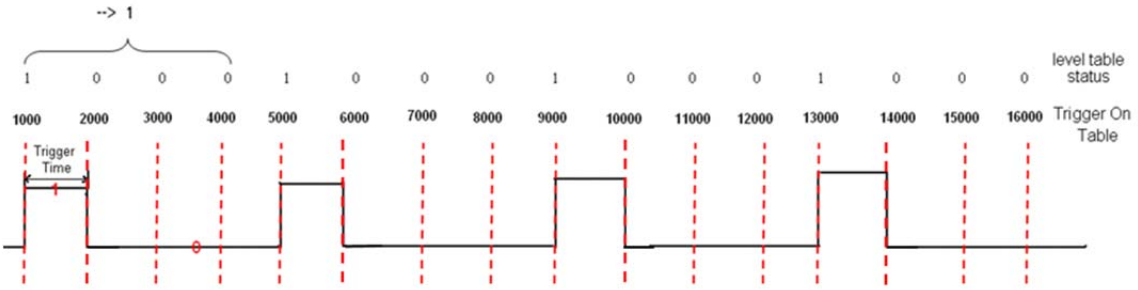


Figure 27.12.4 Output mode of user-defined intervals and trigger level (Correct)

Note:

1. The trigger-on position is set by “Pos_table” of the API “_ECAT_Compare_Set_Channel1_Position_Table”.

The position array of pulse is as follows:

```
I32 pos_table[16] ={1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000,
12000, 13000, 14000, 15000, 16000};
```

As the pulse number to be compared is less than 32, only level_table[0] in level_table is required for triggering the signal as shown in figure 27.12.4.

```
U32 level_table[16] ={0x1111,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
// Hexadecimal value, 0x00001111 = binary array, [0000 0000 0000 0000 0001 0001 0001 0001]
B
// Refer to the pulse array with setting of pos_table, a signal will be triggered only at position 1000,
5000, 9000, and 13000 (pulse).
```

When level_table is set as follows:

```
U32 level_table[16] ={1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0};
// Refer to the pulse array with setting of pos_table, a signal will be triggered only at position 1000
(pulse).
```

Below is the result of the above setting:

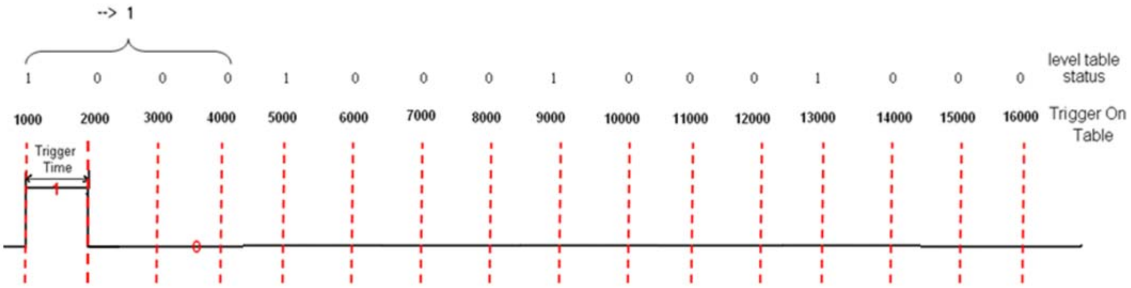


Figure 27.12.4 Output mode of user-defined intervals and trigger level (Incorrect)

2. The trigger retaining time is not influenced by setting of API "_ECAT_Compare_Set_Channel_Trigger_Time. Instead, the trigger time is the time for reaching the next position. (Ex: the time it takes from position 1000 to 2000.)

27.17 _ECAT_Compare_Get_Channel1_Position _Table_Count

■ Syntax

U16 PASCAL _ECAT_Compare_Get_Channel1_Position_Table_Count (U16 CardNo,
U32* cnt)

■ Purpose

Acquire the current trigger counts of channel 1.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Cnt	U32*	Numeric value	Acquire the trigger counts of channel 1.

■ Example

```
U16 CardNo = 0;
U32 cnt;

U16 status = _ECAT_Compare_Get_Channel1_Position_Table_Count (CardNo,& cnt);
```

27.18 _ECAT_Compare_Set_Channel_Polarity

■ Syntax

U16 PASCAL _ECAT_Compare_Set_Channel_Polarity (U16 CardNo, U16 inverse)

■ Purpose

Set the trigger level of the compare function.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Inverse	U16	Option	0: High 1: Low

■ Example

```
U16 CardNo = 0;
U16 inverse = 1;

U16 status = _ECAT_Compare_Set_Channel_Polarity (CardNo, inverse);
```

27.19 _ECAT_Compare_Reuse_Channel1_Position_Table

■ Syntax

U16 PASCAL _ECAT_Compare_Reuse_Channel1_Position_Table (U16 CardNo)

■ Purpose

Execute again the compare function of channel 1 which is set by API "_ECAT_Compare_Set_Channel1_Position_Table".

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 CardNo = 0;

U16 status = _ECAT_Compare_Reuse_Channel1_Position_Table (CardNo);
```

27.20 _ECAT_Compare_Reuse_Channel1_Position _Table_Level

■ Syntax

U16 PASCAL _ECAT_Compare_Reuse_Channel1_Position_Table_Level (U16 CardNo)

■ Purpose

Execute again the user-defined trigger level of compare function for channel 1 which is set by API _ECAT_Compare_Reuse_Channel1_Position_Table_Level”.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.

■ Example

```
U16 CardNo = 0;

U16 status = _ECAT_Compare_Reuse_Channel1_Position_Table_Level (CardNo);
```

(This page is intentionally left blank.)

Information of EtherCAT Dynamic-Link Library (DLL)

28

This chapter introduces the APIs for DLL (Dynamic-link library), such as acquiring information about directory and version of the EtherCat_DLL.dll file.

API list of DDL information

API	Description
_ECAT_Master_Get_DLL_Path	Acquire the directory of the EtherCat_DLL.dll file
_ECAT_Master_Get_DLL_Version	Acquire the version information of the EtherCat_DLL.dll file
_ECAT_Master_Get_DLL_Path_Single	Acquire the directory of the ECAT_RTX_DLL.dll or PCI_L221.dll file
_ECAT_Master_Get_DLL_Version_Single	Acquire the version information of the ECAT_RTX_DLL.dll or PCI_L221.dll file

28.1 _ECAT_Master_Get_DLL_Path

■ Syntax

U16 PASCAL _ECAT_Master_Get_DLL_Path(I8 *lpFilePath, U32 nSize, U32 *nLength)

■ Purpose

Acquire the directory of the EtherCat_DLL.dll file.

■ Parameter

Name	Data type	Property	Description
lpFilePath	I8*	String	Directory of the EtherCat_DLL.dll file
nSize	U32	Value	The string length to be read
nLength	U32*	Value	The string length of the directory

■ Example

```
U16 Status = 0;
I8 FilePath[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Path(FilePath, nSize, &nLength);
```

28.2 _ECAT_Master_Get_DLL_Version

■ Syntax

U16 PASCAL _ECAT_Master_Get_DLL_Version(I8 *lpBuf, U32 nSize, U32 *nLength)

■ Purpose

Acquire the version information of the EtherCat_DLL.dll file.

■ Parameter

Name	Data type	Property	Description
lpBuf	I8*	String	Version of the EtherCat_DLL.dll file.
nSize	U32	Value	The string length to be read
nLength	U32*	Value	The string length of the version information

■ Example

```
U16 Status = 0;
I8 Version[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Version(Version, nSize, &nLength);
```

28.3 _ECAT_Master_Get_DLL_Path_Single

■ Syntax

U16 PASCAL _ECAT_Master_Get_DLL_Path_Single (U16 CardNo, I8 *lpFilePath, U32 nSize, U32 *nLength)

■ Purpose

Acquire the directory of the ECAT_RTX_DLL.dll or PCI_L221.dll file.

Note: These two DLL files (ECAT_RTX_DLL.dll or PCI_L221.dll) are the updated file and the file generated by EtherCAT_DLL.dll automatically.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
lpFilePath	I8*	String	Directory of the ECAT_RTX_DLL.dll or PCI_L221.dll file.
nSize	U32	Value	The string length to be read
nLength	U32*	Value	The string length of the directory

■ Example

```
U16 Status = 0, CardNo=16;
I8 FilePath[128];
U32 nSize=128, nLength=0;

Status= _ECAT_Master_Get_DLL_Path_Single(CardNo, FilePath, nSize, &nLength);
```

28.4 _ECAT_Master_Get_DLL_Version_Single

■ Syntax

U16 PASCAL _ECAT_Master_Get_DLL_Version_Single (U16 CardNo, I8 *lpBuf, U32 nSize, U32 *nLength)

■ Purpose

Acquire the version information of the ECAT_RTX_DLL.dll or PCI_L221.dll file.

Note: These two DLL files (ECAT_RTX_DLL.dll or PCI_L221.dll) are the updated file and the file generated by EtherCAT_DLL.dll automatically.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
lpBuf	I8*	String	Version of the ECAT_RTX_DLL.dll or PCI_L221.dll file.
nSize	U32	Value	The string length to be read
nLength	U32*	Value	The string length of the version information

■ Example

```
U16 Status = 0, CardNo=16;
I8 Version[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Version_Single(CardNo, Version, nSize, &nLength);
```

(This page is intentionally left blank.)

Security of Software Protection

29

This chapter introduces the APIs for security of software protection. With the built-in verification IC on Delta PAC or motion cards, you can protect your own software from being pirated.

API list of software protection

API	Description
_ECAT_Security_Check_Verifykey	Check the verification key
_ECAT_Security_Get_Check_Verifykey_State	Check the verification status of the verification key
_ECAT_Security_Write_Verifykey	Write the verification key into the verification IC
_ECAT_Security_Get_Write_Verifykey_State	Obtain the status and result of writing in the verification key
_ECAT_Security_Check_UserPassword	Check the user password
_ECAT_Security_Get_Check_UserPassword_State	Acquire the status of verifying the user password
_ECAT_Security_Write_UserPassword	Write in the user password into the verification IC
_ECAT_Security_Get_Write_UserPassword_State	Acquire the status and result of writing in the user password

29.1 _ECAT_Security_Check_Verifykey

■ Syntax

U16 PASCAL _ECAT_Security_Check_Verifykey (U16 CardNo, U32 *Verifykey)

■ Purpose

Check the verification key.

The default 8-character verification key is 00000000. Users can reset it via the API "_ECAT_Security_Write_Verifykey" (section 29.3). Please remember to confirm the user password by the API mentioned in 29.5 first. It is suggested to change the verification key and user password together.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Verifykey	U32*	Data	Input an 8-character verification key.

■ Example

```
U16 Status = 0;
U16 CardNo;
U32 Key[8] = {0, 0, 0, 0, 0, 0, 0, 0}

CardNo = 0;
Status= _ECAT_Security_Check_Verifykey ( CardNo, Key );
```

29.2 _ECAT_Security_Get_Check_Verifykey_State

■ Syntax

U16 PASCAL _ECAT_Security_Get_Check_Verifykey_State (U16 CardNo, U16 *State)

■ Purpose

Check the verification status and result of checking the verification key.

Note: The function of verification check will not be completed unless the return code is not 2.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
State	U16*	Status	Return code: 0: Verification sussesful. 1: Verification failed. 2: Verification in process. 3: _ECAT_Security_Check_Verifykey is not executed. 4: Processing error of _ECAT_Security_Check_Verifykey

■ Example

```
U16 Status = 0;
U16 CardNo = 0, State = 0;
U32 Key[8] = {0, 0, 0, 0, 0, 0, 0, 0};
// Check the verification key
Status= _ECAT_Security_Check_Verifykey ( CardNo, Key );
// Wait for the result
While (1)
{
        Status= _ECAT_Security_Get_Check_Verifykey_State ( CardNo, &State );
        if (State != 2)
        {
                // Verification check is done. User can check the result from the return code.
                break;
        }
}
```

29.3 _ECAT_Security_Write_Verifykey

Syntax

U16 PASCAL _ECAT_Security_Write_Verifykey (U16 CardNo, U32 *Verifykey)

Purpose

Write the verification key into the verification IC

Note: Before writing in the verification key, users should use the API function (in section 6.1 and 6.2) to initialize the EtherCAT master. Also, use the API function (section 29.5) to confirm the password. beforehand.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
Verifykey	U32*	Data array	Input an 8-character verification key

Example

```
U16 Status = 0;
U16 CardNo;
U32 Verifykey[8] = {0, 1, 2, 3, 4, 5, 6, 7};

CardNo = 0;
Status= _ECAT_Security_Write_Verifykey ( CardNo, Verifykey);
```

29.4 _ECAT_Security_Get_Write_Verifykey_State

■ Syntax

U16 PASCAL _ECAT_Security_Get_Write_Verifykey_State (U16 CardNo, U16 *State)

■ Purpose

Acquire the status and result of writing in the verification key.

Note: The write-in function cannot be done unless the the return code is not 2.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
State	U16*	Status	Return code: 0: Write in verification sussesful. 1: Write in verification failed. 2: Write in verification in progress. 3: _ECAT_Security_Write_Verifykey is not executed. 4: Processing error of _ECAT_Security_Write_Verifykey

■ Example

```
U16 Status = 0;
U16 CardNo = 0, State = 0;
U32 Verifykey[8] = {0, 1, 2, 3, 4, 5, 6, 7};
// Wrtie in new verification key.
Status= _ECAT_Security_Write_Verifykey ( CardNo, Verifykey);
// Wait unitl the write-in process is complete.
While (1)
{
        Status= _ECAT_Security_Get_Write_Verifykey_State( CardNo, &State );
        if (State != 2)
        {
                // Write-in process is complete. Users can check the result from the return code.
                break;
        }
}
```

29.5 _ECAT_Security_Check_UserPassword

■ Syntax

U16 PASCAL _ECAT_Security_Check_UserPassword (U16 CardNo, U32 *UserPassword)

■ Purpose

Check the user password.

The default user password is 00000000. Users can reset the new password with the API function "_ECAT_Security_Write_UserPassword" (section 29.7).

Note: Before applying this API, users should use the API function (in setion 6.1 and 6.2) to initialize the EtherCAT master in advance.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
UserPassword	U32*	Data array	Input an 8-character User password.

■ Example

```
U16 Status = 0;
U16 CardNo = 0;
U32 UserPassword[8] = {0, 0, 0, 0, 0, 0, 0, 0};

Status= _ECAT_Security_Check_UserPassword ( CardNo, UserPassword);
```

29.6 _ECAT_Security_Get_Check_UserPassword _State

■ Syntax

U16 PASCAL _ECAT_Security_Get_Check_UserPassword_State (U16 CardNo, U16 *State)

■ Purpose

Acquire the status of verifying the user password.

Note: The password check function cannot be done unless the the return code is not 2.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
State	U16*	Status	Return code: 0: Password check is done. User password is valid. 1: Verification failed. User password is invalid. 2: Verification in progress. 3: Please execute _ECAT_Security_Check_UserPassword. 4: Processing error of _ECAT_Security_Check_UserPassword

■ Example

```
U16 Status = 0;
U16 CardNo = 0, State = 0;
U32 UserPassword [8] = {0, 0, 0, 0, 0, 0, 0, 0};
// Check the user password.
Status= _ECAT_Security_Check_UserPassword( CardNo, UserPassword);
// Wait for the result.
While (1)
{
        Status= _ECAT_Security_Get_Check_UserPassword_State ( CardNo, &State );
        if (State != 2)
        {
                // Verification is done. Users can check the result from the return code.
                break;
        }
}
```

29.7 _ECAT_Security_Write_UserPassword

■ Syntax

U16 PASCAL _ECAT_Security_Write_UserPassword (U16 CardNo, U32 *UserPassword)

■ Purpose

Write in the user password into the verification IC.

Note:

1. Before write in the user password, users should use the API function (in setion 6.1 and 6.2) to initialize the EtherCAT master. Also, use the API function in 29.5 to confirm the password beforehand.
2. The new password will be valid immediately. Users will need to use API "_ECAT_Security_Check_UserPassword" (section 29.5) to log in again to use other APIs in this chapter. Once the password is changed, please do remember the new password as Delta can no longer access this verification IC anymore.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
UserPassword	U32*	Data array	Input an 8-character User password.

■ Example

```
U16 Status = 0;
U16 CardNo;
U32 UserPassword [8] = {0, 1, 2, 3, 4, 5, 6, 7};

CardNo = 0;
Status= _ECAT_Security_Write_UserPassword(CardNo, UserPassword);
```

29.8 _ECAT_Security_Get_Write_UserPassword_State

- **Syntax**

U16 PASCAL _ECAT_Security_Get_Write_UserPassword_State (U16 CardNo, U16 *State)

- **Purpose**

Acquire the status and result of writing in the user password.

Note: The password write-in function cannot be done until the the return code is not 2.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
State	U16*	Status	Return code: 0: Verification sussesful. 1: Verification failed. 2: Verification in progress. 3: _ECAT_Security_Write_UserPassword is not executed. 4: Processing error of _ECAT_Security_Write_UserPassword.

- **Example**

```
U16 Status = 0;
U16 CardNo = 0, State = 0;
U32 UserPassword [8] = {0, 1, 2, 3, 4, 5, 6, 7};
// Write in the new password.
Status= _ECAT_Security_Write_UserPassword( CardNo, UserPassword);
// Wait for the write-in process to be completed.
While (1)
{
        Status= _ECAT_Security_Get_Write_UserPassword_State ( CardNo, &State );
        if (State != 2)
        {
                // Password write-in is done. Users can check the result from the return code.
                break;
        }
}
```

Operating MRAM on PAC 30

This chapter introduces the APIs for operating the MRAM on PAC. The high-speed read/write function of MRAM can satisfy the demand of retaining the information when power is cut off, which is more convenient than saving files manually.

API list of MRAM in PAC

API	Description
_ECAT_Master_MRAM_Write_Word_Data	Write the U16 data (Word) to the specified address of MRAM in PAC.
_ECAT_Master_MRAM_Read_Word_Data	Read the U16 data (Word) from the specified address of MRAM in PAC.
_ECAT_Master_MRAM_Write_DWord_Data	Write the U32 data (DWord) into the specified address of MRAM in PAC.
_ECAT_Master_MRAM_Read_DWord_Data	Read the U32 data (DWord) from the specified address of MRAM in PAC.

30.1 _ECAT_Master_MRAM_Write_Word_Data

- **Syntax**

U16 PASCAL _ECAT_Master_MRAM_Write_Word_Data(U16 CardNo, U32 Index, U32 DataNum, U16 *Data)

- **Purpose**

Write the U16 data (Word) to the specified address of MRAM in PAC.

Note: Delta MH1 and MP1 series PAC provide 128K byte retentive memory space. API allows Word type data to access the retentive memory. Index 0 will occupy byte 0 and byte 1, index 1 occupies byte 2 and byte 3 and so on. The index range is between 0 and 65535.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Index	U32	Value	Range: 0 ~ 65535
DataNum	U32	Quantity	Data number to be written into the memory
Data	U16*	Data array	Data array to be written into the memory

- **Example**

```
U16 Status = 0;
U16 CardNo = 16;
U16 data [3] = { 1, 2, 3};
// Write data to the last three data space of retentive memory.
U32 Index=65533, DataNum=3;

Status= _ECAT_Master_MRAM_Write_Word_Data(CardNo, Index, DataNum, &data);
```

30.2 _ECAT_Master_MRAM_Read_Word_Data

■ Syntax

U16 PASCAL _ECAT_Master_MRAM_Read_Word_Data(U16 CardNo, U32 Index, U32 DataNum, U16 *Data)

■ Purpose

Read the U16 data (Word) from the specified address of MRAM in PAC.

Note: Delta MH1 and MP1 series PAC provide 128K byte retentive memory space. API allows Word type data to access the retentive memory. Index 0 will occupy byte 0 and byte 1, index 1 occupies byte 2 and byte 3 and so on. The index range is between 0 and 65535.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Index	U32	Value	Range: 0 ~ 65535
DataNum	U32	Quantity	Data number to be read from the memory
Data	U16*	Data array	Data array to be read from the memory

■ Example

```
U16 Status = 0;
U16 CardNo = 16;
U16 data [3] = {0};
// Read data from the last three data space of retentive memory.
U32 Index=65533, DataNum=3;

Status= _ECAT_Master_MRAM_Read_Word_Data(CardNo, Index, DataNum, data);
```

30.3 _ECAT_Master_MRAM_Write_DWord_Data

Syntax

U16 PASCAL _ECAT_Master_MRAM_Write_DWord_Data(U16 CardNo, U32 Index, U32 DataNum, U32 *Data)

Purpose

Write the U32 data (DWord) to the specified address of MRAM in PAC.

Note: Delta MH1 and MP1 series PAC provide 128K byte retentive memory space. The API allows Double Word type data to access the retentive memory. Index 0 will occupy byte 0, byte 1, byte 2 and byte 3, index 1 occupies byte 2, byte 3, byte 4 and byte 5 and so on. The memory will conflict with the index space, thus, when the index value is odd, error will occur.

Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Index	U32	Value	Range: 0 ~ 65535 Odd numbers are not allowed.
DataNum	U32	Quantity	Data number to be written into the memory
Data	U32*	Data array	Data array to be written into the memory

Example

```
U16 Status = 0;
U16 CardNo = 16;
U32 data [3] = { 1, 2, 3};
// Write data to the last three data space of retentive memory.
U32 Index=65530, DataNum=3;

Status= _ECAT_Master_MRAM_Write_DWord_Data(CardNo, Index, DataNum, data);
```

30.4 _ECAT_Master_MRAM_Read_DWord_Data

■ Syntax

U16 PASCAL _ECAT_Master_MRAM_Read_DWord_Data(U16 CardNo, U32 Index, U32 DataNum, U32 *Data)

■ Purpose

Read the U32 data (DWord) from the specified address of MRAM in PAC.

Note: Delta MH1 and MP1 series PAC provide 128K byte retentive memory space. The API allows Double Word type data to access the retentive memory. Index 0 will occupy byte 0, byte 1, byte 2 and byte 3, index 1 occupies byte 2, byte 3, byte 4 and byte 5 and so on. The memory will conflict with the index space, thus, when the index value is odd, error will occur.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
Index	U32	Value	Range: 0 ~ 65535 Odd numbers are not allowed.
DataNum	U32	Quantity	Data number to be wrote into the memory
Data	U32*	Data array	Data array to be wrote into the memory

■ Example

```
U16 Status = 0;
U16 CardNo = 16;
U16 data [3] = {0};
// Read data from the last three data space of retentive memory.
U32 Index=65530, DataNum=3;

Status= _ECAT_Master_MRAM_Read_DWord_Data(CardNo, Index, DataNum, data);
```

(This page is intentionally left blank.)

Retentive Digital Output of the Module (For 70E2 Series)

31

This chapter will tell you how to use the API for enabling/disabling retentive digital output of the module. Delta R1-EC70E2D0 can retain the output status when EtherCAT is offline. API function in this chapter can be used to enable the function of digital output when EtherCAT is online.

API list of digital output retentive function (for R1-EC70E2D0 series)

API	Description
_ECAT_Slave_R1_EC70E2_Set_Output_Enable	Enable/Disable the digital output of the module.

31.1 _ECAT_Slave_R1_EC70E2_Set_Output_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC70E2_Set_Output_Enable(U16 CardNo, U16 NodeID, U16 SlotNo, U16 Eanble)

■ Purpose

This is for enabling/disabling digital output of Delta R1-EC70E2D0 when EtherCAT is online.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Eanble	U16	Option	0: Disable digital output 1: Enable digital output

■ Example

```
U16 Status = 0;
U16 CardNo = 16, NodeID = 0, SlotNo = 0;
U16 Enable = 1; // Enable digital output of R1-EC70E2D0.

Status = _ECAT_Slave_R1_EC70E2_Set_Output_Enable ( CardNo, NodeID, SlotNo,
Enable );
```

Retentive Digital Output of the Module (For 70X2 Series)

32

This chapter will tell you how to use the API for enabling/disabling digital output of the module. Delta R1-EC70E2D0 and R1-EC70F2D0 provide retentive function, which can keep the output status when EtherCAT is offline. API function in this chapter can be used to enable the digital output when EtherCAT is online.

API list of enabling digitial output (for R1-EC70X2D0 series)

API	Description
_ECAT_Slave_R1_EC70X2_Set_Output_Enable	Enable/Disable digital output of the module

32.1 _ECAT_Slave_R1_EC70X2_Set_Output_Enable

- **Syntax**

U16 PASCAL _ECAT_Slave_R1_EC70X2_Set_Output_Enable(U16 CardNo, U16 NodeID, U16 SlotNo, U16 Eanble)

- **Purpose**

This is for enabling/disabling digital output of Delta R1-EC70E2D0 and R1-EC70F2D0 when EtherCAT is online.

- **Parameter**

Name	Data type	Property	Description
CardNo	U16	Number	Card No.
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Eanble	U16	Option	0: Disable digital output 1: Enable digital output

- **Example**

```
U16 Status = 0;
U16 CardNo = 16, NodeID = 0, SlotNo = 0;
U16 Enable = 1; // Enable digital output of R1-EC70X2D0 series.

Status = _ECAT_Slave_R1_EC70X2_Set_Output_Enable ( CardNo, NodeID, SlotNo,
Enable );
```

MPG Operation (For R1-EC5614D0 Series) 33

This chapter will tell you how to use the APIs for MPG operation. Delta R1-EC5641D0 series provides MPG function, which allows users to set the specified axis to jog. You can start using the MPG function just by enabling it when programming.

API List of MPG operation (for R1-EC5614D0 series)

API	Description
_ECAT_Slave_R1_EC5614_Set_MJ_Config	Set the parameters of MPG function
_ECAT_Slave_R1_EC5614_Set_MJ_Enable	Enable/Disable the MPG function
_ECAT_Slave_R1_EC5614_Get_IO_Status	Acquire the I/O contact status of the MPG module
_ECAT_Slave_R1_EC5614_Get_MPG_Counter	Acquire the value of the MPG counter

33.1 _ECAT_Slave_R1_EC5614_Set_MJ_Config

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5614_Set_MJ_Config (U16 CardNo, U16 MJNo, U16 MJType, U16 NodeID, U16 SlotNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, I32 *MaxSpeedArray, F64 *TaccArray, F64 *RatioArray)

■ Purpose

This is for setting the parameters of MPG function.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
MJNo	U16	Number	Group number of MPG function. It supports max. 8 groups, numbered from 0 to 7.
MJType	U16	Option	Setting of MPG mode : 0: JOG mode 1: Apply MPG (x1) 2: Apply MPG (x4)
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
AxisNum	U16	Number	The axis number reqruied by MPG/JOG (MPG: max. 6 axes; JOG: max. 2 axes)
AxisArray	U16*	Array of numbers	Data array of node number for the axis controlled by the MPG module.
SlotArray	U16*	Array of numbers	Data array of slot number for the axis controlled by the MPG module.
MaxSpeedArray	I32*	Array of speed limit	Data array of the max. speed for each axis. (Unit: pps)
TaccArray	F64*	Array of acceleration time	Data array of the max. acceleration for each axis contolled by MPG module. (Unit:sec) (The speed is set by parameter MaxSpeedArray)
RatioArray	F64*	Array of output ratio	For MPG mode only. It sets the ratio of MPG’s pulse output.

■ Example

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0, AxisNum = 2, MJNo = 2, MJType= 1;
U16 Enable = 1;
U16 AxisArray[2] = {1, 2};
U16 SlotArray[2] = {0, 0};
I32 MaxSpeedArray[2] = {100000, 200000};
F64 TaccArray[2] = {0.1, 0.1};
F64 RatioArray[2] = {1, 1};

// Setting relevant parameters is required.
```

```
Status = _ECAT_Slave_R1_EC5614_Set_MJ_Config( CardNo, NodeID, SlotNo, MJNo,
MJType, AxisNum, AxisArray, SlotArray, MaxSpeedArray, TaccArray, RatioArray );

// It can be enabled when setting is complete.
If(Status == 0)
Status = _ECAT_Slave_R1_EC5614_Set_MJ_Enable(CardNo, MJNo, Enable);
```

33.2 _ECAT_Slave_R1_EC5614_Set_MJ_Enable

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5614_Set_MJ_Enable (U16 CardNo, U16 MJNo, U16 Enable)

■ Purpose

This is for enabling/disabling the MPG function.

Note: Before enabling the MPG function, please set the parameters of _ECAT_Slave_R1_EC5614_Set_MJ_Config in section 33.1.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
MJNo	U16	Number	Group number of MPG function. It supports max. 8 groups, numbered from 0 to 7.
Enable	U16	Option	0: Disable MPG function 1: Enable MPG function

■ Example

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0, AxisNum = 2, MJNo = 1, MJType= 1;
U16 Enable = 1;
U16 AxisArray[2] = {1, 2};
U16 SlotArray[2] = {0, 0};
I32 MaxSpeedArray[2] = {100000, 200000};
F64 TaccArray[2] = {0.1, 0.1};
F64 RatioArray[2] = {1, 1};

// Relevant parameters setting is required.
Status = _ECAT_Slave_R1_EC5614_Set_MJ_Config( CardNo, NodeID, SlotNo, MJNo,
MJType, AxisNum, AxisArray, SlotArray, MaxSpeedArray, TaccArray, RatioArray );

// It can be enabled after setting complete.
If(Status == 0)
    Status = _ECAT_Slave_R1_EC5614_Set_MJ_Enable(CardNo, MJNo, Enable);
```

33.3 _ECAT_Slave_R1_EC5614_Get_IO_Status

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5614_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

■ Purpose

This is for acquiring the DI/O contact status of MPG module.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
IOStatus	U16*	Value	DI/O status of MPG module

■ Example

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0;
U16 IOStatus = 0;

Status = _ECAT_Slave_R1_EC5614_Get_IO_Status (CardNo, NodeID, SlotNo, &IOStatus);
```

■ Status description

Bit	Definition	Bit	Definition
0	JY-	8	X-Axis is selected.
1	JY+	9	Y-Axis is selected.
2	JX-	10	Z-Axis is selected.
3	JX+	11	U-Axis is selected.
4	Scale: ×1	12	W-Axis is selected.
5	Scale: ×10	13	C-Axis is selected.
6	Scale: ×100	14	Reserved
7	Reserved	15	MPG Enabled

33.4 _ECAT_Slave_R1_EC5614_Get_MPG_Counter

■ Syntax

U16 PASCAL _ECAT_Slave_R1_EC5614_Get_MPG_Counter (U16 CardNo, U16 NodeID, U16 SlotNo, I32 *Counter)

■ Purpose

This is for acquiring accumulative value of the MPG counter.

■ Parameter

Name	Data type	Property	Description
CardNo	U16	Number	Card number
NodeID	U16	Number	Node ID
SlotNo	U16	Number	Slot ID
Counter	I32*	Value	Value of the MPG Counter (Unit: pulse)

■ Example

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0;
I32 Counter = 0;

Status = _ECAT_Slave_R1_EC5614_Get_MPG_Counter(CardNo, NodeID, SlotNo,
&Counter);
```

(This page is intentionally left blank.)

Error Code Description 34

If an error code is returned after an API is executed, you can refer to this chapter to find more information about the cause and troubleshooting method.

34.1 List of error code

When using the API of EtherCAT_Dll, the API function will return a value that shows the execution result. Please refer to the following table for the description of those values. For example, if the API function returns 0, it means the API is successfully executed. On the other hand, a non-zero value is regarded as an error code. It means an error has occurred during operation or hardware connection. Please troubleshoot the problem according to the description below.

List of error code

Returned value (Decimal)	Name of the error code	Description
0	ERR_ECAT_NO_ERROR	No error
1	ERR_ECAT_HW_NO_INITIALIZE	Hardware has not been initialized.
2	ERR_ECAT_HW_PWM_INITIAL	Fail to initialize the hardware.
3	ERR_ECAT_HW_HAS_INITIALIZED	Hardware is initialized repeatedly.
16	ERR_ECAT_EEPROM_READ	Fail to read the EEPROM of the module.
17	ERR_ECAT_EEPROM_WRITE	Fail to write the data to the EEPROM of the module.
18	ERR_ECAT_ENVIRONMENT_RECORD_DISABLE	The Autoconfig file is not found.
19	ERR_ECAT_ENVIRONMENT_RECORD_NO_MATCH	The hardware connection configuration saved by Autoconfig file does not match the one for Fieldbus.
20	ERR_ECAT_ENVIRONMENT_RECORD_FILE_OPEN	Fail to open the Autoconfig file.
21	ERR_ECAT_ENVIRONMENT_RECORD_NOT_CREATE	The Autoconfig file for saving the hardware connection configuration has not been created.
23	ERR_ECAT_DEVICE_OPEN	The setting for initializing EtherCAT Master is in error.
24	ERR_ECAT_NO_DEVICE	The module data that is loaded during the initial stage of EtherCAT Master initialization is in error.
25	ERR_ECAT_NO_MASTER	EtherCAT Master not found.
26	ERR_ECAT_NO_SLAVE	EtherCAT Slave not found.
27	ERR_ECAT_UNKNOWN_SLAVE	There is an unknown module type on the EtherCAT bus.
28	ERR_ECAT_IST_CREATE	Fail to create IST.
29	ERR_ECAT_MASTER_CREATE	Fail to create EtherCAT Master.

Returned value (Decimal)	Name of the error code	Description
30	ERR_ECAT_MASTER_REQUEST_ STATE	Wrong communication mode of EtherCAT Master.
31	ERR_ECAT_MASTER_OPERATION_NOT _READY	The EtherCAT Master is not in OP status yet.
32	ERR_ECAT_DELTA_NODE_ID_ALIAS_RE AD	Fait to read the verification code of Delta servo drive.
33	ERR_ECAT_MASTER_GET_ SERIAL_NO_WRONG	Fail to acquire the serial number of Delta PAC or EtherCAT motion card.
34	ERR_ECAT_MASTER_GET_ SERIAL_NO_TIMEOUT	Timeout error occurs when acquiring the serial number of Delta PAC or EtherCAT motion card.
128	ERR_ECAT_PIPELINE_CORE _TIMER_CREATE	EtherCAT kernel – The scheduler fails to create the event.
129	ERR_ECAT_PIPELINE_CREATE	EtherCAT kernel – Fail to create the scheduler function.
130	ERR_ECAT_COMMAND_ENQUEUE	EtherCAT kernel – Fail to create the command buffer for the scheduler.
131	ERR_ECAT_API_BUFFER_ ENQUEUE	EtherCAT kernel –The scheduler fails to add the command to the buffer.
256	ERR_ECAT_NODE_ID	Node ID of EtherCAT is in error.
257	ERR_ECAT_SLOT_ID	Slot ID of EtherCAT is in error.
258	ERR_ECAT_SDO_DOWNLOAD	An error has occurred when setting the data value through SDO.
259	ERR_ECAT_SDO_UPLOAD	An error has occurred when acquiring the data through SDO.
260	ERR_ECAT_GET_PROCESS_DATA	Unable to acquire the specified PDO data.
3840	ERR_ESI_INITIAL	Initialization failure of ESI (EtherCAT Slave Information) file or the ESI file has not been initialized.
3841	ERR_ESI_OPEN_DEVICE	Cannot find the matched hardware setting.
3842	ERR_ESI_CREATE_CANOPEN_OD_LIST	The supported CANOpen OD list cannot be created.
3843	ERR_ESI_NO_DATA_TYPE_INFO	The data of Data Type section cannot be created.
3844	ERR_ESI_NO_OBJECT_INFO	The data of Object section cannot be created.
3845	ERR_ESI_CREATE_SYNC_MANAGER	The data of SM section or Rx / Tx section cannot be created.

Returned value (Decimal)	Name of the error code	Description
3846	ERR_ESI_CREATE_FMMU_CONTROL	The data of Fmmu section cannot be created.
3847	ERR_ESI_NO_PDO_CHANNEL	The specified PDO channel number does not exist.
3848	ERR_ESI_NO_PDO_MAPPING	No object dictionary (OD) is in the specified PDO channel.
3849	ERR_ESI_PDO_MAPPING_INSERT	CANopen OD cannot be created.
3850	ERR_ESI_PDO_MAPPING_DELETE	CANopen OD cannot be deleted.
3851	ERR_ESI_CREATE_DISTRIBUTED_CLOCK	The DC time information in the ESI file cannot be set.
4080	ERR_ESI_ENI_INFORMATION_INITIAL	Unable to generate process data or initial command.
4081	ERR_ESI_ENI_FILE_INITIAL	The information required by ESI (EtherCAT Slave Information) file cannot be generated.
4082	ERR_ESI_ENI_FILE_SAVE	ESI (EtherCAT Slave Information) file cannot be saved.
4096	ERR_ECAT_NO_SLAVE_FOUND	No connection or no slave device is detected.
4097	ERR_ECAT_INITIAL_TIMEOUT	Waiting time for initialization is too long.
4098	ERR_ECAT_MODE_CHANGE_ FAILED	Unable to switch to OP mode or Init mode.
4099	ERR_ECAT_SLAVE_ID	The issued number of slave station is wrong.
4100	ERR_ECAT_ALIAS_SLAVE_ID	Repeated station name or the axis number exceeds the range.
4352	ERR_ECAT_NEED_INITIAL	Initialization is required before starting this operation.
4353	ERR_ECAT_NEED_RESET	This API cannot be executed after the EtherCAT Master is initialized.
4354	ERR_ECAT_NEED_CONNECT	Connection has to be created before starting the operation.
4355	ERR_ECAT_NEED_DC_OP	The execution is only allowed after DC time calibration is completed and the EtherCAT Master switches to OP mode.
4356	ERR_ECAT_NEED_RALM	The API cannot be executed as the current alarm is not cleared.
4357	ERR_ECAT_NEED_SVON	The motor has to be enabled before executing the API.
4358	ERR_ECAT_NEED_HOMECONFIG	Setting homing parameters is required before start homing.

Returned value (Decimal)	Name of the error code	Description
4359	ERR_ECAT_NEED_STOP	This API can be executed only when the specified axis is not moving.
4608	ERR_ECAT_RING_BUFFER_FULL	The cycle communication command buffer of mailbox is full.
4609	ERR_ECAT_API_PARAMETER	The input API parameter is in error.
4610	ERR_ECAT_SLAVE_TYPE	This API is not supported by the module.
4611	ERR_ECAT_TARGET_REACHED	The “target reached” signal is not triggered after the waiting time is up.
4612	ERR_ECAT_MODE_NOT_ SUPPORT	This API is not supported in this motion mode.
4613	ERR_ECAT_MOTION_TYPE	This axis is not supported in this motion mode. (It is probably restricted by the group function.)
4614	ERR_ECAT_PDO_NOT_MAPPING	The OD code of this motion mode is not included the PDO mapping list.
4615	ERR_ECAT_MODULE_REVISION	The firmware version of this module is not supported.
4616	ERR_ECAT_SPEED_CONTINUE_ MODE	One of the axes is not set to continuous speed mode.
4617	ERR_ECAT_HOME_MODE	Parameter setting for homing method is in error.
4618	ERR_ECAT_HOME_OFFSET	The homing offset setting of the homing parameter is in error.
4619	ERR_ECAT_HOME_FIRST_SPEED	The speed setting of looking for the origin during homing is in error.
4620	ERR_ECAT_HOME_SECOND_ SPEED	In API homing parameter, after the origin is found, the speed setting is wrong when it looks for Z pulse.
4621	ERR_ECAT_HOME_ACC	Acceleration setting of the homing parameter is in error.
4622	ERR_ECAT_MRAM_INDEX	The location setting of non-volatile memory is in error.
4623	ERR_ECAT_MRAM_INDEX_OUT_ RANGE	Range setting of the non-volatile memory is in error.
4864	ERR_ECAT_PDO_TX_FAILED	Fail to send the PDO type command.
4865	ERR_ECAT_SDO_TIMEOUT	SDO communication timeout.

Returned value (Decimal)	Name of the error code	Description
4866	ERR_ECAT_SDO_RETURN	The slave device returns SDO error code. Use API "_ECAT_Slave_CANopen_Get_ErrorCode" to acquire the error code.
4867	ERR_ECAT_PDO_RX_FAILED	Fail to return a PDO command.
4868	ERR_ECAT_MAILBOX	Sending failure of Mailbox
4869	ERR_ECAT_SDO_BUFFER_FULL	SDO command buffer is full.
5120	ERR_ECAT_GROUP_NUMBER	The input number of grouped axes is over 10 the maximum.
5121	ERR_ECAT_GROUP_ENABLE	The group function is disabled or in hold state.
5122	ERR_ECAT_GROUP_PAUSE	This action can only be done when group function is in hold state.
5123	ERR_ECAT_GROUP_SLAVE	The slave device has been used by other group or in this group.
5124	ERR_ECAT_GROUP_MODE	The group function is not supported for now.
5125	ERR_ECAT_GROUP_ALREADY_ USED	This group number is in use.
5126	ERR_ECAT_GROUP_TYPE	The type setting has to be done before group function is enabled. Once the function is enabled, setting the type is not allowed.
5127	ERR_ECAT_GROUP_SVON	Please enable all motors in the group.
5128	ERR_ECAT_GROUP_ALM	An alarm has occurred on one of the axes.
5129	ERR_ECAT_GROUP_DATA_ BUFFER	The buffer is full. (799 data)
5130	ERR_ECAT_GROUP_TIMEOUT	No response is sent from the EtherCAT kernel during group operation.
5376	ERR_ECAT_SERVO_PARA_EMPTY	This servo parameter does not exist.
5377	ERR_ECAT_SERVO_PARA_RO	This servo parameter is read-only.
5378	ERR_ECAT_SERVO_COMPARE_ ENABLE	An error has occurred when setting the pulse compare function of the servo drive.
5632	ERR_ECAT_RECORD_TYPE	When the auto recording function is enabled, the acquired data cannot be modified.
5888	ERR_ECAT_MPG_ENABLE	This MPG group has been enabled. Please disable it first before executing the API.
5889	ERR_ECAT_MPG_CONFIG	Setting of this MPG group is not completed.
12288	ERR_ECAT_SECURITY_ OPERATING	Verification procedure is in progress. Please wait and then execute the API.
12289	ERR_ECAT_SECURITY_NEED_ LOGIN	Login is required to proceed with the operation.

Returned value (Decimal)	Name of the error code	Description
12290	ERR_ECAT_SECURITY_CONNECT	This function fails to access the security kernel.
32825	ERR_PATH_ECAT_NEED_ENABLE	This API requires to be enabled.
32826	ERR_PATH_ECAT_ECAM_ENABLE	This API cannot be executed when E-cam is enabled.
32827	ERR_PATH_ECAT_ECAM_ MASTERSOURCE	The master axis of E-cam is not set.
53248	ERR_RTX_RTSS_LOAD	RTSS cannot be enabled/disabled correctly.
53249	ERR_RTX_CONNECT_LINK_ FAILED	Fail to access RTSS shared memory or the license verification is failed.
53250	ERR_RTX_EVENT_FAILED	Fail to access the RTSS event.
53251	ERR_RTX_CONNECT_FAILED	Failure of handshaking with RTSS shared memory.
53252	ERR_RTX_CONFIG_EDITED	Restarting the computer is required to activate the default setting of RTX.
53253	ERR_RTX_SECURITY_FAILED	The RTX license you are using might be a piracy.
53254	ERR_RTX_COMMANDING	No response when RTX special command is issued.
53255	ERR_RTX_SYSTEM_NOT_ SUPPORT	This API is not supported by RTSS.
53256	ERR_RTX_NOT_SUPPORT	The API is not supported by this RTX version.
53257	ERR_RTX_THREAD_CREATE_ FAILED	Fail to enable communication status of RTX thread
53258	ERR_RTX_RTSS_START_FAILED	Fail to enable the RTX system.
53504	ERR_RTX_WIN32_SYSTEM_NOT_ SUPPORT	The callback function is not supported by Win32 system.
53505	ERR_RTX_CALLBACK_CLOSE	Modifying the callback function is not allowed when it is enabled.
53506	ERR_RTX_CALLBACK_FUNCTION	Callback function cannot be enabled due to setting error.
53507	ERR_RTX_CALLBACK_THREAD	When callback function is enabled, the thread is not correctly executed.
53760	ERR_RTX_ERRORLOG_NOT_ ENABLE	Auto recording function is not enabled.
53761	ERR_RTX_ERRORLOG_COUNT_ ERROR	The specified index of error count is wrong.
57344	ERR_CARD_NO_FOUND	Motion card is not found.

Returned value (Decimal)	Name of the error code	Description
57345	ERR_CARD_NO_RESPONSE	No response is sent from the motion card after the command is sent.
57346	ERR_CARD_CONNECT_FAILED	Connection error of the motion card and the driver.
57347	ERR_CARD_MEMORY_NOT_ENOUGH	When recording function is applied, number of motion cards exceeds the limit. (Max. is 24).
57348	ERR_CARD_LOAD_AUTOCONFIG_FILE	Fail to load the Autoconfig file.
57349	ERR_CARD_SECURITY_FAILED	Security verification failed.
57350	ERR_CARD_UPGRADE_CREATE_ THREAD_FAILED	Fail to open the update window.
57351	ERR_CARD_UPGRADE_NO_ RESPONSE	No response returned during the update.
57352	ERR_CARD_UPGRADE_NO_ RESOURSE	No update file (EtherCAT_DLL) is found.
57353	ERR_CARD_UPGRADE_LOAD_ RESOURSE	Fail to access the updating resource (EtherCAT_DLL).
57354	ERR_CARD_UPGRADE_TIMEOUT	System update timeout.
57355	ERR_CARD_UPGRADE_FAILED	System update failure
61440	ERR_ECAT_DLL_IS_USED	EtherCAT_DLL file has been opened.
61441	ERR_ECAT_NO_DLL_FOUND	Connection error of EtherCAT_DLL file, RTSS and DDL file of motion card.
61442	ERR_ECAT_NO_RTSS_DLL_ FOUND	Connection failure of EtherCAT_DLL and RTSSDLL.
61443	ERR_ECAT_NO_CARD_DLL_ FOUND	Connection between EtherCAT_DLL file and DLL file of motion card is in error.
61444	ERR_ECAT_NO_ESI_DLL_FOUND	Connection between EtherCAT DLL file and ESI file (EtherCAT Slave Information) is in error.
61445	ERR_ECAT_SAME_CARD_ NUMBER	Repeated RTSS or motion card number.
61446	ERR_ECAT_CARDNO_ERROR	A non-existing card number of EtherCAT is used.
61447	ERR_ECAT_GET_DLL_PATH	Unable to acquire the directory information of DLL file.
61448	ERR_ECAT_GET_DLL_VERSION	Cannot acquire the version information of DLL.
61449	ERR_ECAT_NOT_SUPPORT	DMCNET type API is currently not supported by EtherCAT.

Returned value (Decimal)	Name of the error code	Description
65535	ERR_ECAT_LOADLIB_EMPTY	Fail to call DLL resource in RTSS.

34.2 Error code description

Code in DEC	0	Code in HEX	0x0
Name of error code	ERR_ECAT_NO_ERROR		
Description	No error.		
Troubleshooting	N/A		

Code in DEC	1	Code in HEX	0x1
Name of error code	ERR_ECAT_HW_NO_INITIALIZE		
Description	Another API is called before EtherCAT master is initialized completely.		
Troubleshooting	Initialize EtherCAT master before calling the API.		

Code in DEC	2	Code in HEX	0x2
Name of error code	ERR_ECAT_HW_PWM_INITIAL		
Description	Fail to initialize EtherCAT master.		
Troubleshooting	Communication for EtherCAT master cannot be created. Please check if the hardware connection is normal.		

Code in DEC	3	Code in HEX	0x3
Name of error code	ERR_ECAT_HW_HAS_INITIALIZED		
Description	EtherCAT master is has been initialized already.		
Troubleshooting	Remove the initialization command in the program.		

Code in DEC	16	Code in HEX	0x10
Name of error code	ERR_ECAT_EEPROM_READ		
Description	Fail to read EEPROM of the module.		
Troubleshooting	Communication for the hardware cannot be created. Please check if the hardware connection is normal.		

Code in DEC	17	Code in HEX	0x11
Name of error code	ERR_ECAT_EEPROM_WRITE		
Description	Fail to write the data to the EEPROM of the module.		
Troubleshooting	Communication for the hardware cannot be created. Please check if the hardware connection is normal.		

Code in DEC	18	Code in HEX	0x12
Name of error code	ERR_ECAT_ENVIRONMENT_RECORD_DISABLE		
Description	The function of the Autoconfig file is not working due to the system configuration error.		
Troubleshooting	N/A		

Code in DEC	19	Code in HEX	0x13
Name of error code	ERR_ECAT_ENVIRONMENT_RECORD_NO_MATCH		
Description	The hardware connection configuration saved by Autoconfig file does not match the one for Fieldbus.		
Troubleshooting	Communication for the hardware cannot be created. Please check if the connection is normal.		

Code in DEC	20	Code in HEX	0x14
Name of error code	ERR_ECAT_ENVIRONMENT_RECORD_FILE_OPEN		
Description	Fail to open the Autoconfig file.		
Troubleshooting	Please check if the Autoconfig file is saved in the given directory.		

Code in DEC	21	Code in HEX	0x15
Name of error code	ERR_ECAT_ENVIRONMENT_RECORD_NOT_CREATE		
Description	The Autoconfig file for saving the hardware connection configuration has not been created.		
Troubleshooting	Please execute the initialization procedure correctly.		

Code in DEC	23	Code in HEX	0x17
Name of error code	ERR_ECAT_DEVICE_OPEN		
Description	The setting for initializing EtherCAT Master is in error.		
Troubleshooting	An error has occurred when initializing EtherCAT Master. Please check if the hardware connection is normal.		

Code in DEC	24	Code in HEX	0x18
Name of error code	ERR_ECAT_NO_DEVICE		

Description	The module information that is loaded during the initial stage of EtherCAT Master initialization is in error.
Troubleshooting	Check if the specific DAT file does exist.

Code in DEC	25	Code in HEX	0x19
Name of error code	ERR_ECAT_NO_MASTER		
Description	EtherCAT Master not found.		
Troubleshooting	Please set the system of EtherCAT Master first.		

Code in DEC	26	Code in HEX	0x1A
Name of error code	ERR_ECAT_NO_SLAVE		
Description	EtherCAT Slave not found.		
Troubleshooting	Hardware communication cannot be created. Please check if the connection is normal.		

Code in DEC	27	Code in HEX	0x1B
Name of error code	ERR_ECAT_UNKNOWN_SLAVE		
Description	There is an unknown module type on the EtherCAT Fieldbus.		
Troubleshooting	Please remove the unsupported module.		

Code in DEC	28	Code in HEX	0x1C
Name of error code	ERR_ECAT_IST_CREATE		
Description	Fail to create IST due to system configuration error.		
Troubleshooting	N/A		

Code in DEC	29	Code in HEX	0x1D
Name of error code	ERR_ECAT_MASTER_CREATE		
Description	Fail to create EtherCAT Master.		
Troubleshooting	Initialization error of EtherCAT Master system. Please check if the hardware connection is normal.		

Code in DEC	30	Code in HEX	0x1D
Name of error code	ERR_ECAT_MASTER_REQUEST_STATE		

Description	Wrong status of EtherCAT Master.
Troubleshooting	Switch to the correct status.

Code in DEC	31	Code in HEX	0x1F
Name of error code	ERR_ECAT_MASTER_OPERATION_NOT_READY		
Description	The EtherCAT Master is not in OP status yet.		
Troubleshooting	Please wait for the EtherCAT Master to switch to OP status.		

Code in DEC	32	Code in HEX	0x20
Name of error code	ERR_ECAT_DELTA_NODE_ID_ALIAS_READ		
Description	Fait to read the verification code of Delta servo drive.		
Troubleshooting	Communication cannot be created, please check if hardware connection is normal. Make sure the firmware version of the Delta servo drive is correct.		

Code in DEC	33	Code in HEX	0x21
Name of error code	ERR_ECAT_MASTER_GET_SERIAL_NO_WRONG		
Description	Fail to acquire the serial number of Delta PAC or EtherCAT motion card.		
Troubleshooting	Please contact Delta to check if the latest FPGA firmware is used on your Delta PAC or EtherCAT motion card.		

Code in DEC	34	Code in HEX	0x22
Name of error code	ERR_ECAT_MASTER_GET_SERIAL_NO_TIMEOUT		
Description	Timeout error occurs when acquiring the serial number of Delta PAC or EtherCAT motion card.		
Troubleshooting	Please contact Delta to check if the latest FPGA firmware is used on your Delta PAC or EtherCAT motion card.		

Code in DEC	128	Code in HEX	0x80
Name of error code	ERR_ECAT_PIPELINE_CORE_TIMER_CREATE		
Description	EtherCAT kernel – The scheduler fails to create the event due to system configuration error.		
Troubleshooting	N/A		

Code in DEC	129	Code in HEX	0x81
Name of error code	ERR_ECAT_PIPELINE_CREATE		
Description	EtherCAT kernel – Fail to create the scheduler function due to system configuration error.		
Troubleshooting	N/A		

Code in DEC	130	Code in HEX	0x82
Name of error code	ERR_ECAT_COMMAND_ENQUEUE		
Description	EtherCAT kernel – Fail to create the command buffer for the scheduler.		
Troubleshooting	This command is not supported by the module.		

Code in DEC	131	Code in HEX	0x82
Name of error code	ERR_ECAT_API_BUFFER_ENQUEUE		
Description	EtherCAT kernel –The scheduler fails to add the command to the buffer.		
Troubleshooting	The add command is not supported by the remote module.		

Code in DEC	256	Code in HEX	0x100
Name of error code	ERR_ECAT_NODE_ID		
Description	Node ID of EtherCAT is in error.		
Troubleshooting	Please check the setting parameters.		

Code in DEC	257	Code in HEX	0x101
Name of error code	ERR_ECAT_SLOT_ID		
Description	Slot ID of EtherCAT is in error.		
Troubleshooting	Please check the setting parameters.		

Code in DEC	258	Code in HEX	0x102
Name of error code	ERR_ECAT_SDO_DOWNLOAD		
Description	An error has occurred when setting the data value through SDO.		
Troubleshooting	Please check the setting parameters.		

Code in DEC	259	Code in HEX	0x103
Name of error code	ERR_ECAT_SDO_UPLOAD		
Description	An error has occurred when acquiring the data through SDO.		
Troubleshooting	Please check the setting parameters.		

Code in DEC	260	Code in HEX	0x104
Name of error code	ERR_ECAT_GET_PROCESS_DATA		
Description	The specified PDO data cannot be acquired.		
Troubleshooting	Please check the setting parameters.		

Code in DEC	3840	Code in HEX	0xF00
Name of error code	ERR_ESI_INITIAL		
Description	Initialization failure of ESI (EtherCAT Slave Information) file or the ESI file has not been initialized.		
Troubleshooting	Re-initialize the EtherCAT master.		

Code in DEC	3841	Code in HEX	0xF01
Name of error code	ERR_ESI_OPEN_DEVICE		
Description	Cannot find the matched hardware setting.		
Troubleshooting	Change the directory destination of ESI (EtherCAT Slave Information) file.		

Code in DEC	3842	Code in HEX	0xF02
Name of error code	ERR_ESI_CREATE_CANOPEN_OD_LIST		
Description	The supported CANOpen OD list cannot be created.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	3843	Code in HEX	0xF03
Name of error code	ERR_ESI_NO_DATA_TYPE_INFO		
Description	The data of Data Type section cannot be created.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	3844	Code in HEX	0xF04
Name of error code	ERR_ESI_NO_OBJECT_INFO		
Description	The data of Object section cannot be created.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	3845	Code in HEX	0xF05
Name of error code	ERR_ESI_CREATE_SYNC_MANAGER		
Description	The data of SM section or Rx / Tx section of PDO cannot be created.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	3846	Code in HEX	0xF06
Name of error code	ERR_ESI_CREATE_FMMU_CONTROL		
Description	The data of Fmmu section cannot be created.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	3847	Code in HEX	0xF07
Name of error code	ERR_ESI_NO_PDO_CHANNEL		
Description	The specified PDO channel number does not exist.		
Troubleshooting	Check if the input value is correct.		

Code in DEC	3848	Code in HEX	0xF08
Name of error code	ERR_ESI_NO_PDO_MAPPING		
Description	No object dictionary (OD) is in the specified PDO channel.		
Troubleshooting	Check if the input value is correct.		

Code in DEC	3849	Code in HEX	0xF09
Name of error code	ERR_ESI_PDO_MAPPING_INSERT		
Description	CANopen OD cannot be created.		
Troubleshooting	PDO channel number is wrong or number of OD exceeds the limit.		

Code in DEC	3850	Code in HEX	0xF0A
Name of error code	ERR_ESI_PDO_MAPPING_DELETE		
Description	CANopen OD cannot be deleted.		
Troubleshooting	PDO channel number is wrong or number of OD number is 0.		

Code in DEC	3851	Code in HEX	0xF0B
Name of error code	ERR_ESI_CREATE_DISTRIBUTED_CLOCK		
Description	The DC time information in the ESI file cannot be set.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	4080	Code in HEX	0xFF0
Name of error code	ERR_ESI_ENI_INFORMATION_INITIAL		
Description	Unable to generate process data or initial command.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	4081	Code in HEX	0xFF1
Name of error code	ERR_ESI_ENI_FILE_INITIAL		
Description	The information required by ESI (EtherCAT Slave Information) file cannot be generated.		
Troubleshooting	Check if the file format of ESI (EtherCAT Slave Information) is correct.		

Code in DEC	4082	Code in HEX	0xFF2
Name of error code	ERR_ESI_ENI_FILE_SAVE		
Description	ESI (EtherCAT Slave Information) file cannot be saved.		
Troubleshooting	Check if the file directory is correct.		

Code in DEC	4096	Code in HEX	0x1000
Name of error code	ERR_ECAT_NO_SLAVE_FOUND		
Description	No connection or no slave device is detected.		
Troubleshooting	Make sure the wiring is correct and all the modules conform to the supported specifications Please update the files relevant to the Autoconfig file.		

Code in DEC	4097	Code in HEX	0x1001
Name of error code	ERR_ECAT_INITIAL_TIMEOUT		
Description	Waiting time for initialization is too long.		
Troubleshooting	Please check the following: 1. Quit the current program and check if RTSS is shutdown correctly. If not, please manually shutdown RTSS. 2. Make sure the set DC time is supported by the servo drive or module. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or the module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4098	Code in HEX	0x1002
Name of error code	ERR_ECAT_MODE_CHANGE_FAILED		
Description	Unable to switch to OP mode or Init mode.		
Troubleshooting	Please check the following: 1. Quit the current program and check if RTSS is shutdown correctly. If not, please manually shutdown RTSS. 2. Make sure the set DC time is supported by the servo drive or module. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or the module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4099	Code in HEX	0x1003
Name of error code	ERR_ECAT_SLAVE_ID		
Description	The issued number of slave station is wrong.		
Troubleshooting	Please check if the salve number exists.		

Code in DEC	4100	Code in HEX	0x1004
Name of error code	ERR_ECAT_ALIAS_SLAVE_ID		
Description	Repeated station name or the axis number exceeds the range.		
Troubleshooting	Make sure each Slave has its unique name and the axis number should be set		

	within the allowable range. Refer to Section 2.1 for the Max. axis number of the Salve.

Code in DEC	4352	Code in HEX	0x1100
Name of error code	ERR_ECAT_NEED_INITIAL		
Description	Initialization is required before starting this operation.		
Troubleshooting	Please refer to Section 3.1 to execute EtherCAT initialization procedure.		

Code in DEC	4353	Code in HEX	0x1101
Name of error code	ERR_ECAT_NEED_RESET		
Description	This API cannot be executed after the EtherCAT Master is initialized.		
Troubleshooting	Please refer to the example in Section 3.1 to execute EtherCAT initialization for motion card.		

Code in DEC	4354	Code in HEX	0x1102
Name of error code	ERR_ECAT_NEED_CONNECT		
Description	Connection has to be created before starting the operation.		
Troubleshooting	Please refer to the example in Section 3.1 to execute EtherCAT initialization procedure.		

Code in DEC	4355	Code in HEX	0x1103
Name of error code	ERR_ECAT_NEED_DC_OP		
Description	The execution is only allowed after DC time calibration is completed and the EtherCAT Master switches to OP mode.		
Troubleshooting	Please refer to the example in Section 3.1 to execute EtherCAT initialization procedure.		

Code in DEC	4356	Code in HEX	0x1104
Name of error code	ERR_ECAT_NEED_RALM		
Description	The API cannot be executed as the current alarm is not cleared.		
Troubleshooting	Please check if there is any alarm occurrence on the Slave. If yes, please follow the troubleshooting procedure and use the API for alarm reset first before executing this API.		

Code in DEC	4357	Code in HEX	0x1105
Name of error code	ERR_ECAT_NEED_SVON		
Description	The motor has to be enabled before executing the API.		
Troubleshooting	Use “ECAT_Slave_Motion_Set_Svon” to enable the motor.		

Code in DEC	4358	Code in HEX	0x1106
Name of error code	ERR_ECAT_NEED_HOMECONFIG		
Description	Setting homing parameters is required before start homing.		
Troubleshooting	Please check if the homing parameters are set before homing.		

Code in DEC	4359	Code in HEX	0x1107
Name of error code	ERR_ECAT_NEED_STOP		
Description	This API can be executed only when the specified axis is not moving.		
Troubleshooting	Please refer to the description of this API in the manual.		

Code in DEC	4608	Code in HEX	0x1200
Name of error code	ERR_ECAT_RING_BUFFER_FULL		
Description	The cycle communication command buffer of Mailbox is full.		
Troubleshooting	The number of register of API has exceeded the limit. Please use “_ECAT_Master_Get_Api_BufferLength” to acquire the current count.		

Code in DEC	4609	Code in HEX	0x1201
Name of error code	ERR_ECAT_API_PARAMETER		
Description	The input API parameter is in error.		
Troubleshooting	Please check if the parameter of the API is correct.		

Code in DEC	4610	Code in HEX	0x1202
Name of error code	ERR_ECAT_SLAVE_TYPE		
Description	This API is not supported by the module.		
Troubleshooting	Please refer to the manual and check if this API is applicable to the module or servo drive.		

Code in DEC	4611	Code in HEX	0x1203

Name of error code	ERR_ECAT_TARGET_REACHED
Description	The “target reached” signal is not triggered after the waiting time is up.
Troubleshooting	Make sure the connection is normal and no alarm has occurred on the servo drive.

Code in DEC	4612	Code in HEX	0x1204
Name of error code	ERR_ECAT_MODE_NOT_SUPPORT		
Description	This API is not supported in this motion mode.		
Troubleshooting	This API is not supported in this motion mode; Please refer to the manual to check if the motion mode is correct. If you are using a regular API function (Neither a user-defined PDO nor a user-defined PDO list), please contact Delta.		

Code in DEC	4613	Code in HEX	0x1205
Name of error code	ERR_ECAT_MOTION_TYPE		
Description	This axis is not supported in this motion mode. (It is probably restricted by the group function.)		
Troubleshooting	The motion axis is not supported in this motion mode or this axis has been grouped. Please refer to the manual to check if this motion is supported by the axis. If the error is caused by the grouping of the axes, please disable the group function first.		

Code in DEC	4614	Code in HEX	0x1206
Name of error code	ERR_ECAT_PDO_NOT_MAPPING		
Description	The OD code of this motion mode is not included the PDO mapping list.		
Troubleshooting	Please create the DAT file again or make sure the module supports this motion mode.		

Code in DEC	4615	Code in HEX	0x1207
Name of error code	ERR_ECAT_MODULE_REVISION		
Description	The firmware version of this module is not supported.		
Troubleshooting	The current version of the module or servo drive is not included in the DAT file. Please contact Delta to get the latest xml file and DAT file		

Code in DEC	4616	Code in HEX	0x1208
Name of error code	ERR_ECAT_SPEED_CONTINUE_MODE		
Description	One of the axes is not set to continuous speed mode.		
Troubleshooting	Please enable the continuous speed mode for the axis.		

Code in DEC	4617	Code in HEX	0x1209
Name of error code	ERR_ECAT_HOME_MODE		
Description	Parameter setting for homing method is in error.		
Troubleshooting	Check the parameter setting.		

Code in DEC	4618	Code in HEX	0x120a
Name of error code	ERR_ECAT_HOME_OFFSET		
Description	The homing offset setting of the homing parameter is in error.		
Troubleshooting	Check the parameter setting.		

Code in DEC	4619	Code in HEX	0x120b
Name of error code	ERR_ECAT_HOME_FIRST_SPEED		
Description	The speed setting of looking for the origin during homing is in error.		
Troubleshooting	Check the parameter setting.		

Code in DEC	4620	Code in HEX	0x120c
Name of error code	ERR_ECAT_HOME_SECOND_SPEED		
Description	In API homing parameter, after the origin is found, the speed setting is wrong when it looks for Z pulse.		
Troubleshooting	Check the parameter setting.		

Code in DEC	4621	Code in HEX	0x120d
Name of error code	ERR_ECAT_HOME_ACC		
Description	Acceleration setting of the homing parameter is in error.		
Troubleshooting	Please check the parameter setting.		

Code in DEC	4622	Code in HEX	0x120e
Name of error code	ERR_ECAT_MRAM_INDEX		
Description	The address of retentive memory is in error.		
Troubleshooting	Please check the parameter setting.		

Code in DEC	4623	Code in HEX	0x120f
Name of error code	ERR_ECAT_MRAM_INDEX_OUT_RANGE		
Description	Range setting of the retentive memory is in error.		
Troubleshooting	Please check the parameter setting.		

Code in DEC	4864	Code in HEX	0x1300
Name of error code	ERR_ECAT_PDO_TX_FAILED		
Description	Fail to send the PDO type command.		
Troubleshooting	Please check the following: 1. Quit the current program. Check if RTSS is correctly shut down; if not, manually shut down RTSS. 2. Check if the servo drive or the module supports the DC time you set. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4865	Code in HEX	0x1301
Name of error code	ERR_ECAT_SDO_TIMEOUT		
Description	SDO communication timeout.		
Troubleshooting	Please check the following: 1. Quit the current program. Check if RTSS is correctly shut down; if not, manually shut down RTSS. 2. Check if the servo drive or the module supports the DC time you set. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4866	Code in HEX	0x1302
Name of error code	ERR_ECAT_SDO_RETURN		
Description	The slave device returns SDO error code. Use API "_ECAT_Slave_CANopen_Get_ErrorCode" to acquire the error code.		
Troubleshooting	Please make sure the OD code, data content, and data size match the specification of the slave station.		

Code in DEC	4867	Code in HEX	0x1303
Name of error code	ERR_ECAT_PDO_RX_FAILED		
Description	Fail to return a PDO command.		
Troubleshooting	Please check the following: 1. Quit the current program. Check if RTSS is correctly shut down; if not, manually shut down RTSS. 2. Check if the servo drive or the module supports the DC time you set. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4868	Code in HEX	0x1304
Name of error code	ERR_ECAT_MAILBOX		
Description	Sending failure of Mailbox		
Troubleshooting	Please check the following: 1. Quit the current program. Check if RTSS is correctly shut down; if not, manually shut down RTSS. 2. Check if the servo drive or the module supports the DC time you set. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	4869	Code in HEX	0x1305
Name of error code	ERR_ECAT_SDO_BUFFER_FULL		

Description	SDO command buffer is full.
Troubleshooting	Please wait until enough SDO buffer space is available and execute the API again.

Code in DEC	5120	Code in HEX	0x1400
Name of error code	ERR_ECAT_GROUP_NUMBER		
Description	The input number of grouped axes is over 10 the maximum.		
Troubleshooting	The maximum number of the grouped axes is 10. The number should not exceed the limit.		

Code in DEC	5121	Code in HEX	0x1401
Name of error code	ERR_ECAT_GROUP_ENABLE		
Description	The group function is disabled or in hold state.		
Troubleshooting	Before start the operation, please make sure the group function is enabled or is switched to hold state.		

Code in DEC	5122	Code in HEX	0x1402
Name of error code	ERR_ECAT_GROUP_PAUSE		
Description	This action can only be done when group function is in hold state.		
Troubleshooting	N/A		

Code in DEC	5123	Code in HEX	0x1403
Name of error code	ERR_ECAT_GROUP_SLAVE		
Description	The slave device has been used by other group or in this group.		
Troubleshooting	A slave station can only be used in one group at a time.		

Code in DEC	5124	Code in HEX	0x1404
Name of error code	ERR_ECAT_GROUP_MODE		
Description	The group function is not supported for now.		
Troubleshooting	The mode you selected is not supported for now, please refer to the manual and document for version update.		

Code in DEC	5125	Code in HEX	0x1405
Name of error code	ERR_ECAT_GROUP_ALREADY_USED		
Description	This group number is in use.		
Troubleshooting	The group you selected has been enabled or stopped; this command is not supported. Please make sure the group number is correct or disable the group setting first.		

Code in DEC	5126	Code in HEX	0x1406
Name of error code	ERR_ECAT_GROUP_TYPE		
Description	The type setting has to be done before group function is enabled. Once the function is enabled, setting the type is not allowed.		
Troubleshooting	Before enabling the group function, please execute API "_ECAT_Slave_User_Motion_Control_Set_Type" and make sure no error has occurred. Please note that the group function is disabled when using the API "_ECAT_Slave_User_Motion_Control_Set_Type".		

Code in DEC	5127	Code in HEX	0x1407
Name of error code	ERR_ECAT_GROUP_SVON		
Description	Please enable all motors in the group.		
Troubleshooting	Before enabling the group, please make sure each motor has been enabled.		

Code in DEC	5128	Code in HEX	0x1408
Name of error code	ERR_ECAT_GROUP_ALM		
Description	An alarm has occurred on one of the axes.		
Troubleshooting	Before enabling the group, please make sure no alarm occurs on any of the axes.		

Code in DEC	5129	Code in HEX	0x1409
Name of error code	ERR_ECAT_GROUP_DATA_BUFFER		
Description	The buffer is full. (799 data)		
Troubleshooting	The Max. buffer size for user-defined data is 800. Acquire the current data count by using API "_ECAT_Slave_User_Motion_Control_Get_DataCnt". Make sure the number is smaller than 799 or a command cannot be issued successfully.		

Code in DEC	5130	Code in HEX	0x140A
Name of error code	ERR_ECAT_GROUP_TIMEOUT		
Description	No response is sent from the EtherCAT kernel during group operation.		
Troubleshooting	Please check if the communication and operation is normal. If you cannot find the cause, please contact Delta.		

Code in DEC	5376	Code in HEX	0x1500
Name of error code	ERR_ECAT_SERVO_PARA_EMPTY		
Description	This servo parameter does not exist.		
Troubleshooting	Please check if this parameter code is supported by Delta servo drive.		

Code in DEC	5377	Code in HEX	0x1501
Name of error code	ERR_ECAT_SERVO_PARA_RO		
Description	This servo parameter is read-only.		
Troubleshooting	Writing in is not allowed as this parameter is read-only. Please refer to the servo drive user manual.		

Code in DEC	5378	Code in HEX	0x1502
Name of error code	ERR_ECAT_SERVO_COMPARE_ENABLE		
Description	An error has occurred when setting the pulse compare function of the servo drive.		
Troubleshooting	Make sure the firmware version of the servo drive is the latest.		

Code in DEC	5632	Code in HEX	0x1600
Name of error code	ERR_ECAT_RECORD_TYPE		
Description	When the auto recording function is enabled, the acquired data cannot be modified.		
Troubleshooting	Please disable the auto recording function and then execute the API again.		

Code in DEC	5888	Code in HEX	0x1700
Name of error code	ERR_ECAT_MPG_ENABLE		
Description	This MPG group has been enabled. Please disable it first before executing the API.		
Troubleshooting	The same MPG group can only be enabled once at a time. Please disable it and carry on the execution.		

Code in DEC	5889	Code in HEX	0x1701
Name of error code	ERR_ECAT_MPG_CONFIG		
Description	Setting of this MPG group is not completed.		
Troubleshooting	The MPG group has to set before it is enabled		

Code in DEC	12288	Code in HEX	0x3000
Name of error code	ERR_ECAT_SECURITY_OPERATING		
Description	Verification procedure is in progress. Please wait and then execute the API.		
Troubleshooting	Only one verification procedure can be executed at a time.		

Code in DEC	12289	Code in HEX	0x3001
Name of error code	ERR_ECAT_SECURITY_NEED_LOGIN		
Description	Login is required to proceed with the operation.		
Troubleshooting	The verification requires logging in. Please refer to the description in Chapter 29.		

Code in DEC	12290	Code in HEX	0x3002
Name of error code	ERR_ECAT_SECURITY_CONNECT		
Description	This function fails to access the security kernel.		
Troubleshooting	Restart the PAC or PC and try again. If this error occurs frequently, please contact Delta.		

Code in DEC	32825	Code in HEX	0x8039
Name of error code	ERR_PATH_ECAT_NEED_ENABLE		
Description	This API requires to be enabled.		
Troubleshooting	Enable this API before using it.		

Code in DEC	32826	Code in HEX	0x803a
Name of error code	ERR_PATH_ECAT_ECAM_ENABLE		
Description	This API cannot be executed when E-cam is enabled.		
Troubleshooting	N/A		

Code in DEC	32827	Code in HEX	0x803b
Name of error code	ERR_PATH_ECAT_ECAM_MASTERSOURCE		
Description	The master axis of E-cam is not set.		
Troubleshooting	Set the source of the master before E-cam is enabled.		

Code in DEC	53248	Code in HEX	0xD000
Name of error code	ERR_RTX_RTSS_LOAD		
Description	RTSS cannot be enabled/disabled correctly.		
Troubleshooting	Make sure the RTX environment is correctly created. Restart the PAC if necessary.		

Code in DEC	53249	Code in HEX	0xD001
Name of error code	ERR_RTX_CONNECT_LINK_FAILED		
Description	Fail to access RTSS shared memory or the license verification is failed.		
Troubleshooting	Make sure RTX environment is correctly created. Restart the device if necessary before executing the API.		

Code in DEC	53250	Code in HEX	0xD002
Name of error code	ERR_RTX_EVENT_FAILED		
Description	Fail to access the RTSS event.		
Troubleshooting	Make sure RTX environment is correctly created. Restart the device if necessary.		

Code in DEC	53251	Code in HEX	0xD003
Name of error code	ERR_RTX_CONNECT_FAILED		
Description	Failure of handshaking with RTSS shared memory.		
Troubleshooting	Please make sure the RTX environment is correctly created. Restart the device if necessary before executing the API.		

Code in DEC	53252	Code in HEX	0xD004
Name of error code	ERR_RTX_CONFIG_EDITED		
Description	Restarting the computer is required to activate the default setting of RTX.		
Troubleshooting	As some parameters of RTX correlates with EtherCAT Master, DO NOT adjust the relevant setting. If this error occurs, please restart the PAC and the parameter will be set automatically.		

Code in DEC	53253	Code in HEX	0xD005
Name of error code	ERR_RTX_SECURITY_FAILED		
Description	The RTX license you are using might be a piracy.		
Troubleshooting	Software verification error. Please check if you are using the right software.		

Code in DEC	53254	Code in HEX	0xD006
Name of error code	ERR_RTX_COMMANDING		
Description	No response when RTX special command is issued.		
Troubleshooting	Please check the following: 1. Quit the current program. Check if RTSS is correctly shut down; if not, manually shut down RTSS. 2. Check if the servo drive or the module supports the DC time you set. 3. Check if the communication of the servo drive or module is normal. 4. Restart the servo drive or module and try again. 5. Use the default setting of EtherCAT Master through EcNavi and then try again. 6. If the issue persists, please contact Delta.		

Code in DEC	53255	Code in HEX	0xD007
Name of error code	ERR_RTX_RTSS_SYSTEM_NOT_SUPPORT		
Description	This API is not supported by RTSS.		
Troubleshooting	N/A		

Code in DEC	53256	Code in HEX	0xD008
Name of error code	ERR_RTX_NOT_SUPPORT		
Description	The API is not supported by this RTX version.		
Troubleshooting	N/A		

Code in DEC	53257	Code in HEX	0xD009
Name of error code	ERR_RTX_THREAD_CREATE_FAILED		
Description	Fail to enable communication status of RTX thread		
Troubleshooting	Please check the following: 1. Power on the PAC again and execute the API. 2. If the issue persists, please contact Delta.		

Code in DEC	53258	Code in HEX	0xD00a
Name of error code	ERR_RTX_RTSS_START_FAILED		
Description	Fail to enable the RTX system.		
Troubleshooting	Please check the following: 1. Power on the PAC again and execute the API. 2. If the issue persists, please contact Delta.		

Code in DEC	53504	Code in HEX	0xD100
Name of error code	ERR_RTX_WIN32_SYSTEM_NOT_SUPPORT		
Description	The callback function is not supported by Win32 system.		
Troubleshooting	Make sure you are using the RTX system and calling ECAT_RTX_RTDLL.rtdll.		

Code in DEC	53505	Code in HEX	0xD101
Name of error code	ERR_RTX_CALLBACK_CLOSE		
Description	Modifying the callback function is not allowed when it is enabled.		
Troubleshooting	Disable the callback function before setting it.		

Code in DEC	53506	Code in HEX	0xD102
Name of error code	ERR_RTX_CALLBACK_FUNCTION		
Description	Callback function cannot be enabled because its setting is in error.		

Troubleshooting	Use API “ECAT_Master_Callback_Set_Function” to set the function to be called with the callback function.

Code in DEC	53507	Code in HEX	0xD103
Name of error code	ERR_RTX_CALLBACK_THREAD		
Description	When callback function is enabled, the thread is not correctly executed.		
Troubleshooting	Make sure the RTX environment is correctly created. Restart the PAC if necessary.		

Code in DEC	53760	Code in HEX	0xD200
Name of error code	ERR_RTX_ERRORLOG_NOT_ENABLE		
Description	Auto recording function is not enabled.		
Troubleshooting	Enable the auto recording function.		

Code in DEC	53761	Code in HEX	0xD201
Name of error code	ERR_RTX_ERRORLOG_COUNT_ERROR		
Description	The specified index of error count is wrong.		
Troubleshooting	Check if the error count you specified exceeds the range.		

Code in DEC	57344	Code in HEX	0xE000
Name of error code	ERR_CARD_NO_FOUND		
Description	Motion card is not found.		
Troubleshooting	Make sure the motion card and driver are correctly installed. Check if they are shown in the Device Manager in Windows.		

Code in DEC	57345	Code in HEX	0xE001
Name of error code	ERR_CARD_NO_RESPONSE		
Description	No response is sent from the motion card after the command is sent.		
Troubleshooting	API timeout is usually caused by software error. Please contact Delta.		

Code in DEC	57346	Code in HEX	0xE002
Name of error code	ERR_CARD_CONNECT_FAILED		
Description	Connection error of the motion card and the driver.		
Troubleshooting	Make sure the driver is successfully installed. Restart the PC if necessary.		

Code in DEC	57347	Code in HEX	0xE003
Name of error code	ERR_CARD_MEMORY_NOT_ENOUGH		
Description	When recording function is applied, number of motion cards exceeds the limit. (Max. is 24).		
Troubleshooting	This is caused by the hardware specifications. To use more than 24 motion cards for recording, please purchase RTX version.		

Code in DEC	57348	Code in HEX	0xE004
Name of error code	ERR_CARD_LOAD_AUTOCONFIG_FILE		
Description	Fail to load the AutoConfig file.		
Troubleshooting	Make sure the AutoConfig file is in the correct directory and re-initialize EtherCAT Master.		

Code in DEC	57349	Code in HEX	0xE005
Name of error code	ERR_CARD_SECURITY_FAILED		
Description	Security verification failed.		
Troubleshooting	Please contact your distributor.		

Code in DEC	57350	Code in HEX	0xE006
Name of error code	ERR_CARD_UPGRADE_CREATE_THREAD_FAILED		
Description	Fail to open the update window.		
Troubleshooting	Re-initialize the system for the update.		

Code in DEC	57351	Code in HEX	0xE007
Name of error code	ERR_CARD_UPGRADE_NO_RESPONSE		
Description	No response returned during the update.		
Troubleshooting	Re-initialize the system. If the issue persists, please contact Delta.		

Code in DEC	57352	Code in HEX	0xE008
Name of error code	ERR_CARD_UPGRADE_NO_RESOURSE		
Description	No update file (EtherCAT_DLL) is found.		
Troubleshooting	Please check the DLL version and operating system.		

Code in DEC	57353	Code in HEX	0xE009
Name of error code	ERR_CARD_UPGRADE_LOAD_RESOURSE		
Description	Fail to access the updating resource (EtherCAT_DLL).		
Troubleshooting	Please check the DLL version and the operating system.		

Code in DEC	57354	Code in HEX	0xE00A
Name of error code	ERR_CARD_UPGRADE_TIMEOUT		
Description	System update timeout.		
Troubleshooting	Re-initialize the system and it will be updated. If the issue persists, please contact Delta.		

Code in DEC	57355	Code in HEX	0xE00B
Name of error code	ERR_CARD_UPGRADE_FAILED		
Description	System update failure		
Troubleshooting	Firmware update failure. Please contact Delta.		

Code in DEC	61440	Code in HEX	0xF000
Name of error code	ERR_ECAT_DLL_IS_USED		
Description	EtherCAT_DLL file has been opened.		
Troubleshooting	Please check if the EtherCAT_DLL is being used by multiple programs.		

Code in DEC	61441	Code in HEX	0xF001
Name of error code	ERR_ECAT_NO_DLL_FOUND		
Description	Connection error of EtherCAT_DLL file, RTSS and DDL file of motion card.		
Troubleshooting	Make sure either RTSS environment or PCI motion card exists.		

Code in DEC	61442	Code in HEX	0xF002
Name of error code	ERR_ECAT_NO_RTSS_DLL_FOUND		
Description	Connection failure of EtherCAT_DLL and RTSSDLL.		
Troubleshooting	Check if RTSS environment exists.		

Code in DEC	61443	Code in HEX	0xF003
Name of error code	ERR_ECAT_NO_CARD_DLL_FOUND		
Description	Connection between EtherCAT_DLL file and DLL file of motion card is in error.		
Troubleshooting	Check if PCI motion card exists.		

Code in DEC	61444	Code in HEX	0xF004
Name of error code	ERR_ECAT_NO_ESI_DLL_FOUND		
Description	Connection between EtherCAT DLL file and ESI file (EtherCAT Slave Information) is in error.		
Troubleshooting	Please check if the files are in the same directory.		

Code in DEC	61445	Code in HEX	0xF005
Name of error code	ERR_ECAT_SAME_CARD_NUMBER		
Description	Repeated RTSS or motion card number.		
Troubleshooting	Make sure no card number is repeated.		

Code in DEC	61446	Code in HEX	0xF006
Name of error code	ERR_ECAT_CARDNO_ERROR		
Description	A non-existing card number of EtherCAT is used.		
Troubleshooting	Please check if you have switch to the correct card number on the knob.		

Code in DEC	61447	Code in HEX	0xF007
Name of error code	ERR_ECAT_GET_DLL_PATH		
Description	Unable to acquire the directory information of DLL file.		
Troubleshooting	Please check the directory of the DLL file.		

Code in DEC	61448	Code in HEX	0xF008
Error Code Name	ERR_ECAT_GET_DLL_VERSION		
Description	Cannot acquire the version information of DLL.		
Troubleshooting	Please check the version of DLL.		

Code in DEC	61449	Code in HEX	0xF009
Name of error code	ERR_ECAT_NOT_SUPPORT		
Description	This DMCNET type API is not supported by EtherCAT.		
Troubleshooting	Use the EtherCAT type API that serves the similar function.		

Code in DEC	65535	Code in HEX	0xFFFF
Name of error code	ERR_ECAT_LOADLIB_EMPTY		
Description	Fail to call DLL resource in RTSS.		
Troubleshooting	Please check the following: 1. Power on the PAC and try again. 2. If the issue persists, please contact Delta.		

Revision History

Release date	Version	Chapter	Revision Contents
March, 2017	V1.0 (First edition)	N/A	

(This page is intentionally left blank.)