NEXCOM

EtherCAT Master

CiA402 servo control API

User Manual

Manual Rev.: V1.4

Revision Date: February 24th, 2016

NEXCOBOT Inc.
www.nexcobot.com

Revise note:

Ver	Description
V1.4	2020/2/4: Add Ch2.3.2 NEC_CoE402SetSlaveProcessCountPerCycle Add Ch 2.2.8. NEC_CoE402GetTxPdoMapping Add Ch 2.2.9. NEC_CoE402GetRxPdoMapping Add Ch 2.2.10. NEC_CoE402UpdatePdoMappingEni
V1.3	2016/2/24: Add Ch2.7.4 NEC_CoE402ClearHomeStartBit() API Add Ch2.7.5 NEC_CoE402CheckHomeStatus() API
V1.2	2015/8/14: Add Ch2.4.11 NEC_CoE402SetTargetVelocity() API Add Ch2.4.12 NEC_CoE402GetActualVelocity() API
V1.1	2015/7/13: Add description about direction of moving to Jog and JogA API.
V1.0	2014/8/19: English version released.

Index

NEXCOM 1

Revise note: 2

Index i

1. CoE CiA402 Operation Description 1

1.1. Function Call Flowchart 1

1.2. Basic Specifications 1

1.3. Notice 1

2. CoE CiA402 Library 2

2.1. API Overview 2

2.2. Functions for Initialization 6

2.2.1. NEC_CoE402Reset 6

2.2.2. NEC_CoE402Close 7

2.2.3. NEC_CoE402GetAxisId 8

2.2.4. NEC_CoE402GetDefaultMapInfo 9

2.2.5. NEC_CoE402GetAxisIdEx 11

2.2.6. NEC_CoE402ResetPdoMapping 13

2.2.7. NEC_CoE402AddPdoMapping .... 錯誤! 尚未定義書籤。

2.2.8. NEC_CoE402UpdatePdoMapping .... 錯誤! 尚未定義書籤。

2.3. CiA402 State Control Functions 18

2.3.1. NEC_CoE402CyclicProcess 18

2.3.2. NEC_CoE402SetCtrlWord / NEC_CoE402GetCtrlWord .. 錯誤! 尚未定義書籤。

2.3.3. NEC_CoE402GetStatusWord 23

2.3.4. NEC_CoE402ChangeState 24

2.3.5. NEC_CoE402SetState 25

2.3.6. NEC_CoE402GetState 26

2.3.7. NEC_CoE402FaultReset 27

2.3.8. NEC_CoE402ServoOn 28

2.4. CiA402 Servo Basic Operation Functions 29

2.4.1. NEC_CoE402SetOperationMode 29

2.4.2. NEC_CoE402GetOperationModeDisplay 31

2.4.3. NEC_CoE402SetParameter 32

2.4.4. NEC_CoE402GetParameter 33

2.4.5. NEC_CoE402GetActualPosition 34

2.4.6. NEC_CoE402SetTargetPosition / NEC_CoE402GetTargetPosition. 錯誤! 尚未定義書籤。

2.4.7. NEC_CoE402SetQuickStopDec 36

2.4.8. NEC_CoE402SetSoftPosLimit 37

2.4.9. NEC_CoE402SetMaxVelLimit 38

2.4.10. NEC_CoE402SetTargetVelocity 39

2.4.11. NEC_CoE402GetActualVelocity 40

2.5. Profile Position mode 41

2.5.1. NEC_CoE402Ptp / NEC_CoE402PtpV / NEC_CoE402PtpA錯誤! 尚未定義書籤。

2.5.2. NEC_CoE402WaitTargetReached 43

2.6. Profile Velocity mode 44

2.6.1. NEC_CoE402Jog / NEC_CoE402JogA .... 錯誤! 尚未定義書籤。

2.6.2. NEC_CoE402Halt 46

2.7. Homing mode 47

2.7.1. NEC_CoE402Home / NEC_CoE402HomeEx .... 錯誤! 尚未定義書籤。

2.7.2. NEC_CoE402WaitHomeFinished 49

2.7.3. NEC_CoE402ClearHomeStartBit 50

2.7.4. NEC_CoE402CheckHomeStatus 51

2.8. Torque Control 52

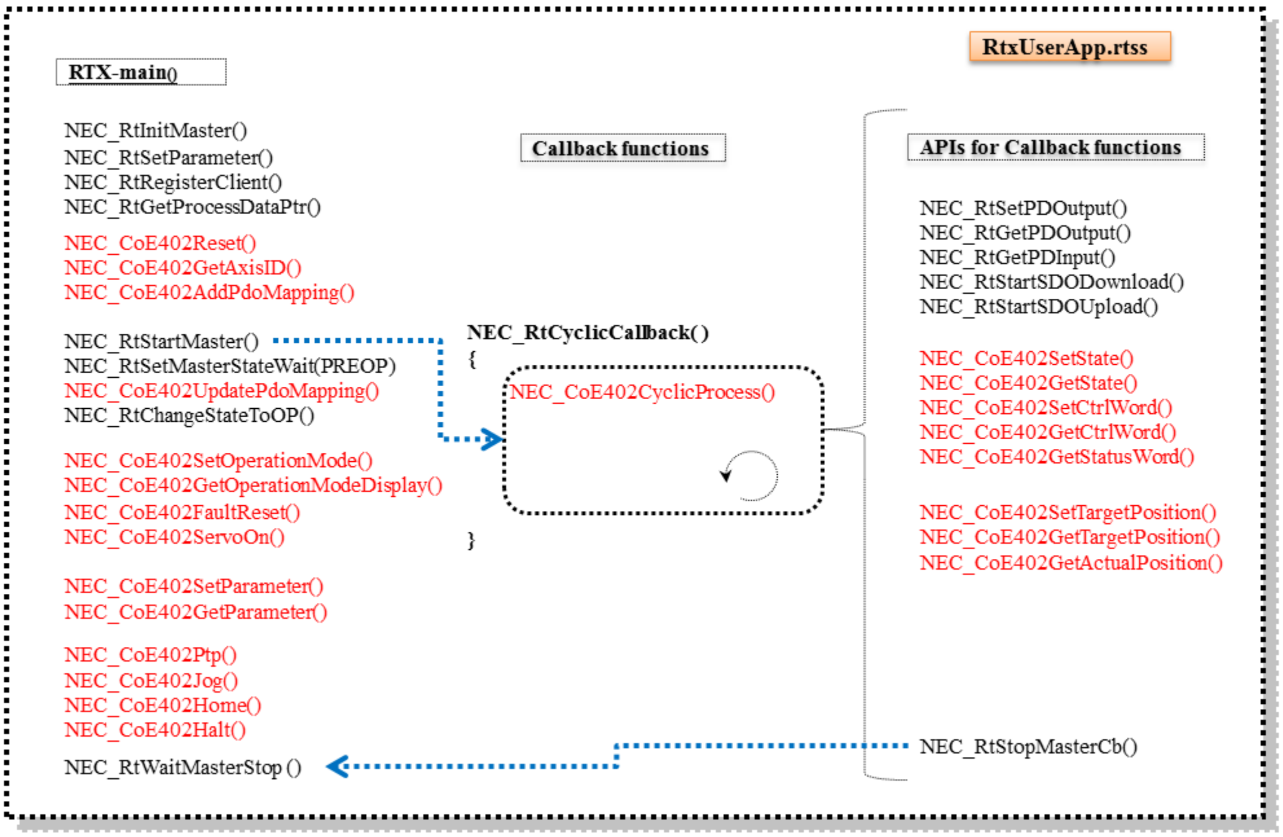
2.8.1. NEC_CoE402SetTargetTorque / NEC_CoE402GetTargetTorque 錯誤! 尚未定義書籤。

2.8.2. NEC_CoE402GetActualTorque 53

2.8.3. NEC_CoE402SetTorqueProfile.. 54

1. CoE CiA402 Operation Description

1.1. Function Call Flowchart



1.2. Basic Specifications

1. Supporting up to 32 axes.
2. Support standard CoE402 EtherCAT Servo Driver.

1.3. Notice

Notice of parameter setting for using the library:

1. The object of “ControlWord (0x6040)” must be assigned as “RxPDO” and the offset of RxPDO mapping is not limited (suggested to be “0”).
2. The object of “StatusWord (0x6041)” must be assigned as “TxPDO”. and the offset of TxPDO mapping is not limited (suggested to be “0”).
3. The object of “TargetPosition (0x607A)” is suggested to assigned as “RxPDO”. “Offset” suggested value is “2”.
4. The object of “Position actual value (0x6064)” is suggested to assigned as “TxPDO”. “Offset” suggested value is “2”.

2. CoE CiA402 Library

2.1. API Overview

All APIs of CiA402 Library are listed. The definition of API is located at the header file "NexCoEMotion.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		
NEC_CoE402Reset	Reset CiA402 library	X
NEC_CoE402Close	Close CiA402 library	X
NEC_CoE402GetAxisId	Registry and get ID of CiA servo axis	X
NEC_CoE402GetDefaultMapInfo	Initialize "MapInfo" data structure	X
NEC_CoE402GetAxisIdEx	Registry and get ID of CiA servo axis (Single device multi-axes)	X
NEC_CoE402ResetPdoMapping	Reset PDO mapping	X
NEC_CoE402AddPdoMapping	Add PDO mapping	X
NEC_CoE402UpdatePdoMapping	Update PDO mapping	X
CiA402 State Control Functions		
NEC_CoE402CyclicProcess	Cyclic CiA402 State Control	C
NEC_CoE402SetCtrlWord	Set CiA402 ControlWord (0x6040)	B
NEC_CoE402GetCtrlWord	Get CiA402 ControlWord (0x6040)	B
NEC_CoE402GetStatusWord	Get CiA402 StatusWord (0x6041)	B
NEC_CoE402ChangeState	Change State of Control Axis	X
NEC_CoE402SetState	Set Servo State	B
NEC_CoE402GetState	Get Servo State	B

NEC_CoE402FaultReset	Clear/Reset Servo Driver Fault	X
NEC_CoE402ServoOn	Set Servo ON/OFF	X
CiA402 Servo Basic Operation Functions		
NEC_CoE402SetOperationMode	Set Servo Operation mode (0x6060)	X
NEC_CoE402GetOperationModeDisplay	Get Servo Operation mode (0x6061)	X
NEC_CoE402SetParameter	Set Servo Parameter of CiA402	X
NEC_CoE402GetParameter	Get Servo Parameter of CiA402	X
NEC_CoE402GetActualPosition	Get Actual Position of Servo Motor (0x6064)	*B
NEC_CoE402SetTargetPosition	Set Target Position (0x607A,PP,CSP)	*B
NEC_CoE402GetTargetPosition	Get Target Position (0x607A,PP,CSP)	*B
NEC_CoE402SetQuickStopDec	Set Quick Stop Deceleration Rate (0x6085)	X
NEC_CoE402SetSoftPosLimit	Set Software Position Limitation (0x607D)	X
NEC_CoE402SetMaxVelLimit	Set Maximum Velocity Limitation (0x607F)	X
Profile Position mode		
NEC_CoE402Ptp	Perform Point-to-point operation	X
NEC_CoE402PtpV	Perform Point-to-point operation with velocity parameter	X
NEC_CoE402PtpA	Perform Point-to-point operation with velocity and acceleration parameters	X
NEC_CoE402WaitTargetReached	Wait until Target position is reached	X
Profile Velocity mode		
NEC_CoE402Jog	Perform jog operation	X
NEC_CoE402JogA	Perform jog operation with acceleration parameters	X

NEC_CoE402Halt	Stop/Halt Operation	X
	Homing mode	
NEC_CoE402Home	Perform Homing operation	X
NEC_CoE402HomeEx	Perform Homing operation with relative parameter settings	X
NEC_CoE402WaitHomeFinished	Wait until Homing is Finished	X
NEC_CoE402ClearHomeStartBit	Clear the home start bit	B
NEC_CoE402CheckHomeStatus	Get status of homing	B
	Torque Control	
NEC_CoE402SetTargetTorque	Set Target Torque (0x6071,PT,CST)	*B
NEC_CoE402GetTargetTorque	Get Target Torque (0x6071,PT,CST)	*B
NEC_CoE402GetActualTorque	Get Torque actual value (0x6077)	*B
NEC_CoE402SetTorqueProfile	Set Target torque and slope (0x6071, 0x6087, PT)	X

(*B means the data value must set as “PDO mapping” for Callback functions)

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

Type	C/C++ Primitive	format	Byte	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. NEC_CoE402Reset

Reset CiA402 Library

C/C++ Syntax:

RTN_ERR NEC_CoE402Reset();

Parameters:

<no Parameters>

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

The Error code is defined in header file "EcErrors.h".

Usage:

Call the function for initializing the library internally, just before using other libraries of CoE402.

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

Reference:

NEC_CoE402Close ();

2.2.2. NEC_CoE402Close

Close CiA402 library.

C/C++ Syntax:

RTN_ERR NEC_CoE402Close ();

Parameters:

<no Parameters>

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” （0） is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

The function is called to release the internal resources of CiA402 library before the program is finished.

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

Reference:

NEC_CoE402Reset();

2.2.3. NEC_CoE402GetAxisId

Registry and get ID of CiA servo axis

C/C++ Syntax:

RTN_ERR NEC_CoE402GetAxisId(U16_T MasterId, U16_T SlaveAddr, CANAxis_T *pAxis);

Parameters:
U16_T MasterId: Assign target ID of EC-Master. For singular EC-Master, please set ID as 0.
U16_T SlaveAddr: Assign target ID of EC-Slave. The ID of a slave is increasingly from 0 and follows the order of connection.
*CANAxis_T *pAxis:* Return the control identification from corresponding CiA servo axis. The ID can be used in other function then.

Returned Values:
Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.
The Error code is defined in header file "EcErrors.h".

Usage:
The function is used to register a sCiA402 slave and get the control ID as a CiA servo axis after the CiA402 library is initialized. The returned ID of axis is then used in other CiA402 library.
This API only for single axis device, for multi-axes device using *NEC_CoE402GetAxisIdEx*()

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

Reference:
NEC_CoE402CyclicProcess();NEC_CoE402GetAxisId()

2.2.4. NEC_CoE402GetDefaultMapInfo

Initialize “MapInfo” data structure

C/C++ Syntax:

RTN_ERR NEC_CoE402GetDefaultMapInfo(U16_T TypeOfSlave, U16_T SlaveAddr, EcCiA402AxisMapInfo_T *pMapInfo);

Parameters:

U16_T TypeOfSlave: Slave type

EC_CIA402_SERVO_DRIVE (0):Single standard CiA402 axis slave device

EC_MULTIPLE_CIA402_SLAVE (1): Multiple standard CiA402 axes slave device

U16_T SlaveAddr: Assign target ID of EC-Slave. The ID of a slave is increasingly from 0 and follows the order of connection.

EcCiA402AxisMapInfo_T *pMapInfo: CiA axis object map information，This structure will be initialized as follow:

```
typedef struct
{
    U16_T TypeOfSlave;
    U16_T SlaveAddr;
    U16_T SlaveSlotNum;
    U16_T Reserved;         // Reserved, set zero.
    U16_T CoeObjectOffset; // ex. 0x0800
    U16_T pdoMapOffset;     // ex. 0x0010
    U16_T MinRxPdoIndex;   // ex. 0x1600
    U16_T MaxRxPdoIndex;   // ex. 0x1603
    U16_T MinTxPdoIndex;   // ex. 0x1A00
    U16_T MaxTxPdoIndex;   // ex. 0x1A03
}EcCiA402AxisMapInfo_T;
```

U16_T TypeOfSlave: Slave type, Initialized as input above

U16_T SlaveAddr: Target ID of EC-Slave, Initialized as input above

U16_T SlaveSlotNum: Local axis number in a slave, zero based.

U16_T Reserved: Reserved, be set zero.

U16_T CoeObjectOffset: CoE object offset，be initialized as 0x0800

U16_T pdoMapOffset; PDO map object offset，be initialized as 0x0010

U16_T MinRxPdoIndex: First axis minimum RxPDO index，be initialized as 0x1600

U16_T MaxRxPdoIndex: First axis maximum RxPDO index，be initialized as 0x1603

U16_T MinTxPdoIndex: First axis minimum TxPDO index，be initialized as 0x1A00

U16_T MaxTxPdoIndex: First axis maximum TxPDO index，be initialized as 0x1A03

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” （0） is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

Before calling *NEC_CoE402GetAxisIdEx*() to register an axis with parameter ” EcCiA402AxisMapInfo_T” which record the information that how object be mapped for an axis. Using this API to initialize the structure then calling *NEC_CoE402GetAxisIdEx*() to register an axis.

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

Reference:

NEC_CoE402GetAxisIdEx()

2.2.5. NEC_CoE402GetAxisIdEx

Registry and get ID of CiA servo axis (Single device multi-axes)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetAxisIdEx(U16_T MasterId, EcCiA402AxisMapInfo_T *pInfo, CANAxis_T *pAxis);

Parameters:

U16_T MasterId: Assign target ID of EC-Master. For singular EC-Master, please set ID as 0.

EcCiA402AxisMapInfo_T *pMapInfo: CiA axis object map information，This structure will be initialized as follow:

```
typedef struct
{
    U16_T TypeOfSlave;
    U16_T SlaveAddr;
    U16_T SlaveSlotNum;
    U16_T Reserved;         // Reserved, set zero.
    U16_T CoeObjectOffset; // ex. 0x0800
    U16_T pdoMapOffset;     // ex. 0x0010
    U16_T MinRxPdoIndex;   // ex. 0x1600
    U16_T MaxRxPdoIndex;   // ex. 0x1603
    U16_T MinTxPdoIndex;   // ex. 0x1A00
    U16_T MaxTxPdoIndex;   // ex. 0x1A03
}EcCiA402AxisMapInfo_T;
```

U16_T TypeOfSlave: Slave type

EC_CIA402_SERVO_DRIVE (0): Single standard CiA402 axis slave device

EC_MULTIPLE_CIA402_SLAVE (1): Multiple standard CiA402 axes slave device

U16_T SlaveAddr: Assign target ID of EC-Slave. The ID of a slave is increasingly from 0 and follows the order of connection.

U16_T SlaveSlotNum: Local axis number in a slave, zero based.

U16_T Reserved: Reserved, be set zero.

U16_T CoeObjectOffset: CoE object offset，be initialized as 0x0800

U16_T pdoMapOffset; PDO map object offset，be initialized as 0x0010

U16_T MinRxPdoIndex: First axis minimum RxPDO index，be initialized as 0x1600

U16_T MaxRxPdoIndex: First axis maximum RxPDO index，be initialized as 0x1603

U16_T MinTxPdoIndex: First axis minimum TxPDO index，be initialized as
0x1A00

U16_T MaxTxPdoIndex: First axis maximum TxPDO index，be initialized as
0x1A03

*CANAxis_T *pAxis:* Return the control identification from corresponding CiA servo axis. The ID can be used in other function then.

Returned Values:

Error Code is returned.

“ECERR_SUCCESS”（0） is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

The function is used to register a sCiA402 slave and get the control ID as a CiA servo axis after the CiA402 library is initialized. The returned ID of axis is then used in other CiA402 library.

This API can be registered multiple axes device and single axis device.

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

Reference:

NEC_CoE402CyclicProcess(); NEC_CoE402GetAxisId()

2.2.6. NEC_CoE402ResetPdoMapping

Reset PDO mapping

C/C++ Syntax:
RTN_ERR NEC_CoE402ResetPdoMapping(CANAxis_T Axis);

Parameters:
CANAxis_T Axis: Assign ID of control axis. The axis ID is returned by function call to NEC_CoE402GetAxisId().

Returned Values:
Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.
The Error code is defined in header file "EcErrors.h".

Usage:
The function is to reset PDO mapping. It will also be called/executed automatically when NEC_CoE402Reset() function is called. User must assign the PDO mapping correctly before the state of CiA402 slave becomes "OP". There are two setting methods:
1. Assign PDO mapping pattern to the library by *NEC_CoE402AddPdoMapping*()
2. Retrieve the PDO mapping pattern to the library by *NEC_CoE402UpdatePdoMapping*().
Recommend using method 2.

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

Reference:
NEC_CoE402AddPdoMapping()

2.2.7. NEC_CoE402UpdatePdoMapping

Update PDO mapping

C/C++ Syntax:

RTN_ERR NEC_CoE402UpdatePdoMapping(CANAxis_T Axis);

Parameters:

CANAxis_T Axis: assign ID of control axis. The axis ID is returned from function call to NEC_CoE402GetAxisId()

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

PDO mapping data is updated automatically from EC-Slave of CiA402 by function call to *NEC_CoE402UpdatePdoMapping*(). Note that the function can be called only when EC-Slave is in the state of “PREOP”, “SAFEOP”, or “OP”.

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

Reference:

NEC_CoE402AddPdoMapping();

2.2.8. NEC_CoE402GetTxPdoMapping

Get Tx PDO mapping configuration.

C/C++ Syntax:

RTN_ERR NEC_CoE402GetTxPdoMapping(CANAxis_T Axis, U32_T *PRetEntryArray, U32_T *PRetInArraySizeOutActualSize);

Parameters:

CANAxis_T Axis: Assign ID of control axis. The axis ID is returned by function call to NEC_CoE402GetAxisId().
U32_T *PRetEntryArray: return Entry array.
U32_T *PRetInArraySizeOutActualSize: return actual size of array.

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.
The Error code is defined in header file "EcErrors.h".

Usage:

Get axis mapping info by calling *NEC_ CoE402GetTxPdoMapping*().TO update axis mapping info, call *NEC_CoE402UpdatePdoMapping() or NEC_CoE402UpdatePdoMappingEni().*

Reference:

NEC_CoE402GetAxisId();NEC_CoE402GetAxisIdEx();
NEC_CoE402UpdatePdoMapping();NEC_CoE402UpdatePdoMappingEni()

2.2.9. NEC_CoE402GetRxPdoMapping

Get Rx PDO mappingg configuration.

C/C++ Syntax:

RTN_ERR NEC_CoE402GetRxPdoMapping(CANAxis_T Axis, U32_T *PRetEntryArray, U32_T *PRetInArraySizeOutActualSize);

Parameters:

CANAxis_T Axis: Assign ID of control axis. The axis ID is returned by function call to NEC_CoE402GetAxisId().
U32_T *PRetEntryArray: return Entry array.
U32_T *PRetInArraySizeOutActualSize: return actual size of array.

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.
The Error code is defined in header file "EcErrors.h".

Usage:

Get axis mapping info by calling *NEC_ CoE402GetTxPdoMapping*().TO update axis mapping info, call *NEC_CoE402UpdatePdoMapping() or NEC_CoE402UpdatePdoMappingEni().*

Reference:

NEC_CoE402GetAxisId();NEC_CoE402GetAxisIdEx();
NEC_CoE402UpdatePdoMapping();NEC_CoE402UpdatePdoMappingEni()

2.2.10. NEC_CoE402UpdatePdoMappingEni

Update setting of PDO mapping

C/C++ Syntax:

RTN_ERR NEC_CoE402UpdatePdoMappingEni(CANAxis_T Axis, const char *PEniPath);

Parameters:

CANAxis_T Axis: Assign ID of control axis. The axis ID is returned by function call to NEC_CoE402GetAxisId().

Const char *PEniPath: location of Eni File

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

Update EC-Slave PDO mapping from ENI file by calling *NEC_CoE402UpdatePdoMappingEni* ().Using this function before other CiA402 function call.

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

Reference:

NEC_CoE402GetAxisId();NEC_CoE402GetAxisIdEx();

2.3. CiA402 State Control Functions

2.3.1. NEC_CoE402CyclicProcess

Cycle State Control

C/C++ Syntax:
RTN_ERR NEC_CoE402CyclicProcess();

Parameters:
<no Parameters>

Returned Values:
Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.
The Error code is defined in header file "EcErrors.h".

Usage:
The state machine of CiA402 slave is controlled through the ControlWord (0x6040) and StatusWord (0x6041). Users can maintain the state machine themselves by control the ControlWord(0x6040) directly or using this function to maintain the state machine.
The function should be executed periodically. In general, the function is executed in Callback function. The function is used to control all states of CiA402 EC-Slaves, and perform state-switching procedure internally. User can assign new CiA402 state directly without handling the switching state. For the state-switching related function, please refer to *NEC_CoE402ChangeState*(), *NEC_CoE402SetState*() and NEC_CoE402ServoOn()

When this function is used to control the state machine of CiA402 slaves, The bit0~3 of ControlWord(0x6040) is controlled by this function. If user need to access the controlword with this function is executed. User must avoid accessing the bit0~3 or race condition could happen.

Bit NO.	Function
0	Switch on
1	Enable voltage
2	Quick Stop

3	Enable operation

(CiA402 ControlWord bit0~3)

Note that ControlWord(0x6040) must be set as RxPDO mapping and StatusWord (0x6041) must be set as TxPDO mapping.

Reference:

NEC_CoE402ChangeState(); NEC_CoE402SetState(); NEC_CoE402GetState(); NEC_CoE402ServoOn()

2.3.2. NEC_CoE402SetSlaveProcessCountPerCycle

Set the maximum access number of axis control every cycle time.

C/C++ Syntax:

RTN_ERR NEC_CoE402SetSlaveProcessCountPerCycle(U16_T AxisCount);

Parameters:

U16_T AxisCount: maximum access number of axis.

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

The Error code is defined in header file "EcErrors.h".

Usage:

Cycle time is setted by this function.Control the maximum access number of axis.For instance,we have system with 16 axes.if AxisCount was setted as 8.Updating all axes on system spend two cycle.Default value of AxisCount is 8.

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

Reference:

2.3.3. NEC_CoE402SetCtrlWord / NEC_CoE402GetCtrlWord

Set/Get CiA402 ControlWord (0x6040)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetCtrlWord(CANAxis_T Axis, U16_T CtrlInBit);

RTN_ERR NEC_CoE402GetCtrlWord(CANAxis_T Axis, U16_T *CtrlInBit);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId()

U16_T CtrlInBit: Set CiA402 ControlWord, Object index = 0x6040

U16_T *CtrlInBit: Get CiA402 ControlWord, Object index = 0x6040

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

The Error code is defined in header file "EcErrors.h".

Usage:

The function is used to control CiA402 ControlWord (object index is 0x6040). The function can be used in a Callback function. Note that ControlWord(0x6040) must be set as RxPDO mapping.

Bit definition of ControlWord in CiA402:

Bit NO.	Function
0	Switch on
1	Enable voltage
2	Quick Stop
3	Enable operation
4 ~ 6	<Depend on operation mode> (*)
7	Fault reset
8	Halt
9	<Depend on operation mode> (*)
10	Reserved
11~15	<Vendor specific> (*)

(*)Please refer to CiA402 device manual

If using *NEC_CoE402CyclicProcess* () to control CiA402 state switch, the function will control ControlWord (0x6040) of Bit0 ~ 3. User should avoid modifying bit0 ~ 3 when *NEC_CoE402CyclicProcess* () is executed.
Typically, Read the ControlWord first and configure the bits then write it back.

Reference:
NEC_CoE402CyclicProcess()

2.3.4. NEC_CoE402GetStatusWord

Get CiA402 StatusWord (0x6041)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetStatusWord(CANAxis_T Axis, U16_T *StatusInBit);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T *StatusInBit: Get CiA402 StatusWord, Object index = 0x6041

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

The function is used to get CiA402 StatusWord (object index 0x6041). The function can be used in a Callback function. Note that StatusWord (0x6041) must be set as TxPDO mapping.

The bit definition of StatusWord in CiA402:

Bit NO.	Function	Bit NO.	Function
0	Ready to switch on	8	<Vendor specific> (*)
1	switch on	9	Remote
2	Operatiion enable	10	<Depend on operation mode> (*)
3	Fault	11	Internal limit active
4	Voltage enable	12	<Depend on operation mode> (*)
5	Quick stop	13	
6	Swithc on disabled	14	<Vendor specific> (*)
7	Warning	15	

Reference:

2.3.5. NEC_CoE402ChangeState

Change state of control axis

C/C++ Syntax:

RTN_ERR NEC_CoE402ChangeState(CANAxis_T Axis, U16_T TargetState, I32_T TimeoutMs);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T TargetState: Target state of CiA402:

#define	CiA402 state	Note
COE_STA_DISABLE(0)	Switch on disable	Servo OFF
COE_STA_READY_TO_SWITCH_ON(1)	Ready to switch on	
COE_STA_SWITCH_ON (2)	Switched on	
COE_STA_OPERATION_ENABLE (3)	Operation enable	Servo ON
COE_STA_QUICK_STOP_ACTIVE (4)	Quick stop active	

I32_T TimeoutMs: Duration of switching timeout in millisecond. When TimeoutMs value is set as “0”, it is the same as function call to *NEC_CoE402SetState*().

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” （0） is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

The function is to change the state of CiA402 slaves. The state is switched successfully if the function call is returned successfully. To switch the motor state during Callback, user must execute the function call to *NEC_CoE402SetState*() instead. This function must be used with *NEC_CoE402CyclicProcess* ().

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

Reference:

NEC_CoE402CyclicProcess(); NEC_CoE402SetState();

2.3.6. NEC_CoE402SetState

Set Servo State

C/C++ Syntax:

RTN_ERR NEC_CoE402SetState(CANAxis_T Axis, U16_T State);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T State: Target State of CiA402:

#define	CiA402 state	Note
COE_STA_DISABLE(0)	Switch on disable	
COE_STA_READY_TO_SWITCH_ON(1)	Ready to switch on	
COE_STA_SWITCH_ON (2)	Switched on	
COE_STA_OPERATION_ENABLE (3)	Operation enable	Servo ON
COE_STA_QUICK_STOP_ACTIVE (4)	Quick stop active	

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

The Error code is defined in header file "EcErrors.h".

Usage:

The function is to set motor target state, it can be used in Callback function. The function call will be returned immediately without waiting for completion of motor switching state. User should apply the function call to *NEC_CoE402GetState*() to confirm motor state.

This function must be used with *NEC_CoE402CyclicProcess* ().

Reference:

NEC_CoE402CyclicProcess(); NEC_CoE402ChangeState();

2.3.7. NEC_CoE402GetState

Get Servo State

C/C++ Syntax:

RTN_ERR NEC_CoE402GetState(CANAxis_T Axis, U16_T *State);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T *State: Return Motor State:

#define	CiA402 state	Note
COE_STA_DISABLE(0)	Switch on disable	
COE_STA_READY_TO_SWITCH_ON(1)	Ready to switch on	
COE_STA_SWITCH_ON (2)	Switched on	
COE_STA_OPERATION_ENABLE (3)	Operation enable	Servo ON
COE_STA_QUICK_STOP_ACTIVE (4)	Quick stop active	
COE_STA_FAULT(5)	Fault	Error
COE_STA_FAULT_REACTION_ACTIVE (6)	Fault reaction active	Error

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

The function is to get current motor state (CiA402 State), it can be used in Callback function.

Reference:

NEC_CoE402SetState();

2.3.8. NEC_CoE402FaultReset

Clear/Reset servo driver fault

C/C++ Syntax:

RTN_ERR NEC_CoE402FaultReset(CANAxis_T Axis, I32_T TimeoutMs);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I32_T TimeoutMs: Timeout time for waiting fault reset. When TimeoutMs value is set as “0” or less than “0”, the timeout will be “5000ms”.

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” （0） is returned if function call is successful, while “Error Code” is returned when failed.

The Error code is defined in header file “EcErrors.h”.

Usage:

When servo is in “Fault” state, please check the servo failure state and make fault corrected, then use this function to clear/reset the “Fault” state afterward.

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

Reference:

2.3.9. NEC_CoE402ServoOn

Set Servo On/Off

C/C++ Syntax:

RTN_ERR NEC_CoE402ServoOn(CANAxis_T Axis, U16_T OnOff);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T OnOff:

0: Servo off (Switch on disable)

1 Servo ON (Operation enable)

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

To set servo on and off.

ServoOn correspond to CiA state: Operation Enable

ServoOff correspond to CiA state: Switch on disable

This function must be used with *NEC_CoE402CyclicProcess* ().

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

Reference:

NEC_CoE402CyclicProcess();NEC_CoE402SetState();NEC_CoE402ChangeState();

2.4. CiA402 Servo Basic Operation Functions

2.4.1. NEC_CoE402SetOperationMode

Set Servo Operation Mode (0x6060)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetOperationMode(CANAxis_T Axis, I8_T MotionMode, I32_T CheckTimeoutMs);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I8_T MotionMode: Set Operation Mode. For supported modes, please refer to servo manual.

Define	CiA402 Operation Mode
CiA402_OP_MODE_PROFILE_POSITION (1)	Profile Position mode
CiA402_OP_MODE_PROFILE_VELOCITY (3)	Profile Velocity mode
CiA402_OP_MODE_TORQUE_PROFILE (4)	Torque Profile mode
CiA402_OP_MODE_HOMING (6)	Homing mode
CiA402_OP_MODE_INTERPOLATED_POSITION (7)	Interpolated Position mode
CiA402_OP_MODE_CYCLIC_POSITION (8)	Cyclic Sync Position mode
CiA402_OP_MODE_CYCLIC_VELOCITY (9)	Cyclic Sync Velocity mode
CiA402_OP_MODE_CYCLIC_TORQUE (10)	Cyclic Sync Torque mode

I32_T CheckTimeoutMs: Duration of timeout (in millisecond).

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

Set servo mode of operation, with object index 0x6060.

The operation mode is set/switched successfully if the function call is returned successfully as well.

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

Reference:

2.4.2. NEC_CoE402GetOperationModeDisplay

Get Servo Operation Mode (0x6061)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetOperationModeDisplay(CANAxis_T Axis, I8_T *MotionMode);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I8_T *MotionMode: return current operation mode of servo:

Define	CiA402 Operation Mode
CiA402_OP_MODE_PROFILE_POSITION (1)	Profile Position mode
CiA402_OP_MODE_PROFILE_VELOCITY (3)	Profile Velocity mode
CiA402_OP_MODE_TORQUE_PROFILE (4)	Torque Profile mode
CiA402_OP_MODE_HOMING (6)	Homing mode
CiA402_OP_MODE_INTERPOLATED_POSITION (7)	Interpolated Position mode
CiA402_OP_MODE_CYCLIC_POSITION (8)	Cyclic Sync Position mode
CiA402_OP_MODE_CYCLIC_VELOCITY (9)	Cyclic Sync Velocity mode
CiA402_OP_MODE_CYCLIC_TORQUE (10)	Cyclic Sync Torque mode

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

Get servo mode of operation display, with object index 0x6061.

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

Reference:

2.4.3. NEC_CoE402SetParameter

Set Servo Parameters of CiA402

C/C++ Syntax:

RTN_ERR NEC_CoE402SetParameter(CANAxis_T Axis, U16_T Index, U8_T SubIndex, U8_T LenOfByte, I32_T Value);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
U16_T Index: CiA object parameters index, please refer to servo manual.
U8_T SubIndex: CiA object parameters sub-index, please refer to servo manual.
U8_T LenOfByte: The length of object (in bytes), ranged from 1 to 4 bytes.
I32_T Value: Object parameter value.

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

Set CiA402 servo parameters. For related servo parameters please refer to servo manual. Use this function to access the object parameters of CiA402 slaves, this function will automatically use PDO or SDO access the module according to the current PDO mapping table. If the object parameters are PDO mapping objects, this function access the parameter via PDO access, If not PDO mapping parameters are accessed using SDO way. Note that mapping table must be built by call *NEC_CoE402UpdatePdoMapping*() in the program initialization phase.

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

Reference:

NEC_CoE402GetParameter (); NEC_CoE402UpdatePdoMapping();

2.4.4. NEC_CoE402GetParameter

Get Servo Parameters of CiA402

C/C++ Syntax:

RTN_ERR NEC_CoE402GetParameter(CANAxis_T Axis, U16_T Index, U8_T SubIndex, U8_T LenOfByte, void *pRetValue);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
U16_T Index: CiA object parameters index, please refer to servo manual.
U8_T SubIndex: CiA object parameters sub-index, please refer to servo manual.
U8_T LenOfByte: The length of object (in bytes), ranged from 1 to 4 bytes.
void *pRetValue: Return object parameter value, in order to avoid memory access violation, the actual data length of this pointer should match with "LenOfByte".

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

Get CiA402 servo parameters. For related servo parameters please refer to servo manual. Use this function to access the object parameters of CiA402 slaves, this function will automatically use PDO or SDO access the module according to the current PDO mapping table. If the object parameters are PDO mapping objects, this function access the parameter via PDO access, If not PDO mapping parameters are accessed using SDO way. Note that mapping table must be built by call *NEC_CoE402UpdatePdoMapping*() in the program initialization phase.

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

Reference:

NEC_CoE402SetParameter (); NEC_CoE402UpdatePdoMapping();

2.4.5. NEC_CoE402GetActualPosition

Get Actual Position of Servo Motor (0x6064)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetActualPosition(CANAxis_T Axis, I32_T *Position);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I32_T *Position: Return motor actual position. (object index:0x6040)

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

The function can be used in a Callback function. User must assign the object 0x6064 as mapping object of TxPDO.

Reference:

2.4.6. NEC_CoE402SetTargetPosition / NEC_CoE402GetTargetPosition

Set/Get Target Position (0x607A,PP,CSP)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetTargetPosition(CANAxis_T Axis, I32_T TargetPos);
RTN_ERR NEC_CoE402GetTargetPosition(CANAxis_T Axis, I32_T *TargetPos);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
I32_T TargetPos: Set Target Position(0x607A)
I32_T *TargetPos: Get Target Position(0x607A)

Returned Values:

Error Code is returned.
“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

The function can be used in a Callback function. User must assign the object 0x607A as mapping object of RxPDO. “TargetPositon” will be used in Profile Position Mode and Cyclic Sync Position mode.

Reference:

2.4.7. NEC_CoE402SetQuickStopDec

Set Quick Stop Deceleration Rate (0x6085)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetQuickStopDec(CANAxis_T Axis, U32_T QuickStopDec);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U32_T QuickStopDec: Set Quick Stop Deceleration Rate (0x6085)

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

This function is equivalent to calling NEC_CoE402SetParameter(Axis, 0x6085, 0, 4, (I32_T) QuickStopDec);

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

Reference:

NEC_CoE402SetParameter ();

2.4.8. NEC_CoE402SetSoftPosLimit

Set Software Position Limitation (0x607D)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetSoftPosLimit(CANAxis_T Axis, I32_T MinPositionLimit, I32_T MaxPositionLimit);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
I32_T MinPositionLimit: Minimum Position Limitation of Software (0x607D : 01)
I32_T MaxPositionLimit: Maximum Position Limitation of Software (0x607D : 02)

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

To release the limitation that software position were set, just set both minimum and maximum values to "0".
This function is equivalent to calling NEC_CoE402SetParameter(Axis, 0x607D, 1, 4, MinPositionLimit) for minimum position limit and NEC_CoE402SetParameter(Axis, 0x607D, 2, 4, MaxPositionLimit) for maximum position limit.

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

Reference:

NEC_CoE402SetParameter ();

2.4.9. NEC_CoE402SetMaxVelLimit

Set Maximum Velocity Limitation (0x607F)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetMaxVelLimit(CANAxis_T Axis, U32_T MaxVelocityLimit);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U32_T MaxVelocityLimit: Set Maximum Velocity Limitation (0x607F)

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

Set maximum velocity limitation in "Profile Position mode" (P2P) or "Profile Velocity mode"(Jog). This function is equivalent to calling NEC_CoE402SetParameter(Axis, 0x6085, 0, 4, (I32_T) MaxVelocityLimit);

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

Reference:

NEC_CoE402SetParameter ();

2.4.10. NEC_CoE402SetTargetVelocity

Set Target Velocity (0x60FF,PV,CSV)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetTargetVelocity (CANAxis_T Axis, I32_T TargetVel);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I32_T TargetVel: Set Target Velocity (0x60FF)

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

The function can be used in a Callback function. User must assign the object 0x60FF as mapping object of RxPDO. “TargetVelocity” will be used in Profile Velocity Mode and Cyclic Sync Velocity mode.

Reference:

2.4.11. NEC_CoE402GetActualVelocity

Get Actual Velocity (0x606C,PV,CSV)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetActualVelocity(CANAxis_T Axis, I32_T *ActualVel)

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I32_T *ActualVel: Get Actual Velocity (0x606C)

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

The function can be used in a Callback function. User must assign the object 0x606C as mapping object of TxPDO. “ActualVelocity” will be used in Profile Velocity Mode and Cyclic Sync Velocity mode.

Reference:

2.5. Profile Position mode

2.5.1. NEC_CoE402Ptp / NEC_CoE402PtpV / NEC_CoE402PtpA

Perform a Point-to-Point Operation (Profile Position mode).

C/C++ Syntax:

RTN_ERR NEC_CoE402Ptp(CANAxis_T Axis, U32_T Option, I32_T TargetPos);
RTN_ERR NEC_CoE402PtpV(CANAxis_T Axis, U32_T Option, I32_T TargetPos, U32_T MaxVel);
RTN_ERR NEC_CoE402PtpA(CANAxis_T Axis, U32_T Option, I32_T TargetPos, U32_T MaxVel, U32_T Acc, U32_T Dec);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
U32_T Option: Options (in bit format). For two or more options assigning at the same time, use Boolean function of "OR".

OPT_ABS (0x00000000): TargetPos is absolute target position coordinates
OPT_REL(0x00000040): TargetPos is relative target position coordinates
OPT_WMC (0x00010000): Wait until PTP operation target is reached
OPT_IMV(0x10000000): Ignore parameters MaxVel inputs
OPT_IAC (0x20000000) :ignore parameters Acc inputs
OPT_IDC (0x4000000): ignore parameters Dec inputs

I32_T TargetPos: Absolute or relative target position (CoE: Target Position 0x607A), depend on options.
U32_T MaxVel: Maximum Velocity (CoE: Profile Velocity 0x6081)
U32_T Acc: Acceleration Rate (CoE: Profile Acceleration 0x6083)
U32_T Dec: Deceleration Rate(CoE: Profile Deceleration 0x6084)

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

This function performs a PTP move operation using "Profile Position mode", therefore required to confirm whether CiA402 module supports Profile Position (PP) mode. Switch to PP mode automatically when calling this function. The Differences between Ptp, PtpV and PtpA is that have different input parameters.

NEC_CoE402Ptp() only set the Target Position “TargetPos” (CoE: Target Position 0x607A) and starting PTP motion. Maximum velocity “MaxVel” (CoE: Profile Velocity 0x6081) , acceleration “Acc” (CoE: Profle Acceleration 0x6083) and deceleration “Dec” (CoE: Profile Deceleration 0x6084) are based on current setting of slave.

NEC_CoE402PtpV() set the Target Position “TargetPos” (CoE: Target Position 0x607A) and maximum velocity “MaxVel” (CoE: Profile Velocity 0x6081) and starting PTP motion. Acceleration “Acc” (CoE: Profle Acceleration 0x6083) and deceleration “Dec” (CoE: Profile Deceleration 0x6084) are in accordance with CiA402 module current settings.

NEC_CoE402PtpA() Start a PTP motion with all relevant parameters include Target Position “TargetPos” (CoE: Target Position 0x607A), maximum velocity “MaxVel” (CoE: Profile Velocity 0x6081), Acceleration “Acc” (CoE: Profle Acceleration 0x6083) and deceleration “Dec” (CoE: Profile Deceleration 0x6084).

Option “OPT_REL (0x00000040)” indicated the input of “TargetPos” is a relative distance from current position.

When option “OPT_WMC (Wait motion complete)” enable, this function will be block until motion is done or error is occurred. The behavior is equivalent to calling NEC_CoE402WaitTargetReached ()

OPT_IMV(0x10000000), OPT_IAC (0x20000000)和 OPT_IDC (0x4000000) only for *NEC_CoE402PtpA*()，which means ignore the input parameter setting but using current slave setting.

Calling *NEC_CoE402Halt*() to stop the operation.

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

Reference:
NEC_CoE402WaitTargetReached (); NEC_CoE402Halt();

2.5.2. NEC_CoE402WaitTargetReached

Wait Until Target Position is Reached

C/C++ Syntax:

RTN_ERR NEC_CoE402WaitTargetReached(CANAxis_T Axis);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

The function is a synchronous function, usually be called after *NEC_CoE402Ptp*(), *NEC_CoE402PtpV*(), *NEC_CoE402PtpA*(), NEC_CoE402Jog() and NEC_CoE402JogA() functions.
If after PTP, block API until PTP operation is finished.
If after Jog, block API until Target Velocity is reached.
If the "Error Code" is returned, it means the operation is fault or being suspended.

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

Reference:

NEC_CoE402Ptp(); NEC_CoE402PtpV();
NEC_CoE402PtpA();NEC_CoE402Jog();NEC_CoE402JogA();NEC_CoE402Home();
NEC_CoE402HomeEx()

2.6. Profile Velocity mode

2.6.1. NEC_CoE402Jog / NEC_CoE402JogA

Perform a jog move (Profile velocity mode)

C/C++ Syntax:

RTN_ERR NEC_CoE402Jog(CANAxis_T Axis, U32_T Option, I32_T MaxVel);
RTN_ERR NEC_CoE402JogA(CANAxis_T Axis, U32_T Option, I32_T MaxVel, I32_T Acc, I32_T Dec);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
U32_T Option: Options (in bit format). For two or more options assigning at the same time, use Boolean function of “OR”.

OPT_WMC (0x00010000): Wail until target velocity is reached
OPT_IAC (0x20000000) : ignore parameters from Acc inputs
OPT_IDC (0x4000000): ignore parameters from Dec inputs

I32_T MaxVel: Target velocity (CoE: Target Velocity 0x60FF)
Plus or minus value decides different direction of moving.

U32_T Acc: Acceleration Rate (CoE: Profile Acceleration 0x6083)
U32_T Dec: Deceleration Rate (CoE: Profile Deceleration 0x6084)

Returned Values:

Error Code is returned.
“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

This function performs a Jog move operation using “Profile Velocity mode”, therefore required to confirm whether CiA402 module supports Profile Velocity (PV) mode. Switch to PV mode automatically when calling this function. The Differences between Jog and JogA is that have different input parameters:

NEC_CoE402Jog() only set the target velocity “MaxVel” (CoE: Target Velocity 0x60FF) and start a Jog motion, acceleration “Acc” and deceleration “Dec” (CoE: Profile Acceleration and Deceleration: 0x6083, 0x6084) are based on current setting of slave.

NEC_CoE402JogA() Start a jog motion with all relevant parameters include target velocity “MaxVel” (CoE: Target Velocity 0x607A), acceleration “Acc” (CoE: Profile Acceleration 0x6083) and deceleration “Dec” (CoE: Profile Deceleration 0x6084).

When option “OPT_WMC ” is enable, this function will be block until target velocity is reached or error is occurred. The behavior is equivalent to calling *NEC_CoE402WaitTargetReached*()

OPT_IAC (0x20000000)和 OPT_IDC (0x4000000) only for *NEC_CoE402JogA*()，which means ignore the input parameter setting but using current slave setting.

Calling *NEC_CoE402Halt*() to stop the operation.

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

Reference:
NEC_CoE402WaitTargetReached(); NEC_CoE402Halt()

2.6.2. NEC_CoE402Halt

Stop/Halt Operation

C/C++ Syntax:

RTN_ERR NEC_CoE402Halt(CANAxis_T Axis, I32_T OnOff);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I32_T OnOff: 0: release，1:Stop or Halt

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

This API mainly control the “Controlword” bit8: Halt (CoE: 0x6040), is used to stop or halt the current motion operation. The behavior of halt operation is usually accordint to “Halt option code” (CoE: 0x605D) parameter. The detailed specifications need to refer to slave device’s user manual.

NEC_CoE402Halt() can be applied to all modes of operation. When the parameter “OnOff” is set to 1 means stop or halt the motion operation. When you want to restart the operation, you should set “OnOff” to zero to release the halt command.

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

Reference:

2.7. Homing mode

2.7.1. NEC_CoE402Home / NEC_CoE402HomeEx

Perform a homing operation

C/C++ Syntax:

RTN_ERR NEC_CoE402Home(CANAxis_T Axis, U32_T Option);
RTN_ERR NEC_CoE402HomeEx(CANAxis_T Axis, U32_T Option, I8_T Method, I32_T Offset, U32_T MaxVel, U32_T ZeroVel, U32_T Acc);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
U32_T Option: Options (in bit format). For two or more options assigning at the same time, use Boolean function of "OR".
OPT_WMC (0x00010000): Wail until Homing is finished
OPT_MTD (0x08000000) : ignore parameters from Method inputs
OPT_IMV(0x10000000): ignore parameters from MaxVel inputs
OPT_IAC (0x20000000) : ignore parameters from Acc inputs
OPT_IZV (0x40000000): ignore parameters from ZeroVel inputs
OPT_IOF (0x80000000) : ignore parameters from Offset inputs

I8_T Method: Homing method (CoE: 0x6098). Please refer to CiA402 servo manual.
I32_T Offset: Offset of original position (CoE: 0x607C)
U32_T MaxVel: Speed during search for switch (CoE: 0x6099:1)
U32_T ZeroVel: Speed during search for zero (CoE: 0x6099:2)
U32_T Acc: Homing Acceleration (CoE: 0x609A)

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

This function performs a homing operation using "Homing mode", therefore required to confirm whether CiA402 module supports Homing (HM) mode. Switch to HM mode automatically when calling this function. The Differences between Home and HomeEx is that have different input parameters:

NEC_CoE402Home() only start a homing operation, those related homing parameters such as home method "Method" (CoE: 0x6098), home offset "Offset" (CoE: 0x607C) etc. are based on current setting of slave. You can use *NEC_CoE402SetParameter*() to set each homing parameters individually.

NEC_CoE402HomeEx() Start a homing operation with all relevant parameters include home method "Method" (CoE: 0x6098), home offset "Offset" (CoE: 0x607C), home speed "MaxVel" (CoE: 0x6099:1), Search zero speed "ZeroVel" (CoE: 0x6099:2) and homing acceleration "Acc" (CoE: 0x609A).

When option "OPT_WMC " is enable, this function will be block until homing operation is done or error is occurred. The behavior is equivalent to calling *NEC_CoE402WaitHomeFinished*()

Option: OPT_MTD (0x08000000), OPT_IMV (0x10000000), OPT_IAC (0x20000000), OPT_IZV (0x40000000), OPT_IOF (0x80000000) only for *NEC_CoE402HomeEx*()， which means ignore the input parameter setting but using current slave setting.

Calling *NEC_CoE402Halt*() to stop the operation.

Attention!
This function is not allowed to be used in Callback function.
This function doesn't clear the start homing bit when "OPT_WMC" option isn't used.
It is suggested to clear the bit by calling NEC_CoE402ClearHomeStartBit after the homing procedure is finished.

Reference:
NEC_CoE402WaitHomeFinished(); NEC_CoE402Halt();
NEC_CoE402ClearHomeStartBit();

2.7.2. NEC_CoE402WaitHomeFinished

Wait until homing is finished.

C/C++ Syntax:

RTN_ERR NEC_CoE402WaitHomeFinished(CANAxis_T Axis);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

The function is a synchronous function, it will not return until Homing is finished. If the "Error Code" is returned, it means the operation is fault or being suspended.
This function clears the start homing bit before returning.

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

Reference:

NEC_CoE402Home();NEC_CoE402HomeEx();

2.7.3. NEC_CoE402ClearHomeStartBit

Clear the start homing bit.

C/C++ Syntax:

RTN_ERR NEC_CoE402ClearHomeStartBit (CANAxis_T Axis);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

Returned Values:

Error Code is returned.
“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

The function is used to clear the start homing bit. When motor is at moving state, the behavior of motor stop or keep moving is dependent on the drive’s design, after calling this function.

Reference:

NEC_CoE402Home();NEC_CoE402HomeEx();

2.7.4. NEC_CoE402CheckHomeStatus

Get the status of homing.

C/C++ Syntax:

RTN_ERR NEC_CoE402CheckHomeStatus (CANAxis_T Axis, U16_T *pStatus);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

U16_T *pStatus: pointer to the space for returned homing status.

Returned Values:

Error Code is returned.

"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

The function is used to get the status of homing. After executing the homing procedure, user can call this function to know which state present homing procedure is on at the moment.

There are six possible states will be returned:

```
#define HOMING_IN_PROGRESS                (0)
#define HOMING_TARGET_NOT_REACH           (1)
#define HOMING_COMPLETE                   (2)
#define HOMING_INTERRUPTED                (3)
#define HOMING_ERR_VEL_ZERO               (4)
#define HOMING_ERR_VEL_NON_ZERO           (5)
```

This function also clears the start homing bit, except the homing procedure is on "HOMING_IN_PROGRESS" state.

Reference:

NEC_CoE402Home();NEC_CoE402HomeEx();

2.8. Torque Control

2.8.1. NEC_CoE402SetTargetTorque / NEC_CoE402GetTargetTorque

Set / Get Target Torque (CoE: 0x6071, PT, CST)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetTargetTorque(CANAxis_T Axis, I16_T TargetTorque);
RTN_ERR NEC_CoE402GetTargetTorque(CANAxis_T Axis, I16_T *TargetTorque);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
I16_T TargetTorque: Set target torque (CoE: 0x6071)
I16_T * TargetTorque: Return target torque (CoE: 0x6071)

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

The function can be used in a Callback function. User must assign the object 0x607A as mapping object of RxPDO.
TargetTorque usually be used in "Profile Torque mode" (PT) and "Cyclic Sync Torque mode" (CST). Switch operation mode please refer to *NEC_CoE402SetOperationMode*()

The unit of torque is depended on the power of servo drive. 0.1% per tick In general, detail specifications please refer to the user manual of servo drive.

Reference:

NEC_CoE402SetOperationMode(); NEC_CoE402GetActualTorque()

2.8.2. NEC_CoE402GetActualTorque

Get Torque actual value (CoE: 0x6077)

C/C++ Syntax:

RTN_ERR NEC_CoE402GetActualTorque(CANAxis_T Axis, I16_T *TorqueActualValue);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().

I16_T *TorqueActualValue: Return toaque actual value (CoE: 0x6077)

Returned Values:

Error Code is returned.

“ECERR_SUCCESS” (0) is returned if function call is successful, while “Error Code” is returned when failed.

Usage:

he function can be used in a Callback function. User must assign the object 0x6077 as mapping object of TxPDO.

The unit of torque is depended on the power of servo drive. 0.1% per tick In general, detail specifications please refer to the user manual of servo drive.

Reference:

NEC_CoE402SetTargetTorque(); NEC_CoE402GetTargetTorque();

2.8.3. NEC_CoE402SetTorqueProfile

Set Target torque and slope (0x6071, 0x6087, PT)

C/C++ Syntax:

RTN_ERR NEC_CoE402SetTorqueProfile(CANAxis_T Axis, I16_T TargetTorque, U32_T TorqueSlope);

Parameters:

CANAxis_T Axis: Assign control axis ID. The axis ID value is returned by function call to NEC_CoE402GetAxisId().
I16_T TargetTorque: Set target torque (CoE: 0x6071)
U32_T TorqueSlope: Set torque slope (CoE: 0x6087)

Returned Values:

Error Code is returned.
"ECERR_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed.

Usage:

This function controls the torque under "Profile Torque mode", therefore required to confirm whether CiA402 module supports Profile Torque (PT) mode. Switch to PT mode automatically when calling this function.

This API set both target torque "TargetTorque" (CoE: 0x6071) and torque slope "TorqueSlope" (CoE: 0x6087) and access those objects by PDO or SDO is automatically determined.

The unit of torque is depended on the power of servo drive. 0.1% per tick In general, detail specifications please refer to the user manual of the CiA402 slaves.

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

Reference:

NEC_CoE402SetOperationMode(); NEC_CoE402SetTargetTorque();
NEC_CoE402GetActualTorque()