NEXCOM

NexRobotEduAPI

Reference Manual

Manual Rev.: V1.3

Revision Date: 2016/03/15

Revise note:

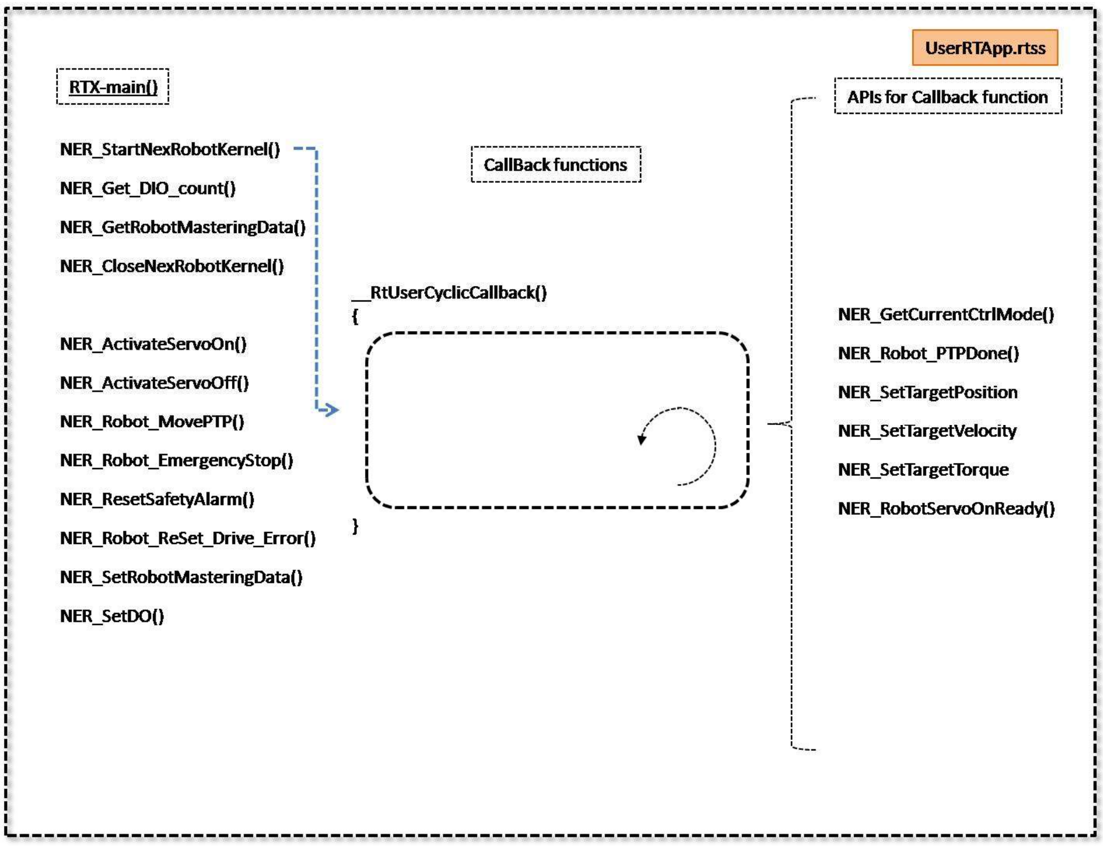
Ver	Description
V1.0	2015/6/15: English version released.
V1.1	1. Add argument command type to NER_Robot_MovePTP().
V1.2	1. NexRobot1.0.0.4 Release 2. Add new APIs (1) NER_Robot_Get_Error_Driver_NO (2) NER_Robot_ChangePTPVel (3) NER_Robot_GetAlarmCode (4) NER_SetTargetTorque (5) NER_Robot_Change_To_CSP_Mode (6) NER_Robot_Change_To_CSV_Mode (7) NER_Robot_Change_To_CST_Mode (8) NER_GetDO (9) NER_Get_AIO_count (10) NER_Get_AIO_Info 3. NER_Robot_MovePTP now support target override.
V1.3	1. Modified NER_StarNexRobotKernel 2. Modified name of NER_Robot_ChangePTPVel to NER_Robot_ChangeSpdRatio 3. Modified name of NER_Robot_PTPDone to NER_Robot_MotDone 4. Add new APIs (1) NER_GetNexRobotKernelVersion (2) NER_ReInitializeDriverParameters (3) NER_Robot_MoveLine 5. NER_Robot_MovePTP support change feedrate override on the fly. 6. NER_Robot_MoveLine support change feedrate override on the fly.

1. NexRobotEdu Library Overview

1.1. Basic Specification

1. NexRobotEdu library support Microsoft® 7 (32 bit) with RTX 2012 update 3.
2. NexRobotEdu library only support the robot which provided by NEXCOM.

1.2. Function Call Flowchart



2. API Reference

2.1. API Overview

All APIs of NexRobotEdu Library are listed. The definition of API is located at the header file “NexRobotKernel.h”.

T : Type (of function call)
C : Callback only
X :not for Callback
B : both

(**T**: Type → C: Callback only, X: Not for callback, B:Both)

Function Name	Description	T
Initialization Functions		
NER_StartNexRobotKernel	Start NexRobotKernel	X
NER_CloseNexRobotKernel	Close NexRobotKernel	X
NER_GetRobotMasteringData	Get robot mastering data	X
NER_SetRobotMasteringData	Set robot mastering data	X
NER_GetNexRobotKernelVersion	Get NexRobotKernel Version	B
NER_ReInitialDriveParameters	Reinitialize driver parameters	X
Robot Servo Basic Operation Functions		
NER_ActivateServoOn	Start to servo on all robot axis	B
NER_ActivateServoOff	Start to servo off all robot axis	B
NER_RobotServoOnReady	Check robot axis servo on state	B
NER_Robot_ReSet_Drive_Error	Reset all robot axis servo error	X
NER_Robot_Get_Error_Driver_NO	Get Error Driver’s index	B
Robot Basic Operation Functions		
NER_Robot_MovePTP	Robot PTP move	B
NER_Robot_ChangeSpdRatio	Change Moving velocity of PTP and LINE	B
NER_Robot_EmergencyStop	Emergency stop the Robot PTP movement	X
NER_Robot_MotDone	Check robot PTP or LINE movement is done or not	B
NER_Robot_GetAlarmCode	Get Alarm Code of NexRobot	B
User Mode Operation Functions		
NER_SetTargetPosition	Set cyclic position command	C
NER_SetTargetVelocity	Set cyclic velocity command	C
NER_SetTargetTorque	Set cyclic Torque command	C
NER_ResetSafetyAlarm	Reset user safety alarm	B

NER_GetCurrentCtrlMode	Get current control mode	B
NER_Robot_Change_To_CSP_Mode	Change operation mode to CSP	B
NER_Robot_Change_To_CSV_Mode	Change operation mode to CSV	B
NER_Robot_Change_To_CST_Mode	Change operation mode to CST	B
Digital I/O Operation Functions		
NER_Get_DIO_count	Get DIO number of current EtherCAT configuration	B
NER_Get_DIO_Info	Get each DIO information	B
NER_SetDO	Set DO	B
NER_GetDI	Get DI data	B
Analog I/O Operation Functions		
NER_Get_AIO_count	Get DIO number of current EtherCAT configuration	B
NER_Get_AIO_Info	Get each DIO information	B

The C/C++ data types for API is defined in “nex_type.h” and listed as follows:

Type	C/C++ Primitive	format	Byte Length	Value Range
BOOL_T	int	Boolean	4	0:False, 1:True
U8_T	unsigned char	Unsigned Integer	1	0 ~ 255
U16_T	unsigned short	Unsigned Integer	2	0 ~ 65535
U32_T	unsigned int	Unsigned Integer	4	0 ~ 4294967295
U64_T	unsigned __int64	Unsigned Integer	8	0 ~ 18446744073709551615
I8_T	char	Signed Integer	1	-128 ~ 127
I16_T	short	Signed Integer	2	-32768 ~ 32767
I32_T	int	Signed Integer	4	-2147483648 ~ 2147483647
I64_T	__int64	Signed Integer	8	-9223372036854775808 ~ 9223372036854775807
F32_T	float	Floating-point number	4	IEEE-754, accurate to the seventh decimal place
F64_T	double	Double-precision floating-point number	8	IEEE-754, accurate to the fifteenth decimal place
RTN_ERR	int	Error code	4	-2147483648 ~ 2147483647

2.2. Functions for Initialization

2.2.1. NER_StartNexRobotKernel

C/C++ Syntax:

BOOLNER_StartNexRobotKernel(NER_RtCyclicCallback__UserCyclicCallback, NER_ROBOT_AXIS_CONTROL_MODE ctrl_mode);

Parameters:

NER_RtCyclicCallback__UserCyclicCallback:

Cyclic callback data pointer.

NER_ROBOT_AXIS_CONTROL_MODEctrl_mode :

The mode for control:

```
typedef enum _NER_ROBOT_AXIS_CONTROL_MODE
{
        NER_CYCLIC_POSITION         = 0x08,
        NER_CYCLIC_VELOCITY         = 0x09,
        NER_CYCLIC_TORQUE           = 0x0A,
}NER_ROBOT_AXIS_CONTROL_MODE;
```

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for initializing and start the NexRobotKernel.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_CloseNexRobotKernel();

2.2.2. NER_CloseNexRobotKernel

C/C++ Syntax:

BOOL NER_CloseNexRobotKernel();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for close the NexRobotKernel.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_CloseNexRobotKernel();

2.2.3. NER_GetRobotMasteringData

C/C++ Syntax:

```
BOOL NER_GetRobotMasteringData(F64_T *MasteringData, NER_ROBOT_TYPE
&type);
```

Parameters:

F64_T *MasteringData:

Robot's Mastering Data.

NER_ROBOT_TYPE &type:

Current robot type:

```
typedef enum _NER_ROBOT_TYPE
{
        NER_6AXIS           = 0x00,
        NER_DELTA           = 0x01,
        NER_SCARA           = 0x02,
} NER_ROBOT_TYPE;
```

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for getting the mastering data of current robot.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_SetRobotMasteringData();

2.2.4. NER_SetRobotMasteringData

C/C++ Syntax:

BOOL NER_SetRobotMasteringData(F64_T *MasteringData, NER_ROBOT_TYPE type);

Parameters:

F64_T *MasteringData :

Robot's Mastering Data.

NER_ROBOT_TYPE type:

Current robot type:

```
typedef enum _NER_ROBOT_TYPE
{
        NER_6AXIS           = 0x00,
        NER_DELTA           = 0x01,
        NER_SCARA           = 0x02,
} NER_ROBOT_TYPE;
```

Attention! Currently only support NER_6AXIS.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for setting the mastering data of current robot.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_GetRobotMasteringData();

2.2.5. NER_GetNexRobotKernelVersion

C/C++ Syntax:

```
U32_T NER_GetNexRobotKernelVersion();
```

Parameters:

<No Parameters>

Returned Values:

Version is returned.

The newest version is 1005.

Usage:

Call the function for getting the version of NexRobot.

2.2.6. NER_ReInitialDriveParameters

C/C++ Syntax:

BOOL NER_ReInitialDriveParameters();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

If robot moving makes some noise of motors, call the function for setting the initial parameters of each driver. But be sure drives is servo off.

Attention! The function is not allowed to be used in Callback function.

Attention! Only support 6R robot currently.

Reference:

NER_ActivateServoOff();

2.3. Functions for Robot Servo Basic Operation

2.3.1. NER_ActivateServoOn

C/C++ Syntax:

```
BOOL NER_ActivateServoOn();
```

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for activating the servo on procedure of all robot axes, it will take some time to do it.

Reference:

NER_ActivateServoOff();

NER_RobotServoOnReady();

2.3.2. NER_ActivateServoOff

C/C++ Syntax:

BOOL NER_ActivateServoOff();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for activating the servo off procedure of all robot axes.

Reference:

NER_ActivateServoOn();

NER_RobotServoOnReady();

2.3.3. NER_RobotServoOnReady

C/C++ Syntax:

BOOL NER_RobotServoOnReady();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if all robot axes are at servo on state, while FALSE is returned when at least one of robot axes is not at servo on state.

Usage:

Call the function for checking all robot axes are at servo on state.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_ActivateServoOn();

NER_ActivateServoOff();

2.3.4. NER_Robot_ReSet_Drive_Error

C/C++ Syntax:

```
BOOL NER_Robot_ReSet_Drive_Error();
```

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

When at least one of robot axes is in “Fault” state, please check the servo failure state and make all fault servos corrected, then use this function to clear/reset the “Fault” state afterward.

Attention! The function is not allowed to be used in Callback function.

2.3.5. NER_Robot_Get_Error_Driver_NO

C/C++ Syntax:

BOOL NER_Robot_Get_Error_Driver_NO(U8_T &AXIS_Number);

Parameters:

U8_T &AXIS_Number:

Get Error Driver's Index.

AXIS_Number:

Bit	7	6	5	4	3	2	1	0
AXIS	X	X	Axis6	Axis5	Axis4	Axis3	Axis2	Axis1

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for gettingwhich driver is in fault status.

Reference:

NER_Robot_GetAlarmCode();

2.4. Functions for Robot Basic Operation

2.4.1. NER_Robot_MovePTP

C/C++ Syntax:

BOOL NER_Robot_MovePTP(U8_T u8CmdType,F64_T *target_data,F64_T *max_vel,F64_T *acc);

Parameters:

U8_T u8CmdType:

Set input parameter format of PTP(1: Axis angle; 2: TCP).

F64_T *target_data:

Set Target data (deg) of each robot axis angle.

F64_T *max_vel:

Set maximum velocity (deg/s) of each robot axis

F64_T *acc:

Set acceleration (deg/s^2) of each robot axis

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for moving all robot axes to desire destination using the fastest speed.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_Robot_EmergencyStop();

NER_Robot_MotDone ();

2.4.2. NER_Robot_MoveLine

C/C++ Syntax:

BOOL NER_Robot_MoveLine(U8_T u8CmdType, F64_T *target_data, F64_T *max_vel, F64_T *acc);

Parameters:

U8_T u8CmdType:

Set input parameter format of line (1: Axis angle; 2: TCP).

F64_T *target_data:

Set Target data (deg) of each robot axis angle, if input parameter format of line is axis angle; Set Target data (mm) of robot TCP, if input parameter format of line is TCP.

F64_T *max_vel:

Set maximum velocity (mm/s) of robot TCP.

F64_T *acc:

Set acceleration (mm/s^2) of each robot TCP

Returned Values:

Error code is returned.

Zero is returned if function call is successful, while nonzero is returned when create line buffer failed.

Usage:

Call the function for moving robot TCP to desire destination using the fastest speed.

Attention! The function is not allowed to be used in Callback function.

Reference:

NER_Robot_EmergencyStop();

NER_Robot_MotDone();

2.4.3. NER_Robot_ChangeSpdRatio

C/C++ Syntax:

BOOL NER_Robot_ChangeSpdRatio(U8_T speed_ratio)

Parameters:

U8_T speed_ratio：

Moving speed ratio of PTP or LINE (0~100%).

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for change moving speed ratio of PTP or LINE.

Reference:

NER_Robot_EmergencyStop();

NER_Robot_MotDone ();

2.4.4. NER_Robot_EmergencyStop

C/C++ Syntax:

BOOL NER_Robot_EmergencyStop();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for stopping the moving robot using the default maximum deceleration.

Reference:

NER_Robot_MovePTP();

NER_Robot_MotDone ();

2.4.5. NER_Robot_MotDone

C/C++ Syntax:

```
BOOL NER_Robot_MotDone ();
```

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if the robot PTP movement is done, while FALSE is returned when the robot PTP movement is not done.

Usage:

Call the function for checking the PTP movement of robot is done or not.

Reference:

NER_Robot_MovePTP();

NER_Robot_EmergencyStop();

2.4.6. NER_Robot_GetAlarmCode

C/C++ Syntax:

I32_TNER_Robot_GetAlarmCode();

Parameters:

<No Parameters>

Returned Values:

Error Code is returned.

The error code’s definition are listed in 「NexRobotErrors.h」.

Usage:

Call the function for getting the alarm code of NexRobot.

Reference:

NER_Robot_Get_Error_Driver_NO();

2.5. Functions for User Mode Operation

2.5.1. NER_SetTargetPosition

C/C++ Syntax:

BOOL NER_SetTargetPosition(U16_T Robot_Axis_ind, F64_T t_Position);

Parameters:

U16_T Robot_Axis_ind :

The index of robot axis (0~5).

F64_T t_Position:

Set Cyclic Target Position(deg) to assigned robot axis.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for cyclic moving assigned robot axis to target position.If one of robot axes will over joint position limit, the function will no longer be useful and all robot axes will automatically stop.

Attention! The function is allowed to be used in Callback function only.

Attention! The function only works in “NER_CYCLIC_POSITION” mode.

Reference:

NER_ResetSafetyAlarm();

2.5.2. NER_SetTargetVelocity

C/C++ Syntax:

BOOL NER_SetTargetVelocity(U16_T Robot_Axis_ind, F64_T t_Velocity);

Parameters:

U16_T Robot_Axis_ind :

The index of robot axis (0~5).

F64_T t_Position:

Set Cyclic Target Velocity(deg/s) to assigned robot axis.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for cyclic moving assigned robot axis to target velocity.If one of robot axes will over joint position limit, the function will no longer be useful and all robot axes will automatically stop.

Attention! The function is allowed to be used in Callback function only.
Attention! The function only works in "NER_CYCLIC_VELOCITY" mode.

Reference:

NER_ResetSafetyAlarm();

2.5.3. NER_SetTargetTorque

C/C++ Syntax:

BOOL NER_SetTargetVelocity(U16_T Robot_Axis_ind,I16_T t_Torque);

Parameters:

U16_T Robot_Axis_ind:

The index of robot axis (0~5).

F64_T t_Torque:

Set Cyclic Target Torque(0.1% of rated torque) to assign robot axis.

Robot ARM rated torque Specification:

Axis	Rated Torque	Unit
1	1.3	N.m
2	1.3	N.m
3	0.64	N.m
4	0.32	N.m
5	0.16	N.m
6	0.16	N.m

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for cyclic moving assigned robot axis to target velocity.If one of robot axes will over joint position limit, the function will no longer be useful and all robot axes will automatically stop.

Attention! The function is allowed to be used in Callback function only.
Attention! The function only works in “NER_CYCLIC_TORQUE” mode.

Reference:

NER_ResetSafetyAlarm();

2.5.4. NER_ResetSafetyAlarm

C/C++ Syntax:

BOOL NER_ResetSafetyAlarm();

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if safety alarm has been reset, while FALSE is returned when safety alarm has not been reset.

Usage:

Call the function for resetting the safety alarm. After the safety alarm has been successful reset, the function “NER_SetTargetPosition” and “NER_SetTargetVelocity” can be used again.

Reference:

NER_SetTargetPosition();

NER_SetTargetVelocity();

2.5.5. NER_GetCurrentCtrlMode

C/C++ Syntax:

```
BOOL NER_GetCurrentCtrlMode(NER_ROBOT_AXIS_CONTROL_MODE &mode);
```

Parameters:

NER_ROBOT_AXIS_CONTROL_MODE &mode :

The current control mode:

```
typedef enum _NER_ROBOT_AXIS_CONTROL_MODE
{
   NER_CYCLIC_POSITION      = 0x08,
   NER_CYCLIC_VELOCITY      = 0x09,
   NER_CYCLIC_TORQUE        = 0x0A,
}NER_ROBOT_AXIS_CONTROL_MODE;
```

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for getting current control mode.

Reference:

NER_Robot_Change_To_CSP_Mode();
NER_Robot_Change_To_CSV_Mode();
NER_Robot_Change_To_CST_Mode();

2.5.6. NER_Robot_Change_To_CSP_Mode

C/C++ Syntax:

BOOL NER_Robot_Change_To_CSP_Mode()

Parameters:

<No Parameters>

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for change Operation mode to CSP mode.

Reference:

NER_GetCurrentCtrlMode();

2.5.7. NER_Robot_Change_To_CSV_Mode

C/C++ Syntax:

BOOL NER_Robot_Change_To_CSV_Mode (F64_T *vel_value)

Parameters:

F64_T *vel_value:

The initial velocity value (deg/s) of each joint while change operation mode to CSV.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for change Operation mode to CSV mode.

Reference:

NER_GetCurrentCtrlMode();

2.5.8. NER_Robot_Change_To_CST_Mode

C/C++ Syntax:

BOOL NER_Robot_Change_To_CST_Mode (I32_T *torque_value)

Parameters:

I32_T *torque_value:

The initial torque value (0.1% of motor’s rated torque) of each joint while change operation mode to CST.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for change Operation mode to CST mode.

Reference:

NER_GetCurrentCtrlMode();

2.6. Functions for Digital I/O Operation

2.6.1. NER_Get_DIO_count

C/C++ Syntax:

U32_T NER_Get_DIO_count();

Parameters:

<No Parameters>

Returned Values:

U32_T is returned.

Return how many Digital IO module is founded.

Usage:

Call the function for getting Digital IO module numbers.

Reference:

NER_Get_DIO_Info();

NER_SetDO();

2.6.2. NER_Get_DIO_Info

C/C++ Syntax:

```
BOOL NER_Get_DIO_Info(_NER_ROBOT_DIO *dio_data);
```

Parameters:

_NER_ROBOT_DIO *dio_data:

DIO structure

```
typedef struct
{
        U32_T                   InSizeInByte;
        U32_T                   OutSizeInByte;
        DIO_Command             Input[32];
        DIO_Command             Output[32];
}_NER_ROBOT_DIO;
```

DIO Command

```
typedef enum _DIO_Command
{
        NER_DIO_OFF                 = 0,
        NER_DIO_ON                  = 1,
} DIO_Command;
```

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for getting DIO slave info such as input and output size, input and output status.

Reference:

NER_Get_DIO_count();

NER_SetDO();

2.6.3. NER_SetDO

C/C++ Syntax:

BOOL NER_SetDO(U32_T DO_index, DIO_Command *Output_data);

Parameters:

U32_T DO_index :

Index of DO slave.

DIO_Command *Output_data

Output data of DO slave.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for getting DO slave output data.

Reference:

NER_Get_DIO_count();

NER_Get_DIO_Info;

2.6.4. NER_GetDO

C/C++ Syntax:

BOOL NER_GetDO(U32_T DI_index, DIO_Command *Input_data);

Parameters:

U32_T DI_index:

Index of DI slave.

DIO_Command *Input_data:

Input data of DI slave.

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for setting DI slave Input data.

Reference:

NER_Get_DIO_count();

NER_Get_DIO_Info;

2.7. Functions for analog I/O Operation

2.7.1. NER_Get_AIO_count

C/C++ Syntax:

U32_T NER_Get_AIO_count();

Parameters:

<No Parameters>

Returned Values:

U32_T is returned.

Return how many Analog IO module is founded.

Usage:

Call the function for getting Analog IO module numbers.

Reference:

NER_Get_AIO_Info();

2.7.2. NER_Get_AIO_Info

C/C++ Syntax:

```
BOOL NER_Get_AIO_Info(_NER_ROBOT_AIO    *aio_data);
```

Parameters:

_NER_ROBOT_AIO *aio_data:

AIO structure

```
typedef struct
{
        U32_T                   InSizeInByte;
        U32_T                   OutSizeInByte;
        U32_T                   Input[8];
        U32_T                   Output[8];
        U8_T                    Channel_Num;
}_NER_ROBOT_AIO;
```

Returned Values:

Boolean is returned.

TRUE is returned if function call is successful, while FALSE is returned when failed.

Usage:

Call the function for getting AIO slave info such as input and output size, input and output values.

Reference:

NER_Get_AIO_count();