HIWIN® MIKROSYSTEM





API Library Reference Manual

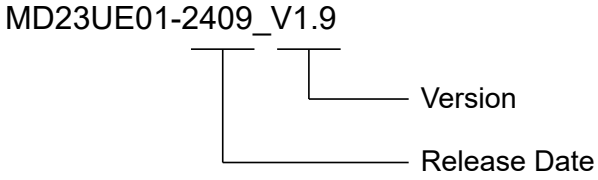
for Servo Drives

www.hiwinmikro.tw

MD23UE01-2409_V1.9

Revision History

The version of the manual is also indicated on the bottom of the front cover.



Release Date	Version	DLL Version	Revision Contents
Sep. 25th, 2024	1.9	2.2.0.0	(1) Add **Related Documents**. (2) Update section 1.1 **About this manual**. (3) Update section 1.3.1 **Available slaves**. (4) Update section 1.4 **Device communication setting**. (5) Update section 2.1 **MPI_CreateAxis**. (6) Update section 3.1 **MPI_ConnectMegaulink**. (7) Update section 3.2 **MPI_Disconnect**. (8) Update section 3.6 **MPI_Reconnect**. (9) Update section 3.9 **MPI_ConnectMegaulinkByNetworkAdapter**. (10) Add section 3.10 **MPI_ConnectUsbById**. (11) Add section 3.11 **MPI_ConnectUsbByInfo**. (12) Add section 3.12 **MPI_GetUsbDevice**. (13) Update section 6.13 **MPI_SetCallBackStates**. (14) Update chapter 10 **Error code**.
Dec. 9th, 2022	1.8	2.1.7.0	(1) Update section 1.3.2 **Prerequisite**.
Mar. 23rd, 2022	1.7	2.1.7.0	(1) Update section 1.3.1 **Available slaves**. (2) Update section 1.3.2 **Prerequisite**. (3) Update section 2.2 **MPI_DestroyAxis**. (4) Add section 3.7 **MPI_GetNetworkAdapterInfo**. (5) Add section 3.8 **MPI_ClearNetworkAdapterInfo**. (6) Add section 3.9 **MPI_ConnectMegaulinkByNetworkAdapter**. (7) Update section 5.1 **MPI_ServoOn**. (8) Update section 5.3 **MPI_MoveAbsolute**. (9) Update section 5.4 **MPI_MoveRelative**. (10) Update section 5.11 **MPI_ClearError**. (11) Update section 6.12 **MPI_WaitMoveOver**.

Release Date	Version	DLL Version	Revision Contents
			(12) Update section 8.1 **MPI_SetTriggerPosition**. (13) Update section 8.2 **MPI_SetTriggerInterval**. (14) Add section 8.5 **MPI_SetTriggerPositionArray**. (15) Add section 8.6 **MPI_SetTriggerPositionStateArray**. (16) Add section 8.7 **MPI_SetTriggerPositionStartIndex**. (17) Add section 8.8 **MPI_SetTriggerPositionEndIndex**. (18) Update section 9.5 **MPI_GetErrorData**. (19) Update section 9.6 **MPI_GetErrorDataArray**. (20) Update section 9.7 **MPI_RunFuncPDL**. (21) Update section 9.8 **MPI_IsTaskRunning**. (22) Update section 9.9 **MPI_KillTask**. (23) Add section 9.11 **MPI_GetVarArray**. (24) Add section 9.12 **MPI_SetVarArray**. (25) Add section 9.13 **MPI_GetErrorMessage**. (26) Update chapter 10 **Error Code**. (27) Add chapter 12 **D Series Servo Drive Error**.
Apr. 28th, 2021	1.6	2.1.3.0	(1) Add section 3.6 **MPI_Reconnect**. (2) Add section 9.10 **MPI_SendRAMtoFlash**. (3) Update chapter 10 **Error Code**. (4) Update chapter 11 **E1 Series Servo Drive Alarm**.
Oct. 30th, 2020	1.5	2.1.2.0	(1) Update section 1.2 **mega-ulink specification**. (2) Update section 1.3.2 **Prerequisite**. (3) Update section 2.1 **MPI_CreateAxis**. (4) Add section 3.5 **MPI_SetCommunicationVar**. (5) Add section 6.13 **MPI_SetCallBackStates**. (6) Add section 9.5 **MPI_GetErrorData**. (7) Add section 9.8 **MPI_IsTaskRunning**. (8) Add section 9.9 **MPI_KillTask**. (9) Update chapter 10 **Error Code**. (10) Update chapter 11 **E1 Series Servo Drive Alarm**.
May 5th, 2020	1.4	2.0	(1) The manual name is changed from *API Library Reference Manual for D series Servo Drives* to *API Library Reference Manual for Servo Drives*. Refer to this manual while using API function to control E1 series servo drives or D series servo drives. (2) Update section 1.1 **About this Manual**. (3) Update section 1.3.1 **Available slaves**.

Release Date	Version	DLL Version	Revision Contents
			(4) Update section 1.3.2 **Prerequisite**. (5) Update section 1.3.3 **Database verification**. (6) Update section 1.3.5 **Drive setting**. (7) Update section 2.1 **MPI_CreateAxis**. (8) Update section 2.3 **MPI_InitialAxis**. (9) Update section 4.17 **MPI_GetSwLimitPos**. (10) Update section 4.18 **MPI_SetSwLimitPos**. (11) Update section 5.1 **MPI_ServoOn**. (12) Update section 5.2 **MPI_ServoOff**. (13) Update section 5.5 **MPI_JogPositive**. (14) Update section 5.6 **MPI_JogNegative**. (15) Update section 5.8 **MPI_StartHome**. (16) Update section 5.12 **MPI_EnableSwLimit**. (17) Update section 5.13 **MPI_DisableSwLimit**. (18) Add section 6.1 **MPI_IsDriveReady**. (19) Update section 6.7 **MPI_IsSwLimitEn**. (20) Update section 6.9 **MPI_IsSwLimit**. (21) Update section 8.1 **MPI_SetTriggerPosition**. (22) Update section 8.2 **MPI_SetTriggerInterval**. (23) Update section 8.3 **MPI_EnableTrigger**. (24) Update section 8.4 **MPI_DisableTrigger**. (25) Add section 9.5 **MPI_GetErrorDataArray**. (26) Update chapter 10 **Error Code**. (27) Add chapter 11 **E1 series Servo Drive Alarm**.
Jan. 8th, 2020	1.3	1.17	Add description in section 1.1: API library does not support full-closed loop (dual-loop) function.
Jun. 27th, 2018	1.2	1.17	Support HIOM.
Aug. 19th, 2016	1.1	1.11	(1) Modify some descriptions. (2) Modify argument type: MPI_GetInputStatus, MPI_GetOutputStatus, MPI_GetState and MPI_SetState. (3) Modify argument unit: MPI_SetTriggerPosition and MPI_SetTriggerInterval. (4) Add figure 3.1.1 and figure 3.1.2.
Feb. 29th, 2016	1.0	1.09	First edition.

Related Documents

Through related documents, users can quickly understand the positioning of this manual and the correlation between manuals and products. Go to HIWIN MIKROSYSTEM's official website → Download → Manual Overview for details (https://www.hiwinmikro.tw/Downloads/ManualOverview_EN.htm).

Table of Contents

1. Preface ····· 1-1
   - 1.1 About this manual ····· 1-2
   - 1.2 mega-ulink specification ····· 1-3
   - 1.3 Before using HIWIN API library ····· 1-4
     - 1.3.1 Available slaves ····· 1-4
     - 1.3.2 Prerequisite ····· 1-4
     - 1.3.3 Database verification ····· 1-5
     - 1.3.4 Database update ····· 1-6
     - 1.3.5 Drive setting ····· 1-7
   - 1.4 Device communication setting ····· 1-9

2. Initialization ····· 2-1
   - 2.1 MPI_CreateAxis ····· 2-2
   - 2.2 MPI_DestroyAxis ····· 2-3
   - 2.3 MPI_InitialAxis ····· 2-4
   - 2.4 MPI_GetVersion ····· 2-5

3. Communication ····· 3-1
   - 3.1 MPI_ConnectMegaulink ····· 3-2
   - 3.2 MPI_Disconnect ····· 3-3
   - 3.3 MPI_GetSlaveAmount ····· 3-4
   - 3.4 MPI_GetComSuccess ····· 3-5
   - 3.5 MPI_SetCommunicationPar ····· 3-6
   - 3.6 MPI_Reconnect ····· 3-7
   - 3.7 MPI_GetNetworkAdapterInfo ····· 3-8
     - 3.7.1 TNetworkAdapterInfo ····· 3-8
     - 3.7.2 TNetworkAdapter ····· 3-8
   - 3.8 MPI_ClearNetworkAdapterInfo ····· 3-9
   - 3.9 MPI_ConnectMegaulinkByNetworkAdapter ····· 3-10
   - 3.10 MPI_ConnectUsbById ····· 3-11
   - 3.11 MPI_ConnectUsbByInfo ····· 3-12
   - 3.12 MPI_GetUsbDevice ····· 3-13

4. Access variables ····· 4-1
   - 4.1 MPI_GetSlaveID ····· 4-2
   - 4.2 MPI_GetAxisName ····· 4-3
   - 4.3 MPI_GetFeedbackPos ····· 4-4

4.4 MPI_GetFeedbackVel ····· 4-5
4.5 MPI_GetVel ····· 4-6
4.6 MPI_SetVel ····· 4-7
4.7 MPI_GetAcc ····· 4-8
4.8 MPI_SetAcc ····· 4-9
4.9 MPI_GetDec ····· 4-10
4.10 MPI_SetDec ····· 4-11
4.11 MPI_GetDecKill ····· 4-12
4.12 MPI_SetDecKill ····· 4-13
4.13 MPI_GetSmoothTime ····· 4-14
4.14 MPI_SetSmoothTime ····· 4-15
4.15 MPI_GetInPosWidth ····· 4-16
4.16 MPI_SetInPosWidth ····· 4-17
4.17 MPI_GetSwLimitPos ····· 4-18
4.18 MPI_SetSwLimitPos ····· 4-19

5. Axis motion ····· 5-1
5.1 MPI_ServoOn ····· 5-2
5.2 MPI_ServoOff ····· 5-3
5.3 MPI_MoveAbsolute ····· 5-4
5.4 MPI_MoveRelative ····· 5-5
5.5 MPI_JogPositive ····· 5-6
5.6 MPI_JogNegative ····· 5-7
5.7 MPI_StopMotion ····· 5-8
5.8 MPI_StartHome ····· 5-9
5.9 MPI_SetZero ····· 5-10
5.10 MPI_ResetDrive ····· 5-11
5.11 MPI_ClearError ····· 5-12
5.12 MPI_EnableSwLimit ····· 5-13
5.13 MPI_DisableSwLimit ····· 5-14

6. Access states ····· 6-1
6.1 MPI_IsDriveReady ····· 6-2
6.2 MPI_IsReady ····· 6-3
6.3 MPI_IsHomed ····· 6-4
6.4 MPI_IsMoving ····· 6-5
6.5 MPI_IsStopped ····· 6-6
6.6 MPI_IsError ····· 6-7
6.7 MPI_IsSwLimitEn ····· 6-8
6.8 MPI_IsHwLimit ····· 6-9
6.9 MPI_IsSwLimit ····· 6-10

6.10 MPI_WaitServoReady ····· 6-11
6.11 MPI_WaitHomeOver ····· 6-12
6.12 MPI_WaitMoveOver ····· 6-13
6.13 MPI_SetCallBackStates ····· 6-14

7. General input/output ····· 7-1
7.1 MPI_GetInputStatus ····· 7-2
7.2 MPI_GetOutputStatus ····· 7-3
7.3 MPI_SetOutput ····· 7-4
7.4 MPI_ClearOutput ····· 7-5
7.5 MPI_ToggleOutput ····· 7-6

8. Specific input/output ····· 8-1
8.1 MPI_SetTriggerPosition ····· 8-2
8.2 MPI_SetTriggerInterval ····· 8-3
8.3 MPI_EnableTrigger ····· 8-4
8.4 MPI_DisableTrigger ····· 8-5
8.5 MPI_SetTriggerPositionArray ····· 8-6
8.6 MPI_SetTriggerPositionStateArray ····· 8-7
8.7 MPI_SetTriggerPositionStartIndex ····· 8-8
8.8 MPI_SetTriggerPositionEndIndex ····· 8-9

9. Access specific variable/state ····· 9-1
9.1 MPI_GetVar ····· 9-2
9.2 MPI_SetVar ····· 9-3
9.3 MPI_GetState ····· 9-4
9.4 MPI_SetState ····· 9-5
9.5 MPI_GetErrorData ····· 9-6
9.6 MPI_GetErrorDataArray ····· 9-7
9.7 MPI_RunFuncPDL ····· 9-8
9.8 MPI_IsTaskRunning ····· 9-9
9.9 MPI_KillTask ····· 9-10
9.10 MPI_SendRAMtoFlash ····· 9-11
9.11 MPI_GetVarArray ····· 9-12
9.12 MPI_SetVarArray ····· 9-13
9.13 MPI_GetErrorMessage ····· 9-14

10. Error code ····· 10-1

11. E1 series servo drive alarm ····· 11-1

12. D series servo drive error ····· 12-1

(This page is intentionally left blank.)

1. Preface

1. Preface ····· 1-1
   1.1 About this manual ····· 1-2
   1.2 mega-ulink specification ····· 1-3
   1.3 Before using HIWIN API library ····· 1-4
      1.3.1 Available slaves ····· 1-4
      1.3.2 Prerequisite ····· 1-4
      1.3.3 Database verification ····· 1-5
      1.3.4 Database update ····· 1-6
      1.3.5 Drive setting ····· 1-7
   1.4 Device communication setting ····· 1-9

1.1 About this manual

This manual is applicable to HIWIN E1 series and D series servo drives supporting mega-ulink or USB communication. The hardware suitable for USB communication is the same as the servo drive hardware supported by Thunder. mega-ulink utilizes EtherCAT bus type hardware, which can chain SubDevices to a local controller. The streamlined hardware and software with standard components provide a simple system integration in localized operation or setup. Such system maintains availability of the components and distributed multi-axis system at low cost. The controller can be a computer with Microsoft Visual C++, Visual Basic .NET, C# interface/tool or customized applications.

HIWIN provides API (Application Protocol Interface) library to facilitate simplified motor control with specific functions and commands. API library is an extension of MPI (mega-ulink Protocol Interface) library, a programming, communication and utility oriented software interface. In conjunction with PDL (Process Description Language), HIWIN application programming software, an efficient customization is at hand for users.

API library comprises three files, **Axis.h**, **Axis.dll** and **Axis.lib**. It is combined with MPI library, which comprises three files, **mpi.lib**, **mpi.dll** and **canlib32.dll**. API library can be applied to Microsoft Visual C++, Visual Basic .NET or C# platform and in Windows XP/7/10 Professional operating system.

The following things can be achieved via API library.

- Communicate with multiple servo drives via mega-ulink or USB.
- Servo on/off servo drive.
- Process servo drive's motion parameters (e.g., velocity, acceleration, deceleration…).
- Process motion status (e.g., servo ready, servo errors, moving…).
- Drive the motor by jogging with absolute/relative parameters, aborting motor motion and executing homing procedure.
- Monitor input/output status and set on/off states at outputs.

Note:

(1) For D series servo drives, API library does not support full-closed loop (dual-loop) function.
(2) When using USB communication, ensure that there are enough USB ports. Up to 5 USB connections can be established.
(3) USB driver does not support ARM architecture, so the connection may not be normal in Windows on ARM architecture.
(4) mega-ulink does not support Windows 11 or the newer operating systems.

1.2 mega-ulink specification

Table 1.2.1

Item	Specification
Physical layer	Ethernet
Baud rate	100 Mbps
Transmission cycle time	> 500 us
Number of slaves	Max. 32
Topology	Bus or Line
Distributed clocks	Not support (Non synchronized)
Basic functions	(1) Drive the motor to the absolute or relative target position (2) Homing procedure (3) Software limit protection (4) General/Specific input/output
Advanced functions	(1) Any user-defined variable/state access (2) Any user-defined PDL function execution

Note: Transmission cycle time varies as the application load of the host controller changes.

1.3 Before using HIWIN API library

1.3.1 Available slaves

- **mega-ulink**

  In HIWIN D series servo drives, the model designation of mega-ulink communication option is **F**. In HIWIN E1 series servo drives, the model designation of mega-ulink communication option is **H**. The possible combinations are listed in table 1.3.1.1.

  API library does not support the simultaneous control of E1 series and D series servo drives or the mixed use of different communications. It does not support E2 series servo drives or above either.

Table 1.3.1.1

Slave	Model Number
E1	ED1F-**H**□-□□□□-□□
D1	D1□-□□-**F**□-□-□
D2	D2□-□□□□-**F**-□□
D1-N	D1-N-□□-**F**□-□-□-□□
HIOM (HIWIN mega-ulink IO module)	HIOM-D32-□-□□

- **USB**

  All HIWIN E1 series servo drives have USB communication interface, and the connection with up to 5 USB SubDevices is supported. It does not support the simultaneous use with mega-ulink communication.

1.3.2 Prerequisite

Install the following software before using HIWIN API library.

- **Npcap 0.05 r16** (For Windows 10 operating system)

  It is the industry-standard packet capture tool for link-layer network access in Windows environment. Download it from https://github.com/nmap/npcap/releases and select **Npcap 0.05 r16**, the installation file **npcap-nmap-0.05-r16.exe** can be downloaded from **Assets**. Please refer to the license of **Npcap 0.05 r16**.

- **WinPcap** (For Windows 7 operating system)
  It is the industry-standard packet capture tool for link-layer network access in Windows environment. Download it from WinPcap official website: http://www.winpcap.org/install/default.htm
  Please refer to the license of WinPcap.

- **Lightening**
  It is the human-machine interface for HIWIN D series servo drives.
  Download it from HIWIN MIKROSYSTEM official website: https://www.hiwinmikro.tw/en/download

- **Thunder**
  It is the human-machine interface for HIWIN E1 series servo drives.
  Download it from HIWIN MIKROSYSTEM official website: https://www.hiwinmikro.tw/en/download

- **MFC library**
  Install it when installing Visual Studio, or download it from Microsoft official website.

1.3.3 Database verification

Database, the main program in servo drive, contains MDP and PDL files. It is used for the reference between registers and variables of each servo drive. Database in servo drive should be the same as that in PC.

User can utilize Wizalg program (tool in software installation package) to verify the database of servo drives. The operation procedure is shown as below. (Note: E1 series servo drives do not support database verification.)

Step 1. Open Wizalg program through the directory “C:\HIWIN\dce\toolswin\winkmi”.

Step 2. Connect PC to servo drive.

Step 3. Select **Verify data base…** in **File** to open **firmware’s** window.

Step 4. Click every servo drive in the slave tree on the left side of **firmware’s** window. A table with two columns and three rows will be displayed, as figure 1.3.3.1 shows. The 1st column is about the firmware in the servo drive, while the 2nd column is about the files in the working directory of PC. The checksums in each Csum sub-row should be the same. (The 1st Csum sub-row, Boot Dsp Program, is not applicable.)

Step 5. Blue text indicates checksums match (as figure 1.3.3.1 shows), while red text indicates checksums do not match (as figure 1.3.3.2 shows). If checksums do not match, refer to section 1.3.4 to update database.

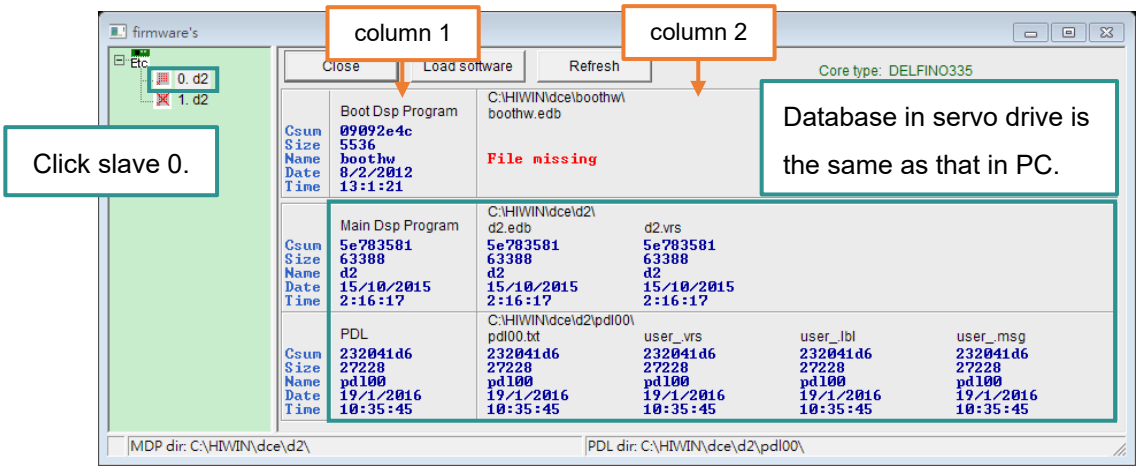


Figure 1.3.3.1 Checksums match

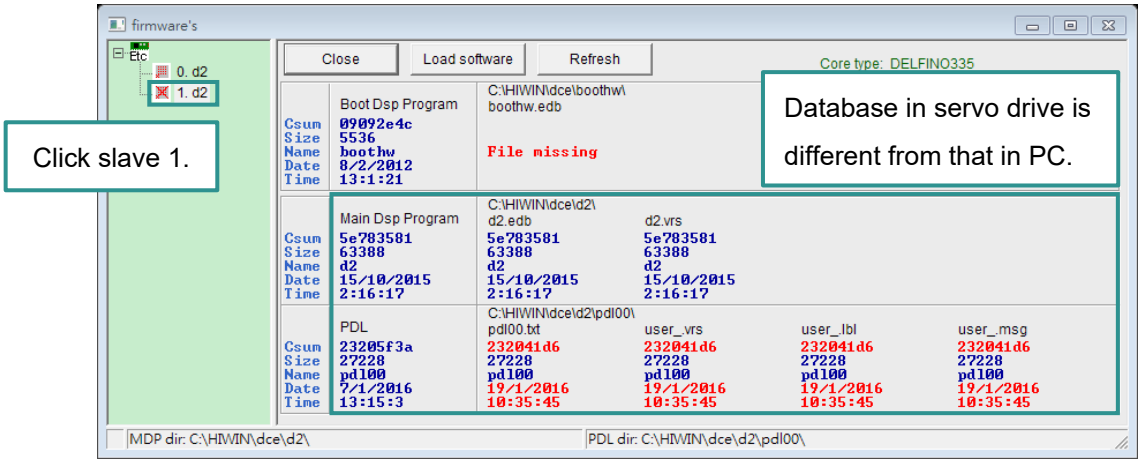


Figure 1.3.3.2 Checksums do not match

1.3.4 Database update

User can utilize Wizalg program to update MDP and PDL files. The operation procedure is shown as below. (Note: E1 series servo drives do not support this function. If multiple servo drives are connected, their firmware versions must be the same.)

Step 1. Open Wizalg program through the directory “C:\HIWIN\dce\toolswin\winkmi”.

Step 2. Connect PC to servo drive.

Step 3. Select **Auto load programs…** in **Boot** to open **Auto load programs** window.

Step 4. Uncheck **Compile PDL** and click **Run** button.

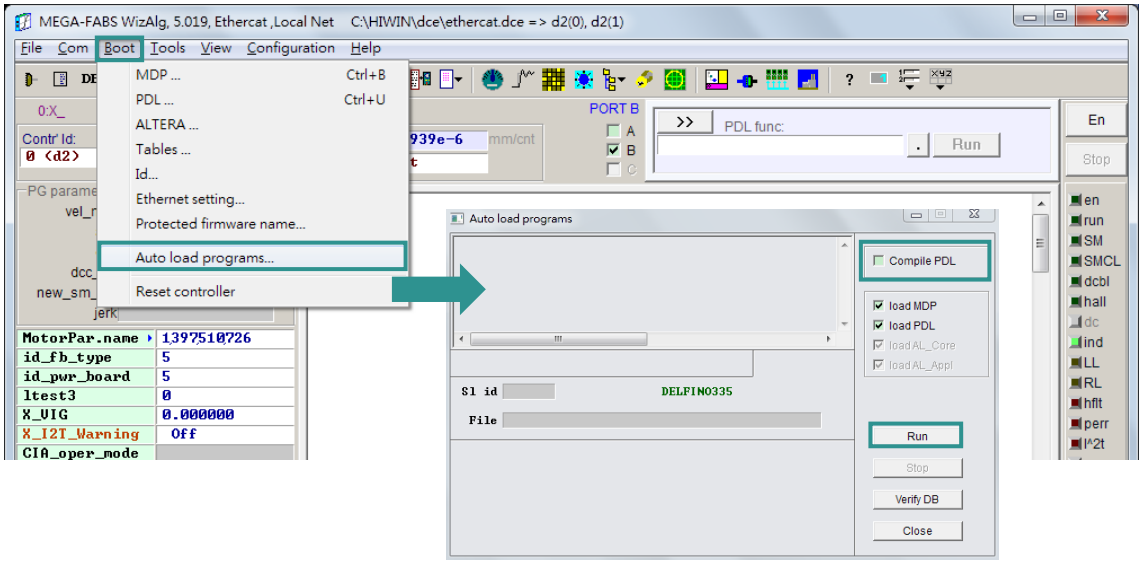


Figure 1.3.4.1

1.3.5 Drive setting

Before implementing API (Axis.dll), D series servo drives should complete configuration, tuning and movement optimization. When the servo drive completes the above setting, the HMI panel of Lightening will look like figure 1.3.5.1. The status indicator “Software enabled” should light up in green, and the axis name of the servo drive should be not only unique but also the same as the axis name in function `MPI_CreateAxis`.

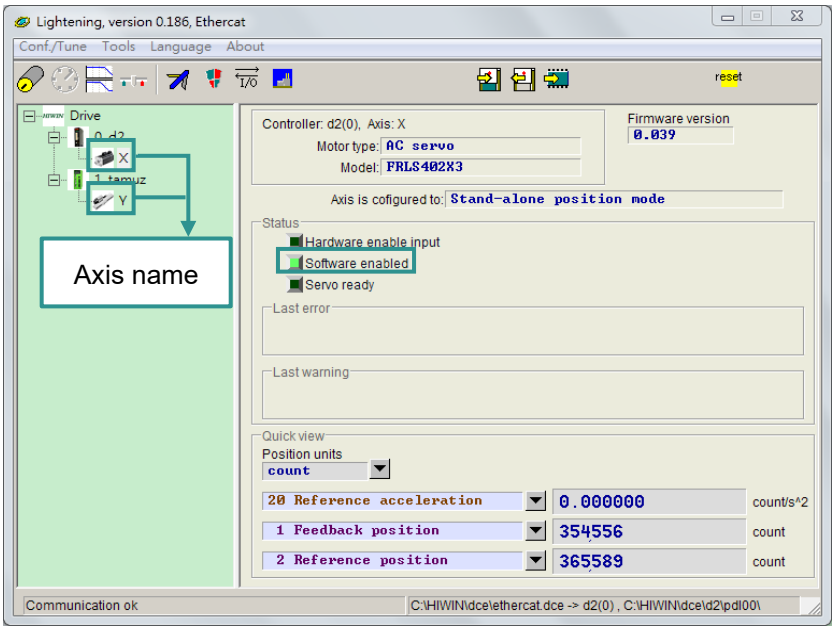


Figure 1.3.5.1

E1 series servo drives should complete configuration, tuning and movement optimization. After the servo drive completes the above setting, the HMI panel of Thunder will look like figure 1.3.5.2. The status indicator “Drive ready” should light up in green.



Figure 1.3.5.2

For E1 series servo drives, please complete display unit setting first. Set the corresponding control unit of 1 rev or 1 mm.

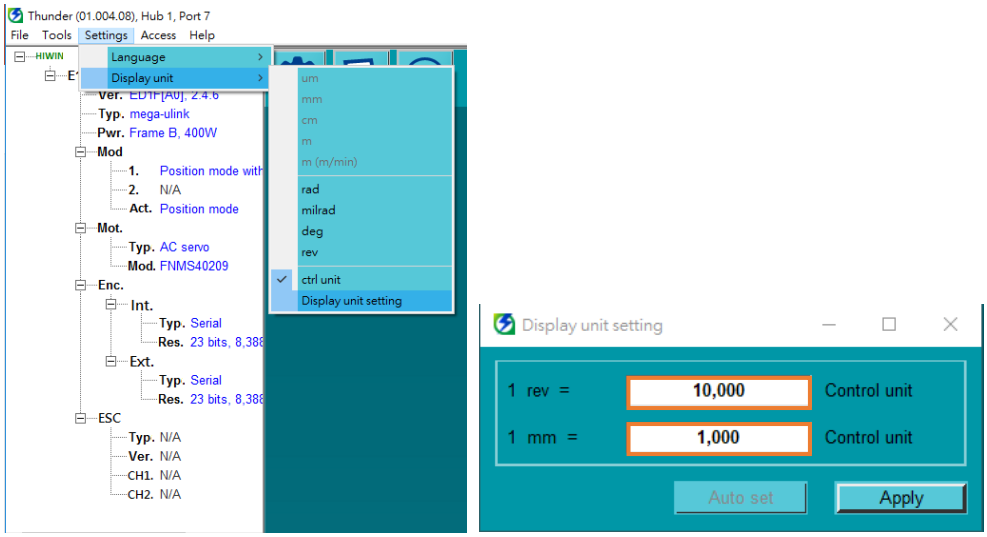


Figure 1.3.5.3

■ Precautions for E1 series servo drives

1. Pay attention to the setting range of each parameter.
2. While writing parameter into the servo drive by API function, if the unit of the servo drive is different from that of the parameter, API function will perform unit conversion. The parameter value will be rounded before being written into the servo drive.

1.4 Device communication setting

Manually select the device used in PC, as the following figure shows.

For mega-ulink communication, call function `MPI_ConnectMegaulink(blNetWorkSetting=true)` first. Then, press **SetUp…** button to open wizalg Communication Setup window, and press **EtherCat…** button to select the desired network. After that, press **Apply** button to save the setting. The setting can be restored every time users connect to EtherCAT mega-ulink.

For USB communication, call function `MPI_ConnectUsbById(*pAxis, blNetWorkSetting=true, nId)`, press **SetUp…** button, and select **USB(Dragonfly)**.

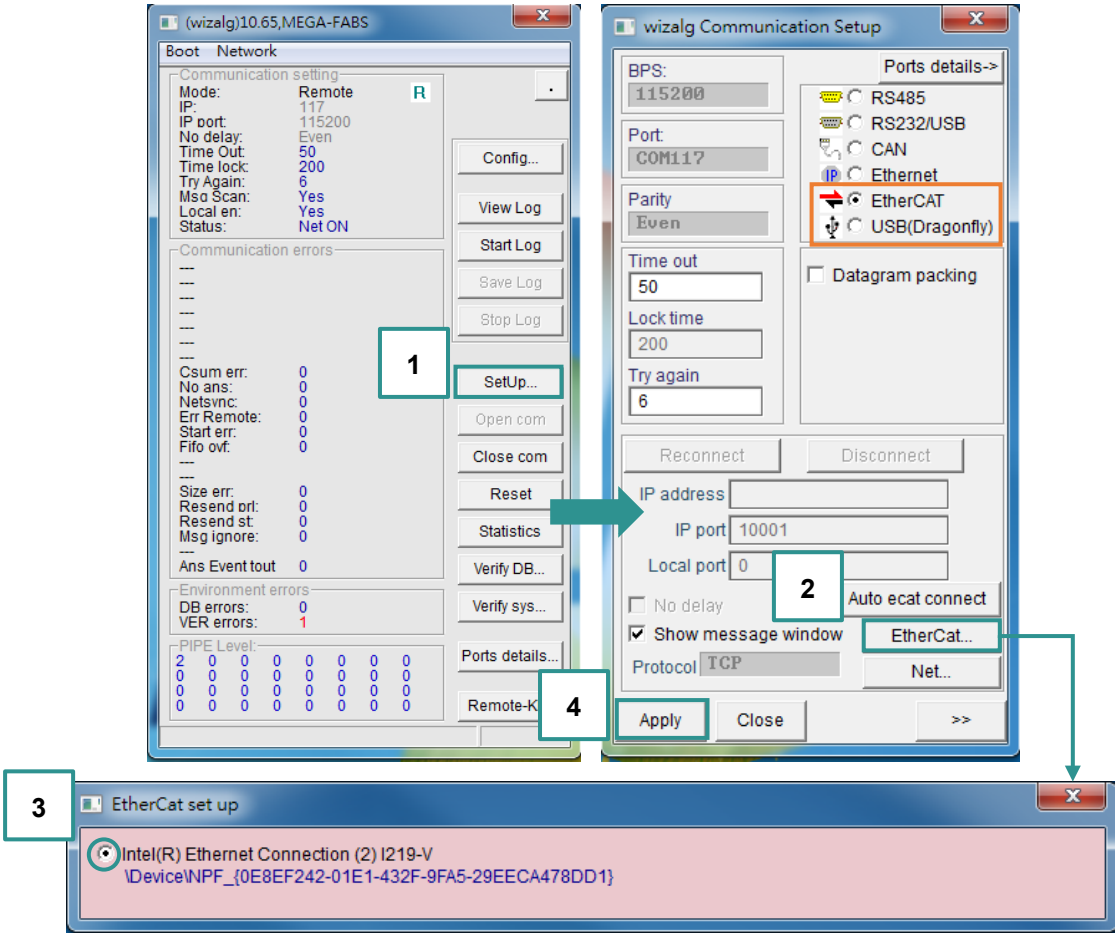


Figure 1.4.1

(This page is intentionally left blank.)

2. Initialization

2. Initialization ····· 2-1
2.1 MPI_CreateAxis ····· 2-2
2.2 MPI_DestroyAxis ····· 2-3
2.3 MPI_InitialAxis ····· 2-4
2.4 MPI_GetVersion ····· 2-5

2.1 MPI_CreateAxis

Purpose

Create an object pointer for a slave. If multiple slaves will be controlled via mega-ulink, users have to create multiple object pointers for all of them.

Prototype

```
CAxis* MPI_CreateAxis(char *pszAxisName, ENUM_UNIT_TYPE enumUnitType,
                      ENUM_DRIVE_TYPE enumDriveType,
                      ENUM_ROOT_FOLDER enumRootFolder=eC_HIWIN);
```

Parameter

*pszAxisName	Axis name.
enumUnitType	Unit type. eUT_ROTARY: rotary unit type (Unit: deg, deg/sec, deg/sec^2) eUT_LINEAR: linear unit type (Unit: mm, mm/sec, mm/sec^2)
enumDriveType	Slave type. eDT_D1: D1 servo drive eDT_D1N: D1-N servo drive eDT_D2: D2 servo drive eDT_HIOM: HIOM eDT_E1: E1 servo drive eDT_E2: E2 servo drive (only support USB communication)
enumRootFolder	Root folder for software installation. eC_HIWIN: “C:\HIWIN”; available for D series servo drive and E1 series servo drive with Thunder version below 1.5.10.0 (not included). eC_Thunder: “C:\Thunder”; available for E1 series servo drive with Thunder version 1.5.10.0 (included) or above.

Return

Pointer of CAxis class object.

Remarks

(1) This function is also available for HIOM.

(2) **Axis name should be the same as that in the slave.** Slave type should also be selected according to the axis name. Otherwise, errors will occur when `MPI_ConnectMegaulink` is called.

2.2 MPI_DestroyAxis

Purpose

Destroy the object pointer of the slave.

Prototype

```
void MPI_DestroyAxis(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

N/A

Remarks

(1) This function is also available for HIOM.

(2) Call this function before the application window is closed.

2.3 MPI_InitialAxis

Purpose

Initialize the motion parameters of the servo drive.

Prototype

```
int MPI_InitialAxis(CAxis *pAxis, double dVel, double dAcc, double dDec,
                    double dDecKill, int nSmoothTime, double dInPosWidth,
                    double dPosLimitPos, double dNegLimitPos);
```

Parameter

*pAxis	Pointer of CAxis class object.
dVel	Maximum velocity. (Unit: mm/sec or deg/sec)
dAcc	Acceleration. (Unit: mm/sec^2 or deg/sec^2)
dDec	Deceleration. (Unit: mm/sec^2 or deg/sec^2)
dDecKill	Emergency stop deceleration. (Unit: mm/sec^2 or deg/sec^2)
nSmoothTime	Smooth time. (Unit: msec)
dInPosWidth	In-position width. (Unit: mm or deg)
dPosLimitPos	Positive software limit position. (Unit: mm or deg)
dNegLimitPos	Negative software limit position. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function must be called in the beginning. Call this function after function `MPI_ConnectMegaulink` is called.

(2) To modify the specific motion condition, it is recommended to call function `MPI_SetVel`, `MPI_SetAcc`, `MPI_SetDec`, `MPI_SetDecKill`, `MPI_SetSmoothTime`, `MPI_SetInPosWidth`, `MPI_SetSwLimitPos`.

(3) E1 series servo drives do not support software limit function, for parameters *dPosLimitPos* and *dNegLimitPos*, please write -1. Functions `MPI_EnableSwLimit`, `MPI_DisableSwLimit`, `MPI_SetSwLimitPos` and `MPI_GetSwLimitPos` are not supported.

2.4 MPI_GetVersion

Purpose

Get the information of HIWIN API version.

Prototype

```
void MPI_GetVersion(char *pszVer);
```

Parameter

*pszVer The information of HIWIN API version.

Return

N/A

Remarks

(1) This function is also available for HIOM.

(2) The information of HIWIN API version will be updated from time to time.

(This page is intentionally left blank.)

3. Communication

3. Communication ····· 3-1
3.1 MPI_ConnectMegaulink ····· 3-2
3.2 MPI_Disconnect ····· 3-3
3.3 MPI_GetSlaveAmount ····· 3-4
3.4 MPI_GetComSuccess ····· 3-5
3.5 MPI_SetCommunicationPar ····· 3-6
3.6 MPI_Reconnect ····· 3-7
3.7 MPI_GetNetworkAdapterInfo ····· 3-8
3.7.1 TNetworkAdapterInfo ····· 3-8
3.7.2 TNetworkAdapter ····· 3-8
3.8 MPI_ClearNetworkAdapterInfo ····· 3-9
3.9 MPI_ConnectMegaulinkByNetworkAdapter ····· 3-10
3.10 MPI_ConnectUsbById ····· 3-11
3.11 MPI_ConnectUsbByInfo ····· 3-12
3.12 MPI_GetUsbDevice ····· 3-13

3.1 MPI_ConnectMegaulink

Purpose

Start mega-ulink communication.

Prototype

```
int MPI_ConnectMegaulink(bool blNetWorkSetting=true);
```

Parameter

blNetWorkSetting true: Show the configuration window of device communication setting.
false: Do not show the configuration window of device communication setting.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is also available for HIOM, but it does not support E2 series servo drives.

(2) Figure 3.1.1 shows the procedure of starting mega-ulink communication.

(3) To know more about device communication setting, refer to section 1.4.

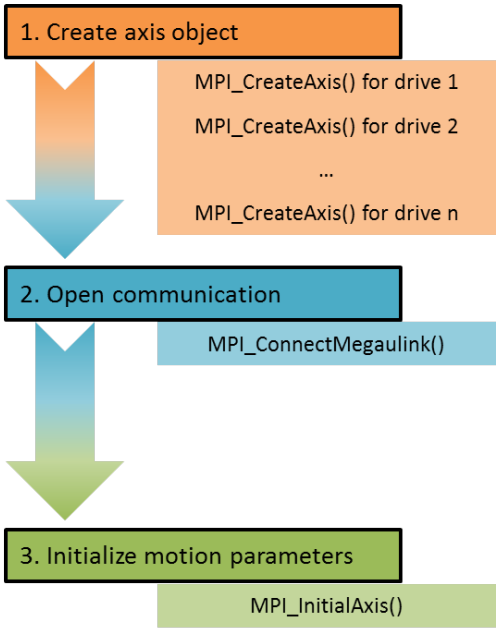


Figure 3.1.1

3.2 MPI_Disconnect

Purpose

Terminate mega-ulink communication.

Prototype

```
int MPI_Disconnect(CAxis *pAxis=nullptr);
```

Parameter

*pAxis Pointer of CAxis class object.
When mega-ulink communication is terminated, this parameter is null pointer.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is also available for HIOM.

(2) Figure 3.2.1 shows the procedure of terminating mega-ulink communication.

(3) Call this function before the application window is closed.

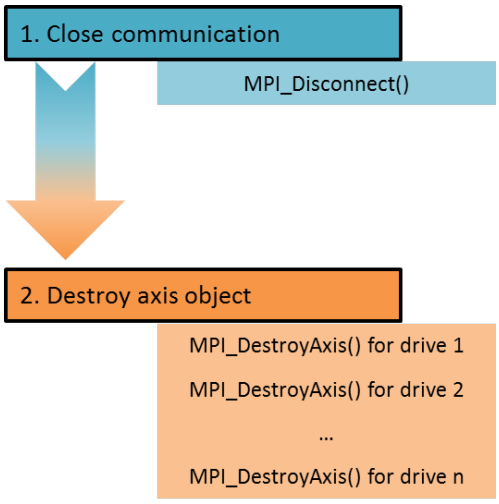


Figure 3.2.1

3.3 MPI_GetSlaveAmount

Purpose

Get the amount of the slaves, which are connected via mega-ulink.

Prototype

```
int MPI_GetSlaveAmount(int *pnSlaveAmount);
```

Parameter

*pnSlaveAmount The amount of the slaves.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

3.4 MPI_GetComSuccess

Purpose

Confirm the state of communication.

Prototype

```
void MPI_GetComSuccess(int *pnComState);
```

Parameter

*pnComState	0: Communication fails.
	1: Communication succeeds.

Return

N/A

Remarks

This function is also available for HIOM.

3.5 MPI_SetCommunicationPar

Purpose

Set mega-ulink communication parameters.

Prototype

```
void MPI_SetComunicationPar(int nTimeOut=200, int nLockTime=200, int nIterNum=6);
```

Parameter

nTimeOut	Time to wait for the servo drive to send back messages. The default is 200. (Unit: msec)
nLockTime	Time to allow the communication error to exist. The default is 200. (Unit: msec)
nIterNum	Maximum number of times to repeatedly send messages to servo drive. The default is 6.

Return

N/A

Remarks

Call this function before calling function `MPI_ConnectMegaulink`.

3.6 MPI_Reconnect

Purpose

Disconnect and restart mega-ulink communication.

Prototype

```
int MPI_Reconnect(CAxis *pAxis=nullptr, unsigned long nWaitTime=100);
```

Parameter

*pAxis	Pointer of CAxis class object. When mega-ulink communication is reconnected, this parameter is null pointer.
nWaitTime	Time to wait for the connection to be rebuilt after it is disconnected. The default is 100. (Unit: msec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

Do not call this function before calling function `MPI_ConnectMegaulink`, `MPI_ConnectUsbById` or `MPI_ConnectUsbByInfo`.

3.7 MPI_GetNetworkAdapterInfo

Purpose

Get struct TNetworkAdapterInfo (which stores the information of all network interface cards in PC).

Prototype

```
int MPI_GetNetworkAdapterInfo(TNetworkAdapterInfo *ptNetworkAdapterInfo);
```

Parameter

*ptNetworkAdapterInfo Pointer of TNetworkAdapterInfo.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

3.7.1 TNetworkAdapterInfo

Defined at **Axis.h**, struct TNetworkAdapterInfo gets the information of all network interface cards in PC and stores it with an array. The members are as follows.

`TNetworkAdapter *ptNetworkAdapter`	Array which stores the information of all network interface cards in PC.
`int iCount`	Array size of storing the information of all network interface cards in PC.
`char szNetworkAdapterErrorMessage`	Error message returned during the process of getting the information of all network interface cards in PC; its string length is 256 characters.

3.7.2 TNetworkAdapter

Defined at **Axis.h**, struct TNetworkAdapter stores the name and the description of one of the network interface cards in PC. The members are as follows.

`char szNetworkAdapterName`	Name of one of the network interface cards in PC; its string length is 250 characters.
`char szNetworkAdapterDescription`	Description of one of the network interface cards in PC; its string length is 250 characters.

3.8 MPI_ClearNetworkAdapterInfo

Purpose

Clear struct TNetworkAdapterInfo.

Prototype

```
int MPI_ClearNetworkAdapterInfo(TNetworkAdapterInfo *ptNetworkAdapterInfo);
```

Parameter

*ptNetworkAdapterInfo Pointer of TNetworkAdapterInfo.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

Calling this function equals completing the following three things:

(1) Delete pointer array `TNetworkAdapter *ptNetworkAdapter`.

(2) Set `int iCount` as 0.

(3) Set `char szNetworkAdapterErrorMessage` as empty string.

3.9 MPI_ConnectMegaulinkByNetworkAdapter

Purpose

Start mega-ulink communication by assigning one of the network interface cards in PC.

Prototype

```
int MPI_ConnectMegaulinkByNetworkAdapter(TNetworkAdapter *ptNetworkAdapter);
```

Parameter

*ptNetworkAdapter Pointer of TNetworkAdapter.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) It is the same as function MPI_ConnectMegaulink.
The difference is that users do not need to manually select network device in this function.

(2) This function must be used with function MPI_GetNetworkAdapterInfo.

(3) Refer to section 3.7.2 for the details of struct TNetworkAdapter.

(4) This function does not support E2 series servo drives.

Example

```
//  Declare the variable of TNetworkAdapterInfo
TNetworkAdapterInfo tNetworkAdapterInfo = {0};
//  Get the information of all network interface cards in PC
MPI_GetNetworkAdapterInfo(&tNetworkAdapterInfo);
//  Assign the first network interface card in PC to start mega-ulink communication
MPI_ConnectMegaulinkByNetworkAdapter(&(tNetworkAdapterInfo.ptNetworkAdapter[0]));
//  Clear the content of tNetworkAdapterInfo
MPI_ClearNetworkAdapterInfo(&tNetworkAdapterInfo);
```

3.10 MPI_ConnectUsbById

Purpose

Start USB communication by USB's ID.

Prototype

```
int MPI_ConnectUsbById(CAxis *pAxis, bool blNetWorkSetting, int nId);
```

Parameter

*pAxis	Pointer of CAxis class object.
blNetWorkSetting	true: Show the configuration window of device communication setting. When multiple connections are executed, it only shows the setting of the last connection. false: Do not show the configuration window of device communication setting.
nId	Connection ID for USB device. One USB device has one Port ID, starting from 0 and the same as the sequence of USB Port in Thunder connection window.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is not available for HIOM.

(2) Figure 3.10.1 shows the procedure of starting USB communication by USB's ID.
It accepts up to 5 connections.

(3) To know more about device communication setting, refer to section 1.4.

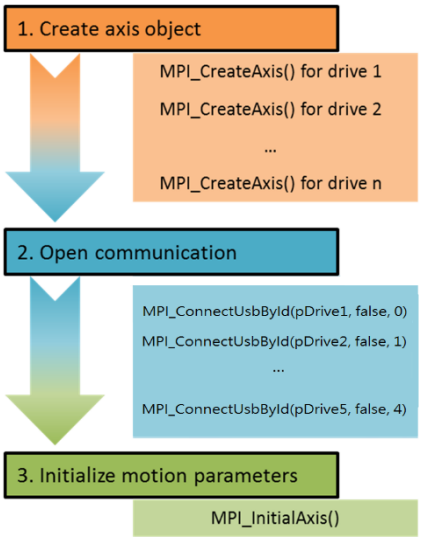


Figure 3.10.1

3.11 MPI_ConnectUsbByInfo

Purpose

Start USB communication by USB's information.

Prototype

```
int MPI_ConnectUsbByInfo(CAxis *pAxis, const char* pszUsbInfo);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszUsbInfo	Connection information for USB device. One USB port has the fixed connection information, which can be got from function `MPI_GetUsbDevice`.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is not available for HIOM.

(2) Figure 3.11.1 shows the procedure of starting USB communication by USB's information. It accepts up to 5 connections.

(3) To know more about device communication setting, refer to section 1.4.

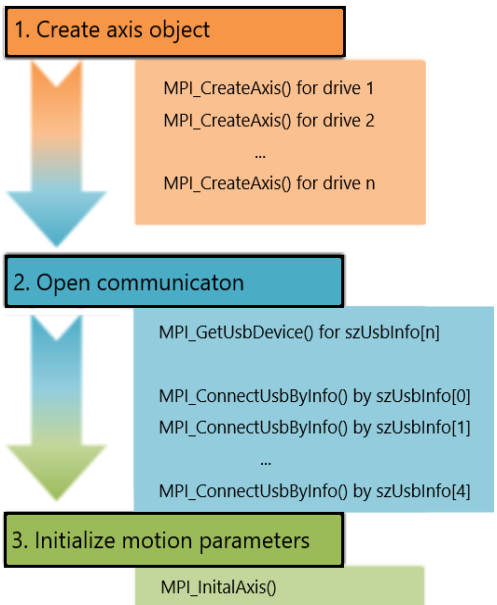


Figure 3.11.1

3.12 MPI_GetUsbDevice

Purpose

Get USB's connection information.

Prototype

```
int MPI_GetUsbDevice(char* szDeviceList[], const int nSize=MAX_USB_DEV_AMOUNT,
                     cont int nStringLength=MAX_USB_DEV_STR,
                     int* pnCount=nullptr);
```

Parameter

*szDeviceList[]	List of USB devices.
nSize	Number of USB devices to be got.
nStringLength	String length of the names of USB devices.
*pnCount	USB devices actually got.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is not available for HIOM.

(2) It can list up to 10 connected USB devices.

(3) To know more about device communication setting, refer to section 1.4.

(This page is intentionally left blank.)

4. Access variables

4. Access variables ····· 4-1
- 4.1 MPI_GetSlaveID ····· 4-2
- 4.2 MPI_GetAxisName ····· 4-3
- 4.3 MPI_GetFeedbackPos ····· 4-4
- 4.4 MPI_GetFeedbackVel ····· 4-5
- 4.5 MPI_GetVel ····· 4-6
- 4.6 MPI_SetVel ····· 4-7
- 4.7 MPI_GetAcc ····· 4-8
- 4.8 MPI_SetAcc ····· 4-9
- 4.9 MPI_GetDec ····· 4-10
- 4.10 MPI_SetDec ····· 4-11
- 4.11 MPI_GetDecKill ····· 4-12
- 4.12 MPI_SetDecKill ····· 4-13
- 4.13 MPI_GetSmoothTime ····· 4-14
- 4.14 MPI_SetSmoothTime ····· 4-15
- 4.15 MPI_GetInPosWidth ····· 4-16
- 4.16 MPI_SetInPosWidth ····· 4-17
- 4.17 MPI_GetSwLimitPos ····· 4-18
- 4.18 MPI_SetSwLimitPos ····· 4-19

4.1 MPI_GetSlaveID

Purpose

Get slave ID.

Prototype

```
int MPI_GetSlaveID(CAxis *pAxis, int *pnSlaveID);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pnSlaveID	Slave ID.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is also available for HIOM.

(2) To display slave ID on human-machine interface, call this function.

4.2 MPI_GetAxisName

Purpose

Get axis name.

Prototype

```
int MPI_GetAxisName(CAxis *pAxis, char *pszAxisName);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszAxisName	Axis name.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is also available for HIOM.

(2) To display axis name on human-machine interface, call this function.

4.3 MPI_GetFeedbackPos

Purpose

Get feedback position.

Prototype

```
int MPI_GetFeedbackPos(CAxis *pAxis, double *pdPos);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdPos	Feedback position. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.4 MPI_GetFeedbackVel

Purpose

Get feedback velocity.

Prototype

```
int MPI_GetFeedbackVel(CAxis *pAxis, double *pdVel);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdVel	Feedback velocity. (Unit: mm/sec or deg/sec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.5 MPI_GetVel

Purpose

Get the maximum velocity from the servo drive.

Prototype

```
int MPI_GetVel(CAxis *pAxis, double *pdVel);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdVel	Maximum velocity. (Unit: mm/sec or deg/sec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.6 MPI_SetVel

Purpose

Set the maximum velocity to the servo drive.

Prototype

```
int MPI_SetVel(CAxis *pAxis, double dVel);
```

Parameter

*pAxis	Pointer of CAxis class object.
dVel	Maximum velocity to be set. (Unit: mm/sec or deg/sec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set maximum velocity will immediately become effective.

4.7 MPI_GetAcc

Purpose

Get acceleration from the servo drive.

Prototype

```
int MPI_GetAcc(CAxis *pAxis, double *pdAcc);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdAcc	Acceleration. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.8 MPI_SetAcc

Purpose

Set acceleration to the servo drive.

Prototype

```
int MPI_SetAcc(CAxis *pAxis, double dAcc);
```

Parameter

*pAxis	Pointer of CAxis class object.
dAcc	Acceleration to be set. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set acceleration will immediately become effective.

4.9 MPI_GetDec

Purpose

Get deceleration from the servo drive.

Prototype

```
int MPI_GetDec(CAxis *pAxis, double *pdDec);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdDec	Deceleration. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.10 MPI_SetDec

Purpose

Set deceleration to the servo drive.

Prototype

```
int MPI_SetDec(CAxis *pAxis, double dDec);
```

Parameter

*pAxis	Pointer of CAxis class object.
dDec	Deceleration to be set. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set deceleration will immediately become effective.

4.11 MPI_GetDecKill

Purpose

Get emergency stop deceleration from the servo drive.

Prototype

```
int MPI_GetDecKill(CAxis *pAxis, double *pdDecKill);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdDecKill	Emergency stop deceleration. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.12 MPI_SetDecKill

Purpose

Set emergency stop deceleration to the servo drive.

Prototype

```
int MPI_SetDecKill(CAxis *pAxis, double dDecKill);
```

Parameter

*pAxis	Pointer of CAxis class object.
dDecKill	Emergency stop deceleration to be set. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set emergency stop deceleration will immediately become effective.

4.13 MPI_GetSmoothTime

Purpose

Get smooth time from the servo drive.

Prototype

```
int MPI_GetSmoothTime(CAxis *pAxis, int *pnSmoothTime);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pnSmoothTime	Smooth time. (Unit: msec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.14 MPI_SetSmoothTime

Purpose

Set smooth time to the servo drive.

Prototype

```
int MPI_SetSmoothTime(CAxis *pAxis, int nSmoothTime);
```

Parameter

*pAxis	Pointer of CAxis class object.
nSmoothTime	Smooth time to be set. (Unit: msec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set smooth time will immediately become effective.

4.15 MPI_GetInPosWidth

Purpose

Get in-position width from the servo drive.

Prototype

```
int MPI_GetInPosWidth(CAxis *pAxis, double *pdInPosWidth);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdInPosWidth	In-position width. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

4.16 MPI_SetInPosWidth

Purpose

Set in-position width to the servo drive.

Prototype

```
int MPI_SetInPosWidth(CAxis *pAxis, double dInPosWidth);
```

Parameter

*pAxis	Pointer of CAxis class object.
dInPosWidth	In-position width to be set. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

The set in-position width will immediately become effective.

4.17 MPI_GetSwLimitPos

Purpose

Get the positive/negative position of the software limit protection from the servo drive.

Prototype

```
int MPI_GetSwLimitPos(CAxis *pAxis, double *pdPosLimitPos, double *pdNegLimitPos);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pdPosLimitPos	Positive software limit position. (Unit: mm or deg)
*pdNegLimitPos	Negative software limit position. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

E1 series servo drives do not support this function.

4.18 MPI_SetSwLimitPos

Purpose

Set the positive/negative position of the software limit protection to the servo drive.

Prototype

```
int MPI_SetSwLimitPos(CAxis *pAxis, double dPosLimitPos, double dNegLimitPos);
```

Parameter

*pAxis	Pointer of CAxis class object.
dPosLimitPos	Positive software limit position to be set. (Unit: mm or deg)
dNegLimitPos	Negative software limit position to be set. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) *dPosLimitPos* must be larger than *dNegLimitPos*.

(2) The set positive/negative position of the software limit protection will immediately become effective.

(3) To enable software limit protection, call function `MPI_EnableSwLimit`. To disable software limit protection, call function MPI_DisableSwLimit.

(4) E1 series servo drives do not support this function.

(This page is intentionally left blank.)

5. Axis motion

5. Axis motion ····· 5-1
- 5.1 MPI_ServoOn ····· 5-2
- 5.2 MPI_ServoOff ····· 5-3
- 5.3 MPI_MoveAbsolute ····· 5-4
- 5.4 MPI_MoveRelative ····· 5-5
- 5.5 MPI_JogPositive ····· 5-6
- 5.6 MPI_JogNegative ····· 5-7
- 5.7 MPI_StopMotion ····· 5-8
- 5.8 MPI_StartHome ····· 5-9
- 5.9 MPI_SetZero ····· 5-10
- 5.10 MPI_ResetDrive ····· 5-11
- 5.11 MPI_ClearError ····· 5-12
- 5.12 MPI_EnableSwLimit ····· 5-13
- 5.13 MPI_DisableSwLimit ····· 5-14

5.1 MPI_ServoOn

Purpose

Enable servo drive.

Prototype

```
int MPI_ServoOn(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) Check Servo On\Off signal by function `MPI_IsReady`.

(2) When D series servo drive is used, one of the Input signals must be set as “Axis Enable”, or this function will fail.

5.2 MPI_ServoOff

Purpose

Disable servo drive.

Prototype

```
int MPI_ServoOff(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

Check Servo On\Off signal by function `MPI_IsReady`.

5.3 MPI_MoveAbsolute

Purpose

Drive the motor to the absolute target position under the specific motion condition.

Prototype

```
int MPI_MoveAbsolute(CAxis *pAxis, double dPos, double dVel=0, double dAcc=0,
                     double dDec=0);
```

Parameter

*pAxis	Pointer of CAxis class object.
dPos	Target position, which is relative to the zero point of the motor. (Unit: mm or deg)
dVel	Motion velocity. (Unit: mm/sec or deg/sec)
dAcc	Motion acceleration. (Unit: mm/sec^2 or deg/sec^2)
dDec	Motion deceleration. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) If parameters *dVel*, *dAcc* and *dDec* are set as 0 (default value), the motor will be driven under the last set motion condition. MPI_InitialAxis, `MPI_SetVel`, `MPI_SetAcc`, `MPI_SetDec`, `MPI_MoveAbsolute` and `MPI_MoveRelative` are the functions that can set motion condition.

(2) This function should be called after the zero point is successfully defined by function `MPI_StartHome`.

5.4 MPI_MoveRelative

Purpose

Drive the motor to the relative target position under the specific motion condition.

Prototype

```
int MPI_MoveRelative(CAxis *pAxis, double dPos, double dVel=0, double dAcc=0,
                     double dDec=0);
```

Parameter

*pAxis	Pointer of CAxis class object.
dPos	Target position, which is relative to the current encoder position of the motor. (Unit: mm or deg)
dVel	Motion velocity. (Unit: mm/sec or deg/sec)
dAcc	Motion acceleration. (Unit: mm/sec^2 or deg/sec^2)
dDec	Motion deceleration. (Unit: mm/sec^2 or deg/sec^2)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

If parameters *dVel*, *dAcc* and *dDec* are set as 0 (default value), the motor will be driven under the last set motion condition. MPI_InitialAxis, `MPI_SetVel`, `MPI_SetAcc`, `MPI_SetDec`, `MPI_MoveAbsolute` and `MPI_MoveRelative` are the functions that can set motion condition.

5.5 MPI_JogPositive

Purpose

Drive the motor with the specific motion velocity in the positive direction.

Prototype

```
int MPI_JogPositive(CAxis *pAxis, double dVel);
```

Parameter

*pAxis	Pointer of CAxis class object.
dVel	Motion velocity. (Unit: mm/sec or deg/sec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) To drive the motor in the negative direction, call function MPI_JogNegative.

(2) To stop the motor, call function `MPI_StopMotion` or set velocity to 0.

5.6 MPI_JogNegative

Purpose

Drive the motor with the specific motion velocity in the negative direction.

Prototype

```
int MPI_JogNegative(CAxis *pAxis, double dVel);
```

Parameter

*pAxis	Pointer of CAxis class object.
dVel	Motion velocity. (Unit: mm/sec or deg/sec)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) To drive the motor in the positive direction, call function MPI_JogPositive.

(2) To stop the motor, call function `MPI_StopMotion` or set velocity to 0.

5.7 MPI_StopMotion

Purpose

Stop the motor with emergency stop deceleration defined by function MPI_InitialAxis or `MPI_SetDecKill`.

Prototype

```
int MPI_StopMotion(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

To get the current emergency stop deceleration, call function `MPI_GetDecKill`.

5.8 MPI_StartHome

Purpose

Start homing procedure.

Prototype

```
int MPI_StartHome(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

All the parameters used in homing procedure should be firstly set in Lightening or Thunder.

5.9 MPI_SetZero

Purpose

Set the current position of the motor as zero point.

Prototype

```
int MPI_SetZero(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

Generally, zero point is defined by built-in homing procedure. If users decide to find zero point by external homing procedure, this function can be called to set zero point to servo drive.

5.10 MPI_ResetDrive

Purpose

Reset servo drive.

Prototype

```
int MPI_ResetDrive(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

5.11 MPI_ClearError

Purpose

Clear servo drive's errors / alarms.

Prototype

```
int MPI_ClearError(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) When D series servo drive is used, one of the Input signals must be set as "Clear Error", or this function will fail.

(2) This function can only clear the specific errors / alarms. To check whether the specific alarm can be cleared when using E1 series servo drive, please refer to column "Alarm Reset" of table 13.2.1.1 in "E1 Series Servo Drive User Manual".

5.12 MPI_EnableSwLimit

Purpose

Enable software limit protection.

Prototype

```
int MPI_EnableSwLimit(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) To disable software limit protection, call function MPI_DisableSwLimit.

(2) To set the positive/negative position of software limit protection, call function `MPI_SetSwLimitPos`.

(3) E1 series servo drives do not support software limit function.

5.13 MPI_DisableSwLimit

Purpose

Disable software limit protection.

Prototype

```
int MPI_DisableSwLimit(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) To enable software limit protection, call function `MPI_EnableSwLimit`.

(2) To set the positive/negative position of software limit protection, call function `MPI_SetSwLimitPos`.

(3) E1 series servo drives do not support software limit function.

6. Access states

6. Access states ····· 6-1
- 6.1 MPI_IsDriveReady ····· 6-2
- 6.2 MPI_IsReady ····· 6-3
- 6.3 MPI_IsHomed ····· 6-4
- 6.4 MPI_IsMoving ····· 6-5
- 6.5 MPI_IsStopped ····· 6-6
- 6.6 MPI_IsError ····· 6-7
- 6.7 MPI_IsSwLimitEn ····· 6-8
- 6.8 MPI_IsHwLimit ····· 6-9
- 6.9 MPI_IsSwLimit ····· 6-10
- 6.10 MPI_WaitServoReady ····· 6-11
- 6.11 MPI_WaitHomeOver ····· 6-12
- 6.12 MPI_WaitMoveOver ····· 6-13
- 6.13 MPI_SetCallBackStates ····· 6-14

6.1 MPI_IsDriveReady

Purpose

Check if E1 series servo drives are ready.

Prototype

```
int MPI_IsDriveReady(CAxis *pAxis, bool *pblReady);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblReady	true: Servo drive is ready.
	false: Servo drive is not ready.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

D series servo drives do not support this function.

6.2 MPI_IsReady

Purpose

Check if the servo drive is servo ready.

Prototype

```
int MPI_IsReady(CAxis *pAxis, bool *pblReady);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblReady	true: The servo drive is servo ready. false: The servo drive is servo off.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.3 MPI_IsHomed

Purpose

Check if the servo drive is homed.

Prototype

```
int MPI_IsHomed(CAxis *pAxis, bool *pblHomed);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblHomed	true: The servo drive is homed.
	false: The servo drive is not homed yet.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.4 MPI_IsMoving

Purpose

Check if the motor is moving.

Prototype

```
int MPI_IsMoving(CAxis *pAxis, bool *pblMoving);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblMoving	true: The motor is moving. false: The motor is stopped.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.5 MPI_IsStopped

Purpose

Check if the motor is stopped.

Prototype

```
int MPI_IsStopped(CAxis *pAxis, bool *pblStop);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblStop	true: The motor is stopped. false: The motor is moving.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.6 MPI_IsError

Purpose

Check if there is an error / alarm in the servo drive.

Prototype

```
int MPI_IsError(CAxis *pAxis, bool *pblError);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblError	true: There is an error / alarm in the servo drive.
	false: There is no error / alarm in the servo drive.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.7 MPI_IsSwLimitEn

Purpose

Check if the software limit protection is enabled.

Prototype

```
int MPI_IsSwLimitEn(CAxis *pAxis, bool *pblSwLimitEn);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblSwLimitEn	true: The software limit protection is enabled. false: The software limit protection is disabled.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

E1 series servo drives do not support this function.

6.8 MPI_IsHwLimit

Purpose

Check if the hardware limit sensor is triggered.

Prototype

```
int MPI_IsHwLimit(CAxis *pAxis, bool *pblLeftLimit, bool *pblRightLimit);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblLeftLimit	true: The hardware left limit sensor is triggered.
	false: The hardware left limit sensor is not triggered.
*pblRightLimit	true: The hardware right limit sensor is triggered.
	false: The hardware right limit sensor is not triggered.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.9 MPI_IsSwLimit

Purpose

Check if the software limit protection is triggered.

Prototype

```
int MPI_IsSwLimit(CAxis *pAxis, bool *pblLeftLimit, bool *pblRightLimit);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pblLeftLimit	true: The software left limit protection is triggered.
	false: The software left limit protection is not triggered.
*pblRightLimit	true: The software right limit protection is triggered.
	false: The software right limit protection is not triggered.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

E1 series servo drives do not support this function.

6.10 MPI_WaitServoReady

Purpose

Wait for the servo drive to become servo ready.

Prototype

```
int MPI_WaitServoReady(CAxis *pAxis, int nTimeOut);
```

Parameter

*pAxis	Pointer of CAxis class object.
nTimeOut	Timeout value. (Unit: ms; Minimum: 10) If users set it less than 10, it will be automatically set as 10.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.11 MPI_WaitHomeOver

Purpose

Wait for the servo drive to complete homing procedure.

Prototype

```
int MPI_WaitHomeOver(CAxis *pAxis, int nTimeOut);
```

Parameter

*pAxis	Pointer of CAxis class object.
nTimeOut	Timeout value. (Unit: ms; Minimum: 10) If users set it less than 10, it will be automatically set as 10.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

6.12 MPI_WaitMoveOver

Purpose

Wait for the motor to complete moving.

Prototype

```
int MPI_WaitMoveOver(CAxis *pAxis, int nTimeOut);
```

Parameter

*pAxis	Pointer of CAxis class object.
nTimeOut	Timeout value. (Unit: ms; Minimum: 10) If users set it less than 10, it will be automatically set as 10.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

When E1 series servo drive is used, the condition for the motor to complete moving depends on the setting of servo drive's COIN signal. Refer to section 8.4.4 **Positioning completion output (COIN) signal** in "E1 Series Servo Drive User Manual" for details.

6.13 MPI_SetCallBackStates

Purpose

When the state changes, users will be proactively notified and the corresponding process will be performed.

Prototype

```
int MPI_SetCallBackStates(CAxis *pAxis=nullptr, long lFuncAdd);
```

Parameter

*pAxis	Pointer of CAxis class object. When mega-ulink communication is connected, this parameter is null pointer.
lFuncAdd	Function pointer; its prototype is `void (*func)(BYTE, int)`.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

type and *id* are state attributes defined for each state in file <appl name.stt>. When a state is changed, function `PutEventSt` should be called, and at least one of the first two bits in *type* is 1 (which defines the direction of change for sending event). Refer to “PDL Reference Manual for D-series Drives” for details.

Example

```
void PutEventSt(BYTE type, int id)
{
    char str[200];
    sprintf(str, "state change, type=%02x, id=%ld", type, id);
    MessageBox(str);
}
……
MPI_SetCallBackStates(PutEventSt);
//  Set DLL with pointer of callback function
```

7. General input/output

7. General input/output ····· 7-1
7.1 MPI_GetInputStatus ····· 7-2
7.2 MPI_GetOutputStatus ····· 7-3
7.3 MPI_SetOutput ····· 7-4
7.4 MPI_ClearOutput ····· 7-5
7.5 MPI_ToggleOutput ····· 7-6

7.1 MPI_GetInputStatus

Purpose

Get the specific input status.

Prototype

```
int MPI_GetInputStatus(CAxis *pAxis, int nInputIndex, bool *pblStatus);
```

Parameter

*pAxis	Pointer of CAxis class object.
nInputIndex	Index of the specific input.
*pblStatus	Input status. true is on; false is off.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

7.2 MPI_GetOutputStatus

Purpose

Get the specific output status.

Prototype

```
int MPI_GetOutputStatus(CAxis *pAxis, int nOutputIndex, bool *pblStatus);
```

Parameter

*pAxis	Pointer of CAxis class object.
nOutputIndex	Index of the specific output.
*pblStatus	Output status. true is on; false is off.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

7.3 MPI_SetOutput

Purpose

Set on the specific output.

Prototype

```
int MPI_SetOutput(CAxis *pAxis, int nOutputIndex);
```

Parameter

*pAxis	Pointer of CAxis class object.
nOutputIndex	Index of the specific output.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

7.4 MPI_ClearOutput

Purpose

Set off the specific output.

Prototype

```
int MPI_ClearOutput(CAxis *pAxis, int nOutputIndex);
```

Parameter

*pAxis	Pointer of CAxis class object.
nOutputIndex	Index of the specific output.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

7.5 MPI_ToggleOutput

Purpose

Toggle the specific output status.

Prototype

```
int MPI_ToggleOutput(CAxis *pAxis, int nOutputIndex);
```

Parameter

*pAxis	Pointer of CAxis class object.
nOutputIndex	Index of the specific output.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

This function is also available for HIOM.

8. Specific input/output

8. Specific input/output ····· 8-1
8.1 MPI_SetTriggerPosition ····· 8-2
8.2 MPI_SetTriggerInterval ····· 8-3
8.3 MPI_EnableTrigger ····· 8-4
8.4 MPI_DisableTrigger ····· 8-5
8.5 MPI_SetTriggerPositionArray ····· 8-6
8.6 MPI_SetTriggerPositionStateArray ····· 8-7
8.7 MPI_SetTriggerPositionStartIndex ····· 8-8
8.8 MPI_SetTriggerPositionEndIndex ····· 8-9

8.1 MPI_SetTriggerPosition

Purpose

Set the start and end position of fixed interval position trigger function.

Prototype

```
int MPI_SetTriggerPosition(CAxis *pAxis, double dStartPos, double dEndPos,
                           double dInterval=0);
```

Parameter

*pAxis	Pointer of CAxis class object.
dStartPos	Start position of fixed interval position trigger function. (Unit: mm or deg)
dEndPos	End position of fixed interval position trigger function. (Unit: mm or deg)
dInterval	Interval distance. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives and D1-N servo drive.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

8.2 MPI_SetTriggerInterval

Purpose

Set the interval distance of fixed interval position trigger function.

Prototype

```
int MPI_SetTriggerInterval(CAxis *pAxis, double dInterval);
```

Parameter

*pAxis	Pointer of CAxis class object.
dInterval	Interval distance. (Unit: mm or deg)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives and D1-N servo drive.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

8.3 MPI_EnableTrigger

Purpose

Enable position trigger function.

Prototype

```
int MPI_EnableTrigger(CAxis *pAxis, ENUM_POLARITY_TYPE enumPolarity=ePT_LOW,
                      int nPulseWidth=0);
```

Parameter

*pAxis	Pointer of CAxis class object.
enumPolarity	Output polarity. ePT_LOW: Normal low ePT_HIGH: Normal high Do not support E1 series servo drives.
nPulseWidth	Pulse width. (Unit: 25 ns; Range: 1~4095) If users set it larger than 4095, it will be automatically set as 4095. Do not support E1 series servo drives.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives and D1-N servo drive.

(2) Before calling this function to enable position trigger function in E1 series servo drives, please set output polarity and pulse width in Thunder first. Refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

(3) This function should be called after the zero point is successfully defined by function `MPI_StartHome`.

8.4 MPI_DisableTrigger

Purpose

Disable position trigger function.

Prototype

```
int MPI_DisableTrigger(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives and D1-N servo drive.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

(3) If the motor position exceeds *dEndPos* set by function `MPI_SetTriggerPosition`, the position trigger function will be automatically disabled.

8.5 MPI_SetTriggerPositionArray

Purpose

Set the position array of random interval position trigger function.

Prototype

```
int MPI_SetTriggerPositionArray(CAxis *pAxis, int iIndex, double dPosition);
```

Parameter

*pAxis	Pointer of CAxis class object.
iIndex	Index.
dPosition	Position data.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

8.6 MPI_SetTriggerPositionStateArray

Purpose

Set the status array of random interval position trigger function (status output).

Prototype

```
int MPI_SetTriggerPositionStateArray(CAxis *pAxis, int iIndex, int iState);
```

Parameter

*pAxis	Pointer of CAxis class object.
iIndex	Index.
iState	Status data.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

(3) For the details of status data, please refer to table 8.13.13 in “E1 Series Servo Drive User Manual”.

8.7 MPI_SetTriggerPositionStartIndex

Purpose

Set the start index of random interval position trigger function.

Prototype

```
int MPI_SetTriggerPositionStartIndex(CAxis *pAxis, int iIndex);
```

Parameter

*pAxis	Pointer of CAxis class object.
iIndex	Start index. Random interval position trigger function starts at this position data.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

8.8 MPI_SetTriggerPositionEndIndex

Purpose

Set the end index of random interval position trigger function.

Prototype

```
int MPI_SetTriggerPositionEndIndex(CAxis *pAxis, int iIndex);
```

Parameter

*pAxis	Pointer of CAxis class object.
iIndex	End index. Random interval position trigger function ends at this position data.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives.

(2) For detailed information of position trigger function of E1 series servo drives, please refer to section 8.13 **Setting position trigger function** in “E1 Series Servo Drive User Manual”.

(This page is intentionally left blank.)

9. Access specific variable/state

9. Access specific variable/state ····· 9-1
- 9.1 MPI_GetVar ····· 9-2
- 9.2 MPI_SetVar ····· 9-3
- 9.3 MPI_GetState ····· 9-4
- 9.4 MPI_SetState ····· 9-5
- 9.5 MPI_GetErrorData ····· 9-6
- 9.6 MPI_GetErrorDataArray ····· 9-7
- 9.7 MPI_RunFuncPDL ····· 9-8
- 9.8 MPI_IsTaskRunning ····· 9-9
- 9.9 MPI_KillTask ····· 9-10
- 9.10 MPI_SendRAMtoFlash ····· 9-11
- 9.11 MPI_GetVarArray ····· 9-12
- 9.12 MPI_SetVarArray ····· 9-13
- 9.13 MPI_GetErrorMessage ····· 9-14

9.1 MPI_GetVar

Purpose

Get the specific variable from the servo drive.

Prototype

```
int MPI_GetVar(CAxis *pAxis, char *pszVarName, double *pdValue);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszVarName	Variable name.
*pdValue	Variable value.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.2 MPI_SetVar

Purpose

Set the specific variable to the servo drive.

Prototype

```
int MPI_SetVar(CAxis *pAxis, char *pszVarName, double dValue);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszVarName	Variable name.
dValue	Variable value to be set. It will be automatically converted to the correct variable type in the servo drive.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.3 MPI_GetState

Purpose

Get the specific state from the servo drive.

Prototype

```
int MPI_GetState(CAxis *pAxis, char *pszStateName, bool *pblStatus);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszStateName	State name.
*pblStatus	Variable state. true is on; false is off.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.4 MPI_SetState

Purpose

Set the specific state to the servo drive.

Prototype

```
int MPI_SetState(CAxis *pAxis, char *pszStateName, bool blStatus);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszStateName	State name.
blStatus	Variable state to be set. true is for on; false is for off.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.5 MPI_GetErrorData

Purpose

Get the error of D series servo drives.

Prototype

```
int MPI_GetErrorData(CAxis *pAxis, int pnErrorData);
```

Parameter

*pAxis	Pointer of CAxis class object.
pnErrorData	Error number.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for D series servo drives.

(2) Refer to chapter 12 for the name of error.

9.6 MPI_GetErrorDataArray

Purpose

Get the alarm of E1 series servo drives.

Prototype

```
int MPI_GetErrorDataArray(CAxis *pAxis, int pnErrorData[3]);
```

Parameter

*pAxis	Pointer of CAxis class object.
pnErrorData[0]	0~31 bit stand for alarms 1~32 in chapter 11.
pnErrorData[1]	0~31 bit stand for alarms 33~64 in chapter 11.
pnErrorData[2]	0~12 bit stand for alarms 65~77 in chapter 11.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) This function is only available for E1 series servo drives.

(2) Refer to chapter 11 for the name and the description of alarm.

(3) This function can only get the alarms listed in chapter 11. To get other alarms, call function `MPI_GetErrorMessage`.

9.7 MPI_RunFuncPDL

Purpose

Run the specific PDL function from the servo drive.

Prototype

```
int MPI_RunFuncPDL(CAxis *pAxis, char *pszPDLName, int *pnTaskID);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszPDLName	PDL function name. Any label in PDL program which starts with "_" (underscore) will be recognized by the host.
*pnTaskID	PDL's task ID. (Range: 1~3)

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.8 MPI_IsTaskRunning

Purpose

Check if the specific task of servo drive's PDL is running.

Prototype

```
int MPI_IsTaskRunning(CAxis *pAxis, int nTaskID, bool *plRunning);
```

Parameter

*pAxis	Pointer of CAxis class object.
nTaskID	PDL's task ID. (Range: 1~3)
*plRunning	true: The task is running. false: The task is not running.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.9 MPI_KillTask

Purpose

Kill, stop or continue the specific task of servo drive's PDL.

Prototype

```
int MPI_KillTask(CAxis *pAxis, int nTaskID,
                 ENUM_EXECTASK_TYPE enumExecTaskType=eETT_KILL);
```

Parameter

*pAxis	Pointer of CAxis class object.
nTaskID	PDL's task ID. (Range: 1~3)
enumExecTaskType	eETT_KILL: Kill the task.
	eETT_STOP: Stop the task.
	eETT_CONT: Continue the task.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.10 MPI_SendRAMtoFlash

Purpose

Save parameters data to servo drive's memory.

Prototype

```
int MPI_SendRAMtoFlash(CAxis *pAxis);
```

Parameter

*pAxis Pointer of CAxis class object.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.11 MPI_GetVarArray

Purpose

Get the specific array type variable from the servo drive.

Prototype

```
int MPI_GetVarArray(CAxis *pAxis, char *pszVarName, double *pdata, int iSize);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszVarName	Variable name.
*pdata	Variable value.
iSize	Array size.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.12 MPI_SetVarArray

Purpose

Set the specific array type variable to the servo drive.

Prototype

```
int MPI_SetVarArray(CAxis *pAxis, char *pszVarName, double *pdata, int iSize);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszVarName	Variable name.
*pdata	Variable value to be set. It will be automatically converted to the correct variable type in the servo drive.
iSize	Array size.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

9.13 MPI_GetErrorMessage

Purpose

Get the error / alarm of the servo drive.

Prototype

```
int MPI_GetErrorMessage(CAxis *pAxis, char *pszErrorMessage);
```

Parameter

*pAxis	Pointer of CAxis class object.
*pszErrorMessage	Error / alarm; its string length is 256 characters.

Return

0 – Function succeeds.

nonzero value – Function fails. Refer to the description of error code in chapter 10.

Remarks

(1) The alarm of E1 series servo drives is AL□□□. For the cause and corrective action of alarm, please refer to section 13.2.2 in “E1 Series Servo Drive User Manual”.

(2) The error of D series servo drives is the name of error. For the cause and corrective action of error, please refer to the description of error in the related chapter of D Series Servo Drive User Manual.

10. Error code

10. Error code ····· 10-1

List of error codes is shown as follows.

Table 10.1

Code number	Name	Description
0	ERR_NO_ERROR	No error.
1	ERR_NO_OBJECT	Wrong object pointer. Please check the definition.
2	ERR_WRONG_MOTOR_TYPE	Wrong motor type. Please check the definition.
3	ERR_WRONG_UNIT_TYPE	Wrong unit type. Please check the definition.
4	ERR_WRONG_DRIVE_TYPE	Wrong slave type. Please check the definition.
5	ERR_WRONG_ROOT_FOLDER	Wrong software root folder. Please check the software version.
6	ERR_INVALID_ARGUMENT	Invalid argument. Please check the definition.
10	ERR_DISCONNECT	No connection. Please check the wiring.
11	ERR_NO_SLAVE	No slave is connected. Please check the wiring.
12	ERR_DUPLICATE_AXIS_NAME	Two slaves have the same axis name. Please check the drive setting via Lightening or Thunder.
13	ERR_MISMATCH_AXIS_NAME	Fail to match the axis name from the servo drive and the setting. Please check the drive setting via Lightening or Thunder.
14	ERR_NOT_INSTALLED	The software is not installed. Please install the corresponding software based on servo drive type.
15	ERR_CONNECTION_OVERRANGE	The number of connections exceeds the limit.
16	ERR_NO_CONNECTION	The communication of the axis does not exist. Please establish the connection.
20	ERR_GET_VAR_ERROR	Fail to get the variable from the servo drive. Please check the variable name or the firmware version of the servo drive.
21	ERR_SET_VAR_ERROR	Fail to set the variable to the servo drive. Please check the variable name or the firmware version of the servo drive.
22	ERR_GET_STATE_ERROR	Fail to get the state from the servo drive. Please check the state name or the firmware version of the servo drive.
23	ERR_SET_STATE_ERROR	Fail to set the state to the servo drive. Please check the state name or the firmware version of the servo drive.
24	ERR_RUN_PDL_ERROR	Fail to run the PDL function from the servo drive. Please check the PDL function name or the firmware version of the servo drive.

Code number	Name	Description
25	ERR_SET_WRONG_INPUT_FUNC	Fail to call the input function. Please check the input function setting of the servo drive via Lightening or Thunder.
26	ERR_KILL_TASK_ERROR	Fail to kill, stop or continue the task of PDL.
27	ERR_TASK_NUM_OVER_LIMIT	PDL's task ID is out of range. Its maximum value is 3.
28	ERR_TASK_NOT_RUN	The task of PDL is not running. Therefore, it cannot be killed, stopped or continued.
30	ERR_WAIT_TIMEOUT	Timeout. Please check the timeout value or the wiring.
31	ERR_NOT_HOMED_YET	The servo drive is not homed yet. Please check the homed state.
32	ERR_MPD_VERSION_NOT_SUPPORT	The firmware version is too low to support this function. Please check the firmware version of the servo drive. (This error is only available for E1 series servo drives.)
33	ERR_NOT_SUPPORT	Do not support API function, please refer to the previous chapters. (This error is only available for E1 series servo drives.)
34	ERR_RESOLUTION_VALUE_ERROR	Abnormal resolution value. Please check the resolution setting of the servo drive via Lightening or Thunder.
35	ERR_DISPLAY_UNIT_NOT_SET	The display unit is not set. Please set the display unit in Thunder. (This error is only available for E1 series servo drives.)
36	ERR_INVALID_STATE	Invalid state of the servo drive. Please check the state name or the firmware version of the servo drive.
37	ERR_GET_ERROR_MESSAGE_ERROR	Fail to get the alarm of the servo drive.
38	ERR_NETWORK_ADAPTER_ERROR	Fail to get the network interface card.
100	ERR_CANT_SAVE_WHEN_ENABLED	When servo drive is enabled, parameters data cannot be saved to servo drive's memory.
101	ERR_SAVE_TO_FLASH_ERROR	Fail to save parameters data to servo drive's memory.

(This page is intentionally left blank.)

11. E1 series servo drive alarm

11. E1 series servo drive alarm ····· 11-1

List of alarms is shown as follows.

Table 11.1

Alarm Number	Name	Description
1	-	Reserved
2	-	Reserved
3	-	Reserved
4	AL024	System alarm 1
5	AL025	System alarm 2
6	AL030	Main circuit malfunction
7	AL040	Parameter setting error
8	AL050	Combination error
9	-	Reserved
10	-	Reserved
11	-	Reserved
12	AL070	Motor change detected
13	-	Reserved
14	AL0b0	Invalid servo on command
15	AL100	Overcurrent detected
16	AL320	Regenerative energy overflow
17	AL400	Overvoltage
18	AL410	Undervoltage
19	AL510	Overspeed
20	AL511	Encoder pulse output overspeed
21	-	Reserved
22	-	Reserved
23	-	Reserved
24	AL710	Overload (instantaneous maximum load)
25	AL720	Overload (continuous maximum load)
26	-	Reserved
27	-	Reserved
28	-	Reserved
29	-	Reserved
30	AL7A2	Power board temperature error
31	-	Reserved
32	-	Reserved
33	AL800	Encoder data backup error
34	AL810	Encoder battery undervoltage

Alarm Number	Name	Description
35	AL820	Encoder communication error
36	AL830	Encoder data error
37	AL840	Encoder communication crc error
38	AL850	Encoder counting error
39	AL860	Encoder data writing error
40	AL870	Encoder overheating
41	AL880	Incremental encoder signal error
42	AL890	Excellent Smart Cube (ESC) - incremental encoder disconnection
43	AL8A0	First set of encoder - Excellent Smart Cube (ESC) signal error
44	AL8b0	First set of encoder - encoder signal error
45	AL8C0	Second set of encoder - Excellent Smart Cube (ESC) signal error
46	AL8d0	Second set of encoder - encoder signal error
47	AL8E0	Digital encoder disconnection
48	AL8F0	Excellent Smart Cube (ESC) internal error
49	AL861	Motor overheating
50	ALb10	Velocity command A/D converter error
51	-	Reserved
52	ALb20	Torque command A/D converter error
53	ALb33	Current detection malfunction
54	ALC10	Motor out of control
55	ALC20	Phase detection error
56	ALC21	Hall sensor error
57	-	Reserved
58	ALC50	Electrical angle detection failure
59	ALC51	Overtravel detected during electrical angle detection
60	ALC52	Electrical angle detection incomplete
61	-	Reserved
62	ALd00	Position deviation overflow
63	-	Reserved
64	-	Reserved
65	ALd10	Motor-load position deviation overflow
66	-	Reserved
67	ALEb1	Safety function signal input timing error
68	ALEb2	Safety function module error
69	ALF10	Power cable open phase
70	ALF50	Motor main circuit cable disconnection

Alarm Number	Name	Description
71	ALFA0	Encoder power error
72	ALFB0	Fieldbus communication hardware malfunction
73	ALFB1	Fieldbus communication error
74	ALFC0	Communication error of gantry control system
75	ALFC1	Slave axis error in gantry control system
76	AL891	Excellent Smart Cube (ESC) - incremental encoder signal error
77	ALFC2	Fieldbus communication setup error

Note:

For the cause and corrective action of alarm, please refer to section 13.2.2 in “E1 Series Servo Drive User Manual”.

12. D series servo drive error

12. D series servo drive error ····· 12-1

List of errors is shown as follows.

Table 12.1

Error Number (displayed in hex)	Name
0x00000001	Motor short (over current) detected
0x00000002	Over voltage detected
0x00000004	Position error too big
0x00000008	Encoder error
0x00000010	Soft-thermal threshold reached
0x00000020	Motor maybe disconnected
0x00000040	Amplifier over temperature
0x00000080	Motor over temperature sensor activated
0x00000100	Under voltage detected
0x00000200	5V for encoder card fail
0x00000400	Phase initialization error
0x00000800	Serial encoder communication error
0x00001000	Hall sensor error
0x00002000	Hall phase check error
0x00004000	Current control error
0x00008000	-
0x00010000	Hybrid deviation too big
0x00020000	STO active
0x00040000	HFLT inconsistent error
0x00080000	Auto phase center not completed error
0x00100000	-
0x00200000	DC bus voltage abnormal
0x00400000	EtherCAT interface is not detected
0x00800000	CiA-402 homing error
0x01000000	Fan fault error
0x02000000	Drive overload error

Note:

For the cause and corrective action of error, please refer to the description of error in the related chapter of D Series Servo Drive User Manual.