

NexECMRtx

EtherCAT Master for RTX

用戶手冊

Manual Rev.: 1.8

Revision Date: May, 2th, 2016

Part No:

Revise note:

Date	Ver	Description
2013/1/10	1.0	First version release
2013/12/10	1.1	Add NEC_RtSetProcessDataPtrSource function
2014/1/23	1.2	Modify chapter 2.3.1 How to config RTX TCP/IP
2014/6/16	1.3	Add Chapter 7. NexECM Library (Win32API)
2014/11/19	1.4	Ch2.1 Add support platform Ch2.3 Modify NIC driver setup Ch3 Enhance NEXCAT Utility description
2015/5/15	1.5	Typo fix
2015/8/10	1.6	Add Ch6.6.2 NEC_RtGetSlaveInfomation() API Add Ch6.6.3 NEC_RtGetSlaveState() API
2015/12/11	1.7	Add Ch6.8.1 NEC_RtGetConfiguredAddress() API Add Ch6.8.2 NEC_RtGetAliasAddress() API Add Ch6.8.3 NEC_RtNEC_GetSlaveCoeProfileNum() API Add Ch7.9.1 NEC_GetConfiguredAddress() API Add Ch7.9.2 NEC_GetAliasAddress() API Add Ch7.9.3 NEC_GetSlaveCoeProfileNum () API
2016/5/6	1.8	Add Ch5.8.2 CoE – Emergency Add Ch6.7.7 NEC_RtStartGetODListCount Add Ch6.7.8 NEC_RtStartGetODList Add Ch6.7.9 NEC_RtStartGetObjDesc Add Ch6.7.10 NEC_RtStartGetEntryDesc Add Ch6.7.11 NEC_RtEmgDataCount Add Ch6.7.12 NEC_RtEmgData Add Ch7.7.5 NEC_GetODListCount Add Ch7.7.6 NEC_GetODList Add Ch7.7.7 NEC_GetObjDesc Add Ch7.7.8 NEC_GetEntryDesc Add Ch7.7.9 NEC_EmgDataCount Add Ch7.7.10 NEC_EmgData

目錄

NexECMRtx	1
Revise note:	2
目錄	i
1. NexECMRtx 介紹	1
1.1. NexECMRtx 功能特點	2
2. NexECMRtx 安裝說明	4
2.1. 硬體需求	4
2.2. 軟體需求	4
2.3. NexECMRtx 安裝、開發流程	4
2.3.1. 安裝 RTX NIC 驅動程式	5
2.3.2. 設定網路卡"C:\RtxEcNic.ini"	11
2.3.3. 開發 EtherCAT 主站 RTX 應用程式	14
3. EtherCAT 公用程式說明	19
3.1. NexCAT 簡介	19
3.1.1. 軟體需求	19
3.1.2. NexCAT 操作流程說明	20
3.1.3. ESI 和 ENI 檔案	22
3.1.4. NexCAT 主頁面操作	23
3.1.5. 從站模組參數設定頁面 (Set Slave Parameters)	25
3.1.6. EC-Master 參數設定頁面 (Set Master Parameters)	30
3.1.7. ESI 檔案管理表單 (ESI List)	32
3.1.8. DIO 操作頁面	33
3.1.9. CoE-SDO 存取操作頁面	35
3.1.10. Process Image 參數存取操作頁面	36
3.1.11. Network Quality Monitor 網路通訊品質測試頁面	37

3.2. NexECMRtxStarup 啟動程式說明.....	39
4. EtherCAT 技術簡介.....	41
4.1. EtherCAT 通訊協定概述.....	42
4.2. 網路拓樸.....	43
4.3. 高速效能.....	44
4.4. 網路同步.....	45
5. 編程原理.....	47
5.1. 基本編程框架.....	47
5.2. ENI 載入.....	48
5.3. 初始化 NexECMRtx.....	49
5.4. 啟動主站通訊.....	50
5.5. Callback 函式.....	51
5.5.1. 註冊回呼函數(Callback functions).....	51
5.5.2. 週期回呼函數(Cyclic-Callback function).....	51
5.5.3. 事件回呼函數(Event-Callback function).....	53
5.5.4. 錯誤事件回呼函數(Error-Callback function).....	53
5.6. EtherCAT 狀態機.....	54
5.6.1. ESM 狀態變更.....	54
5.7. Process Data 存取.....	56
5.7.1. ProcessData 運作機制.....	56
5.7.2. ProcessData 資料內容.....	57
5.8. Mailbox 通訊.....	59
5.8.1. CoE -SDO 通訊.....	59
5.8.2. CoE -Emergency.....	60
6. NexECMRtx 程式庫.....	61
6.1. RTX API 總覽.....	61
6.2. 初始化相關函式.....	63

6.2.1. NEC_RtGetVersion	63
6.2.2. NEC_RtRetVer	64
6.2.3. NEC_RtInitMaster	65
6.2.4. NEC_RtCloseMaster	66
6.2.5. NEC_RtSetParameter	67
6.2.6. NEC_RtGetParameter	67
6.2.7. NEC_RtRegisterClient.....	69
6.2.8. NEC_RtUnregisterClient.....	71
6.2.9. NEC_RtGetProcessDataPtr.....	72
6.2.10. NEC_RtSetProcessDataPtrSource	73
6.3. EC-Master 控制相關函式	75
6.3.1. NEC_RtStartMaster	75
6.3.2. NEC_RtStopMaster	76
6.3.3. NEC_RtStopMasterCb	77
6.3.4. NEC_RtWaitMasterStop.....	78
6.3.5. NEC_RtGetMasterState	79
6.3.6. NEC_RtSetMasterState	79
6.3.7. NEC_RtChangeStateToOP	81
6.3.8. NEC_RtSetMasterStateWait	82
6.3.9. NEC_RtGetStateError.....	83
6.4. Process data 存取相關函式.....	84
6.4.1. NEC_RtSetProcessDataOutput.....	84
6.4.2. NEC_RtGetProcessDataOutput.....	84
6.4.3. NEC_RtGetProcessDataInput.....	84
6.4.4. NEC_RtSetSlaveProcessDataOutput	85
6.4.5. NEC_RtGetSlaveProcessDataOutput	85
6.4.6. NEC_RtGetSlaveProcessDataInput	85

6.5. Callback 函式.....	87
6.5.1. *NEC_RtCyclicCallback.....	87
6.5.2. *NEC_RtEventCallback.....	88
6.5.3. *NEC_RtErrorCallback.....	89
6.6. ENI 相關函式.....	90
6.6.1. NEC_RtGetSlaveCount	90
6.6.2. NEC_RtGetSlaveInfomation	91
6.6.3. NEC_RtGetSlaveState.....	93
6.7. CoE 通訊相關函式.....	94
6.7.1. NEC_RtStartSDODownload	94
6.7.2. NEC_RtStartSDOUpload.....	94
6.7.3. NEC_RtSDODownload	96
6.7.4. NEC_RtSDOUpload.....	96
6.7.5. NEC_RtSDODownloadEx	96
6.7.6. NEC_RtSDOUploadEx	96
6.7.7. NEC_RtStartGetODListCount	97
6.7.8. NEC_RtStartGetODList	99
6.7.9. NEC_RtStartGetObjDesc	101
6.7.10. NEC_RtStartGetEntryDesc	103
6.7.11. NEC_RtGetEmgDataCount	105
6.7.12. NEC_RtGetEmgData	106
6.8. 存取從站模組硬體資訊相關函式	108
6.8.1. NEC_RtGetConfiguredAddress.....	108
6.8.2. NEC_RtGetAliasAddress.....	109
6.8.3. NEC_RtGetSlaveCoeProfileNum	111
7. NexECM 程式庫 (Win32 APIs)	112
7.1. Win32 API 總覽	113

7.2. RTX 系統相關控制 API.....	115
7.2.1. NEC_LoadRtxApp	115
7.2.2. NEC_KillRtxApp	116
7.3. NexECM 初始化相關函式	117
7.3.1. NEC_GetVersion.....	117
7.3.2. NEC_StartDriver	118
7.3.3. NEC_StopDriver	119
7.4. NexECMRtx Runtime 控制相關函式.....	120
7.4.1. NEC_GetRtMasterId.....	120
7.4.2. NEC_GetRtMasterVersion.....	121
7.4.3. NEC_GetRtMasterPorcessId	122
7.4.4. NEC_ResetEcMaster.....	123
7.4.5. NEC_LoadNetworkConfig.....	124
7.4.6. NEC_StartNetwork.....	125
7.4.7. NEC_StartNetworkEx	126
7.4.8. NEC_StopNetwork	127
7.4.9. NEC_SetParameter	128
7.4.10. NEC_GetParameter.....	128
7.5. 網路狀態存取相關函式	130
7.5.1. NEC_GetSlaveCount.....	130
7.5.2. NEC_GetNetworkState	131
7.5.3. NEC_GetSlaveState	132
7.5.4. NEC_GetStateError	133
7.5.5. NEC_GetErrorMsg.....	134
7.6. DIO 控制相關函式	135
7.6.1. NEC_SetDo	135
7.6.2. NEC_GetDo	136

7.6.3. NEC_GetDi.....	137
7.7. CoE 通訊相關函式.....	138
7.7.1. NEC_SDODownloadEx.....	138
7.7.2. NEC_SDOUploadEx	138
7.7.3. NEC_SDODownload	138
7.7.4. NEC_SDOUpload	138
7.7.5. NEC_GetODListCount.....	140
7.7.6. NEC_GetODList	142
7.7.7. NEC_GetObjDesc.....	144
7.7.8. NEC_GetEntryDesc.....	146
7.7.9. NEC_GetEmgDataCount.....	148
7.7.10. NEC_GetEmgData	149
7.8. Process data 存取相關函式.....	151
7.8.1. NEC_RWProcessImage.....	151
7.8.2. NEC_GetProcessImageSize	152
7.8.3. NEC_RWSlaveProcessImage	153
7.9. 存取從站模組硬體資訊相關函式	154
7.9.1. NEC_GetConfiguredAddress	154
7.9.2. NEC_GetAliasAddress	155
7.9.3. NEC_GetSlaveCoeProfileNum	157

1. NexECMRtx 介紹

NexECMRtx 是一套建構於 RTX (Microsoft Windows 延伸即時子作業系統) 的 EtherCAT 主站(EC-Master) 通訊層解決方案，透過此通訊層和 EtherCAT 從站 (EC-Slave devices)進行即時通訊作業。NexECMRtx 提供豐富高階 C / C++ 應用程式開發介面 (API) 來編程操作。

除 EtherCAT 主站通訊層外，亦可透過圖形化 EtherCAT 公用程式 – NexCAT 進行 EtherCAT 相關工業應用開發，其功能包含：

1. EtherCAT 網路裝置掃描
2. 匯入 ESI 檔案，輸出 ENI 檔案
3. 從站模組(Slaves)網路參數設定
4. EtherCAT 通訊品質監控測試
5. 從站模組(Slaves)功能性測試操作

其詳細功能介紹請參考第三章。

下圖為 NexECMRtx 主站通訊解決方案系統方塊圖，虛線的方塊代表用戶的應用程式開發位置，箭頭代表存取溝通的對象。

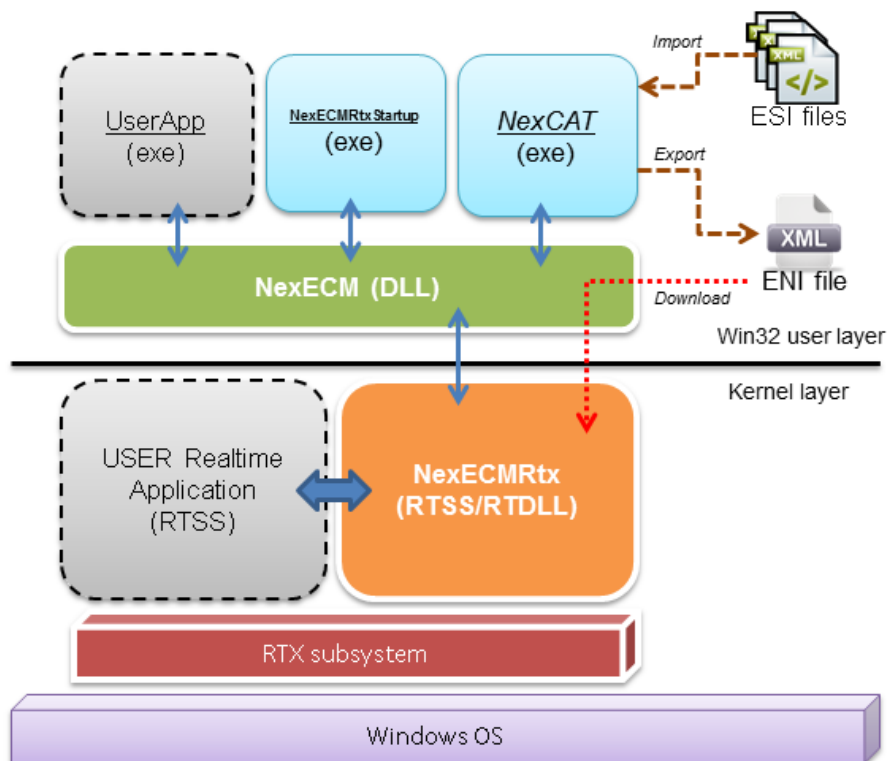


圖 1.1: NEXCOM EtherCAT master solution 系統方塊圖

系統方塊圖個模組功能簡述如下表:

軟體模組	說明	參考章節
運作位置:RTX 子系統		
NexECMRtx.rtss	主站通訊 (EC-Master)核心 RTX 程式庫	CH 5, CH 6
運作位置:Win32 用戶層(UserMode)		
NexCAT.exe	EC-Master 公用程式	CH 3.1
NexECMRtxStartup.exe	EC-Master 啟動協助工具程式	CH 3.2
NexECM.dll	EC-Master Win32 溝通介面程式庫	CH7

1.1. NexECMRtx 功能特點

根據 EtherCAT 標準文件編號 ETG.1500，NexECMRtx 支援 EtherCAT 主站功能如下表所列:

V: 已支援, △: 即將支援

Feature name	Short description	NexECMRtx
Basic Features		
Service Commands	Support of all commands	V
IRQ field in datagram	Use IRQ information from Slave in datagram header	V
Slaves with Device Emulation	Support Slaves with and without application controller	V
EtherCAT State Machine	Support of ESM special behavior	V
Error Handling	Checking of network or slave errors, e.g. Working Counter	V
Process Data Exchange		
Cyclic PDO	Cyclic process data exchange	V
Network Configuration		
Reading ENI	Network Configuration taken from ENI file	V
Compare Network configuration	Compare configured and existing network configuration during boot-up	V
Explicit Device Identification	Identification used for Hot Connect and prevention against cable swapping	V
Station Alias Addressing	Support configured station alias in slave, i.e. enable 2nd Address and use it	V
Access to	Support routines to access EEPROM via ESC	V

EEPROM	register	
Mailbox Support		
Support Mailbox	Main functionality for mailbox transfer	V
Mailbox polling	Polling Mailbox state in slaves	V
CAN application layer over EtherCAT (CoE)		
SDO Up/Download	Normal and expedited transfer	V
Complete Access	Transfer the entire object (with all sub-indices) at Once	V
SDO Info service	Services to read object dictionary	△
Emergency Message	Receive Emergency messages	△
Ethernet over EtherCAT (EoE)		
EoE	Ethernet over EtherCAT	△
File over EtherCAT (FoE)		
FoE	File over EtherCAT	△
Servo over EtherCAT (SoE)		
SoE	Servo over EtherCAT	△
Distributed Clocks		
DC	Support of Distributed Clock	V

2. NexECMRtx 安裝說明

2.1. 硬體需求

標準 X86 PC / IPC，並具備一個以上乙太網路(Ethernet)通訊埠，NexECMRtx 已經在 Nexcom 平台進行驗證及最佳化，請參考 “NexECMRtx hardware support list.pdf” 文件，此文件存放在 NexECMRtx 安裝目錄中

2.2. 軟體需求

依照上述硬體規格，在系統上須預先安裝：

1. 作業系統要求 Microsoft Windows XP SP3 and Window 7 32 位元作業系統
2. IntervalZero RTX 2012
3. Microsoft Visual studio 2005 ~2012
4. NEXCOM NexECMRtx 安裝檔案
5. .Net framework 4.0 (供 NexCAT.exe 公用程式)
6. Visual Basic Power Packs 3.0 (供 NexCAT.exe 公用程式)

- 安裝 RTX 2012，你可以在官方網站中下載 RTX 2012，90 天試用版^(*)：

<http://www.intervalzero.com/rtx-downloads/>

(*) 購買正式版或申請試用版序號請洽 RTX 大中華區總代理-新漢電腦

Tel : +886-2-8226-7786 # ICS 業務處

- Net framework 4.0 官方免費下載：

<http://www.microsoft.com/zh-tw/download/details.aspx?id=17718>

- Visual Basic Power Packs 3.0 官方免費下載：

<http://download.microsoft.com/download/1/2/A/12AA9B28-4F67-42C3-9319-684E8AD6F0AE/VisualBasicPowerPacks3Setup.exe>

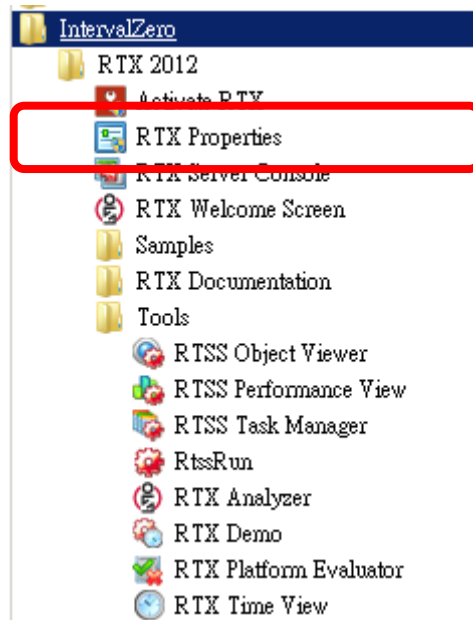
2.3. NexECMRtx 安裝、開發流程

當您取得 IPC 硬體平台，並安裝好 Windows 作業系統和 RTX 子系統，請依照下列步驟完成軟體安裝：

1. 安裝 RTX NIC 驅動程式
2. 設定網路卡“C:\RtxEcNic.ini”
3. 使用 NexCAT 設定 EtherCAT 網路(請參考第三章)
4. 開發 EtherCAT 主站 RTX 應用程式

2.3.1. 安裝 RTX NIC 驅動程式

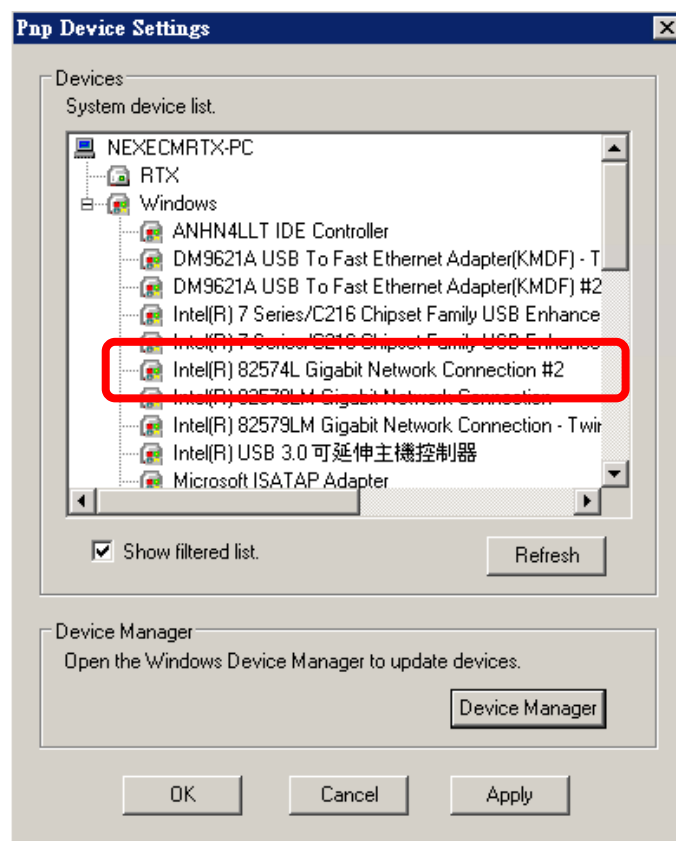
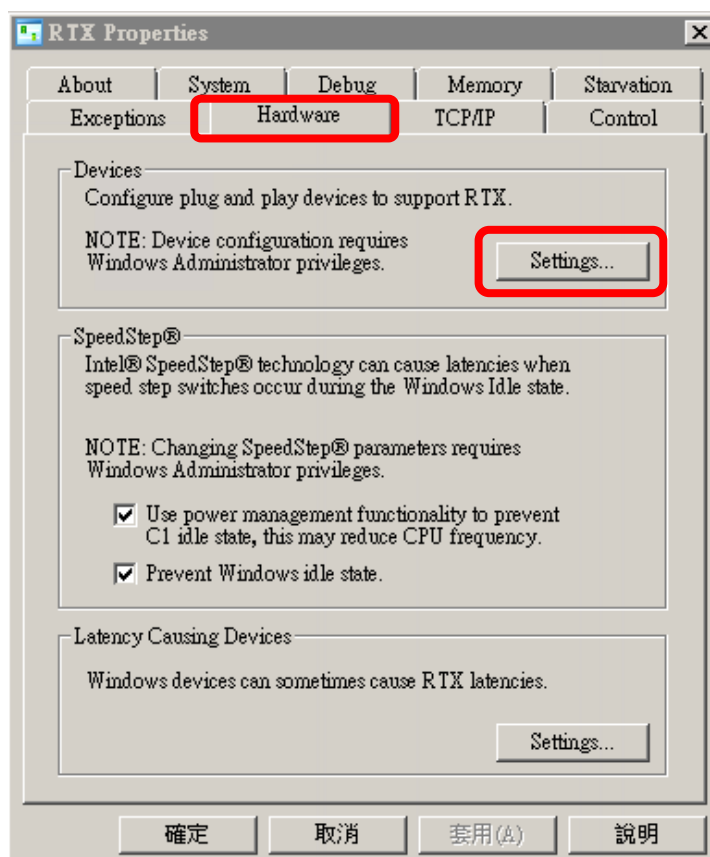
在開始功能表中(如下圖)或桌面上  打開 RTX 工具程式:“RTX properties” 如下圖:

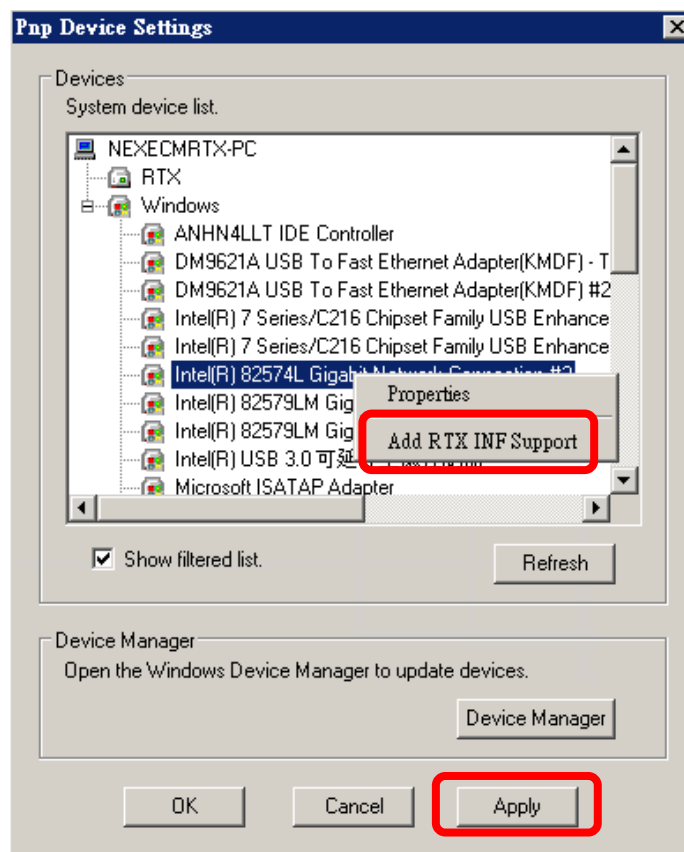


RTX properties tool

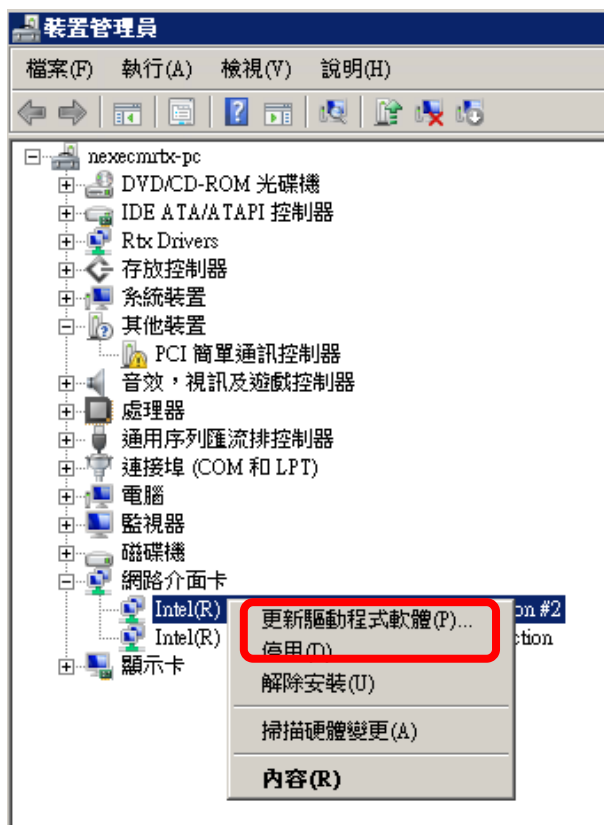
在 “Hardware” 頁面(如下圖)

1. 點選 Device/Setting 按鈕彈出“Pnp Device Settings”頁面
2. 在網路卡上右鍵選擇 “Add RTX INF Support” 並點選 “Apply”

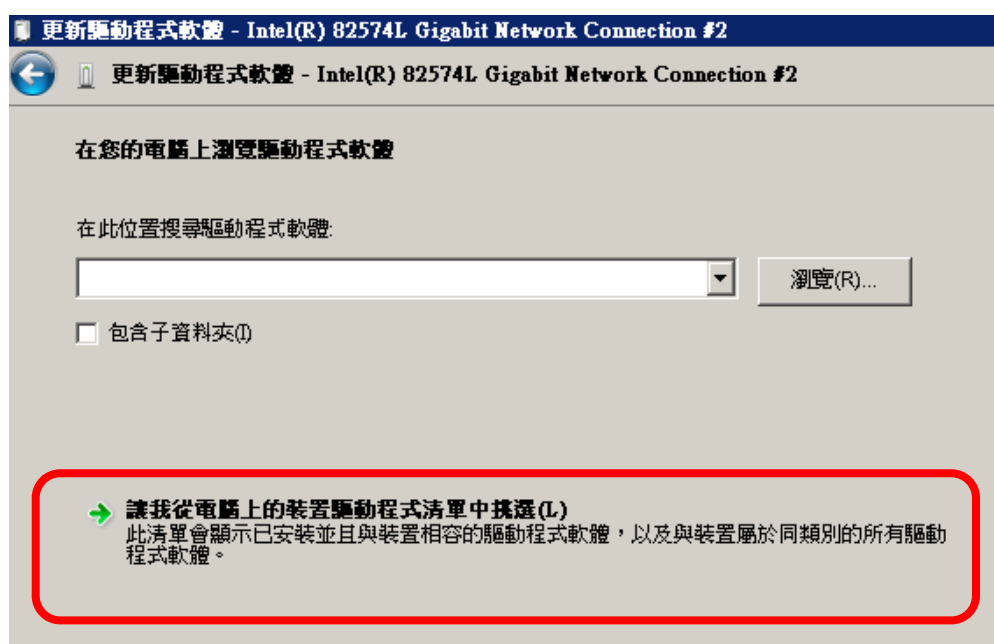
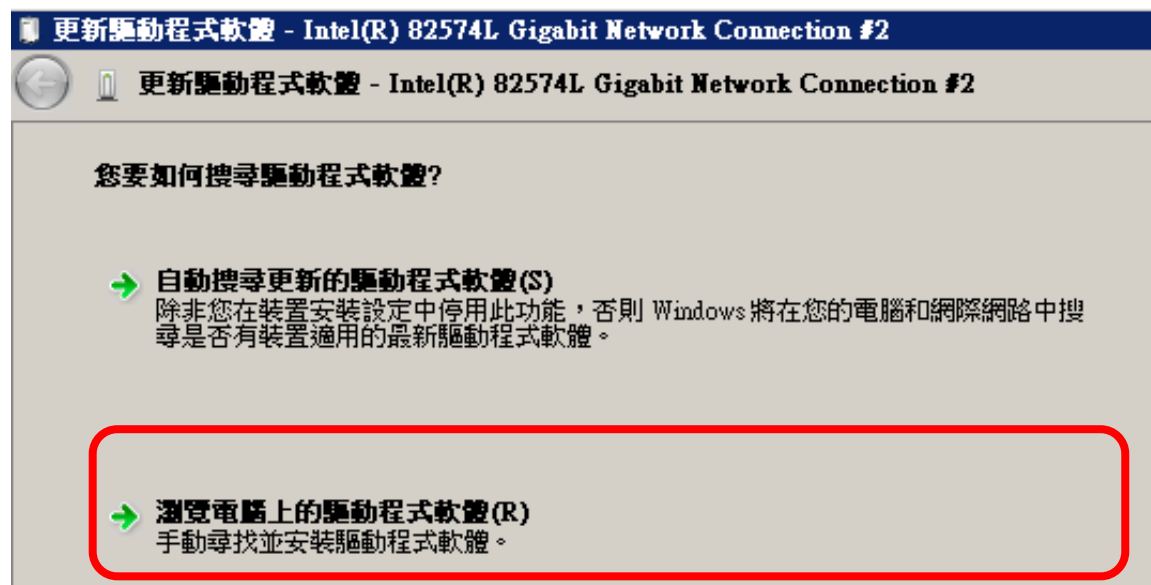


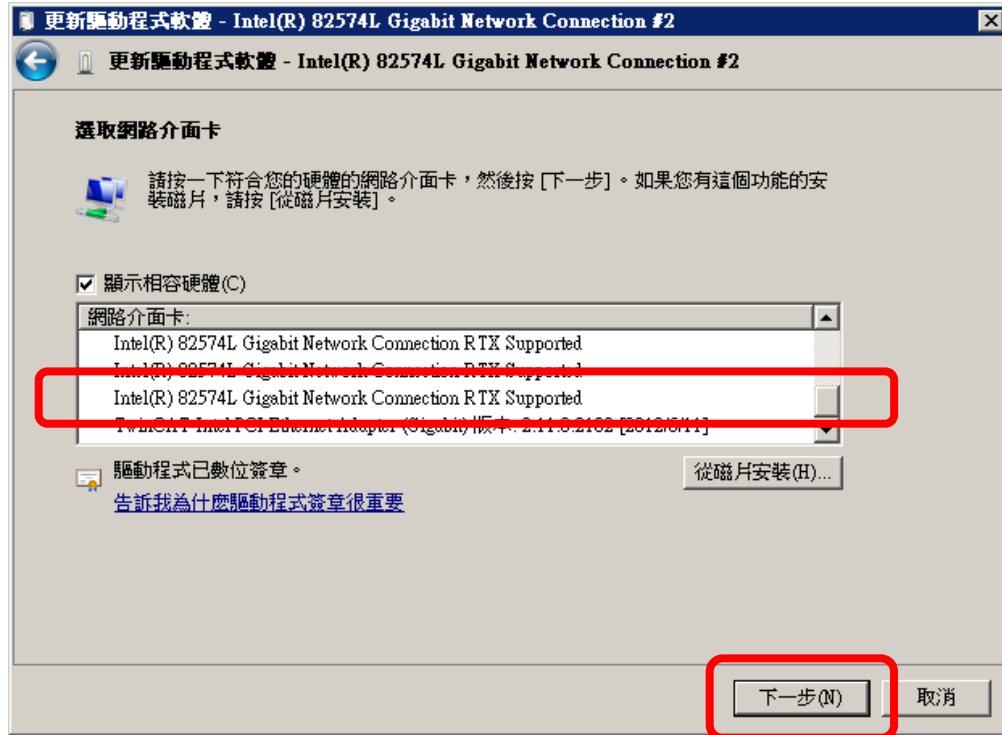


開啟檔案管理員，右鍵點選網路卡“更新驅動程式軟體”

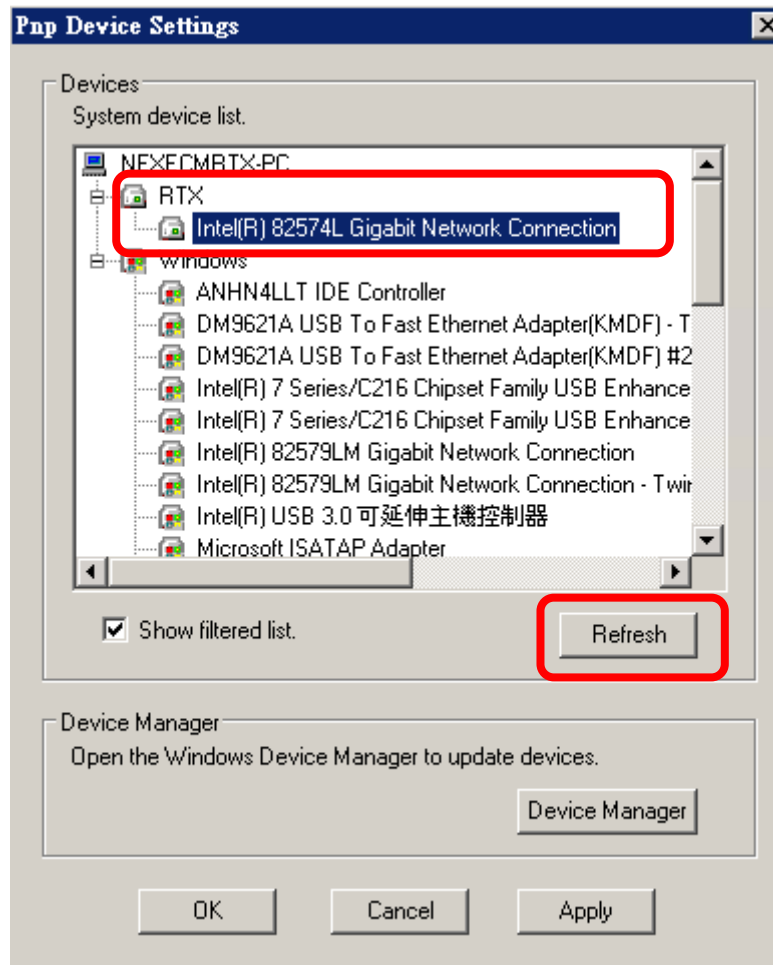


以下列步驟選取 RTX supported 網路卡驅動程式





安裝完成後可在“Pnp Device Settings”頁面確認 RTX 驅動程式是否正常安裝



2.3.2. 設定網路卡“C:\RtxEcNic.ini”

使用記事本開啟“C:\RtxEcNic.ini”檔案如下



```

[Info]
NumOfInterfaces = 2

[Nic0]
BusNum=2
DevNum=0
FunNum=0

[Nic1]
BusNum=5
DevNum=0
FunNum=0
    
```

NumOfInterfaces: EtherCAT 網路卡數量

[Nic0]: 第一張網卡的位置

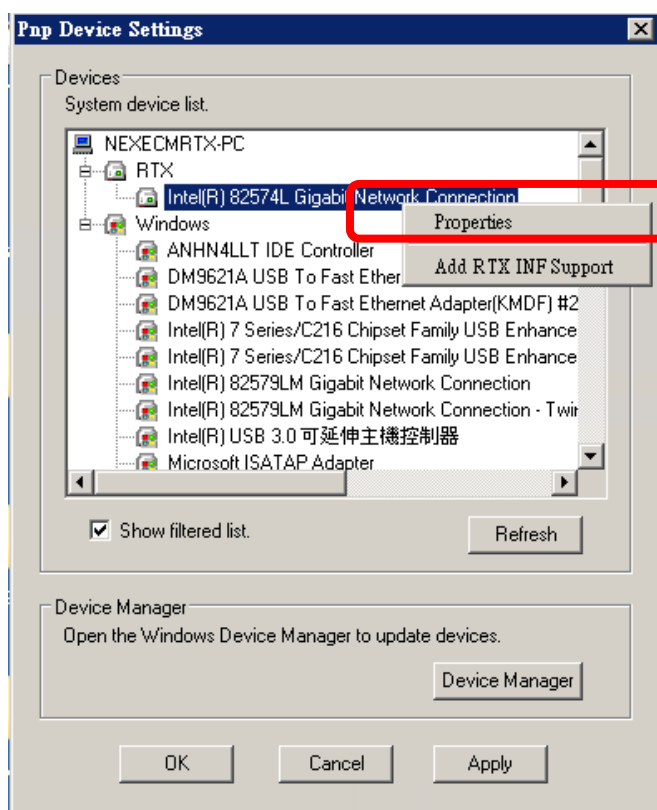
[Nic1]: 第二張網卡的位置

...以此類推

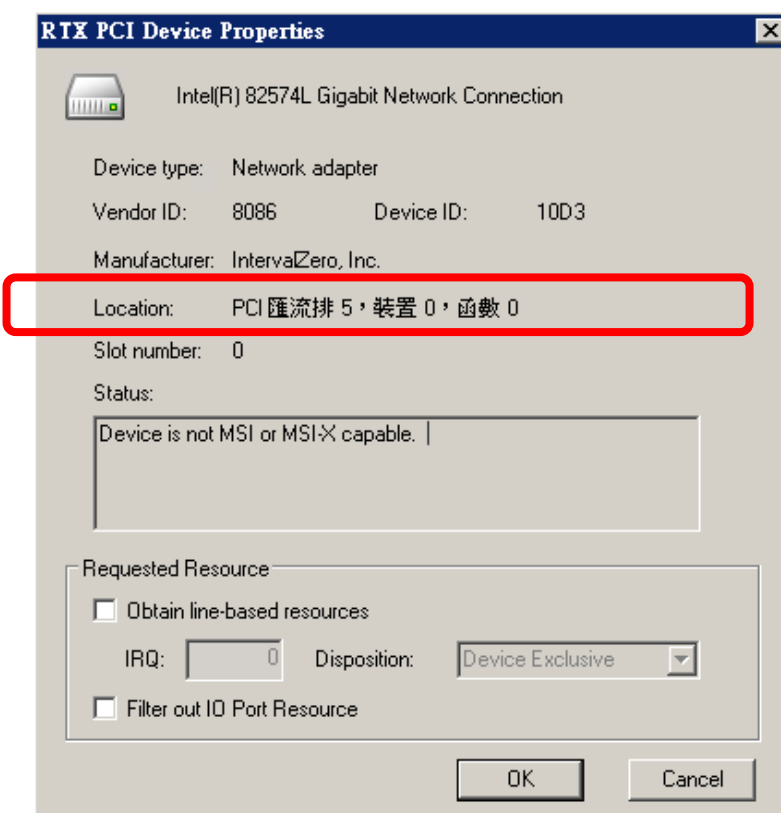
BusNum / DevNum/ FunNum PCI 位置資訊可由下述步驟查詢:

在“Pnp Device Settings”頁面，右鍵網卡選擇“Properties”即彈出“RTX PCI Device Properties”頁面，將“Location”位置紀錄下來填入 INI 檔案中。

特別注意: 此 INI 檔案必須置於“C:\RtxEcNic.ini”路徑



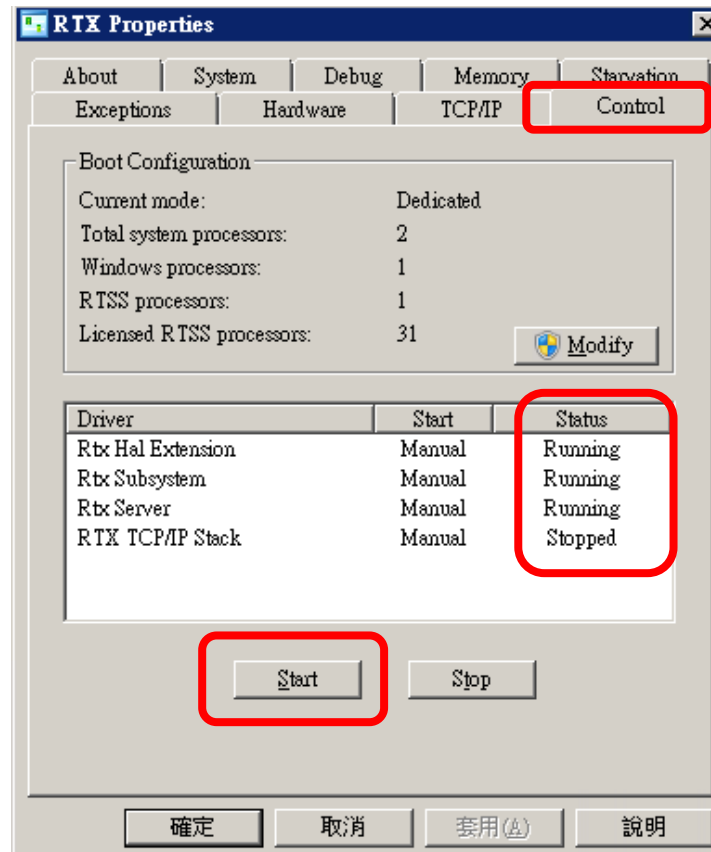
在 “Pnp Device Settings” 頁面，右鍵網卡選擇 “Properties”



將“Location” 位置紀錄下來填入 INI 檔案中

啟動 RTX Subsystem

打開 RTX 工具程式: “RTX properties” ，於“Control” 頁面(如下圖) ，點選“Start” button 啟動 RTX system ， RTX 的 Driver 狀態會切換到 “Running”狀態。

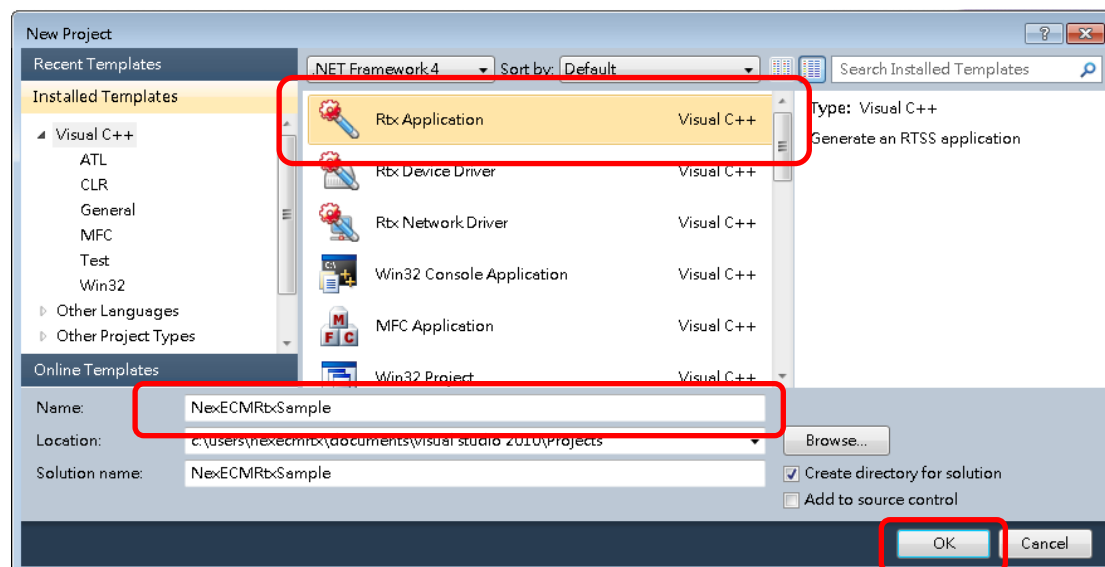


當上述步驟完成，NexECMRtx 的前置安裝步驟已經全部完成，而上述步驟只需第一次安裝 NexECMRtx 完成。後續無須再進行設定。

2.3.3. 開發 EtherCAT 主站 RTX 應用程式

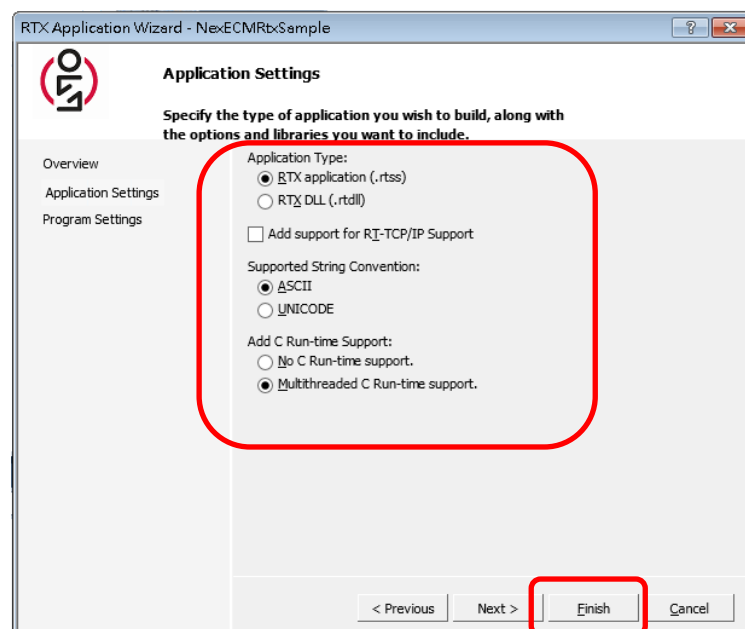
下面步驟說明如何使用 Visual studio 開發 EtherCAT 主站 RTX 應用程式，以 Visual studio 2010 為例：

1. 新增一專案(File/New)

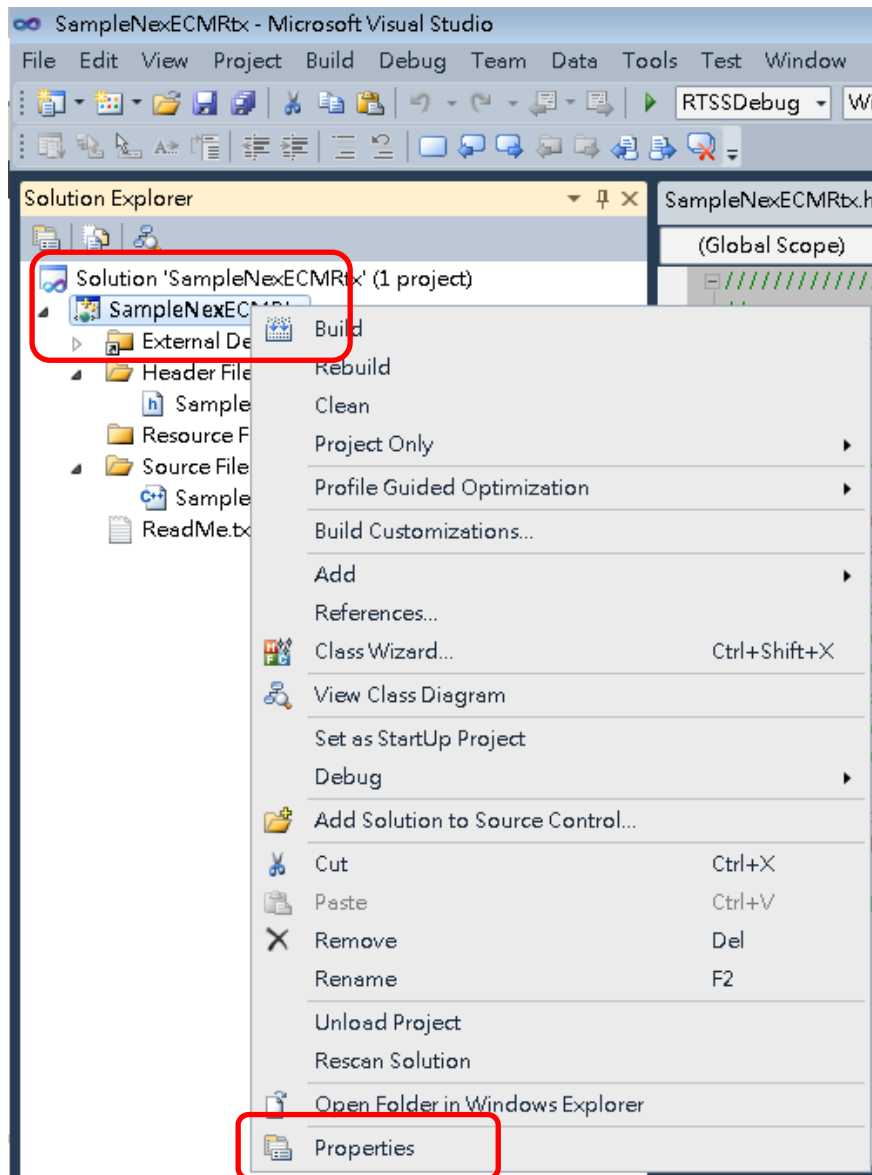


選擇 RTX 專案，輸入專案名稱。

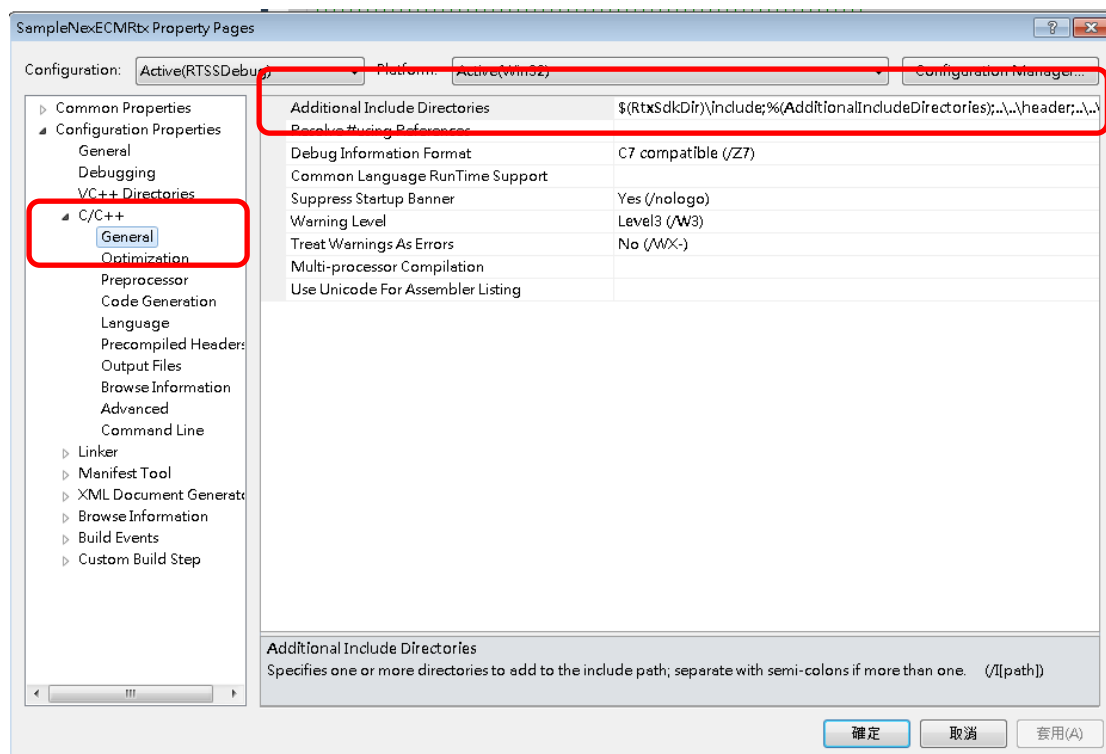
2. 設定 RTX 應用程式專案選項，如下圖



3. 設定使用 NexECMRtx 函式庫，在 Solution Explorer 中專案上右鍵點選 Properties

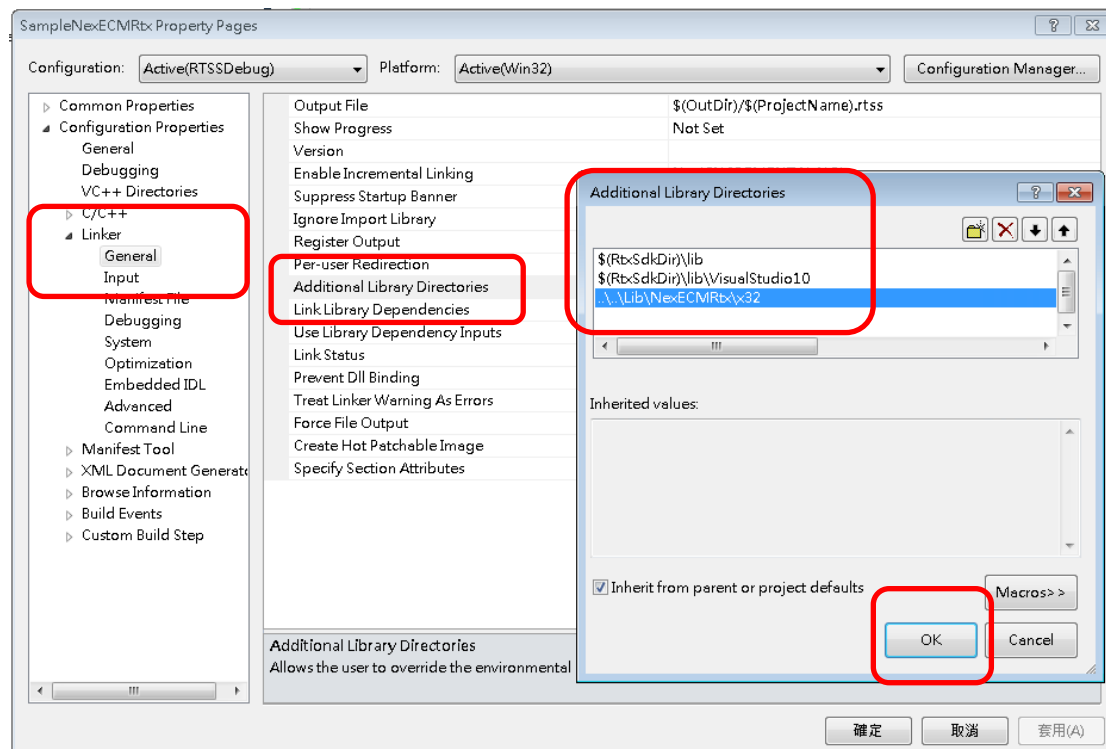


C/C++ \ General 中設定 NexECMRtx 的標頭檔路徑，如下圖：

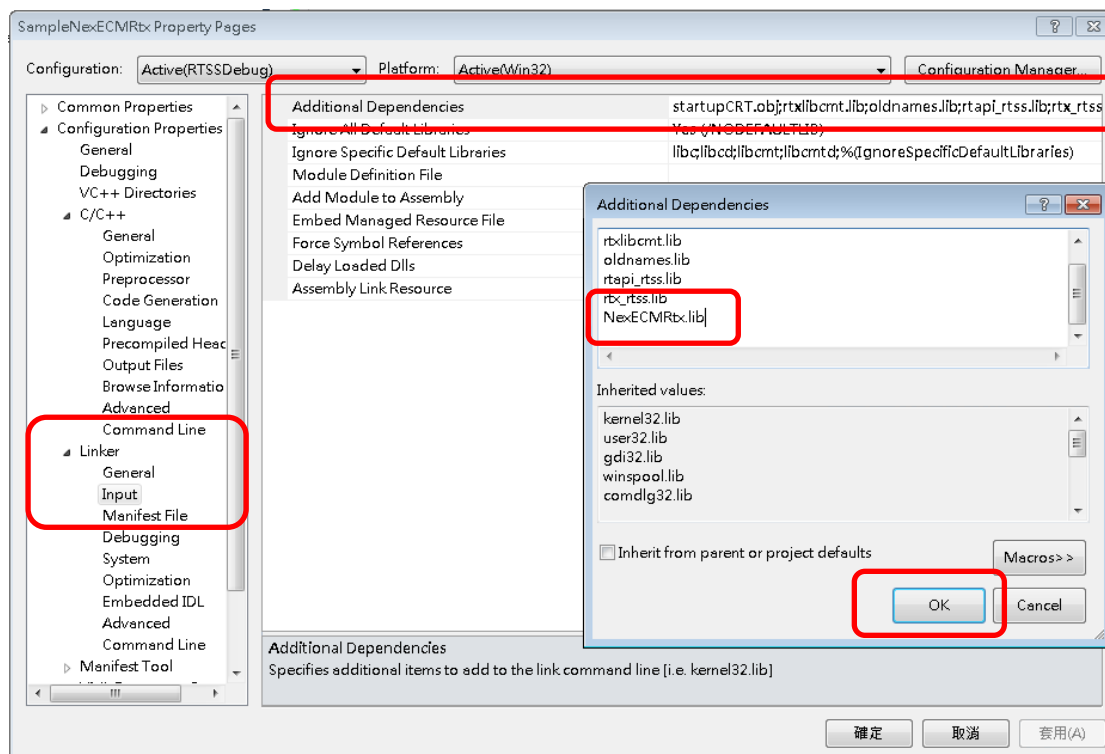


NexECMRtx 的標頭檔置於安裝目錄中。

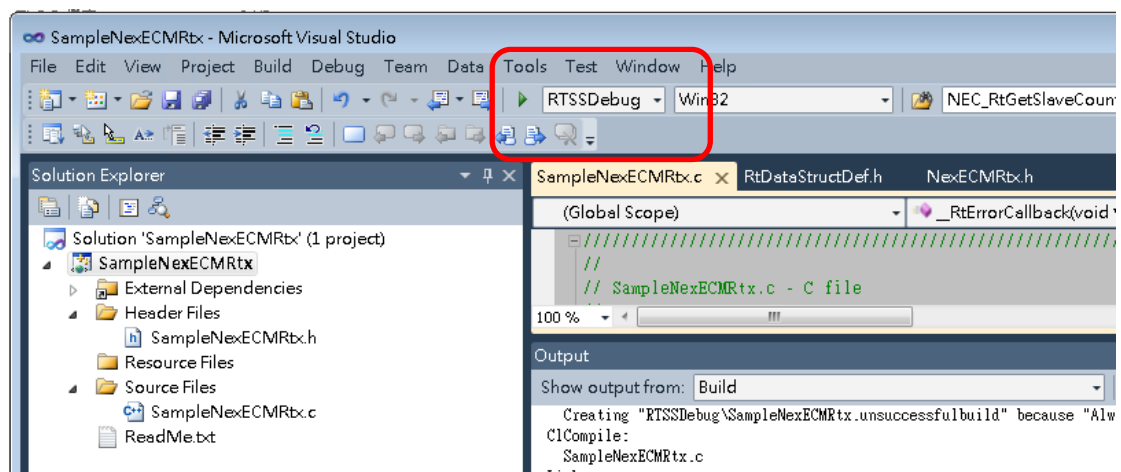
Linker\General 設定 NexECMRtx.lib 的路徑



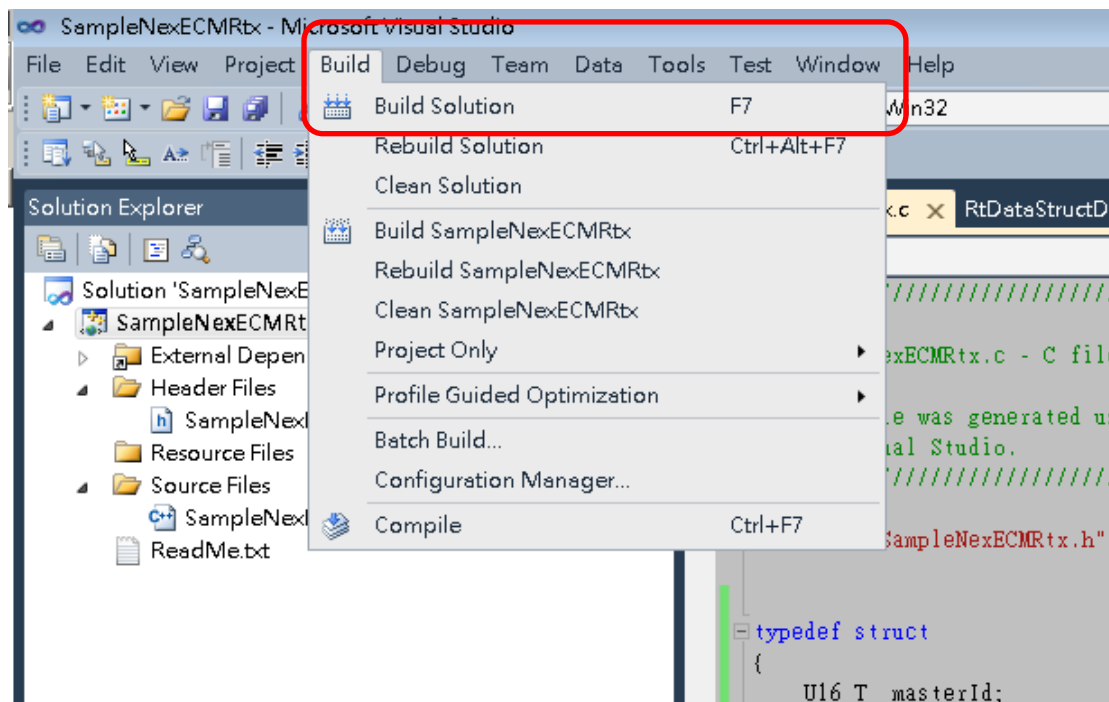
在 Linker\Input 中設定引入的函式庫 NexECMRtx.lib



開發程式，編譯程式碼，注意要選擇 RTSSDebug 或 RTSSRelease



編譯 RTX 應用程式



3. EtherCAT 公用程式說明

3.1. NexCAT 簡介

透過公用程式 NexCAT，可完成下列事項。

1. EtherCAT 網路裝置掃描
2. 匯入 ESI 檔案，輸出 ENI 檔案
3. CoE 從站模組(Slaves) PDO 映射(Mapping)
4. ProcessData 存取
5. CoE 從站模組(Slaves) SDO 通訊測試
6. EtherCAT 通訊品質監控測試
7. 從站模組(Slaves)功能性測試操作

3.1.1. 軟體需求

使用前請先確定 MS-Windows 是否已安裝下列原件:

- Net framework 4.0 微軟官方免費下載:

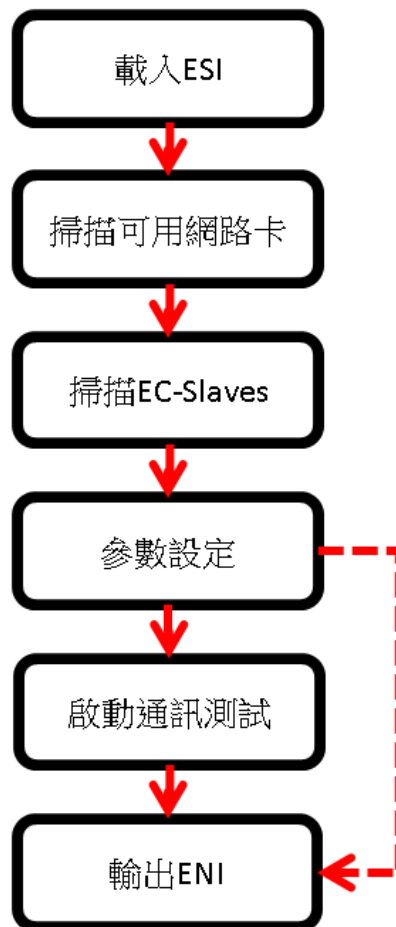
<http://www.microsoft.com/zh-tw/download/details.aspx?id=17718>

- Visual Basic Power Packs 3.0 微軟官方免費下載:

<http://download.microsoft.com/download/1/2/A/12AA9B28-4F67-42C3-9319-684E8AD6F0AE/VisualBasicPowerPacks3Setup.exe>

3.1.2. NexCAT 操作流程說明

NexCAT 的基本操作流程如下圖:



- **載入 ESI:**

由 NexCAT 於程式啟動時完成匯入所有的檔案，其路徑為 NexCAT.exe 安裝路徑下的“ESI”子目錄，其附檔名為*.xml。

- **掃描可用網路卡:**

由 NexCAT 完成，程式可以自動判斷在 Windows RTX 環境下，自動找到對應之網路卡之網路連接埠。關於 RTX 網路卡的安裝設定請參考 Ch2.3 章節。

- **掃描 EC-Slaves:**

NexCAT 可以自動掃描使用者所選擇的網路埠，找到串連之 Slave 模組，當串連之模組無法由對應之 ESI 檔內容找到對應之內容(比對 Vendor ID and Device ID)，該模組名稱顯示為未知 “Unknown”。將滑鼠游標移置“Unknow”裝置可以顯示該模組的硬體資訊(VendorID, DeviceID and RevisionNumber)

- **參數設定:**

NexCAT 依照 ESI 內容自動產生對應之 PDO 與 ProcessData 記憶體規劃，並輸出 ENI 檔案。使用者可以利用內建之 PDO mapping 編輯器來進行規劃。

- **啟動通訊測試:**

規劃完成之後可以直接啟動所有的 Slave 模組，其狀態由初始 INIT 至操作 Operation 狀態。若其中有一個模組無法成功轉至 Operation，主頁面於區域 4、5 顯示狀態及訊息(參考下頁圖)。

- **輸出 ENI:**

當測試完成各模組之動作皆正常時，使用者可以使用 Export ENI 檔案的功能，輸出至儲存裝置，倘若使用者使用啟動網路(Start Network)功能，系統自動輸出當下的設定及網路拓撲至 ENI 檔。

(其預設路徑為 : system root:\ENI_NexCAT_Export.xml)

3.1.3. ESI 和 ENI 檔案

ESI (EtherCAT Slave Information)檔為硬體描述檔案，其內容描述了硬體的資訊及相關的參數及記憶體配置，其中包含該裝置所支援的通訊協定。由提供 EtherCAT slave 模組的硬體廠商提供。

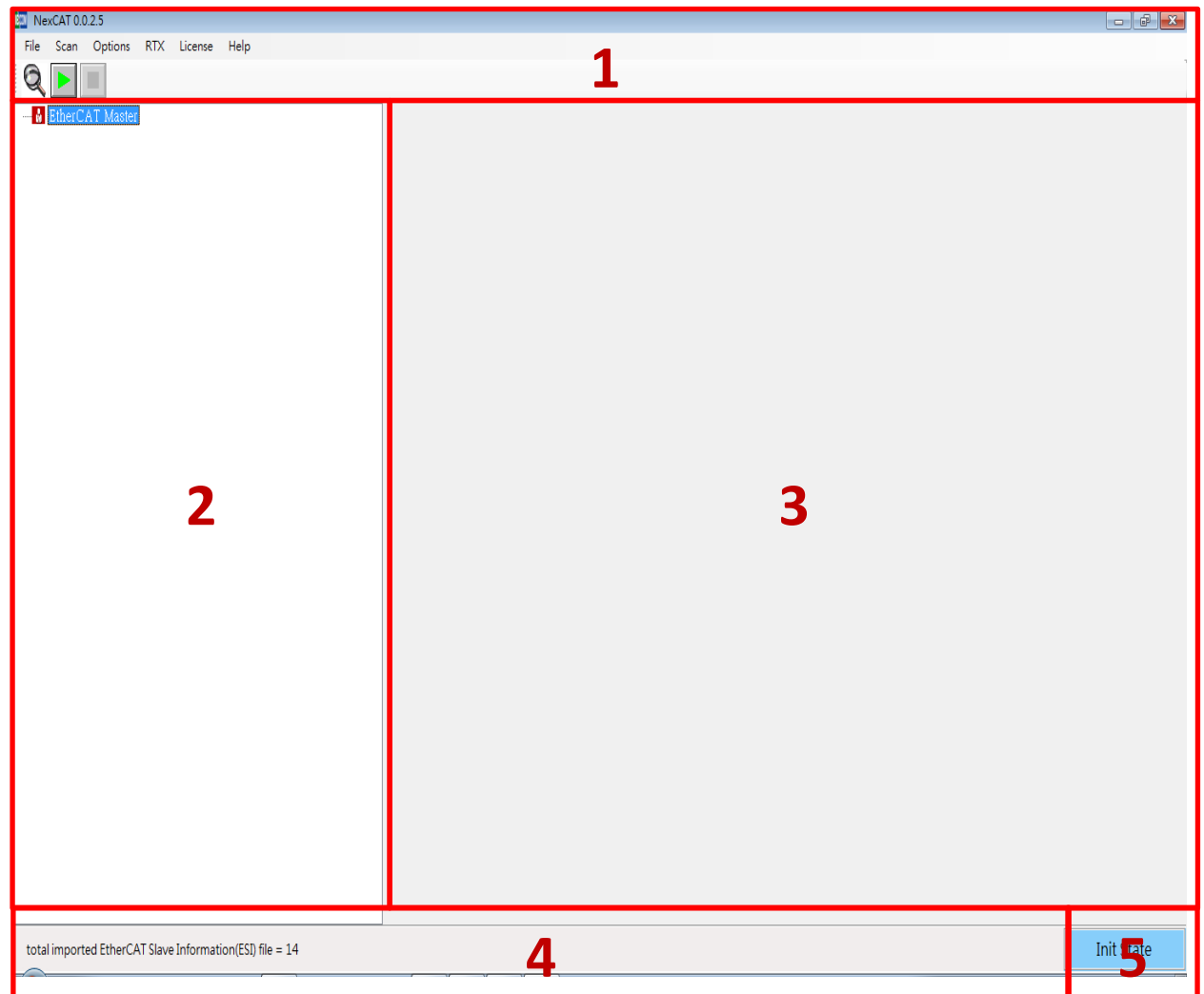
*EtherCAT 公用程式對於模組之間的預設配置，係根據 ESI 的內容而來，因此強烈建議請勿更動 ESI 的內容，並確保所取得的 ESI 檔案為該硬體的最新版本。

ENI(EtherCAT Network Information)為記載一個特定的 EtherCAT 網路拓撲及該網路拓撲下的 Process Data 記憶體配置和 EC-Slave 網路組態的描述檔案，使用者在完成 EC-Slave 模組串連配線之後，必須透過 NexCAT 公用程式，產生出 ENI 檔案之後，EC-Master 便可依據其檔案設定內容運行。




*強烈建議請勿更動 ENI 的內容，若網路配置有異動請另外再輸出更動後的 ENI 檔案。

3.1.4. NexCAT 主頁面操作

主頁面共分成下列 5 個操作區域並分開說明於下列陳述：



- 區域 1:
表單的正上方顯示軟體版本資訊。

Icon	說明
	Scan NIC:自動尋找可以用網路連接埠，並顯示於表單上
	Start Network: 啟動通訊並輸出 ENI 檔案到預設的路徑(C:\)
	Stop Network:停止通訊自動停止所有 ECAT-Slaves

- 區域 2:
顯示整個網路拓撲(Topology)。完成掃描模組(Scan Slave)後所有線上的 EC-Slave 顯示於此。倘若 EC-Slave 模組未能於所有 ESI 檔內找到對應的內容，該模組顯示名稱爲(Unknown)。此時請洽該模組供應商，提供該模組之 ESI 檔，並進行匯入之動作。



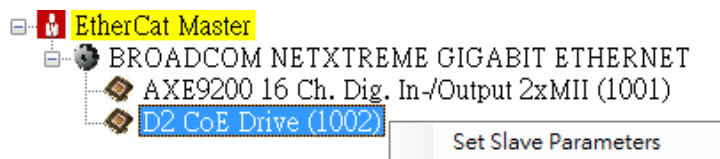
“Unknown”裝置，游標放置可顯示 ID 資訊

- 區域 3:
設定參數選單之顯示區。設定 Slave Parameters 及設定 Master Parameters. 頁面顯示區。
- 區域 4:
訊息及錯誤代碼顯示區。
- 區域 5:
Slave 模組狀態顯示區。目前共有 4 種狀態顯示模組狀態:
 1. Initial(初始狀態):尚未啟動網路時，模組的初始狀態。
 2. Error(錯誤狀態):啟動網路時，Master 無法將所有 Slave 模組由 INIT 狀態切換至 OP 操作狀態。常見的錯誤為 ENI 檔案與實際網路裝置不符，ESI 版本與 Slave 裝置版本不符等。
 3. Retry(重試狀態): 當 ECAT master 的參數 “Link Error Mode”若是設定為“Auto re-connect” (自動重新連接),當 OP 狀態中的 Slave 模組遭遇斷線時呈現 Retry 狀態，Master 會針對斷線模組進行重新連線的動作，其它模組還是可照常運作，此狀態持續顯示到該斷訊模組重新連接工作正常為止。

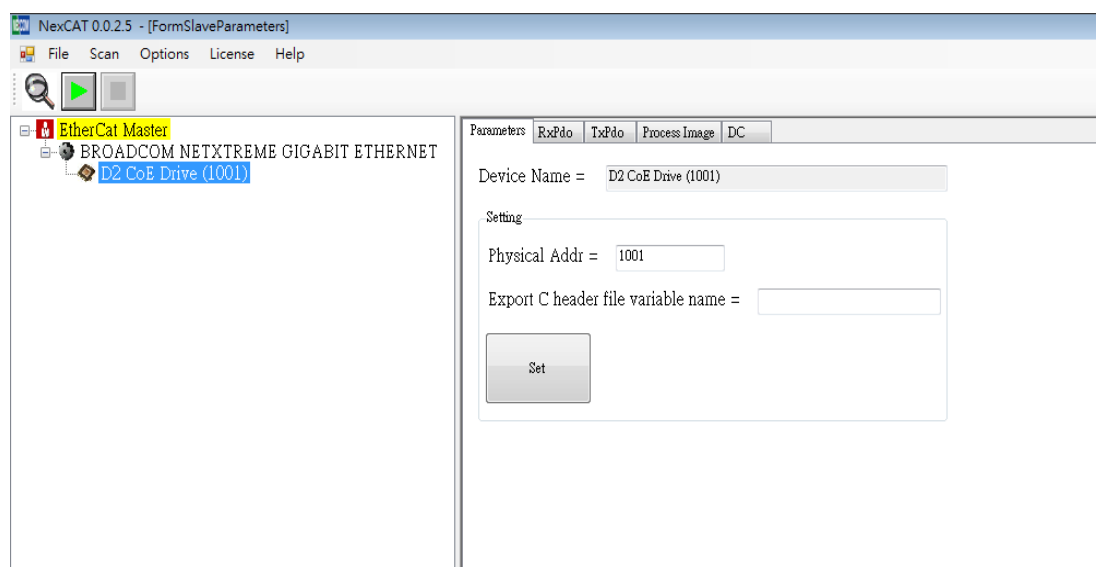
4. Running(工作狀態):啟動網路之後順利轉換至工作狀態(OP)。

3.1.5. 從站模組參數設定頁面 (Set Slave Parameters)

點選模組的名稱，按下滑鼠右鍵出現彈跳選單(pop-up menu)，選擇 set slave parameters 即可。



模組設定頁必須於啟動網路(start network)之前才可以被使用，因為必須於啟動網路之前，將所有參數設定完成之後，設定才會生效，若啟動網路之後，所更改之設定值不會馬上生效，必須重新啟動網路之後才會生效。



1. 參數頁 Parameters Page

Device Name =

Setting

Physical Addr =

Export C header file variable name =

Device Name: 顯示目前設定頁所設定之模組名稱。

Physical Addr: 設定 Slave 模組之 node address (configured address)。

Export C header file variable name: Define 變數名稱，此功能必須配合 Master Parameters 設定之 Export C file 功能一起使用，其功能為輸出各模組之下記憶體配置(Process Image)之變數名稱如下列所示: 請注意內容有無空白字元

#define _Physical Addrsss (+ **variable name**)_ObjectName [記憶體位置]

範例:

Export C header file variable name = “_AXIS”

```
#define _1001_AXIS_Statusword          16777216
#define _1001_AXIS_PositionActualValue 16777218
#define _1001_AXIS_VelocityActualValue 16777222
#define _1001_AXIS_Controlword         16777216
#define _1001_AXIS_TargetPosition      16777218
```

輸出 C header file 請參考下小節說明

2. RxPdo & TxPdo Page

The screenshot shows the NexCAT 0.0.2.5 software interface. The main window is titled 'NexCAT 0.0.2.5 - [FormSlaveParameters]'. It features a menu bar (File, Scan, Options, License, Help) and a toolbar with icons for search, play, and stop. On the left, there is a sidebar with a tree view showing the device structure: 'EtherCat Master' > 'BROADCOM NETXTREME GIGABIT ETHERNET' > 'D2 CoE Drive (1001)'. The main area is divided into two panels. The left panel is titled 'Parameters' and contains a table for RxPdo configuration. The right panel is titled 'TxPdo' and contains a table for TxPdo configuration. Below these tables are buttons for 'save', 'default', and 'clear all'. At the bottom of the window, there is a status bar that says 'found 1 slave module' and an 'Init State' button.

RxPdo Name	Index(Hex)	SM	Mandatory	Fixed
RxPDO 1	1600	2	-1	0
RxPDO 2	1601	-1	-1	0
RxPDO 3	1602	-1	-1	0

Entry Name	Index(Hex)	Sub Index	Bit Len	Data Type
Constantwert	6040	0	16	UINT
Target Position	607A	0	32	DINT
	0000	0	0	
	0000	0	0	
	0000	0	0	

表格欄位說明:

RxPdo(TxPdo) Name: 預設的名稱是由 ESI 的內容所提供，使用者可以於欄位內更改其名稱後，輸出至 ENI 檔。

Index: CoE 提供的參數值，不建議使用者自行更改其設定值。

SM: 對映之 Sync Manager 號碼，可以由使用者自行定義更改。

Mandatory: 定義是否為必要之參數。

Fixed: 定義該參數是否可以被使用者更改。

Entry Name: 其名稱是由 CoE 所提供之名稱，可以更改名稱，輸出至 ENI。

Index: CoE 提供的參數值，不建議使用者自行更改其設定值。

Sub Index: CoE 提供的參數值，不建議使用者自行更改其設定值。

BitLen: CoE 提供的參數值，不建議使用者自行更改其設定值。

DataType: CoE 提供的參數值，不建議使用者自行更改其設定值。

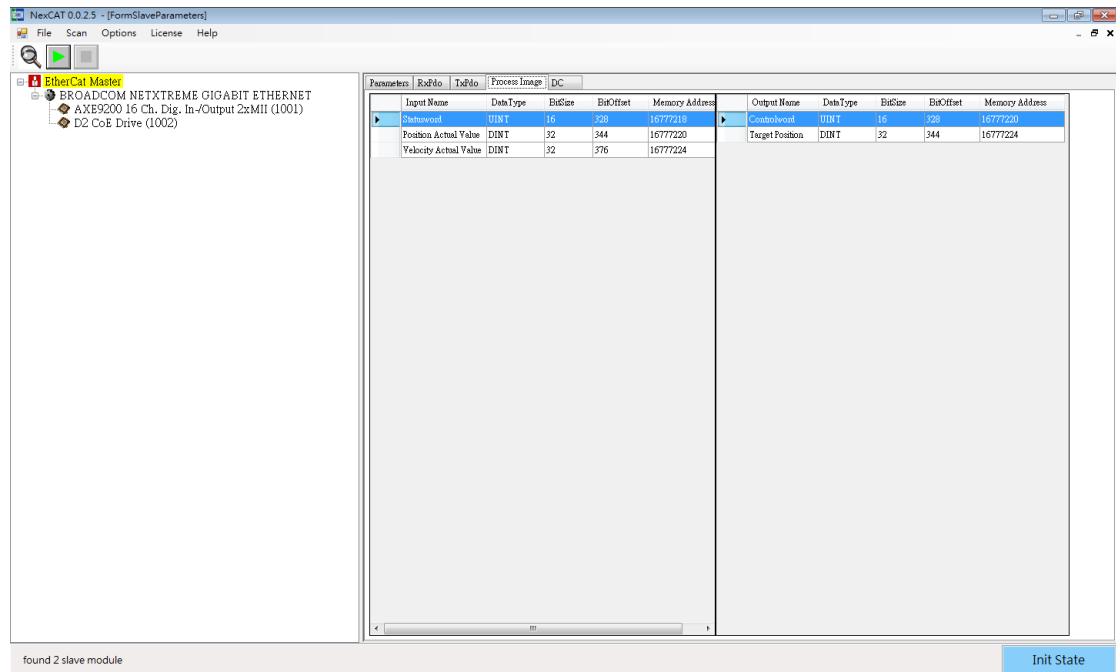
Save Button: 編輯完成之後，必須要儲存後才生效。

Default Button: 可回復成原始 ESI 的設定。

Clear All Button: 清空 PDO 的設定。

3. Process Image Page

使用者可以於 RxPdo 或 TxPdo 編輯頁面，編輯完之後至此頁面觀查它對應之記憶體位置，編輯完之後必須要儲存之後才會生效。



表格欄位說明：

Input(Output) Name: 依據 RxPdo 或 TxPdo 所設定的名稱。

BitSize: 變數所存在之記憶體位置大小。

BitOffset: 依據 RxPdo 或 TxPdo 所設定。

Memory Address: 變數所存在之記憶體位置。

4. DC

設定 Slave 模組的 DC 模式，此頁設定內容的預設值，係由各 Slave 的 ESI file 所提供之資訊

Parameters	RxPdo	TxPdo	Process Image	DC
Setting				
Mode	=	DC		
Description	=	DC SYNC0		
DC SYNC Activation	=	0x0300		
		Apply To Other		Set

Mode (Description) :

選擇 DC 模式。若 Slave 支援 DC 模式，則預設為啟動“DC”。當網路拓樸中有任一 Slave 啟動 DC 功能，EtherCAT Master 將輸出有 DC 資訊(功能)的 ENI File；欲關閉 DC 功能，使用者必須將所有模組的 DC 選項關閉，設定為 FreeRun mode，ENI 檔的輸出內容將沒有 DC 的相關描述。

DC SYNC Activation : (ESC Register 0x0980~0x0981)

0x0000 – Disable SYNC0 & SYNC1 (Free Run)

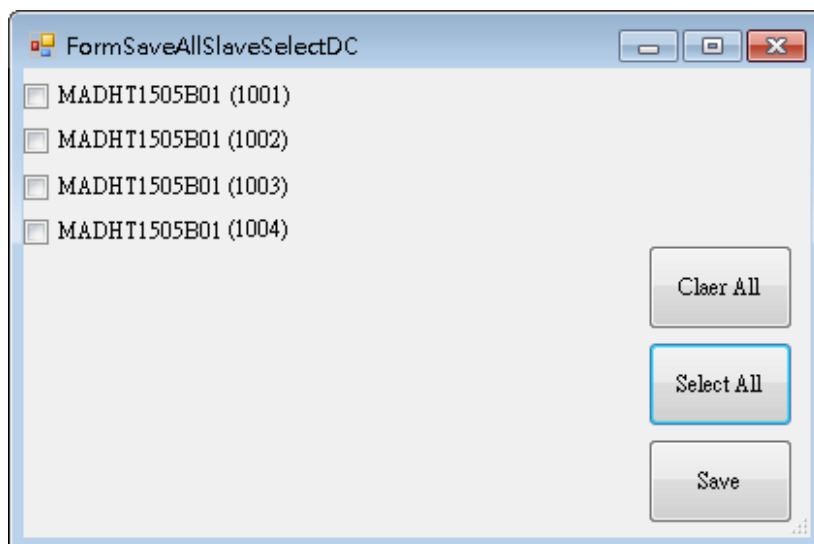
0x0300 – Activate SYNC0 (DC Sync)

0x0700 – Activate SYNC0 & SYNC1

此為進階設定，選擇 DC 模式時此欄會根據 ESI 內容顯示模式對應之設定值，用於控制 Slave 內部 SYNC 中斷訊號，一般依照預設值即可。

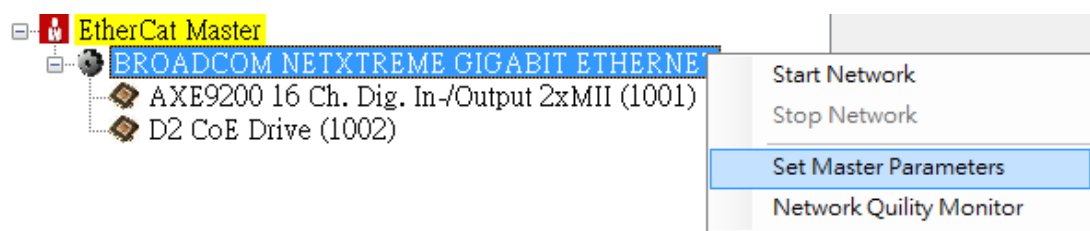
Apply To Other :

可將此設定套用到網路拓樸上其他相同型號之 Slave。(如下圖)



3.1.6. EC-Master 參數設定頁面 (Set Master Parameters)

點選樹狀圖上網路卡的名稱，按下滑鼠右鍵出現彈跳選單(pop-up menu)，選擇 set master parameters。

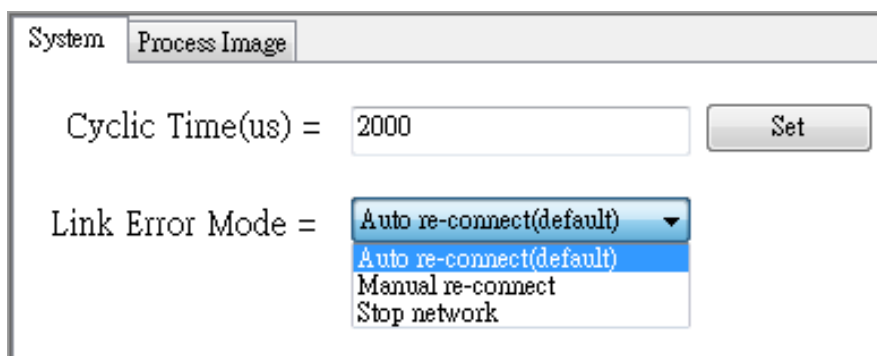


共有兩個子頁面：

1. System
2. ProcessImage

描述如下：

System: 系統設定頁面



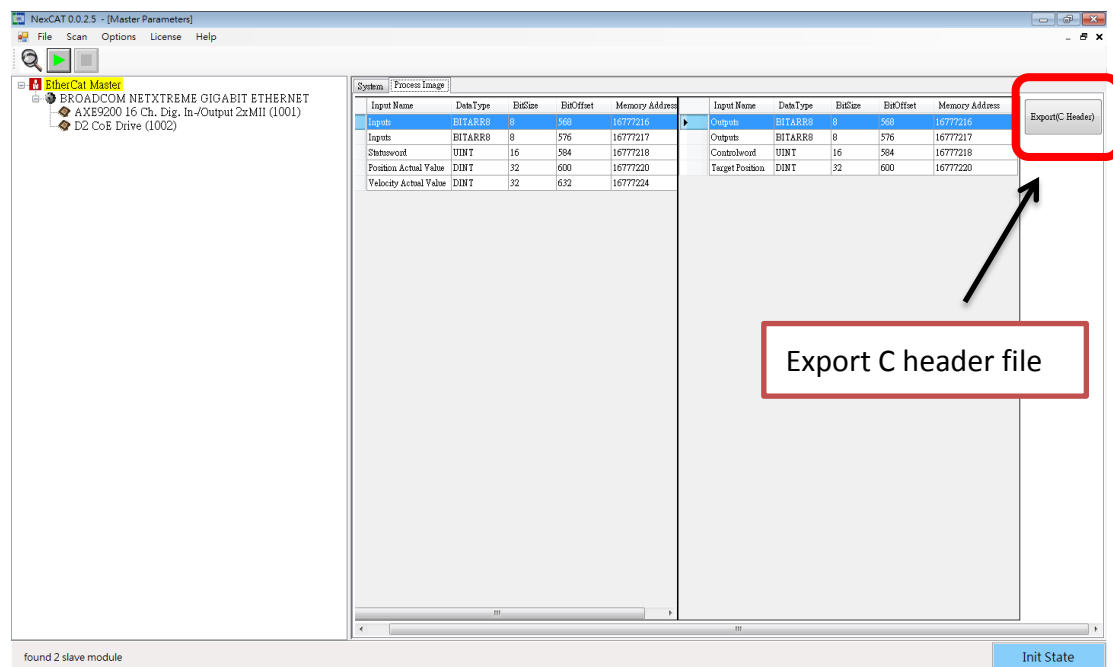
Cyclic Time: EC-Master 通訊週期，單位為 micro second(us)，指定系統的通訊

週期，與模組間資料的溝通時間或更新頻率，最小數值不能超出系統限制，可透過呼叫 API 設定。請參考 6.2.5 NEC_RtSetParameter()

Link Error Mode:連線錯誤處理機制，啟動網路(start network)之後，模組進入操作模式(Operation state)，當 Master 偵測連線中斷時所進行的處理機制。共有三種模式可供選擇，如下：
此模式所對應之 API 請參考 6.2.5 NEC_RtSetParameter()

- **Auto re-connect(default):** 當模組通訊中斷，主頁面區域 5 出現 Slave Retry 訊息，同時系統自動對失去通訊的模組重試連線，一直持續至連線成功為止，其它模組繼續正常工作。
- **Manual re-connect:** 當模組通訊中斷，其它模組繼續正常工作，主頁面區域 5 出現錯誤訊息，持續至下次啟動網路連線成功為止。
- **Stop network:** 當模組通訊中斷，EC-Master 將停止網路連線，主頁面區域 5 出現錯誤訊息，持續至下次啟動網路連線成功為止。

ProcessImage 頁面



- **Network process image map**
與 3.1.5 之章節所描述的 ProcessImage page 內容格式相同，在此可以看到整個網路拓撲產生出之記憶體配置位置

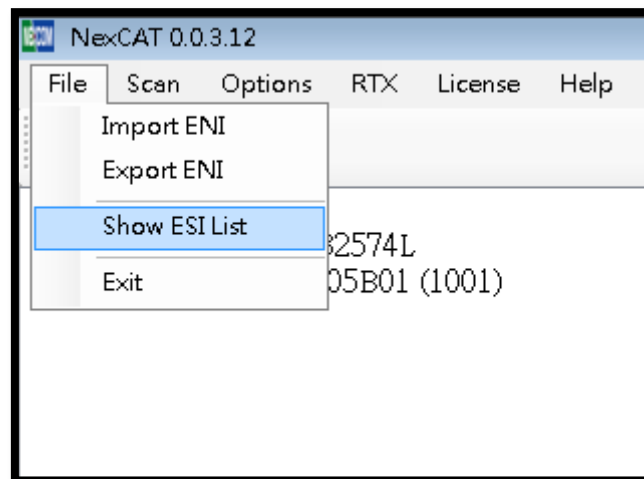
- 輸出 C header file for process image map
點擊 "Export C Header File" 按鈕 可輸出各從站模組之 PDO mapping 在 Process Image 中的偏移量(offset)定義，撰寫程式時透過使用 define symbol 的方式對該記憶體進行存取，當 PDO 順序改變時，只需從新輸出此檔案達成程式碼的易維護性及可讀性。輸出形式可參考 3.1.5 小節從站模組參數設定。

3.1.7. ESI 檔案管理表單 (ESI List)

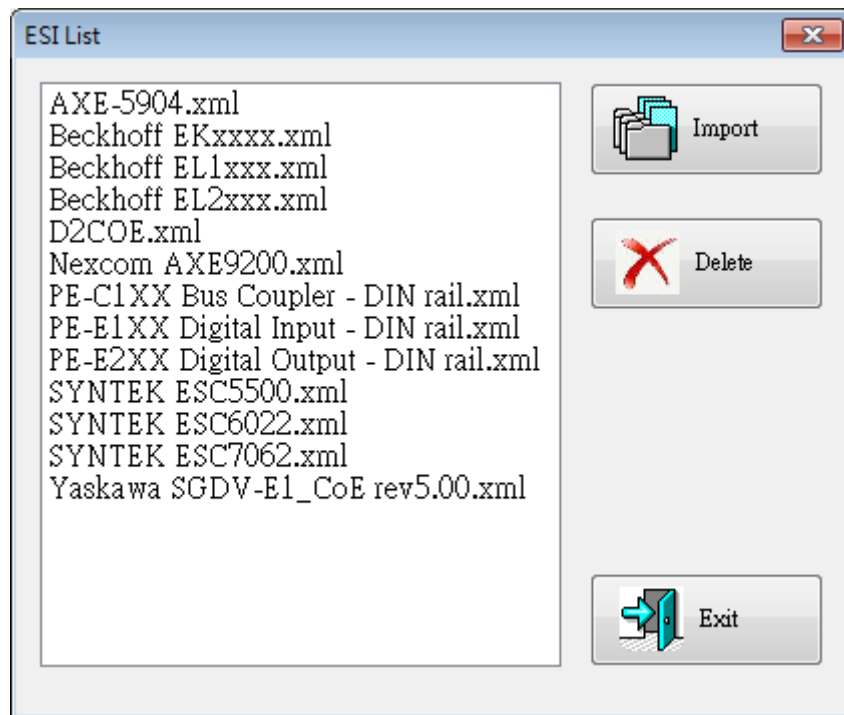
使用 NexCAT 進行掃描網路時，可以得到線路上有多少從站裝置並取得該裝置的 Device ID，透過比對 ESI 檔案來辨識該硬體裝置(可參考 3.1.3 小節)。若使用者取得一新的 ECAT slave 裝置，必須將該 ESI 檔案匯入到 NexCAT 中。

NexCAT 提供了 2 種方式來管理 ESI 檔案：

1. 使用者直接將 ESI 檔案放置在 NexCAT 指定的目錄下，從新啟動 NexCAT，NexCAT 會於啟動過程中載入資料夾中所有的 ESI 檔。
2. 使用 "ESI List" 檔案管理表單：當使用者欲新增/刪除 ESI 檔，可透過此介面來操作。當使用者編輯完成之後按下 Exit 系統自動重新整理 ESI 資料夾內之檔案，直接重新掃描即可，不須重新啟動程式。



開啟 ESI List

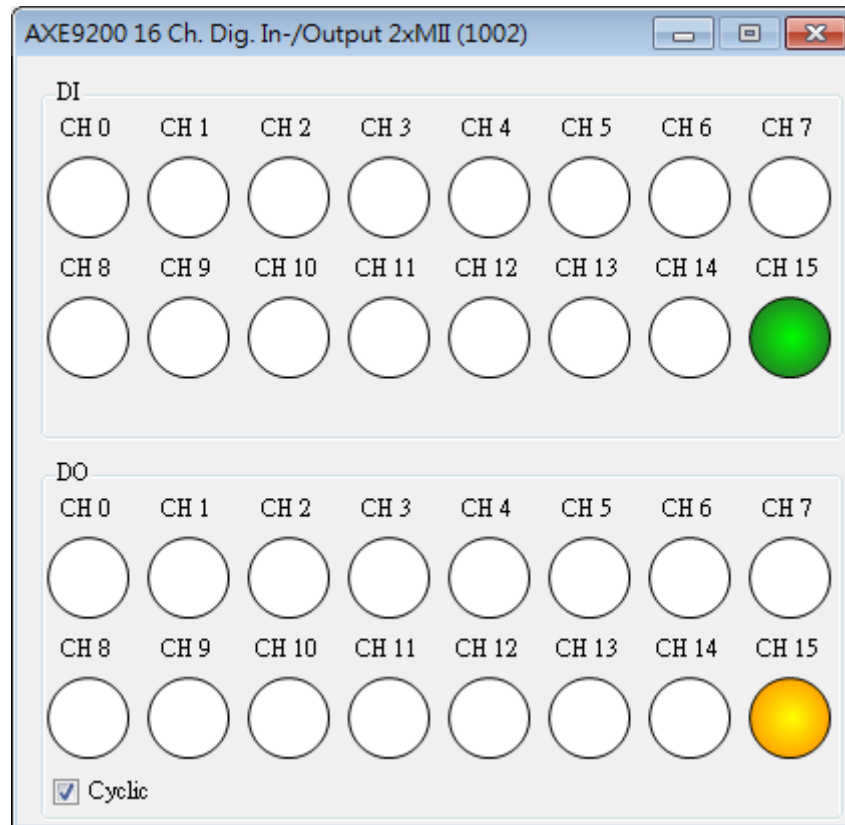


ESI List 畫面

3.1.8. DIO 操作頁面

使用者於主頁面之區域 2，點選須要操作之 DIO 模組，快速點二下滑鼠左鍵則 DIO 操作表單自動出現畫面中央。程式會自動判定該模組為 DI、DO 或是 DIO 模組並自動計算其 IO 數量。於畫面中自動呈現。

當滑鼠游標移至 DO 的按鈕處，使用者可以手動壓按 DO 按鈕，即可操作 DO，亦或者，使用者可以使用 Cyclic 功能，讓 DO 模組自動運行啟動跑馬燈功能，該模組由第 0 個通道(Channel)開始，由小至大重覆運行。當 Cyclic 的框選處被打勾後自動運行，勾選取消則程式自動停止停在運行時執行的最後一個通道。



3.1.1.9. CoE-SDO 存取操作頁面

使用者於主頁面之區域 2，點選須要操作具 CoE 功能之模組，快速點二下滑鼠左鍵則 CoE 操作表單自動出現畫面中央。程式會自動判定該模組為是否支援 CIA 402 之模組。

按下“Refresh” 按鈕後表格內之參數值將自動更新，使用者可自行選擇以 10 進制表示或者以 16 進制表示，若有某參數為浮點數，則該參數不受進位制影響一律以浮點數表示之。

倘若使用者欲改變參數值，可以使用滑鼠，於 Value 的表格內快速點二次滑鼠左鍵，可編輯該參數值，編輯完成後，按下 Enter 鍵或離開該表格即可成功寫入，若寫入失敗或寫入格式不符合規範的資料形態，則該參數值自動回覆成為編輯前之狀態。

Index(Hex)	Name	DataType	Access	Value
1000	Device Type	UDINT	ro	131474
1001	Error Register	USINT	ro	0
1009	Manufacturer Hardware Version	STRING(1)	ro	0
100A	Manufacturer Software Version	STRING(1)	ro	0
1018	Identity Object	DT1018		
1600	Position Mode RxPDO Mapping Parameter 1	DT1600		
1601	Velocity Mode RxPDO Mapping Parameter 2	DT1601		
1602	Torque Mode RxPDO Mapping Parameter 3	DT1602		
1A00	Position Mode TxPDO Mapping Parameter 1	DT1A00		
1A01	Velocity Mode TxPDO Mapping Parameter 2	DT1A01		
1A02	Torque Mode TxPDO Mapping Parameter 3	DT1A02		
1C00	Sync Manager Communication Type	DT1C00		
1C10	Sync Manager 0 PDO Assignment	DT1C10		
1C11	Sync Manager 1 PDO Assignment	DT1C11		
1C12	Sync Manager 2 PDO Assignment	DT1C12		
1C13	Sync Manager 3 PDO Assignment	DT1C13		
1C32	Sync Manager 2 Synchronization	DT1C32		
1C33	Sync Manager 3 Synchronization	DT1C33		
2000	Motor Type	UINT	ro	2
2001	Inner Encoder Resolution	DINT	ro	131072
2002	Outer Encoder Resolution	DINT	ro	131072

CoE parameters

若參數的資料型態為 **DataType** 時，表示該參數包含了子參數(Sub Index)，使用者可以於想要存取之參數快速點滑鼠左鍵二次，若程式判斷其底下確實有子參數，會出現子視窗如下圖所示，其值之讀取與寫入與之前章節所提及到的方式相同。

Identity Object

☒ Value = Hex Refresh

	Sub Index	Name	Data Type	Access	Value
▶	0	number of entries	USINT	ro	04
	1	Vendor Id	UDINT	ro	0000aaaa
	2	Product Code	UDINT	ro	00000003
	3	Revision number	UDINT	ro	00000001
	4	Serial number	UDINT	ro	00000001

Sub parameters

3.1.10. Process Image 參數存取操作頁面

D2 CoE Drive (1001)

CoE Parameters Process Image

☐ Input Data = Hex ☐ Output Data = Hex

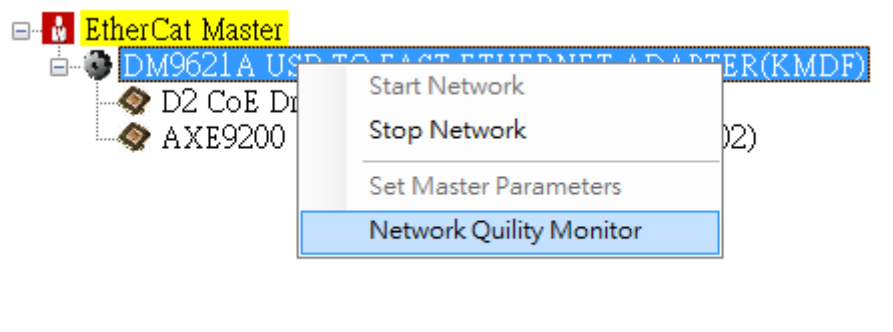
Input Name	Data Type	BitSize	BitOffset	Data
▶ Statusword	UINT	16	568	96
Position Actual Value	DINT	32	584	2342
Velocity Actual Value	DINT	32	616	0

Output Name	Data Type	BitSize	BitOffset	Data
▶ Controlword	UINT	16	568	
Target Position	DINT	32	584	

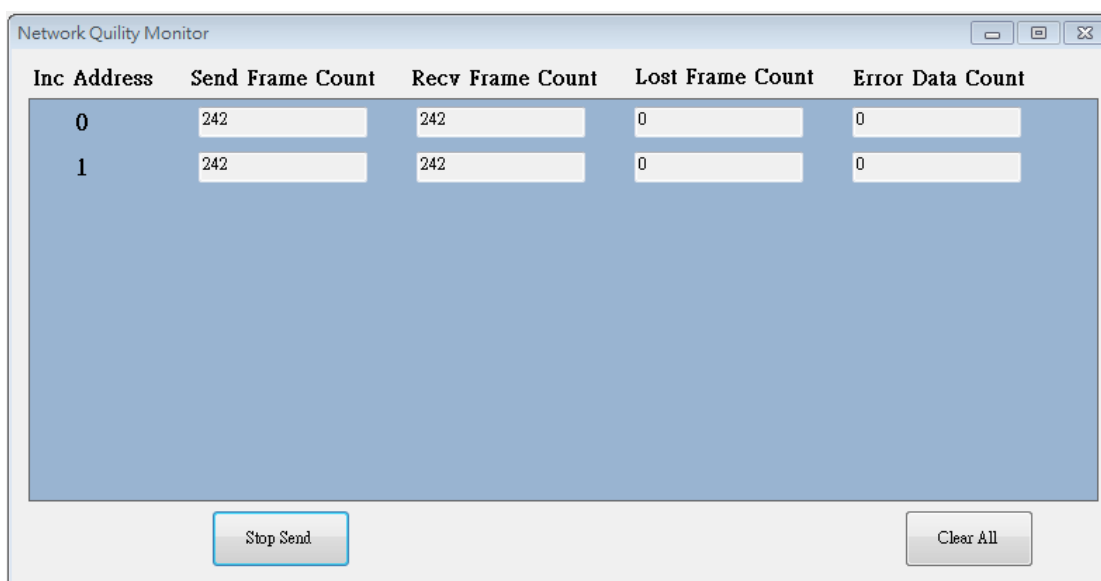
使用者可於成功啟動網路(start network)之後，存取 pdo(process data object)資料。其顯示方式，當 Input data(output data) = hex 的框選欄被勾選時，其資料於表格內所示之數值為 16 進制，反之為 10 進制的表示方式。

3.1.11. Network Quality Monitor 網路通訊品質測試頁面

網路成功啟用之後，使用者可以開啟網路通訊品質測試頁面，進行 Master 端對於各模組的通訊封包測試。欲開啟此頁面，可以使用滑鼠於主頁面的區域 2 之 NIC 的節點上，按下滑鼠右鍵後，將會出現一個彈跳視窗(pop-up menu)，可以點選 Network Quility Monitor 的選項後，出現網路品質測試頁面。



NIC 的節點上，按下滑鼠右鍵



Network Quality Monitor 頁面

- Inc Address：模組之 Slave ID，係依照所掃描到的模組順序排列之。
- Send Frame Count:系統對模組發送之測試封包數量，讀取 Slave 模組的狀態是否為“OP”狀態，系統發封包的頻率為 10 ms，實際的發送速度依照系統當時的效率。
- Recv Frame Count: Slave 模組回應的封包數量，正常的情況之下，每發送一個測試封包 Slave 模組需回應一個封包。

- Lost Frame Count: 遺失的封包，當發送出去封包，其模組無回應該封包的數量。
- Error Frame Count: 檢查 Slave 回應之封包內容資訊，若其狀態不為“OP”狀態則 Error Frame Count + 1。

其關係如下列式

Send Frame count = Recv Frame count + Lost Frame count

Recv Frame count = Normal Frame (state == OP) + Error Data Frame count.

3.2. NexECMRtxStartup 啟動程式說明

NexECMRtxStartup.exe 為提供你在使用 EtherCAT 主站(EC-Master)的便利性。此程式根據 NexECMRtxConfig.ini 的內容，提供以下主要的三個功能：

1. 載入 EtherCAT 主站(EC-Master) - NexECMRtx.rtss
2. 下載 ENI (EtherCAT Network Information) 網路資訊檔
3. 載入用戶 RTX 應用程式 (例如:UserRTXApp.rtss)

你可透過記事本或其它文字編輯軟體修改 NexECMRtxStartup.ini 內容以符合你當前的檔案放置情形；通常須修改 1.應用程式(Application)的位置，2.網路資訊檔(ENI)的位置。你可在 "C:\Program Files\NEXCOM\NexECMRtx\tools" 此位置找到此 ini 檔案，請參考以下圖解說明。



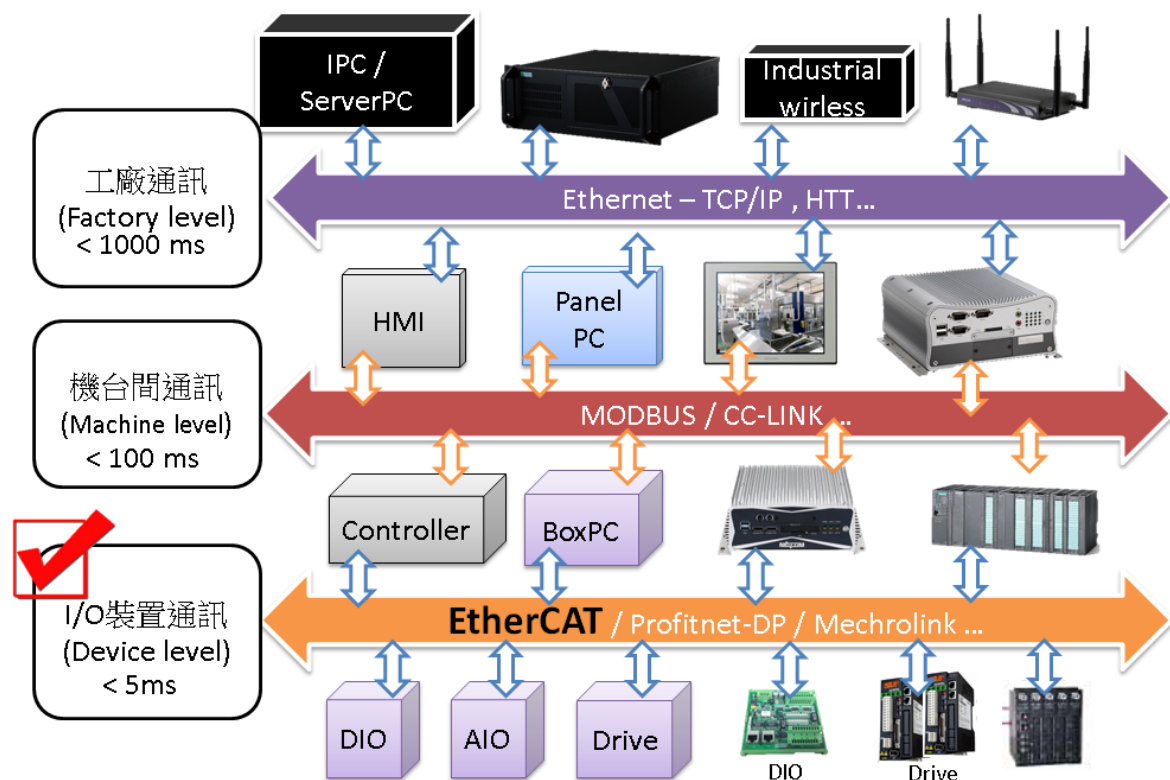
NexECMRtxConfig.ini 的內容

識別字	說明
PATH_ENI	
PATH:	網路資訊檔(ENI)檔案位置。 OPTION: 是否使用 ENI 中，網路卡的資訊 0:使用 ENI 中的網路卡資訊 1:不使用 ENI 資訊，採用 Parameter 的設定
PATH_NEXECMRtx_DRIVER	
PATH:	NexECMRtx.rtss 檔案位置。
PATH_USER_APP (Option)	
PATH:	填入你的 RTX 應用程式(*.rtss)的路徑位置與檔案名稱。

4. EtherCAT 技術簡介

隨著通訊技術的進步，工業通訊現場匯流排(Fildbus)的技術已廣泛的應用在工業控制系統之中。就網路技術而言，Ethernet 已成為最普遍，也是最廣泛被應用在各種領域之上，其相關的硬體和軟體技術也隨技術的普及而高速的發展中。在如此大量地使用之下 Ethernet 已成為最先進且最具成本效益的一項通訊技術之一。

而 EtherCAT (Ethernet Control Automation Technology)是一項基於 Ethernet 被設計應用在自動化(特別是在機台自動化)的工業通訊技術，它挾帶著 Ethernet 的各項優勢(普遍，高速，低成本)，其相關的產品也以驚人的速度在成長。下圖為一工業通訊系統示意圖，EtherCAT 主要被應用在連接高速即時的 I/O 裝置。

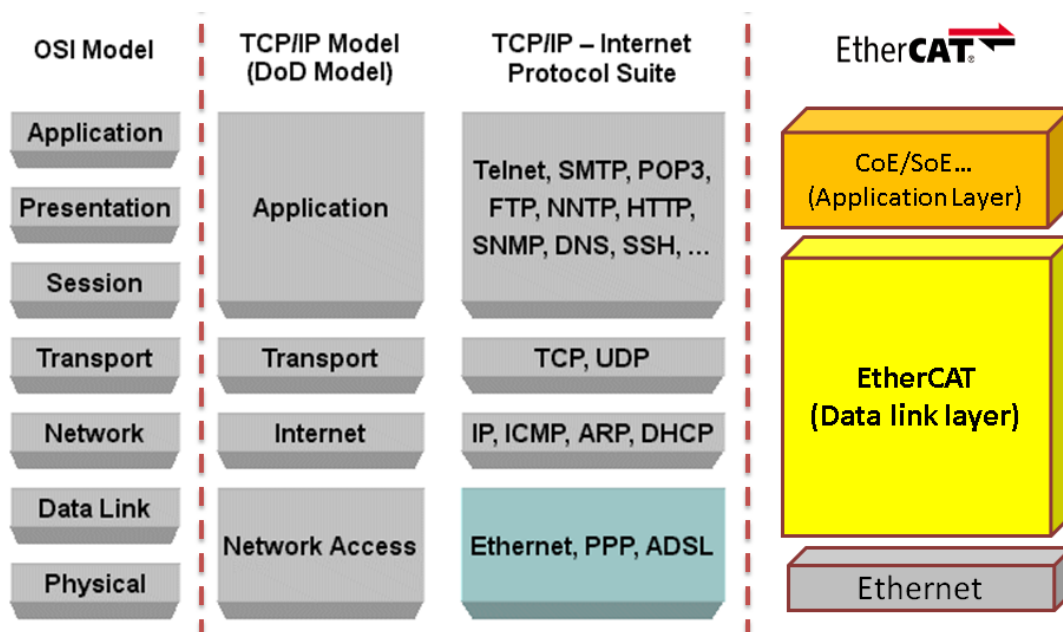


下面幾個小節簡單說明 EtherCAT 通訊技術及其特點。

4.1. EtherCAT 通訊協定概述

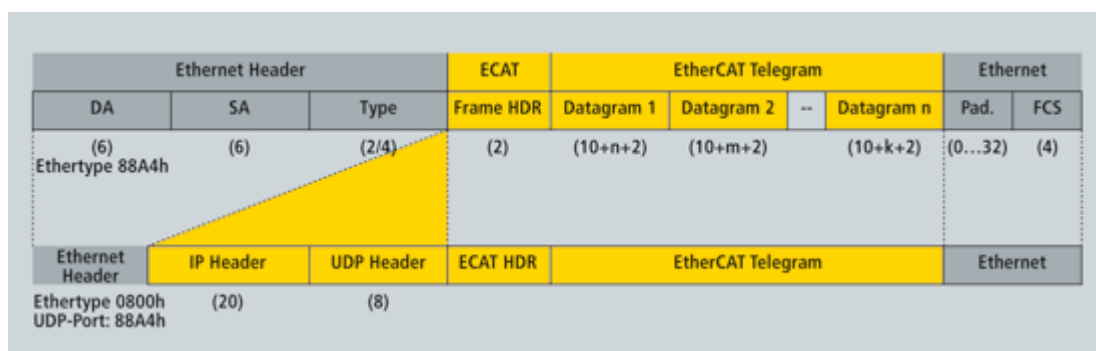
EtherCAT 是一個架構在 Ethernet 物理層(Physical Layer)上具即時性(Real-time)的通訊協定，目前由 ETG (EtherCAT Technology Group)組織來維護及推廣。

以 OSI 網路模型而言，EtherCAT 通訊協定主要定義於資料層(Data link Layer)和應用層(Application Layer)。



EtherCAT 的網路封包使用 IEEE 802.3 Ethernet 格式。EtherType 除了可使用標準 EtherCAT 封包之外(EtherType = 0x88A4)，亦可使用 UDP 格式(EtherType = 0x0800)，如下圖。

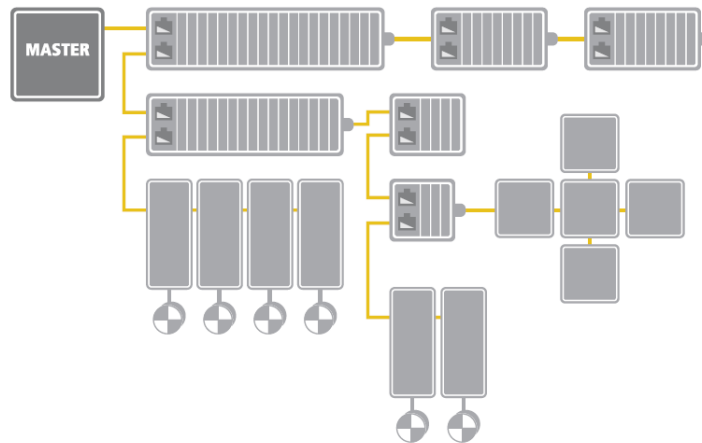
EtherCAT 的內文部分主要包含了 ECAT 的標頭(Frame header)以及後面的報文(Telegrams)。



EtherCAT 網路封包格式 (資料來源: <http://www.ethercat.org/>)

4.2. 網路拓樸

網路拓樸特性關係到現場佈線的可能性。EtherCAT 幾乎支援所有網路拓樸形式包含直線，樹狀及星狀等拓樸，另外，使用標準 100BASE-TX 網路線兩個裝置之間最長的距離可達 100 米，由上述規格來說在設備自動化的應用領域而言幾乎沒有任何限制。



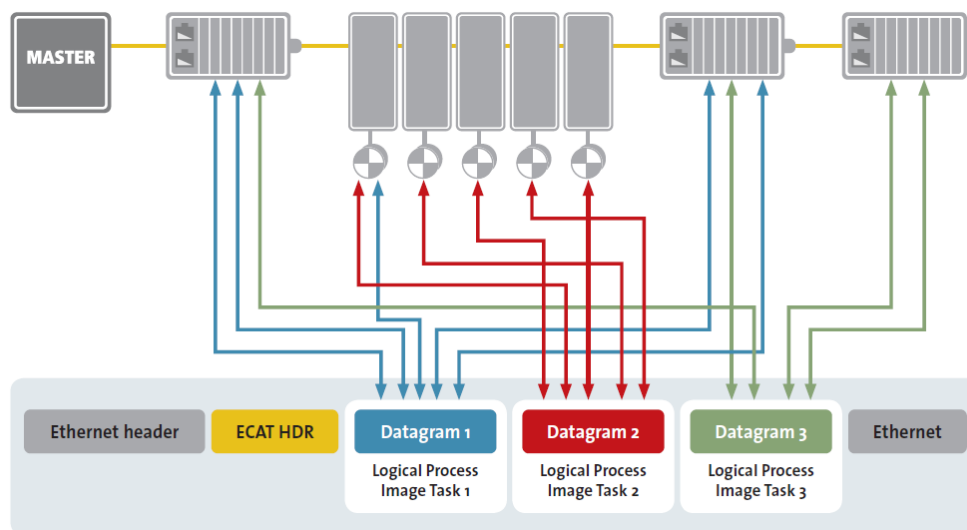
EtherCAT 網路拓樸, Line, Tree and Star 拓樸
(資料來源: <http://www.ethercat.org/>)



使用標準 Ethernet 網路線

4.3. 高速效能

透過 EtherCAT 所定義的定址方式，配合 EC-Slaves 上的硬體所實作的記憶體管理單元: Fieldbus memory manager unit (FMMU)，可完成經由傳送一個網路封包可與線上所有裝置(EC-Slaves)做同步的資料交換，例如 DIO 資料，AIO 資料和伺服馬達位置資料等。如下示意圖：



EtherCAT 網路封包與裝置資料交換 (資料來源: <http://www.ethercat.org/>)

EtherCAT 從站(EC-slaves)裝置的部分，會搭載一 EtherCAT 專用通訊 IC (EtherCAT Slave IC,簡稱 ESC) 來完成，搭配上上述 FMMU 技術，根據 ETG 的資料，1000 點的 I/O 資料更新只需約 30us 可完成交換。100 軸的伺服馬達資料只需 100us。請參閱下表：

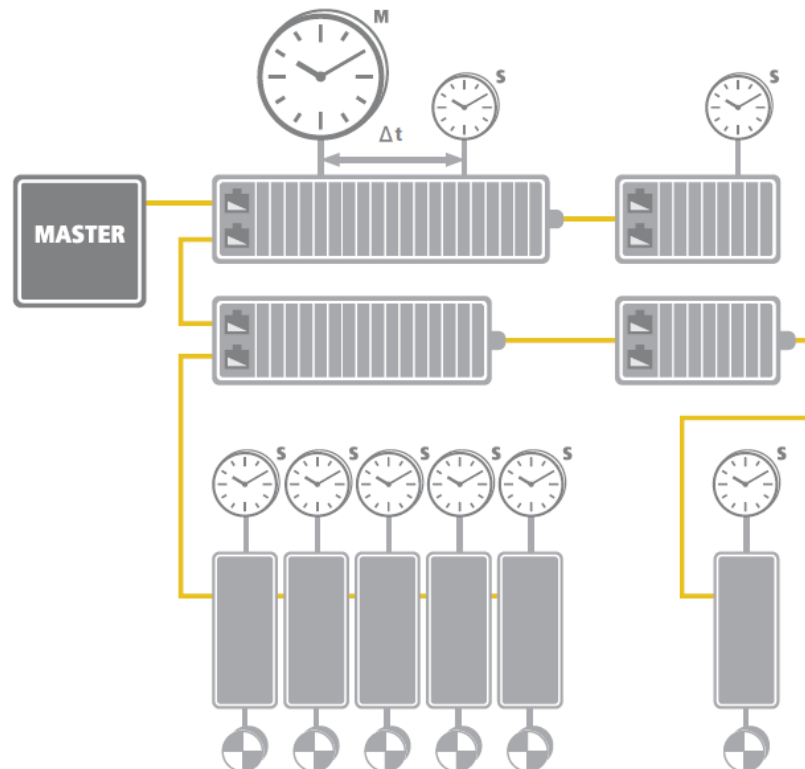
資料量	更新時間
256 點 I/O	11μs
1000 點 I/O	30μs
200 Channels 類比 I/O (16 位元)	50μs (=20kHz)
100 個伺服軸 (8 Bytes Input and output per axis)	100μs
1 個現場匯流排主閘道 (Fieldbus Master-Gateway) (1486 位元組輸入與 1486 位元組輸出資料)	150μs

EtherCAT 通訊資料量與時間關係 (資料來源: <http://www.ethercat.org/>)

若以 100 軸 100us 的資料交換速度而言相當於控制頻寬為 10KHz，此控制頻寬用於速度控制迴路或甚至扭矩(Torque)控制迴路都已足夠。

4.4. 網路同步

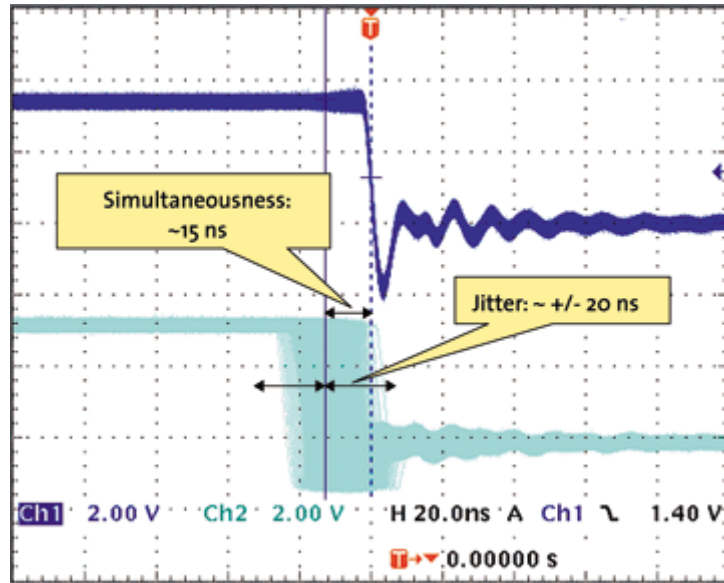
在設備自動化應用中對分散式網路的要求除了網路通訊需具備即時性(Real-time)外，其裝置之間的同步性要求不可或缺，例如控制多軸伺服馬達的直線或圓弧補間運動，龍門(Gantry)軸控制等。這些應用必須具備網路同步性才能確保應用的正確性。



EtherCAT 分散時鐘技術示意圖 (資料來源: <http://www.ethercat.org/>)

EtherCAT 的網路同步機制是根據 IEEE-1588 精確時間同步協定，延伸定義出所謂分散時鐘技術(Distributed-clock 簡稱 DC)。簡單的來說在每個 EtherCAT 的從站裝置上(ESC 內)都自行維護一個硬體時鐘(Local Clock)，其時間最小間格為 1 ns 共 64 位元，這個由 EC-Slave 自行維護的時間稱之為“當地系統時鐘”(Local system time)，在透過精確網路對時機制以及動態時間補償機制後^(*)，EtherCAT 的 DC 技術可以保證所有的從站裝置上的當地系統時鐘(Local system time)的誤差在+/- 20 奈秒(nano-second)之內，如下圖兩個從站間 I/O 信號誤差約為 20 ns。

^(*) 可參考 EtherCAT standard document ETG1000.4

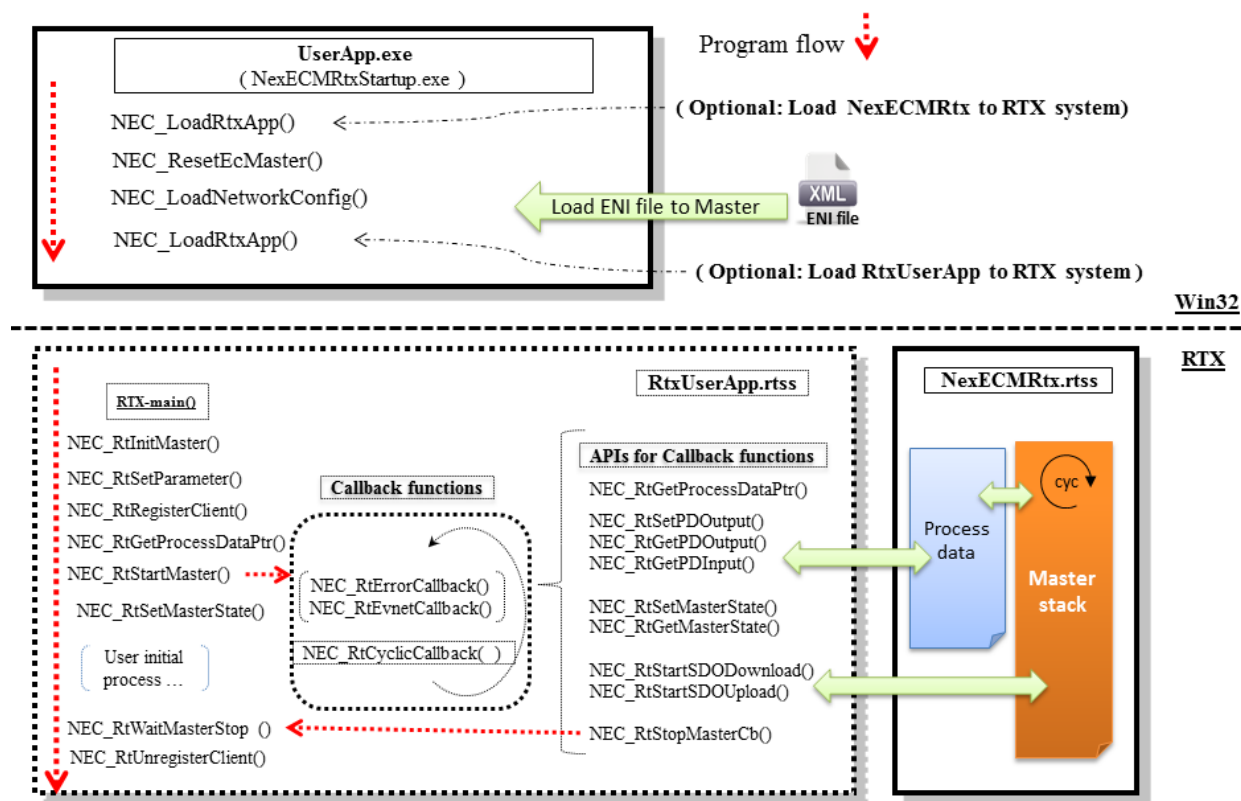


Sync 信號誤差 (資料來源: <http://www.ethercat.org/>)

5. 編程原理

5.1. 基本編程框架

下圖說明 NexECMRtx 基本編程流程。



基本流程如下

步驟	說明	相關函式
您的 Windows 上層(UserMode)應用程式: ex. <u>UserApp.exe</u> (詳細 API 說明請參考 CH.7)		
1	載入 NexECMRtx (EtherCAT 主站通訊層)	NEC_LoadRtxApp()
2	重置 NexECMRtx	NEC_ResetEcMaster()
3	載入 ENI 網路設定檔案	NEC_LoadNetworkConfig()
4	載入您的 RTX 應用程式 (如上圖 RtxUserApp.rtss)	NEC_LoadRtxApp()
您的 RTX 應用程式主程序 main(): ex. <u>RtxUserApp.trss</u> (詳細 API 說明請參考 CH.6)		
5	初始化 EC-Master	NEC_RtInitMaster()
6	設定參數，如主站通訊週期(Cycle-time)等	NEC_RtSetParameter() NEC_RtGetParameter()
7	註冊 Callback 函式	NEC_RtRegisterClient()

		NEC_RtUnregisterClient()
8	取得 ProcessImage 的記憶體指標 (Memory pointer)	NEC_RtGetProcessDataPtr() NEC_RtGetSlaveProcessDataPtr()
9	啟動 EC-master 通訊	NEC_RtStartMaster()
10	將 Ec-Master 的狀態切換至“OP”狀態	NEC_RtSetMasterState()
11	RTX 主程式為一 main() 型態終端應用程序，當程序結束後即整個程序會被卸載，因此當完成上述程序後， 可依照您的應用： 1. 讓主程式進入睡眠等待 EC-master 通訊結束或 2. 您的其他(裝置)控制應用程序	NEC_RtWaitMasterStop()
在周期 Callback 函式中 (詳細 API 說明請參考 CH.6)		
12	你的控制程序進行 1. ProcessData 存取 2. CoE 通訊	NEC_RtSetPDOutput() NEC_RtGetPDOutput() NEC_RtGetPDInput() NEC_RtStartSDODownload() NEC_RtStartSDOUpload()
13	應用程序結束，停止 EC-Master 通訊	NEC_RtStopMasterCb()

上述可流程可參考 NexECMRtx 安裝目錄中的範例程式。

5.2. ENI 載入

ENI 網路檔案(EtherCAT Network Information)須由您的 Win32 程式進行下載，下載前請先將 NexECMRtx.rtss 載入 RTX 系統中。載入方式分為下列 2 種：

1. 手動載入：滑鼠左鍵雙擊“NexECMRtx.rtss”
2. 呼叫 NEC_LoadRtxApp() API 載入

載入 NexECNRtx.rtss 後，呼叫 NEC_LoadNetworkConfig() 下載 ENI 檔。當 ENI 檔案成功載入後，EC-Master 將保留這些載入的資料在其內部記憶體中。當 EC-Master 啟動通訊時會透過這些資料來設定 EtherCAT 從站(EC-Slaves)。啟動過程中 EC-Master 會偵測 ENI 中的資料是否與目前實際網路接線配置是否相同，若有異

常會停止通訊並發出錯誤事件(Error callback) ,

你可以透過 NexCAT 程式來設定網路配置並產生 ENI 檔案，其詳細操作方式請參考第三章。

完成載入 ENI 檔案到 EC-Master 後，你可以接著啟動您的 RTX 應用程式。並在您的程式中對 NexECMRtx 進行通訊的控制及操作，請繼續參與後續小節的說明。

上述的步驟你也可以使用 *NexECMRtxStartup.exe* 工具程式來完成，在程式資料夾中 INI 檔案中設定：

1. ENI 檔案的路徑
2. NexECMRtx 的路徑
3. 您的 RTX 應用程式路徑

詳細的功能設定請參考 3.2 章的說明。

設定好後，執行該程式會自動完成上述載入 ENI 的動作，並啟動您的 RTX 應用程式。

5.3. 初始化 NexECMRtx

在您的 RTX 應用程序中，呼叫 *NEC_RtInitMaster()* 函數來初始化 EC-Master。此動作將不會重置您從 Win32 應用程序中下載的 ENI 的數據資料。

呼叫 *NEC_RtSetParameter()* 函數來完成配置 EC-Master。例如，參數代號 *NEC_PARA_S_ECM_CYCLETIMEUS* 是用來設置主站的通訊週期時間。而其他參數請參閱該函數的說明。

使用 *NEC_RtRegisterClient()* 來註冊一組個客戶端的 Callback 函數，其 Callback 的原型如下：

```
void NEC_RtCyclicCallback( void *UserDataPtr );
void NEC_RtEventCallback ( void *UserDataPtr, U32_T EventCode );
void NEC_RtErrorCallback ( void *UserDataPtr, U32_T ErrorCode );
```

Callback	說明
NEC_RtCyclicCallback	週期性 Callback 函數，實作控制程序
NEC_RtEventCallback	當預先定義的事件發生由 EC-Master 呼叫
NEC_RtErrorCallback	當錯誤事件產生時會被 EC-Master 呼叫

這些 Callback 函數用於與 EC-Master 進行的同步通信與 ProcessData 資料交換。當 EC-Master 在狀態機(State-Machine)^(*)更改為 “SAFEOP” 和 “OP” 的狀態時，EC-Master 開始在每個通訊周其中將 ProcessData 數據更新傳送至 EC-Slaves 並從 EC-Slave 將資料傳回 ProcessData 中。

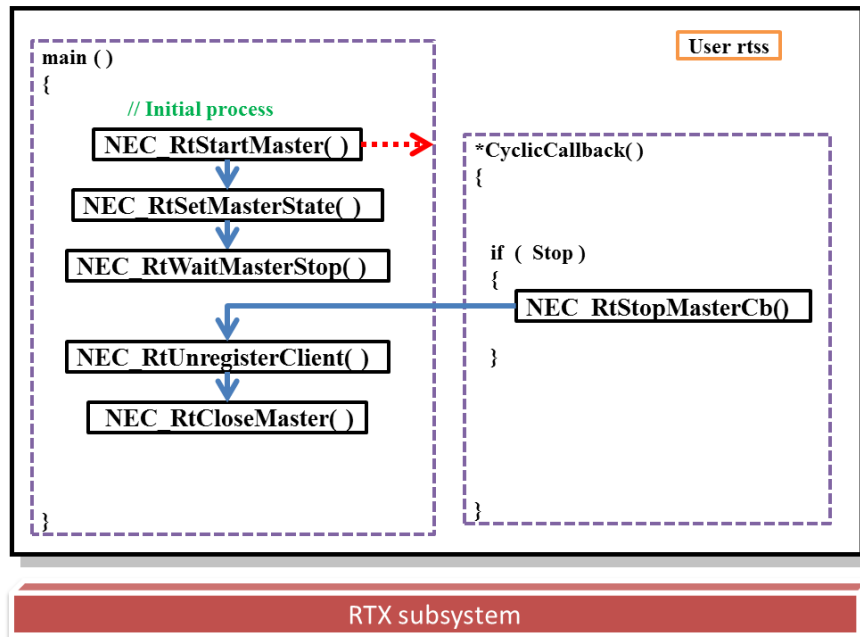
(*)關於 EtherCAT 狀態機，請參考 5.6 小節“EtherCAT 狀態機”的說明。

大多數的 EC-Slave 的通訊應用程序(例如運動控制演算法等)將被放置在週期的 Callback 函數中，在 Callback 函數種編程的基本原則是**避免**在 Callback 中撰寫導致 Callbak 停止或過於攏長的程序使 Callback 的執行時間超出通訊週期。因此儘量避免使用類似 Sleep(), RtSleep()或 while();, do while()的 Polling 等程序在 Callback 函數中。關於 Callback 的使用請參考 5.5 小節 Callback 函式。

5.4. 啟動主站通訊

根據上一小節完成主站的初始化設定程序後，調用 `NEC_RtStartMaster()` 啟動主站 EtherCAT 通訊，若成功的調用此函數，使用者所註冊的週期回呼函數(Callback)會週期的執行。請注意，當啟動通訊後 EC-Master 的狀態機會保持在“INIT” 狀態，直到使用 `NEC_RtSetMasterState()`來送出狀態變更要求。

一般的情況下，用戶會將控制演算法實現在回呼函數(Callback function)之中。然而，RTX 應用程序是一個類似主控制台(Console)編程的方式，RTX 程序有一標準的進入點 `main()`函數。當 `main()`函數返回時，該程序即結束並由 RTX 系統中卸載。為了防止主程序的結束，用戶可使用 `NEC_RtWaitMasterStop()`來阻止該程序結束，使主程序進入休眠狀態並等待 EC-Master 的停止信號。當用戶要結束應用程序並停止通訊，可在回呼函數中使用 `NEC_RtStopMasterCb()`來發送停止信號來喚醒被阻塞的 `NEC_RtWaitMasterStop()`使其返回。



關於回呼函數(Callback function)的使用，請參考下一小節。EC-Master 狀態機將詳述於 4.6 小節

5.5. Callback 函式

5.5.1. 註冊回呼函數(Callback functions)

NexECMRtx 提供三種回呼函數，

1. 週期回呼函數 (Cyclic-Callback function)
2. 事件回呼函數 (Event-Callback function)
3. 錯誤事件回呼函數 (Error-Callback function)

啟動 EC-Master 通訊前，使用 *NEC_RtRegisterClient()* 將 Callback 函數註冊 EC-Master 之中。結束通訊後使用 *NEC_RtUnregisterClient()* 取消 Callback 函數的註冊。必須注意避免在通訊過程中使用 *RtUnregisterClient()* 將 Callback 函數取消註冊。

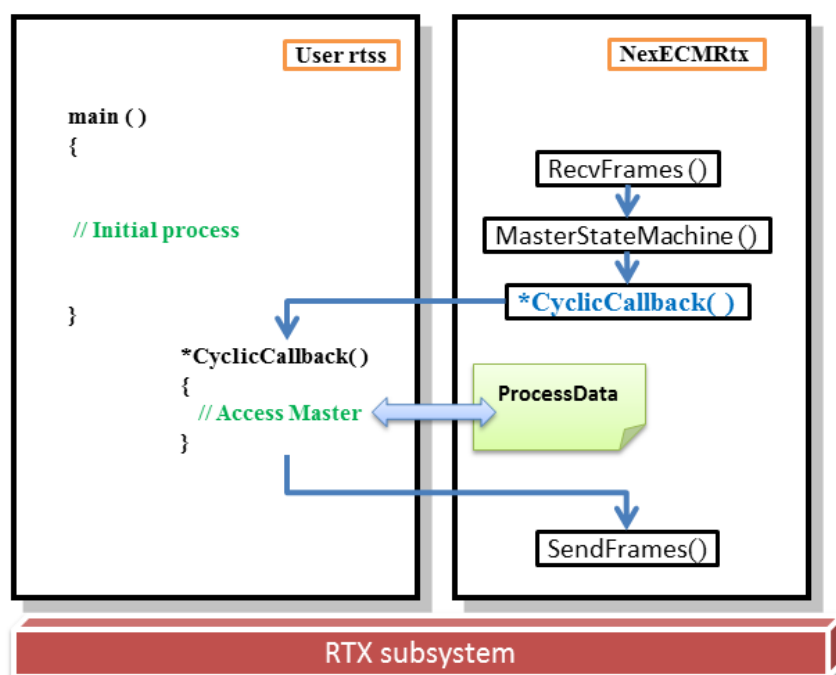
5.5.2. 週期回呼函數(Cyclic-Callback function)

透過 *NEC_RtGetProcessDataPtr()* 函數取得 ProcessData 記憶體指標(Memory pointer)，在取得 ProcessData memory 指標後你可以任意的讀寫 EC-Master 中的 ProcessData，因此我們可以將這個記憶體視為一塊 Share memory，使之成為使用者的程序和 EC-Master 之間的資料交換管道。但 NexECMRtx 的週期程序和使用者的程序基本上可以視作兩個獨立的執行緒(Thread)，因此如何讓使用者的程序

和 EC-Master 的程序之間能同步的存取 ProcessData，保持資料的一致性(Data Consistence)?

另外，在即時工業控制應用中，某些從站應用必須在每個通訊周期中取得主站的資料以完成即時工業控制行為。例如伺服馬達的控制，必須在每個通訊週期中對數個伺服馬達送出該馬達的位置控制資訊，來完成同步位置控制。因此，使用者的即時控制程序必須與 EC-Master 的週期通訊程式必須互相連結。

NexECMRtx 提供 Callback 函數的同步回呼機制(synchronous callbacks)，使您的 Real-time 應用程式與 NexECMRtx 中的 ProcessData 存取同步。透過此機制來完成上述的兩種狀況，保證傳遞資料的一致性以及通訊周期的同步性。下圖表示在 NexECMRtx 中，EC-Master 的資料處理程序與使用者所提供的 Callback 函數運行的關係流程圖。



5.5.3. 事件回呼函數(Event-Callback function)

當 EC-Master 內定的事件發生時，EC-Master 會同步呼叫此函數通知使用者，同時傳入事件代碼(Event code)，使用者可以根據事件代碼來處理對應的事件，或者忽略不處理。

如同週期回呼函數，在函數內存取 ProcessData 可以確保資料的同步性。

5.5.4. 錯誤事件回呼函數(Error-Callback function)

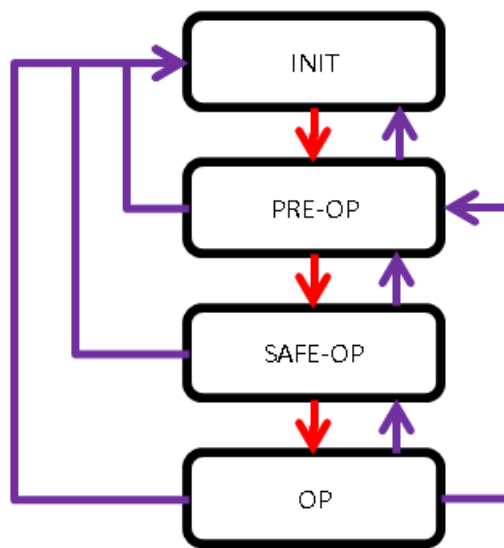
當 EC-Master 發生錯誤事件時，EC-Master 會同步呼叫此函數通知使用者，同時傳入錯誤代碼(Error code)，使用者可以根據錯誤代碼來處理對應的事件，或者忽略不處理。

如同週期回呼函數，在函數內存取 ProcessData 可以確保資料的同步性。

5.6. EtherCAT 狀態機

根據 EtherCAT 標準文件(ETG.1000.6)，EtherCAT 主站必須具備狀態機(EtherCAT State Machine, 簡稱 ESM)，來負責處理主站與各從站(EC-Slaves)之間從初始化狀態到可操作狀態的工作流程。其狀態圖如下圖所示，包含四種狀態：

1. INIT 狀態
2. PREOP 狀態
3. SAFEOP 狀態
4. OP 狀態



EtherCAT State Machine diagram

一般而言在 EtherCAT 的應用中，在進行工業控制流程之前，必須將狀態機的狀態從“INIT”狀態變更為“OP”狀態，此動作所隱含的意義在於初始化 EC-Master 網路組態和 EC-Slaves 模組相關設定，其設定的內容是根據 NexCAT 公用程式所輸出的 ENI (EtherCAT Network Information) 檔案。

5.6.1. ESM 狀態變更

啟動 EC-Master 通訊後，`NEC_RtChangeStateToOP()`將狀態切換至“OP”狀態。若有特殊應用可使用 `NEC_RtSetMasterStateWait()`切換至不同狀態。上述函數不應在 Callback 函數中使用。在 Callback 函數中應使用 `NEC_RtSetMasterState()`函數改變目標 EC-Master 狀態，`NEC_RtSetMasterState()`函數只用於發出狀態改變要求，並不會等待 EC-Master 的狀態是否成功改變，須配合使用 `NEC_RtGetMasterState()`函數檢查狀態是否正確地完成變更。關於函數的用法請參考相關函數說明。

在各個狀態下，EC-Master 所提供的服務以及 EC-Slaves 可以接受的操作指令如下表所示：

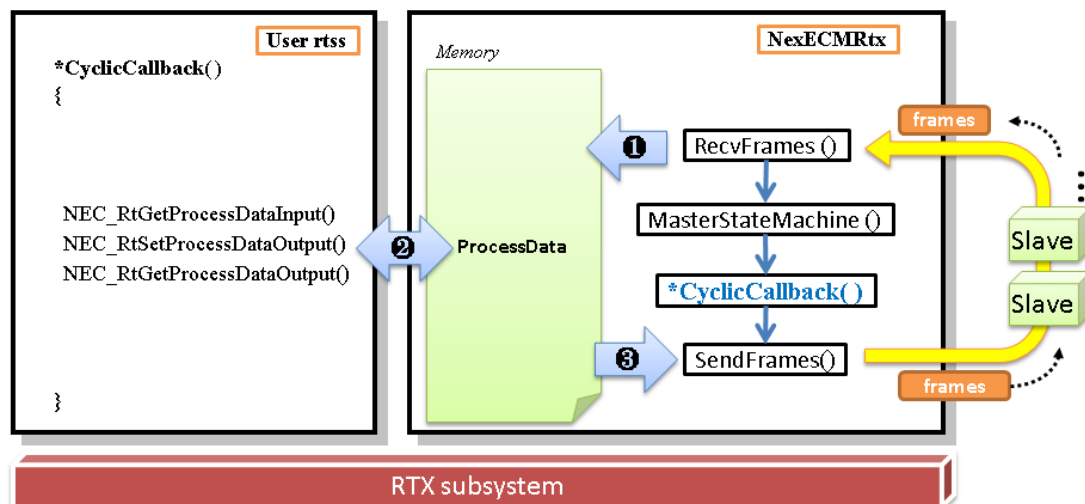
狀態	EC-Master 服務	EC-Slave 服務
INIT	<ul style="list-style-type: none"> ➤ 週期回呼函數啟動 ➤ 無 ProcessData 通訊傳輸(To slaves) ➤ 無 Mailbox 通訊傳輸(To slaves) 	<ul style="list-style-type: none"> ➤ EC-Master 不可控制
PREOP	<ul style="list-style-type: none"> ➤ 週期回呼函數啟動 ➤ Mailbox 通訊傳輸啟動(To slaves) -SDO 通訊啟動 ➤ 無 ProcessData 通訊傳輸(To slaves) 	<ul style="list-style-type: none"> ➤ Mailbox 通訊啟動
SAFEOP	<ul style="list-style-type: none"> ➤ 週期回呼函數啟動 ➤ Mailbox 通訊傳輸啟動(To slaves) -SDO 通訊啟動 ➤ ProcessDataInput 通訊傳輸啟動 ➤ 無 ProcessDataOutput 通訊傳輸 	<ul style="list-style-type: none"> ➤ Mailbox 通訊啟動 ➤ 將 Input 資料更新到 ProcessDataInput
OP	<ul style="list-style-type: none"> ➤ 週期回呼函數啟動 ➤ Mailbox 通訊傳輸啟動(To slaves) -SDO 通訊啟動 ➤ ProcessDataInput 通訊傳輸啟動 ➤ ProcessDataOutput 通訊傳輸啟動 	<ul style="list-style-type: none"> ➤ Mailbox 通訊啟動 ➤ 將 Input 資料更新到 ProcessDataInput ➤ 從 ProcessDataOutput 取得 Output 資料並輸出

5.7. Process Data 存取

5.7.1. ProcessData 運作機制

NexECMRtx 在內部維護一塊記憶體供 EtherCAT 通訊 ProcessData 使用，ProcessData 上的資料會根據通訊週期時間定期的傳輸及更新資料。如下圖所示 EC-Master 定期的執行：

1. 接收 EC-Slaves 資料，寫入 ProcessData 記憶體中
2. EthreCAT 狀態機處理
3. 呼叫週期 Callback 函數
4. 將 ProcessData 的資料傳送資料到 EC-Slaves



可利用 `NEC_RtGetProcessDataPtr()` 函數直接取得 ProcessData 的記憶體指標，用戶的程式可直接訪問此內部記憶體。此方式其優點在於有較高的存取效率，但必須自行注意存取的範圍以及存取的時機，當存取錯誤時可能會導致系統崩潰。另一較安全的存取方式是使用下列 API 方式存取 ProcessData，API 內部會檢查存取的範圍是否正確。

```

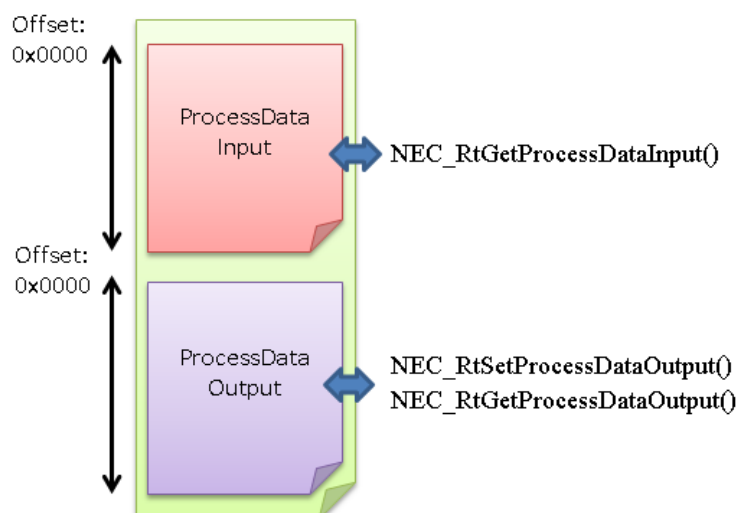
NEC_RtSetProcessDataOutput();
NEC_RtGetProcessDataOutput();
NEC_RtGetProcessDataInput();
  
```

關於 Process Data 的存取時機，建議在 Callback 程序之中來存取，以確保資料全區塊的同步完整性，若在 Callback 函數之外存取 ProcessData，資料的同步性只有 1 Byte 為單位。

ProcessData 在內部實際分為 ProcessDataInput 區域和 ProcessDataOutput 區域。其傳遞資料的方向如下表：

ProcessData	資料方向	Read/Write
ProcessDataInput	EC-Slaves 傳送 EC-Master	Read Only
ProcessDataOutput	EC-Master 傳送至 EC-Slaves	Read/Write

存取所對應的 API 如下圖表示:

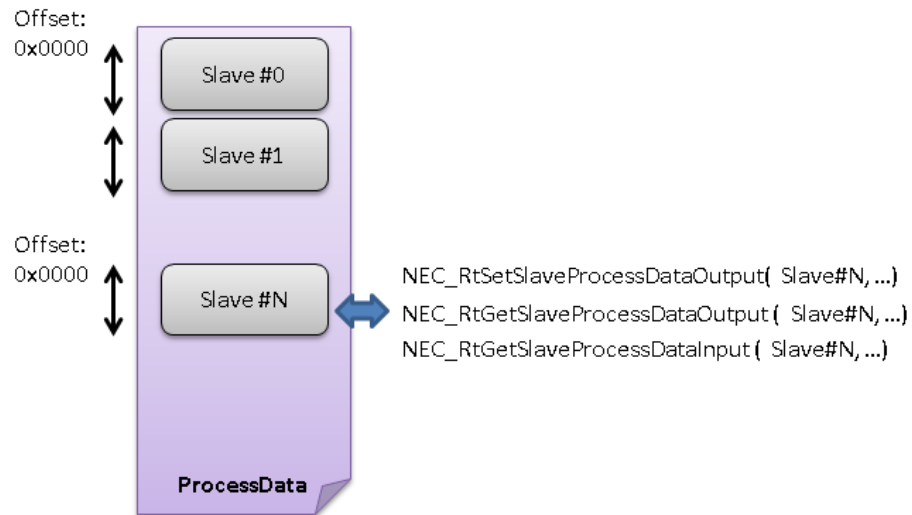


5.7.2. ProcessData 資料內容

ProcessData 內的資料內容根據不同的 EC-Slaves 模組所提供的內容有所差異。例如 DIO 類的 EC-Slave 模組其 ProcessData 為 Digital Input 的狀態值或 Digital Output 的輸出資料。而功能較多的 EC-Slave 模組例如伺服驅動器，其 ProcessData 的資料可以是“目標位置”(Target Position)， “馬達實際位置” (Position actual value)等。

這些資料排放在 ProcessData 上的位置(Offset)根據模組串接的順序和其所提供資料的長度的大小所決定，如下圖所示，每個 EC-Slaves 在 ProcessData 之中會有各自的記憶體區塊。

若要同步輸出資料到不同模組上，可在 Callback 函數中存取這些記憶體即可達到此效果。



可以使用下列函數存取 Slave 各自的 ProcessData 記憶體區域

```
NEC_RtSetSlaveProcessDataOutput ();
NEC_RtGetSlaveProcessDataOutput ();
NEC_RtGetSlaveProcessDataInput ()
```

5.8. Mailbox 通訊

5.8.1. CoE -SDO 通訊

CANOpen 技術在工業自動化應用中已經是一相當普遍且成熟的技術，為了和既有控制技術結合，EtherCAT 提供 CANOpen over EtherCAT (CoE)來實現。在 CANOpen 中主要定義了兩種傳輸方式：

1. PDO (Process data object), 即時同步資料傳輸
2. SDO (Service data object), 非同步資料傳輸

PDO 的資料會直接映射到 EtherCAT 的 ProcessData 區域，而 SDO 部分則使用 EtherCAT 的 Mailbox 機制達成。

根據 CANOpen 的 SDO 傳輸可區分為

1. SDO download: 主站將資料傳輸到從站
2. SDO upload: 資料由從站上傳到主站
3. SDO information: 取得 ObjectDictionary List

SDO 特性如下：

1. SDO 傳輸採用交握(Handshake)的方式
2. 需要數個通訊週期才能完成一次 SDO 傳輸
3. 當 SDO 傳輸成功表示證資料有正確的交換到另一端
4. 一般用在非即時性的參數設定

NexECMRtx 提供下列函數來完成 SDO 的通訊傳輸，

函數名稱	說明	T
NEC_RtStartSDODownload	送出 SDO download 命令資料(Master to slave)	B
NEC_RtStartSDOUpload	送出 SDO upload 命令資料(Slave to Master)	B
NEC_RtSDODownload	執行 SDO download (Structure)	X
NEC_RtSDOUpload	執行 SDO upload (Structure)	X
NEC_RtSDODownloadEx	執行 SDO download (Native data type)	X
NEC_RtSDOUploadEx	執行 SDO upload (Native data type)	X

API 詳細的使用方式請參考 6.7 小節說明。

5.8.2. CoE -Emergency

當 EC-Slaves 有錯誤發生時，會透過 CoE 通訊傳遞 Emergency 訊息至 EC-Master。EC-Master 收到此 Emergency 後，會儲存至內部的 Emergency 訊息容器中，若用戶有註冊事件回呼函數，則事件回呼函數會被 EC-Master 呼叫。Emergency 引發事件回呼函數的規則如下：

1. 當有新的 Emergency 訊息新增至訊息容器時，事件回呼函數會被引用。
2. 同一時間點，EC-Master 接收來自多個 EC-Slaves 的 Emergency 訊息，此時回呼事件函數僅會被呼叫一次。
3. 若無新的事件發生，則雖然 Emergency 訊息容器含有未讀走的訊息，事件回呼函數不會再次被呼叫。

若 Emergency 訊息容器已滿，再來的訊息會覆蓋掉最舊的訊息。

NexECMRtx 提供下列函數來完成讀取 EC-Master 內部中的 Emergency 訊息，

函數名稱	說明	T
NEC_RtEmgDataCount	讀取 Emergency 訊息容器中訊息資料數量	B
NEC_RtEmgData	讀取 Emergency 訊息容器中訊息資料	B

API 詳細的使用方式請參考 6.7 小節說明。

6. NexECMRtx 程式庫

6.1. RTX API 總覽

下表列出 NexECMRtx 函式庫 API 的列表，API 定義於 NexECMRtx.h 標頭檔之中。

“T 欄位(Type)”代表該函式可被呼叫的位置

C :只能在 Callback 函式中呼叫

X :不能再 Callback 函式中呼叫

B :沒有限制

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

函數名稱	說明	T
初始化相關函式		
NEC_RtGetVersion	取得 NexECMRtx 目前版本資訊	B
NEC_RtRetVer	回傳 NexECMRtx 目前版本資訊	B
NEC_RtInitMaster	初始化 NexECMRtx	X
NEC_RtCloseMaster	關閉 NexECMRtx (EC-Master)	X
NEC_RtSetParameter	設置 EC-Master 參數	X
NEC_RtGetParameter	讀取 EC-Master 參數	X
NEC_RtRegisterClient	註冊 Callback clients 函數	X
NEC_RtUnregisterClient	取消 Callback client 的註冊	X
NEC_RtGetProcessDataPtr	取得 EC-Mater 中 ProcessData 記憶體指標	B
NEC_RtSetProcessDataPtrSource	設置 ProcessData 記憶體的來源	X
EC-Master 控制相關函式		
NEC_RtStartMaster	啟動 EC-Master 通訊	X
NEC_RtStopMaster	停止 EC-Master 週期通訊	X
NEC_RtStopMasterCb	發出 EC-Master 通訊中止要求	C
NEC_RtWaitMasterStop	等待 EC-Master 通訊中止要求並停止通訊	X
NEC_RtGetMasterState	取得 EC-Master 目前的狀態	B
NEC_RtSetMasterState	發出 EC-Master 狀態變更要求	B
NEC_RtChangeStateToOP	阻塞式變更 EC-Master 狀態成為“OP”狀態	X
NEC_RtSetMasterStateWait	阻塞式變更 EC-Master 狀態	X
NEC_RtGetStateError	讀取狀態錯誤代碼	B
Process data 存取相關函式		
NEC_RtSetProcessDataOutput	寫入 ProcessDataOutput 資料	B
NEC_RtGetProcessDataOutput	讀取 ProcessDataOutput 資料	B
NEC_RtGetProcessDataInput	讀取 ProcessDataInput 資料	B

NEC_RtSetSlaveProcessDataOutput	寫入 EC-Slave 的 ProcessDataOutput 資料	B
NEC_RtGetSlaveProcessDataOutput	讀取 EC-Slave 的 ProcessDataOutput 資料	B
NEC_RtGetSlaveProcessDataInput	讀取 EC-Slave 的 ProcessDataInput 資料	B
Callback 函式 (函式原型)		
*NEC_RtCyclicCallback	週期回呼函數	C
*NEC_RtEventCallback	事件回呼函數	C
*NEC_RtErrorCallback	錯誤事件回呼函數	C
ENI 相關函式		
NEC_RtGetSlaveCount	讀取 EC-Slaves 個數	B
NEC_RtGetSlaveInformation	讀取某個EC-Slaves 的資訊	B
NEC_RtGetSlaveState	讀取多個EC-Slaves 的狀態	B
CoE 通訊相關函式		
NEC_RtStartSDODownload	送出 SDO download 命令資料(Master to slave)	B
NEC_RtStartSDOUpload	送出 SDO upload 命令資料(Slave to Master)	B
NEC_RtSDODownload	執行 SDO download (Structure)	X
NEC_RtSDOUpload	執行 SDO upload (Structure)	X
NEC_RtSDODownloadEx	執行 SDO download (Native data type)	X
NEC_RtSDOUploadEx	執行 SDO upload (Native data type)	X
存取從站模組硬體資訊相關函式		
NEC_RtGetConfiguredAddress	取得從站 Configured Station Address	X
NEC_RtGetAliasAddress	取得從站 Configured Station Alias	X
NEC_RtGetSlaveCoeProfileNum	取得從站 ProfileNum 資訊	B

API 所使用的 C/C++ 資料型態定義於 nex_type.h 中，說明如下表：

型別	C/C++ 原型	說明	大小 byte	範圍
BOOL_T	int	布林型別	4	0:False, 1:True
U8_T	unsigned char	無號整數	1	0 ~ 255
U16_T	unsigned short	無號整數	2	0 ~ 65535
U32_T	unsigned int	無號整數	4	0 ~ 4294967295
U64_T	unsigned __int64	無號整數	8	0 ~ 18446744073709551615
I8_T	char	有號整數	1	-128 ~ 127
I16_T	short	有號整數	2	-32768 ~ 32767
I32_T	int	有號整數	4	-2147483648 ~ 2147483647
I64_T	__int64	有號整數	8	-9223372036854775808 ~ 9223372036854775807

F32_T	float	浮點數	4	IEEE-754, 有效小數後 7 位
F64_T	double	雙精浮點數	8	IEEE-754, 有效小數後 15 位
RTN_ERR	int	錯誤代碼	4	-2147483648 ~ 2147483647

6.2. 初始化相關函式

6.2.1. NEC_RtGetVersion

取得 NexECMRtx 目前版本資訊

C/C++語法:

```
RTN_ERR NEC_RtGetVersion( U32_T *Version );
```

參數:

U32_T *Version: Return the version of NexECMRtx.

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

沒有限制。

參閱::

NEC_RtRetVer()

6.2.2. NEC_RtRetVer

回傳 NexECMRtx 目前版本資訊

C/C++語法:

```
U32_T NEC_RtRetVer();
```

參數:

<無參數>

回傳值:

回傳NexECMRtx目前版本資訊。

用法:

沒有限制。

參閱::

NEC_RtGetVersion()

6.2.3. NEC_RtInitMaster

初始化 NexECMRtx

C/C++語法:

```
RTN_ERR NEC_RtInitMaster( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

使用其他 NexECMRtx 函式前，先調用此函數進行函市庫內部初始化。

注意! 禁止於 Callback 函式中調用此函數

參閱::

NEC_RtCloseMaster()

6.2.4. NEC_RtCloseMaster

關閉 NexECMRtx (EC-Master)

C/C++語法:

```
RTN_ERR NEC_RtCloseMaster( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

應用程式結束前，調用此函數釋放 NexECMRtx 內部資源。

注意! 禁止於 Callback 函式中調用此函數

參閱::

NEC_RtInitMaste()

6.2.5. NEC_RtSetParameter

6.2.6. NEC_RtGetParameter

這兩個函數用於設置和讀取 EC-Master 參數。

C/C++語法:

```
RTN_ERR NEC_RtSetParameter( U16_T MasterId, U16_T ParaNum, I32_T
ParaData );
```

```
RTN_ERR NEC_RtGetParameter( U16_T MasterId, U16_T ParaNum, I32_T
*ParaData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T ParaNum: 指定參數代碼，請參考用法:

I32_T ParaData: 指定參數值，請參考用法:

I32_T *ParaData: 回傳參數值，請參考用法:

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

用於啟動 EC-Master 通訊之前，設定 EC-Master 通訊使用的參數，其參數列表如下。調用 *NEC_RtInitMaster()* 不會影響此區參數設定。

注意! 禁止於 Callback 函式中調用此函數

參數代碼	說明	參數值
NEC_PARA_S_ECM_CYCLETIMEUS	EC-Master 通訊週期，單位 micro-second	250 ~ 1000000
NEC_PARA_S_NIC_INDEX	指定要使用的 NIC Port，注意, Load ENI 時可能會修改此參數，請參考 CH7.4.5 <i>NEC_LoadNetworkConfig()</i>	0 ~ Max NIC port
NEC_PARA_ECM_RECV_TIMEOUT_US	定義 EtherCAT 網路封包回傳 timeout 時間，單位 micro-second。一般使用預設值即可。	250 ~ 2000000
NEC_PARA_ECM_LINK_ERR_MODE	網路斷線行為模式: LINKERR_AUTO:	LINKERR_AUTO (0) LINKERR_MANUAL

	<p>當 EC-Slave(s) 斷線，EC-Master 自動偵測斷線的裝置。當裝置從新連線自動將 EC-Slaves 初始化並設定為"OP"狀態</p> <p>LINKERR_MANUAL:</p> <p>當某裝置斷線，其狀態會停留在 ERROR 狀態。當斷線裝置恢復連線，不會從新初始化。</p> <p>LINKERR_STOP:</p> <p>只要有一裝置斷線，EC-Master 將網路中斷，並進入 ERROR 狀態。</p>	<p>(1)</p> <p>LINKERR_STOP (2)</p>
NEC_PARA_ECM_DC_CYC_TIME_MODE	<p>DC 時間設定模式:</p> <p>0: 根據 Master cycle time (預設)</p> <p>1: 根據 ENI 資料</p>	0~1

參閱:

NEC_RtGetStateError(); NEC_RtStartMaster()

6.2.7. NEC_RtRegisterClient

註冊 Callback clients 函數

C/C++語法:

```
RTN_ERR NEC_RtRegisterClient( U16_T MasterId, TClintParam *ClientParam );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

TClintParam *ClientParam: Callback client 資料

```
typedef struct
{
    U32_T version;
    void *userDataPtr;
    NEC_RtCyclicCallback cyclicCallback;
    NEC_RtEventCallback eventCallback;
    NEC_RtErrorCallback errorCallback;
    U16_T clientID;
} TClintParam;
```

U32_T version: 傳入前設定版本，使用 *NEC_RtRetVer()*

void *userDataPtr:

傳入用戶自訂的資料結構指標(Pointer)，可設 0 表示不傳入。EC-Master 會記住，並在呼叫 Callback 函式時將該指標傳入。

NEC_RtCyclicCallback cyclicCallback:

傳入 Callback 函數指標。可設定為 0 表示不使用。此 Callback 函數在啟動 EC-Master 後會被周期性的呼叫。

NEC_RtEventCallback eventCallback:

傳入 Callback 函數指標。可設定為 0 表示不使用。此 Callback 函數在啟動 EC-Master 後當預設事件產生後會被呼叫。

NEC_RtErrorCallback errorCallback:

傳入 Callback 函數指標。可設定為 0 表示不使用。此 Callback 函數在啟動 EC-Master 後當預設錯誤事件產生後會被呼叫。

U16_T clientID: 保留 NexECMRtx 內部使用。請勿變更此參數。

(*) 關於 Callback 函數的使用，請參考 5.5 小節的說明。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

必須在啟動 EC-Master 通訊之前調用此函數。若要從新註冊或 EC-Master 通訊結束後，必須呼叫 `NEC_RtUnregisterClient()` 取消原先的 Callback client 註冊。

注意! 禁止於 Callback 函式中調用此函數

參閱::

`NEC_RtStartMaster (); NEC_RtUnregisterClient();NEC_RtStopMaster();`

6.2.8. NEC_RtUnregisterClient

取消 Callback client 的註冊

C/C++語法:

```
RTN_ERR NEC_RtUnregisterClient( U16_T MasterId, TClintParam *ClientParam );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

TClintParam *ClientParam: 請參閱 *NEC_RtRegisterClient()* 函數說明。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

若要重新註冊或 EC-Master 通訊結束後，必須呼叫 *NEC_RtUnregisterClient()* 取消原先的 Callback client 註冊。一般情況下，請在 *NEC_RtWaitMasterStop()*之後取消註冊，請勿在 EC-Master 通訊過程中取銷註冊。

注意! 禁止於 Callback 函式中調用此函數

參閱::

NEC_RtRegisterClient(); *NEC_RtWaitMasterStop()*

6.2.9. NEC_RtGetProcessDataPtr

取得 EC-Mater 中 ProcessData 記憶體指標(Pointer)

C/C++語法:

```
RTN_ERR NEC_RtGetProcessDataPtr( U16_T MasterId, U8_T **InputProcessDataPtr,
U32_T *InPDSIZEInByte, U8_T **OutputProcessDataPtr, U32_T *OutPDSIZEInByte );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U8_T **InputProcessDataPtr:

回傳 process data input (Slaves to master)記憶體指標，設定為 0 忽略

U32_T *InPDSIZEInByte:

回傳 process data input (Slaves to master)記憶體指標可存取長度，設定為 0 忽略

U8_T **OutputProcessDataPtr:

回傳 process data output (master to slave)記憶體指標，設定為 0 忽略

U32_T *OutPDSIZEInByte:

回傳 process data input (Slaves to master)記憶體指標可存取長度，設定為 0 忽略

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

為提高 ProcessData 資料交換效率，使用此功能來獲得記憶體指標，始用者程式可直接訪問內部記憶體。但須注意，用戶必須非常小心地訪問此塊記憶體，當訪問衝突可能會導致系統崩潰。直接存取 ProcessData 資料必須在 Callback 函式中進行，才能確保數據的一致性(Consistance)，相關說明請參考 5.5 callback 函式和 5.7 小節 ProcessData 存取。

參閱::

```
NEC_RtSetSlaveProcessDataOutput (); NEC_RtGetSlaveProcessDataOutput ();
NEC_RtGetSlaveProcessDataInput ();NEC_RtSetProcessDataPtrSource()
```


6.2.10. NEC_RtSetProcessDataPtrSource

設置 ProcessData 記憶體의來源

C/C++語法:

```
RTN_ERR NEC_RtSetProcessDataPtrSource( U16_T MasterId, void
*InputProcessDataPtrSource, U32_T InPDSIZEInByte, void
*OutputProcessDataPtrSource, U32_T OutPDSIZEInByte );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

void *InputProcessDataPtrSource:

設定 Process data input 記憶體由外部提供，若設定 0 表示使用內部記憶體

U32_T InPDSIZEInByte:

Process data input 記憶體的大小，單位 Byte

void *OutputProcessDataPtrSource:

設定 Process data output 記憶體由外部提供，若設定 0 表示使用內部記憶體

U32_T OutPDSIZEInByte:

Process data output 記憶體的大小，單位 Byte

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

EC-Master 內定是以內部記憶體當作 ProcessData 來源可使用

NEC_RtGetProcessDataPtr()取得內部記憶體指標加以存取。另一種方式為採用 NEC_RtSetProcessDataPtrSource()設定外部記憶體當作 ProcessData 的使用空間，此函是必須在啟動通訊前使用，啟動通訊後不能修改。當通訊停止後，ProcessData 自動恢復使用內部記憶體方式，因此若需再此啟動通訊，必須重新呼叫此函數從新設定。

注意! 當 EC-Master 停止通訊後，用戶的 RTX 應用程式卸載之前必須呼叫 NEC_RtSetProcessDataPtrSource() 還原為內部記憶體，以避免記憶體存取錯誤。

直接存取 **ProcessData** 資料必須在 **Callback** 函式中進行，才能確保數據的一致性(Consistence)，相關說明請參考 5.5 callback 函式和 5.7 小節 **ProcessData** 存取。

參閱::

NEC_RtSetSlaveProcessDataOutput (); NEC_RtGetSlaveProcessDataOutput ();

NEC_RtGetSlaveProcessDataInput (); NEC_RtGetProcessDataPtr();

6.3. EC-Master 控制相關函式

6.3.1. NEC_RtStartMaster

啟動 EC-Master 通訊。

C/C++語法:

```
RTN_ERR NEC_RtStartMaster( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

注意! 禁止於 Callback 函式中調用此函數

此函數的使用時機在於使用 *NEC_RtSetParameter()* 設定 EC-Master 參數後啟動通訊，啟動成功後，EC-Master 會建立周期的通訊機制。若有註冊 Callback 函數 *NEC_RtRegisterClient()*，Callback 函數則在啟動成功後即開始週期性的運作。停止通訊的方法:

- 1.在 Callback 函數中呼叫 *NEC_RtStopMasterCb()*或
- 2.非 Callback 函數中使用 *NEC_RtStopMaster()*

參閱:

NEC_RtSetParameter(); NEC_RtRegisterClient(); NEC_RtStopMaster();
*NEC_RtStopMasterCb();*NEC_RtCyclicCallback()*

6.3.2. NEC_RtStopMaster

停止 EC-Master 週期通訊。

C/C++語法:

```
RTN_ERR NEC_RtStopMaster( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

注意! 禁止於 Callback 函式中調用此函數

此函數用於停止 EC-Master 週期通訊。在停止通訊前須將 EC-Master 狀態切換至“INIT”狀態，可調用 *NEC_RtSetMasterState()*。

若要在 Callback 函數中停止 EC-Master 請參考 *NEC_RtStopMasterCb()*。

參閱::

NEC_RtStartMaster(); *NEC_RtStopMasterCb()*

6.3.3. NEC_RtStopMasterCb

發出 EC-Master 通訊停止要求，使用於 Callback 函數之中。

C/C++語法:

```
RTN_ERR NEC_RtStopMasterCb( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

當用戶欲在 Callback 函數中停止 EtherCAT 主站通信。

成功呼叫此函數後並未立即停止通訊。它發出了一個信號來喚醒阻塞函數-

NEC_RtWaitMasterStop()。通訊會在 *NEC_RtWaitMasterStop()*成功返回後停止，過程中 EtherCAT 主站的狀態將變為“INIT”狀態。

參閱:

NEC_RtStartMaster();NEC_RtStopMasterCb();NEC_RtWaitMasterStop()

6.3.4. NEC_RtWaitMasterStop

等待 EC-Master 通訊中止要求並停止通訊，阻塞式(Blocked)函數。

C/C++語法:

```
RTN_ERR NEC_RtWaitMasterStop( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

主功能有二:

1. 讓主程式進入睡眠等待
2. 收到停止訊號後，將 EC-Master 狀態機切換至“INIT”狀態。

多數的情況下，用戶主要將工業控制程序實作在週期的 Callback()函數之中，然而 RTX 的應用程式為 Console 主程式的架構，主程式的進入點為 main()函數，當 main()函數返回後，RTX 的應用程式即結束。為避免應用程式結束，使用者可以呼叫此函數來等待 Callback()函數中的 NEC_RtStopMasterCb()所發出的停止 EC-Master 訊號。

注意! 禁止於 Callback 函式中調用此函數

參閱:

NEC_RtStopMasterCb()

6.3.5. NEC_RtGetMasterState

6.3.6. NEC_RtSetMasterState

NEC_RtGetMasterState()：取得 EC-Master 目前的狀態(狀態機)

NEC_RtSetMasterState()：發出 EC-Master 狀態變更要求

C/C++語法:

```
RTN_ERR NEC_RtGetMasterState( U16_T MasterId, U16_T *State );
```

```
RTN_ERR NEC_RtSetMasterState( U16_T MasterId, U16_T State );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T *State: 回傳目前狀態。請參考下列定義:

ECM_STA_INIT	(1)
ECM_STA_PREOP	(2)
ECM_STA_SAFEOP	(3)
ECM_STA_OPERATION	(4)
ECM_STA_ERROR	(6)

U16_T State：設定目標定義。可設定範圍請參考下列定義:

ECM_STA_INIT	(1)
ECM_STA_PREOP	(2)
ECM_STA_SAFEOP	(3)
ECM_STA_OPERATION	(4)

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

啟動 EC-Master 通訊後，使用 *NEC_RtSetMasterState()*函式改變目標 EC-Master 狀態，*NEC_RtSetMasterState()*函數可在 callback 函式中呼叫，只用於發出狀態改變要求，並不會等待 EC-Master 的狀態是否成功改變，須配合使用

*NEC_RtGetMasterState()*函數檢查狀態是否正確地完成變更。

另外也可使用阻塞式函數來改變 EC-Master 狀態，請參閱

NEC_RtSetMasterStateWait(), *NEC_RtChangeStateToOP()*

關於 EC-Master 狀態機的說明請參考 5.6 小節 “EtherCAT 狀態機” 的說明。

參閱：

`NEC_RtStartMaster(); NEC_RtSetMasterStateWait(), NEC_RtChangeStateToOP()`

6.3.7. NEC_RtChangeStateToOP

阻塞式函數，切換 EC-Master 狀態成為”OP”狀態

C/C++語法:

```
RTN_ERR NEC_RtChangeStateToOP ( U16_T MasterId, I32_T TimeoutMs );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

I32_T TimeoutMs: 逾時等待時間，單位 millisecond。

設定 0 其效果等同 *NEC_RtGetMasterState()*，設定-1 表示永不逾時。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

大部分的應用程序是在 EC-Master 狀態為:”OP”狀態中進行。可使用此函數來切換至”OP”狀態。用法類似 *NEC_RtSetMasterState()* 啟動 EC-Master 通訊後用來改變目標 EC-Master 狀態。不同之處在於可設定逾時等待，函數成功返回代表狀態已成功改變至目標狀態。

注意! 禁止於 Callback 函式中調用此函數

關於 EC-Master 狀態機的說明請參考 5.6 小節 ”EtherCAT 狀態機”的說明。

參閱:

NEC_RtStartMaster();NEC_RtSetMasterState (); RtSetMasterStateWait(),

6.3.8. NEC_RtSetMasterStateWait

阻塞式變更 EC-Master 狀態

C/C++語法:

```
RTN_ERR NEC_RtSetMasterStateWait( U16_T MasterId, U16_T State, I32_T
TimeoutMs );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

I32_T TimeoutMs: 逾時等待時間，單位 millisecond。

設定 0 其效果等同 *NEC_RtGetMasterState()*，設定-1 表示永不逾時。

U16_T State：設定目標定義。請參考下列定義：

```
#define ECM_STA_INIT           (1)
#define ECM_STA_PREOP         (2)
#define ECM_STA_SAFEOP        (3)
#define ECM_STA_OPERATION     (4)
```

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

使用此函數來切換 EC-Master 狀態。用法類似 *NEC_RtSetMasterState()* 啟動 EC-Master 通訊後用來改變目標 EC-Master 狀態。不同之處在於可設定逾時等待，函數成功返回代表狀態已成功改變至目標狀態。

注意! 禁止於 Callback 函式中調用此函數

關於 EC-Master 狀態機的說明請參考 5.6 小節 “EtherCAT 狀態機”的說明。

參閱:

NEC_RtStartMaster();NEC_RtSetMasterState ();NEC_RtChangeStateToOP (),

6.3.9. NEC_RtGetStateError

讀取狀態錯誤代碼

C/C++語法:

```
RTN_ERR NEC_RtGetStateError( U16_T MasterId, I32_T *Code );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

I32_T *Code: 回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

當 EC-Master 狀態異常(如通訊斷線)，狀態會改變為 *ECM_STA_ERROR* 狀態，EC-Master 會記錄狀態異常(錯誤代碼)，可使用此函數讀取狀態異常紀錄，用於除錯。

參閱:

NEC_RtSetMasterState (); *RtSetMasterStateWait()*, NEC_RtGetMasterState()

6.4. Process data 存取相關函式

6.4.1. NEC_RtSetProcessDataOutput

6.4.2. NEC_RtGetProcessDataOutput

6.4.3. NEC_RtGetProcessDataInput

存取 EC-Master 中的 ProcessData 記憶體

C/C++語法:

```
RTN_ERR NEC_RtSetProcessDataOutput( U16_T MasterId, U32_T PDOAddr, U32_T
LengthOfData, U8_T *SetData );
```

```
RTN_ERR NEC_RtGetProcessDataOutput( U16_T MasterId, U32_T PDOAddr, U32_T
LengthOfData, U8_T *RetData );
```

```
RTN_ERR NEC_RtGetProcessDataInput( U16_T MasterId, U32_T PDIAddr, U32_T
LengthOfData, U8_T *RetData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U32_T PDOAddr: ProcessData 記憶體偏移位置，單位 Byte

U32_T LengthOfData: 資料存取大小，單位 Byte

U8_T *SetData: 寫入資料指標

U8_T *RetData: 讀取資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此函數只能用在 Callback 函數之中，用來讀取或寫入資料到 ProcessData 之中。

若在 Callback 函數外使用，則不保證讀取或寫入資料的一致性。

關於 ProcessData 存取，請參考 5.7 小節 Process Data 存取

參閱:

```
NEC_RtGetProcessDataPtr (); NEC_RtSetSlaveProcessDataOutput ();  
NEC_RtGetSlaveProcessDataOutput (); NEC_RtGetSlaveProcessDataInput ()
```

6.4.4. NEC_RtSetSlaveProcessDataOutput

6.4.5. NEC_RtGetSlaveProcessDataOutput

6.4.6. NEC_RtGetSlaveProcessDataInput

存取 EC-Slave 的 ProcessData 資料。

C/C++語法:

```
RTN_ERR NEC_RtSetSlaveProcessDataOutput( U16_T MasterId, U16_T SlaveAddr,  
U16_T Offset, U8_T *Data, U32_T Size );
```

```
RTN_ERR NEC_RtGetSlaveProcessDataOutput( U16_T MasterId, U16_T SlaveAddr,  
U16_T Offset, U8_T *Data, U32_T Size );
```

```
RTN_ERR NEC_RtGetSlaveProcessDataInput( U16_T MasterId, U16_T SlaveAddr,  
U16_T Offset, U8_T *Data, U32_T Size );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T Offset: 該 EC-Slave ProcessData 記憶體偏移量，從 0 開始，單位 Byte

U8_T *Data: 資料指標

U32_T Size: 資料長度，單位 Byte

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此函數只能用在 Callback 函數之中，用來讀取或寫入資料到 EC-Slave 所佔的 ProcessData 記憶體之中。若在 Callback 函數外使用，則不保證讀取或寫入資料的一致性。關於 ProcessData 存取，請參考 5.7 小節 Process Data 存取

參閱:

```
NEC_RtSetProcessDataOutput(); NEC_RtGetProcessDataOutput();
```

NEC_RtGetProcessDataInput()

6.5. Callback 函式

6.5.1. *NEC_RtCyclicCallback

週期回呼函數

C/C++語法:

```
void (*NEC_RtCyclicCallback)( void *UserDataPtr );
```

參數:

void *UserDataPtr: 傳入用戶所定義的資料指標

回傳值:

void 無回傳值

用法:

啟動通訊前必須使用 *NEC_RtRegisterClient()*將此 Callback 函數註冊到 EC-Master 之中。啟動通訊後此函數週期性的被調用，並傳入用戶的資料指標。

結束通訊後使用 *NEC_RtUnregisterClient()*取消 Callback 函數的註冊。避免在通訊過程中取消註冊。

(*) 關於 Callback 函數的使用，請參考 5.5 小節“Callback 函式”的說明。

參閱:

NEC_RtRegisterClient();NEC_RtUnregisterClient();

6.5.2. *NEC_RtEventCallback

事件回呼函數

C/C++語法:

```
void (*NEC_RtEventCallback) ( void *UserDataPtr, U32_T EventCode );
```

參數:

void *UserDataPtr: 傳入用戶的資料指標

U32_T EventCode: 傳入事件代碼，請參考用法說明。

回傳值:

void 無回傳值

用法:

啟動通訊前必須使用 *NEC_RtRegisterClient()*將此 Callback 函數註冊到 EC-Master 之中。啟動通訊後當下表示件發生時，此 Callback 函數會被呼叫，使用者可在 Callback 函數中撰寫事件處理程序。

(*)事件代號定義在 RtDataStructDef.h 之中

事件代號	說明
EVENT_ECM_STATE_CHANGE	EC-Master 狀態改變事件
EVENT_ECM_CNECT_FINISH	所有 EC-Slaves 的狀態“Operational” state.

結束通訊後使用 *NEC_RtUnregisterClient()*取消 Callback 函數的註冊。避免在通訊過程中取消註冊。

(*) 關於 Callback 函數的使用，請參考 5.5 小節“Callback 函式”的說明。

參閱:

NEC_RtRegisterClient();NEC_RtUnregisterClient();

6.5.3. *NEC_RtErrorCallback

錯誤事件回呼函數

C/C++語法:

```
void (*NEC_RtErrorCallback) ( void *UserDataPtr, I32_T ErrorCode );
```

參數:

void *UserDataPtr: 傳入用戶的資料指標

I32_T ErrorCode: 錯誤代碼，定義於 EcErrors.h 標頭檔中

回傳值:

void 無回傳值

用法:

啟動通訊前必須使用 *NEC_RtRegisterClient()*將此 Callback 函數註冊到 EC-Master 之中。啟動通訊後當錯誤事件產生，此 Callback 函數會被呼叫，使用者可在 Callback 函數中撰寫錯誤事件處理程序。

結束通訊後使用 *NEC_RtUnregisterClient()*取消 Callback 函數的註冊。避免在通訊過程中取消註冊。

(*) 關於 Callback 函數的使用，請參考 5.5 小節“Callback 函式”的說明。

參閱:

NEC_RtRegisterClient(); *NEC_RtUnregisterClient()*;

6.6. ENI 相關函式

6.6.1. NEC_RtGetSlaveCount

讀取 EC-Slaves 個數

C/C++語法:

```
RTN_ERR NEC_RtGetSlaveCount( U16_T MasterId, U16_T *Count );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T *Count: 回傳總 EC-Slaves 數量

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

當 ENI 下載至 EC-Master 後，使用此函數讀取 EC-Slave 的個數，此資訊來自 ENI 的內容。

參閱:

<無參閱>

6.6.2. NEC_RtGetSlaveInformation

讀取某個 EC-Slaves 的資訊

C/C++語法:

```
RTN_ERR NEC_RtGetSlaveInformation( U16_T MasterId, U16_T SlaveAddr,
SLAVE_INFO *pSlaveInfo );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

SLAVE_INFO *pSlaveInfo: 回傳 Slave 結構資訊，資料結構定義如下:

結構定義於: RtDataStructDef.h

```
typedef struct
{
    U16_T autoIncAddr;
    U16_T configAddr;
    U32_T vendorId;
    U32_T productCode;
    U32_T revisionNo;
    U32_T serialNo;
    U16_T DL_Status;
    U16_T res; //Reserved set 0
} SLAVE_INFO;
```

U16_T autoIncAddr: EtherCAT auto incremental address

U16_T configAddr: EtherCAT config address

U32_T vendorId: EtherCAT slave vendor ID

U32_T productCode: EtherCAT slave product code

U32_T revisionNo: EtherCAT slave revision number

U32_T serialNo: EtherCAT slave serial number

U16_T DL_Status: ESC register value (offset:0x0110)

注意:當Slave的狀態由INIT轉換至PREOP時DL_status的值才會被更新

U16_T res: 保留內部使用

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

當 ENI 下載至 EC-Master 後，使用此函數讀取 EC-Slave 的相關資訊，此資訊來自 ENI 的內容(DL_Status 除外)。

參閱:

<無參閱>

6.6.3. NEC_RtGetSlaveState

讀取多個 EC-Slaves 的狀態

C/C++語法:

```
RTN_ERR NEC_RtGetSlaveState( U16_T MasterId, U16_T SlaveIndex, U8_T *StateArr,
U16_T *ArrLen );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveIndex: 指定目標 EC-Slave 代號，從 0 開始依序增號，起始的 Slave 代號

U8_T *StateArr: 回傳的狀態陣列：狀態值定義如下：

Define	value	Description
STATE_STOPPED	0	Slave in INIT, PreOP, SafeOP state
STATE_OPERATIONAL	1	Slave in OP state
STATE_ERROR	2	Slave is in error state
STATE_SLAVE_RETRY	3	Slave is in re-connect state

U16_T *ArrLen: 傳入的狀態陣列大小，回傳實際有效的陣列大小

如果傳入的陣列大小 < (小於) 實際 Slave 的個數，會以傳入的大小為主。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函式可一次讀回多個 slave 的狀態，例如:

```
U16_T SlaveIndex = 0; // From first slave (0)
U8_T StateArr[6];     // If you have 6 slaves on line.
U16_T ArrLen = 6;
NEC_RtGetSlaveState( MasterId, SlaveIndex, StateArr, &ArrLen );
```

參閱:

<無參閱>

6.7. CoE 通訊相關函式

6.7.1. NEC_RtStartSDODownload

6.7.2. NEC_RtStartSDOUpload

存取 EC-Slaves CoE 參數

C/C++語法:

```
RTN_ERR NEC_RtStartSDODownload( U16_T MasterId, U16_T SlaveAddr, TSDO
*HSdo );
RTN_ERR NEC_RtStartSDOUpload( U16_T MasterId, U16_T SlaveAddr, TSDO *HSdo );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TSDO *HSdo: SDO 結構資料指標

```
typedef struct
{
    U16_T index;
    U8_T subIndex;
    U8_T ctrlFlag;
    U8_T *dataPtr;
    U16_T dataLenByte;
    U16_T status;
    I32_T abortCode;
} TSDO;
```

U16_T index: CANOpen Object index

U8_T subIndex: CANOpen Object sub-index

U8_T ctrlFlag: Reserved, 請設定為 0

U8_T *dataPtr: Download 或 Upload 的資料指標

U16_T dataLenByte: 資料長度

U16_T status: SDO 傳輸的狀態

- SDO_INIT (0) // SDO 通訊中
- SDO_SUCCESS (1) // SDO 通訊完成
- SDO_FAIL (2) // SDO 通訊失敗, EC-Master 回傳 Error code
- SDO_ABORT (3) // SDO 通訊失敗, EC-Slave 回傳 Abort code

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於送出一個 SDO Download 或 SDO Upload 的要求，送出後函數立即返回。此函數適用於 Callback 函數之中。

SDO 命令一般需要數個通訊週期的時間來完成，使用者必須檢查 SDO 的傳輸狀態(TSDO.status)是否由 SDO_INIT(0)值改變成非(0)SDO_INIT 值。

參閱:

NEC_RtSDODownload(); NEC_RtSDOUpload(); NEC_RtSDODownloadEx();
NEC_RtSDOUploadEx()

6.7.3. NEC_RtSDODownload**6.7.4. NEC_RtSDOUpload****6.7.5. NEC_RtSDODownloadEx****6.7.6. NEC_RtSDOUploadEx**

存取 EC-Slaves CoE 參數

C/C++語法:

```
RTN_ERR NEC_RtSDODownload( U16_T MasterId, U16_T SlaveAddr, TSDO *HSdo );
RTN_ERR NEC_RtSDOUpload( U16_T MasterId, U16_T SlaveAddr, TSDO *HSdo );
RTN_ERR NEC_RtSDODownloadEx( U16_T MasterId, U16_T SlaveAddr
    , U16_T Index, U8_T SubIndex , U8_T CtrlFlag
    , U32_T DataLenByte, U8_T *DataPtr, I32_T *AbortCode );
RTN_ERR NEC_RtSDOUploadEx( U16_T MasterId, U16_T SlaveAddr
    , U16_T Index, U8_T SubIndex , U8_T CtrlFlag
    , U32_T DataLenByte, U8_T *DataPtr, I32_T *AbortCode );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TSDO *HSdo: SDO 結構資料指標。請參閱 6.7.1 *NEC_RtStartSDODownload()* 參數說明

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

注意! 禁止於 Callback 函式中調用此函數

本函數用於完成一個 SDO Download 或 SDO Upload 的要求，當函數成功返回表示 SDO 傳輸已經完成。由於 SDO 命令一般需要數個通訊週期的時間來完成，此函數禁止使用於 Callback 函數之中，避免程序死結(Dead-Locked)

參閱:

NEC_RtStartSDODownload(); NEC_RtStartSDOUpload()

6.7.7. NEC_RtStartGetODListCount

讀取 EC-Slave 五個物件字典(Object Dictionary)種類個別數量

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtStartGetODListCount( U16_T MasterId, U16_T SlaveAddr,
TCoEODListCount *pCoeOdListCount );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoEODListCount * pCoeOdListCount: 結構資料指標

TCoEODListCount 資料結構

```
typedef struct
{
    U16_T numOfAllObj;
    U16_T numOfRxPdoObj;
    U16_T numOfTxPdoObj;
    U16_T numOfBackupObj;
    U16_T numOfStartObj;
    U16_T status;
    I32_T abortCode;
} TCoEODListCount;
```

U16_T numOfAllObj: 所有物件字典數量.

U16_T numOfRxPdoObj: 可映射的 RxPDO 字典數量.

U16_T numOfTxPdoObj: 可映射的 TxPDO 字典數量.

U16_T numOfBackupObj: 可儲存的字典數量.

U16_T numOfStartObj: 開機載入的字典數量.

U16_T status: SDO 傳輸的狀態

- SDO_INIT (0) // SDO 通訊中
- SDO_SUCCESS (1) // SDO 通訊完成
- SDO_FAIL (2) // SDO 通訊失敗, EC-Master 回傳 Error code
- SDO_ABORT (3) // SDO 通訊失敗, EC-Slave 回傳 Abort code

I32_T abortCode: SDO abort code 或 EC-Master error code.

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，

錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於送出一個 SDO information，取得 EC-Slave 五個物件字典種類個別數量，送出後函數立即返回。此函數適用於 Callback 函數之中。

SDO information 命令一般需要數個通訊週期的時間來完成，使用者必須檢查 SDO 的傳輸狀態(TCoEODListCount.status)是否由 SDO_INIT(0)值改變成非(0)SDO_INIT 值。

參閱:

NEC_RtStartGetODList();NEC_RtStartGetObjDesc();NEC_RtStartGetEntryDesc();
NEC_RtGetEmgDataCount();NEC_RtGetEmgData();

6.7.8. NEC_RtStartGetODList

讀取 EC-Slaves 的各種類的物件字典(Object Dictionary)索引值(index)

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtStartGetODList( U16_T MasterId, U16_T SlaveAddr,
TCOEODList *pCoeOdList );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCOEODList * pCoeOdList: 結構資料指標

TCOEODList 資料結構

```
typedef struct
{
    U16_T listType;
    U16_T lenOfList;
    U16_T *plistData;
    U16_T status;
    I32_T abortCode;
} TCOEODList;
```

U16_T listType: 指定物件字典種類.

U16_T lenOfList: 指定回傳資料指標陣列長度.

U16_T *plistData: 指定回傳資料指標.

U16_T status: SDO 傳輸的狀態.

- SDO_INIT (0) // SDO 通訊中
- SDO_SUCCESS (1) // SDO 通訊完成
- SDO_FAIL (2) // SDO 通訊失敗, EC-Master 回傳 Error code
- SDO_ABORT (3) // SDO 通訊失敗, EC-Slave 回傳 Abort code

I32_T abortCode: SDO abort code 或 EC-Master error code.

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件字典種類所有物件索引值 (Index)送出後函數立即返回。此函數適用於 Callback 函數之中。

SDO information 命令一般需要數個通訊週期的時間來完成，使用者必須檢查 SDO 的傳輸狀態(TCoEODList.status)是否由 SDO_INIT(0)值改變成非(0)SDO_INIT 值。

參閱：

```
NEC_RtStartGetODListCount();NEC_RtStartGetObjDesc();NEC_RtStartGetEntryDesc();  
NEC_RtGetEmgDataCount();NEC_RtGetEmgData()
```

6.7.9. NEC_RtStartGetObjDesc

讀取 EC-Slave 單一物件的 **Object Description** 資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtStartGetObjDesc( U16_T MasterId, U16_T SlaveAddr,
TCOEObjDesc *pCoeObjDesc );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCOEODList * pCoeOdList: 結構資料指標

TCOEODList 資料結構

```
typedef struct
{
    U16_T index;
    U16_T dataType;
    U8_T maxNumOfSubIndex;
    U8_T objectCode;
    U16_T sizeOfName;
    U8_T *pName;
    U16_T reserved;
    U16_T status;
    I32_T abortCode;
} TCOEObjDesc;
```

U16_T index: 指定物件索引值

U16_T dataType: 回傳物件資料型別

U8_T maxNumOfSubIndex: 回傳物件最大子物件

U8_T objectCode: 回傳物件資料形式

U16_T sizeOfName: 指定回傳資料指標陣列長度

U8_T pName: 指定回傳資料指標

U16_T reserved: 保留

U16_T status: SDO 傳輸的狀態

- SDO_INIT (0) // SDO 通訊中
- SDO_SUCCESS (1) // SDO 通訊完成
- SDO_FAIL (2) // SDO 通訊失敗, EC-Master 回傳 Error code
- SDO_ABORT (3) // SDO 通訊失敗, EC-Slave 回傳 Abort code

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件 Object Description 資訊，送出後函數立即返回。此函數適用於 Callback 函數之中。

SDO information 命令一般需要數個通訊週期的時間來完成，使用者必須檢查 SDO 的傳輸狀態(TCoEObjDesc.status)是否由 SDO_INIT(0)值改變成非(0)SDO_INIT 值。

參閱:

NEC_RtStartGetODListCount();NEC_RtStartGetODList();NEC_RtStartGetEntryDesc();
NEC_RtGetEmgDataCount();NEC_RtGetEmgData()

6.7.10. NEC_RtStartGetEntryDesc

讀取 EC-Slave 的單一物件 Entry Description 資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtStartGetEntryDesc( U16_T MasterId, U16_T SlaveAddr,
TCoeEntryDesc *pCoeEntryDesc );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoeEntryDesc * pCoeEntryDesc: 結構資料指標

TCoeEntryDesc 資料結構

```
typedef struct
{
    U16_T index;
    U8_T subIndex;
    U8_T valueInfo;
    U16_T dataType;
    U16_T bitLength;
    U16_T objectAccess;
    U16_T sizeOfData;
    U8_T *pData;
    U16_T status;
    I32_T abortCode;
} TCoEEntryDesc;
```

U16_T index: 指定物件索引值

U8_T subIndex: 指定物件子索引值

U8_T valueInfo: 指定回傳資訊(一般設 0)

U16_T dataType: 回傳物件資料型別

U16_T bitLength: 回傳物件資料長度

U8_T objectAccess: 回傳物存取屬性

U16_T sizeOfData: 指定回傳資料指標陣列長度

U8_T pData: 指定回傳資料指標

U16_T status: SDO 傳輸的狀態

- SDO_INIT (0) // SDO 通訊中
- SDO_SUCCESS (1) // SDO 通訊完成
- SDO_FAIL (2) // SDO 通訊失敗, EC-Master 回傳 Error code
- SDO_ABORT (3) // SDO 通訊失敗, EC-Slave 回傳 Abort code

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件 Entry Description 資訊，送出後函數立即返回。此函數適用於 Callback 函數之中。

SDO information 命令一般需要數個通訊週期的時間來完成，使用者必須檢查 SDO 的傳輸狀態(TCoEEntryDesc.status)是否由 SDO_INIT(0)值改變成非(0)SDO_INIT 值。

參閱:

NEC_RtStartGetODListCount();NEC_RtStartGetODList();NEC_RtStartGetObjDesc();
NEC_RtGetEmgDataCount();NEC_RtGetEmgData()

6.7.11. NEC_RtGetEmgDataCount

讀取 EC-Master 中 Emergency 資料數量

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtGetEmgDataCount( U16_T MasterId, U16_T  
*pEmgDataCount );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T *pEmgDataCount: 資料回傳指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於取得 EC-Master Emergency 資料數量。此函數適用於 Callback 函數之中。

參閱:

NEC_RtStartGetODListCount();NEC_RtStartGetODList();NEC_RtStartGetObjDesc();
NEC_RtStartGetEntryDesc();NEC_RtGetEmgData()

6.7.12. NEC_RtGetEmgData

讀取 EC-Master 中 Emergency 資料

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtGetEmgData( U16_T MasterId, TEmgData *pEmgData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

TEmgData _T *pEmgData: 資料回傳結構指標

TEmgData 資料結構

```
typedef struct
{
    U16_T lenOfData;
    U16_T res;
    U16_T *pSlaveAddrDataArr;
    TCoEEmgMsg *pEmgMsgDataArr;
} TEmgData;
```

U16_T lenOfData: 指定資料回傳陣列指標長度.

U16_T res: 保留

U16_T *pSlaveAddrDataArr: 資料回傳結構指標

TCoEEmgMsg *pEmgMsgDataArr: 資料回傳結構指標

TCoEEmgMsg 資料結構

```
typedef struct
{
    U16_T errorCode;
    U8_T errorRegister;
    U8_T data[5];
} TCoEEmgMsg;
```

U16_T errorCode: 回傳錯誤碼

U8_T errorRegister: 回傳錯誤暫存器

U8_T data[5]: 回傳資料陣列

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於讀取 EC-Master 中的 Emergency 資料。此函數適用於 Callback 函數之中。

參閱:

NEC_RtStartGetODListCount();NEC_RtStartGetODList();NEC_RtStartGetObjDesc();
NEC_RtStartGetEntryDesc();NEC_RtGetEmgData()

6.8. 存取從站模組硬體資訊相關函式

6.8.1. NEC_RtGetConfiguredAddress

取得從站模組“Configured Station Address”

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtGetConfiguredAddress( U16_T MasterId, U16_T SlaveAddr,  
U16_T *pConfigAddr );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T *pConfigAddr: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

注意! 禁止於 Callback 函式中調用此函數

此函數用於取得指定從站模組的 Configured Station Address 資訊。例如，使用者可經由底下程式碼，得到從站模組代號 0 其 Configured Station Address

```
U16_T SlaveAddr = 0;    // The first slave  
U16_T ConfigAddr = 0;   // A variable for storing the configured address  
NEC_RtGetConfiguredAddress ( MasterId, SlaveAddr, &ConfigAddr );
```

參閱:

6.8.2. NEC_RtGetAliasAddress

取得從站模組” Configured Station Alias”

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtGetAliasAddress(U16_T MasterId, U16_T SlaveAddr, U16_T
*pAliasAddr);
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T *pAliasAddr: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

注意! 禁止於 Callback 函式中調用此函數

此函數用於取得指定從站模組的 Configured Station Alias 資訊。例如，使用者可經由底下程式碼，得到 Configured Station Alias 為 0x0021 其模組代號

```
U16_T i;
U16_T SlaveAddr;
U16_T RetAddr = 0;
U16_T SlaveCnt = 0;
U16_T const AliasAddr = 0x0021; // A const value for searching.
```

```
NEC_RtGetSlaveCount(MasterId, &SlaveCnt)
for( i = 0; i < SlaveCnt; ++i )
{
    NEC_RtGetAliasAddress(MasterId, i, &RetAddr);
    if( AliasAddr == RetAddr )
    {
        SlaveAddr = i; //find out!
        break;
    }
}
```

}
}

参関:

6.8.3. NEC_RtGetSlaveCoeProfileNum

取得從站模組”CoeProfileNum”資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_RtGetSlaveCoeProfileNum(U16_T MasterId, U16_T SlaveAddr,
U32_T *pCoeProfileNum);
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U32_T * pCoeProfileNum: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

此函數用於取得指定從站模組的 CoeProfileNum 資訊。例如，使用者可經由底下程式碼，得到網路上那些模組為驅動器(CoeProfileNum 等於 402)

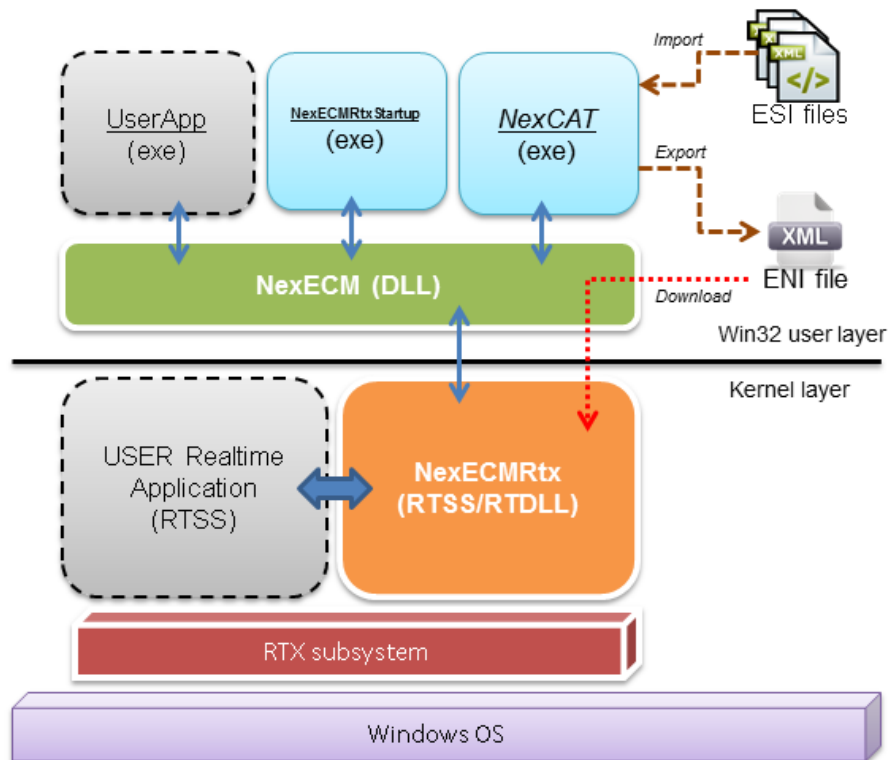
```
U16_T SlaveAddr, i;
U16_T SlaveCnt = 0;
U32_T CoeProfileNum = 0;
```

```
NEC_RtGetSlaveCount(MasterId, &SlaveCnt)
for( i = 0; i < SlaveCnt; ++i )
{
    NEC_RtGetSlaveCoeProfileNum (MasterId, i, &CoeProfileNum);
    if( CoeProfileNum == 402)
    {
        RtPrintf(“SlaveAddr:%d is a drive.\n”, i);
    }
}
```

參閱:

7. NexECM 程式庫 (Win32 APIs)

使用者的 Win32 應用程式(如下圖中 UserApp.exe)可透過 NexECM 動態連結程式庫 (Win32 Dynamically linked library, DLL), 來控制 NexECMRtx runtime 。透過 NexECM Win32 函數庫使用者可以很快速的將 Win32 的應用程式和 NexECMRtx RTX 應用程式做整合。



NexECM 函式庫主要提供下列功能:

1. 載入 / 移除 RTSS 應用程式至 RTX 系統中。
2. 載入 EtherCAT 網路描述檔(ENI)資訊至 EC-Master(NexECMRtx)中
3. ProcessData 存取(透過 NexECMRtx)
4. CoE SDO 存取(透過 NexECMRtx)
5. CiA402 APIs, 請參考 CiA402 APIs 使用手冊

7.1. Win32 API 總覽

下表列出 NexECM 函式庫 API 的列表，API 定義於 NexECM.h 標頭檔之中。

函數名稱	說明	
RTX 系統相關控制 API		
NEC_LoadRtxApp	載入RTSS 應用程式	
NEC_KillRtxApp	移除RTSS應用程式	
NexECM 初始化相關函式		
NEC_GetVersion	讀取NexECM 函式庫 (NexECM.dll) 版本號碼	
NEC_StartDriver	初始化NexECM函式庫	
NEC_StopDriver	關閉NexECM函式庫	
NexECMRtx Runtime 控制相關函式		
NEC_GetRtMasterId	自動偵測NexECMRtx並取得NexECMRtx所屬的控制代號 (MasterID)	
NEC_GetRtMasterVersion	讀取NexECMRtx 版本號碼	
NEC_GetRtMasterProcessId	讀取NexECMRtx Process ID (RTX Process ID)	
NEC_ResetEcMaster	重置 EC-Master (NexECMRtx)	
NEC_LoadNetworkConfig	載入ENI檔案	
NEC_StartNetwork	啟動EC-Master 通訊	
NEC_StartNetworkEx	啟動EC-Master 通訊附帶Option選項	
NEC_StopNetwork	停止EC-Master週期通訊	
NEC_SetParameter	設置 EC-Master 參數	
NEC_GetParameter	讀取 EC-Master 參數	
網路狀態存取相關函式		
NEC_GetSlaveCount	讀取EC-Slaves 個數	
NEC_GetNetworkState	讀取EC-Master當前狀態	
NEC_GetSlaveState	讀取EC-Slave當前狀態	
NEC_GetStateError	讀取EC-Master錯誤狀態	
NEC_GetErrorMsg	讀取EC-Master錯誤訊息字串	
DIO 控制相關函式		
NEC_SetDo	設定EC-Slave輸出值	
NEC_GetDo	取得EC-Slave輸出設定值	
NEC_GetDi	讀取EC-Slave輸入值	
CoE 通訊相關函式		
NEC_SDODownloadEx	送出 SDO download 命令資料 (Master to slave)	

NEC_SDOUploadEx	送出 SDO upload 命令資料 (Slave to Master)	
NEC_SDODownload	執行 SDO download (Structure)	
NEC_SDOUpload	執行 SDO upload (Structure)	
Process data 存取相關函式		
NEC_RWProcessImage	存取 EC-Master 的 ProcessData 資料	
NEC_GetProcessImageSize	取得 EC-Master 的 ProcessData 記憶體長度	
NEC_RWSlaveProcessImage	存取 EC-Slave 的 ProcessData 記憶體	
存取從站模組硬體資訊相關函式		
NEC_GetConfiguredAddress	取得從站 Configured Station Address	
NEC_GetAliasAddress	取得從站 Configured Station Alias	
NEC_GetSlaveCoeProfileNum	取得從站 CoeProfileNum	

API 所使用的 C/C++ 資料型態定義於 nex_type.h 中，說明如下表：

型別	C/C++ 原型	說明	大小 byte	範圍
BOOL_T	int	布林型別	4	0:False, 1:True
U8_T	unsigned char	無號整數	1	0 ~ 255
U16_T	unsigned short	無號整數	2	0 ~ 65535
U32_T	unsigned int	無號整數	4	0 ~ 4294967295
U64_T	unsigned __int64	無號整數	8	0 ~ 18446744073709551615
I8_T	char	有號整數	1	-128 ~ 127
I16_T	short	有號整數	2	-32768 ~ 32767
I32_T	int	有號整數	4	-2147483648 ~ 2147483647
I64_T	__int64	有號整數	8	-9223372036854775808 ~ 9223372036854775807
F32_T	float	浮點數	4	IEEE-754, 有效小數後 7 位
F64_T	double	雙精浮點數	8	IEEE-754, 有效小數後 15 位
RTN_ERR	int	錯誤代碼	4	-2147483648 ~ 2147483647

7.2. RTX 系統相關控制 API

7.2.1. NEC_LoadRtxApp

載入 RTSS 應用程式

C/C++語法:

```
RTN_ERR NEC_LoadRtxApp( const char* RtxFileName );
```

參數:

const char* RtxFileName: RTSS 檔案路徑 C-style 字串

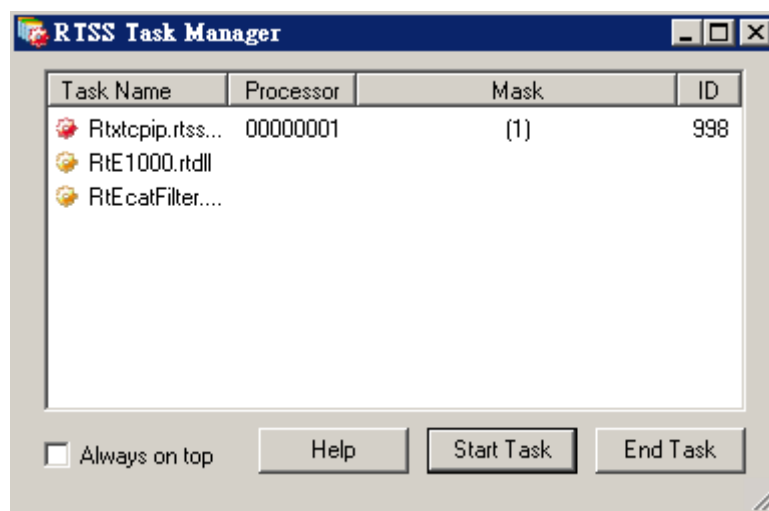
回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

可用本函數將 EC-Master runtime (NexECMRtx.rtss) 或任何 RTSS 應用程式載入到 RTX 系統中。可用 RTSS Task Manager (RTX 的工具程式)觀察被載入的程式。



本函式調用和本函數 *NEC_StartDriver()*無關，可先於 *NEC_StartDriver()*

參閱:

NEC_GetRtMasterId (); NEC_KillRtxApp (); NEC_GetRtMasterPorcessId()

7.2.2. NEC_KillRtxApp

移除 RTSS 應用程式

C/C++語法:

```
RTN_ERR NEC_KillRtxApp( U32_T ProcessId );
```

參數:

U32_T ProcessId: RTSS's Process ID.

回傳值:

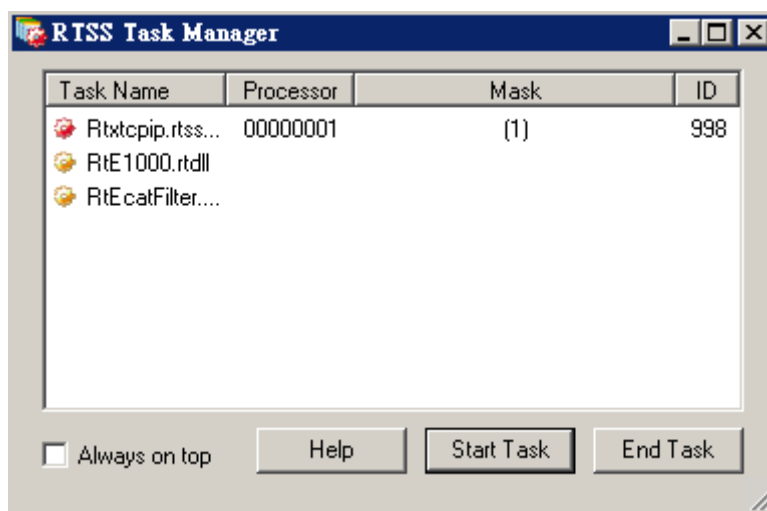
回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

可用本函數將 EC-Master runtime (NexECMRtx.rtss) 或任何 RTSS 應用程式從 RTX 系統中強制移除。ProcessID 可經由 RTX 工具程式: “Task Manager” (如下圖)觀察。亦可用 RTSS Task Manager (RTX 的工具程式)確認 RTSS 的程序是否被移除。

NexECMRtx 的 ProcessID 可以用 NEC_GetRtMasterPorcessId() 取得。



若使用本函數強制移除相關應用程式，可能造成系統不穩定，須注意使用。容易不穩定的狀況可能來自移除應用程式的順序，例如當應用程式直接存取 NexECMRtx 的記憶體時將 NexECMRtx 移除。

本函式調用和函數 NEC_StartDriver()無關，可先於 NEC_StartDriver()

參閱:

NEC_LoadRtxApp(); NEC_GetRtMasterPorcessId(); NEC_GetRtMasterId();

7.3. NexECM 初始化相關函式

7.3.1. NEC_GetVersion

讀取 NexECM 函式庫 (NexECM.dll) 版本號碼

C/C++語法:

```
U32_T NEC_GetVersion();
```

參數:

<無參數>

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

讀取 NexECM 函式庫 (NexECM.dll) 版本號碼

參閱:

<無參閱>

7.3.2. NEC_StartDriver

初始化 NexECM 函式庫

C/C++語法:

```
RTN_ERR NEC_StartDriver();
```

參數:

<無參數>

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

在使用 NexECM 函式庫中所有函式(API)前，需先呼叫此函數進行初始化。本函數亦同時偵測 NexECMRtx 是否載入，因此必須確定 NexECMRtx 是否已經被載入到 RTX 系統中。載入 NexECMRtx 可參考 *NEC_LoadRtxApp()*

參閱:

NEC_LoadRtxApp();NEC_StopDriver();

7.3.3. NEC_StopDriver

關閉 NexECM 函式庫

C/C++語法:

```
void NEC_StopDriver();
```

參數:

<無參數>

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

通常在 Win32 應用程式結束前呼叫本函數來釋放系統資源。

參閱:

NEC_StartDriver ()

7.4. NexECMRtx Runtime 控制相關函式

7.4.1. NEC_GetRtMasterId

偵測 NexECMRtx 是否已載入並取得 NexECMRtx 所屬的控制代號(MasterID)

C/C++語法:

```
RTN_ERR NEC_GetRtMasterId( U16_T *pMasterId );
```

參數:

U16_T *pMasterId: 回傳 NexECMRtx 的控制代號(MasterID)

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

Win32 NexECM在操作(或控制) NexECMRtx之前必須先使用本函數取得控制代號 (MasterID)，若本函數還處錯誤代碼(ECERR_DRIVER_NOT_FOUND)可能表示 RTX系統中尚未載入NexECMRtx。

參閱:

NEC_LoadRtxApp(); NEC_KillRtxApp (); NEC_GetRtMasterPorcessId()

7.4.2. NEC_GetRtMasterVersion

讀取 NexECMRtx 版本號碼

C/C++語法:

```
RTN_ERR NEC_GetRtMasterVersion( U16_T MasterId, U32_T *pVersion );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

U32_T *pVersion: NexECMRtx 版本號碼

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

讀取 NexECMRtx (RTX EtherCAT master) 版本號碼。同 *NEC_RtGetVersion()*

參閱:

7.4.3. NEC_GetRtMasterPorcessId

讀取 NexECMRtx Process ID (RTX Process ID)

C/C++語法:

```
RTN_ERR NEC_GetRtMasterPorcessId( U16_T MasterId, U32_T *pProcessID );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

U32_T *pProcessID: 回傳 RTX Process ID

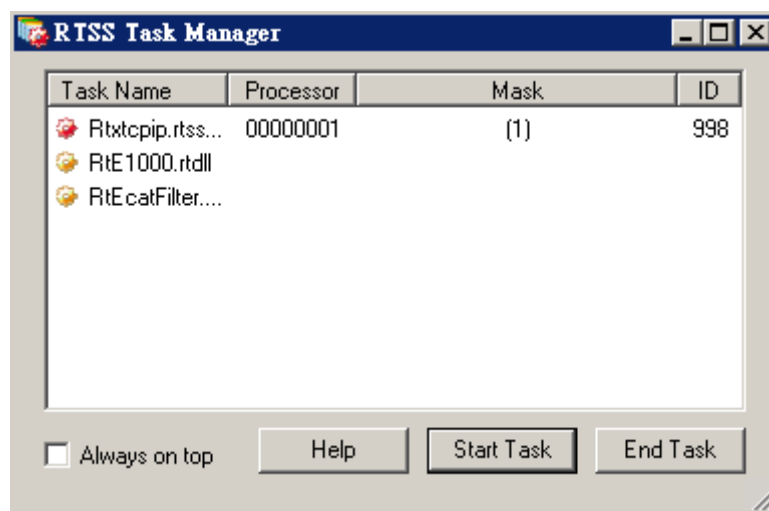
回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

RTX ProcessID 可經由工具程式 Task manager 觀察。用本函數可取得 NexECMRtx 的 ProcessID. 可用於 *NEC_KillRtxApp()*。



RTSS Task Manager

參閱:

NEC_KillRtxApp();

7.4.4. NEC_ResetEcMaster

重置 EC-Master (NexECMRtx)

C/C++語法:

```
RTN_ERR NEC_ResetEcMaster( U16_T MasterId );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用來重置 EC-Master (NexECMRtx)。一般使用本函數的時機是在 *NEC_GetRtMasterId()*取得 MasterID 後，載入 ENI 檔案前。使用本函數先將 EC-Master 內部變數進行重置。若先前已經有下載過 ENI 檔案，ENI 檔案資料也會被清除。

參閱:

NEC_GetRtMasterId(); *NEC_LoadNetworkConfig()*

7.4.5. NEC_LoadNetworkConfig

載入 ENI 檔案

C/C++語法:

```
RTN_ERR NEC_LoadNetworkConfig( U16_T MasterId, const char *ConfigurationFile,
U32_T Option );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

const char *ConfigurationFile: ENI 檔案路徑 C-style 字串

U32_T Option: 載入選項。

Bit 號	31 ~ 1	0
說明	保留(設定為 0)	網路卡(NIC)選擇 0:使用 ENI 設定 1:使用內部參數

當 Bit 0 設定為 1 時，EC-Master 所使用的網路卡(NIC)由參數設定，請參考 *RtSetParameter()*。

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此還是一般用在 *ResetEcMaster()*之後，本函式用於載入目前 EtherCAT 網路組態資訊(ENI) XML 檔案。ENI 檔案必須符合 ETG.2100 規範。ENI 檔案可由 NexCAT 工具程式產生(請參閱第三章)

參閱:

NEC_ResetEcMaster()

7.4.6. NEC_StartNetwork

啟動 EC-Master 通訊

C/C++語法:

```
RTN_ERR NEC_StartNetwork ( U16_T MasterId, const char *ConfigurationFile, I32_T  
TimeoutMs );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

const char *ConfigurationFile: ENI 檔案路徑 C-style 字串

I32_T TimeoutMs: 逾時等待時間，單位 millisecond

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，
錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此函數的使用時機在於使用 *NEC_SetParameter()* 設定 EC-Master 參數後啟動通訊，
啟動成功後，EC-Master 會建立周期的通訊機制。呼叫 *NEC_StopNetwork()*停止通
訊 EC-Master 通訊。

參閱:

NEC_GetRtMasterId(); *NEC_SetParameter()*; *NEC_StopNetwork()*;
NEC_StartNetworkEx()

7.4.7. NEC_StartNetworkEx

啟動 EC-Master 通訊；此 API 同 NEC_StartNetwork，多了一個設定參數。

C/C++語法:

```
RTN_ERR NEC_StartNetwork ( U16_T MasterId, U16_T MasterId, const char
*ConfigurationFile, U32_T Option, I32_T TimeoutMs );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

const char *ConfigurationFile: ENI 檔案路徑 C-style 字串

U32_T Option: 啟動選項。

Bit 號	31 ~ 1	0
說明	保留(設定為 0)	網路卡(NIC)選擇 0:使用 ENI 設定 1:使用內部參數

當 Bit 0 設定為 1 時，EC-Master 所使用的網路卡(NIC)由參數設定，請參考 RtSetParameter()。

I32_T TimeoutMs: 逾時等待時間，單位 millisecond

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此函數的使用時機在於使用 NEC_SetParameter() 設定 EC-Master 參數後啟動通訊，啟動成功後，EC-Master 會建立周期的通訊機制。呼叫 NEC_StopNetwork 停止通訊 EC-Master 通訊。

參閱:

NEC_GetRtMasterId();NEC_SetParameter();NEC_StartNetwork();NEC_StopNetwork()

7.4.8. NEC_StopNetwork

停止 EC-Master 通訊

C/C++語法:

```
RTN_ERR NEC_StopNetwork( U16_T MasterId, I32_T TimeoutMs );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

I32_T TimeoutMs: 逾時等待時間，單位 millisecond

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”（0），反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

此函數用於停止 EC-Master 週期通訊。停止通訊過程中，EC-Master 會切回 “INIT” 狀態。

參閱:

NEC_GetRtMasterId();NEC_SetParameter();NEC_StartNetwork();

NEC_StartNetworkEx()

7.4.9. NEC_SetParameter

7.4.10. NEC_GetParameter

這兩個函數用於設置和讀取 EC-Master 參數。

C/C++語法:

```
RTN_ERR NEC_SetParameter( U16_T MasterId, U16_T ParaNum, I32_T ParaData );
```

```
RTN_ERR NEC_GetParameter( U16_T MasterId, U16_T ParaNum, I32_T *ParaData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()*取得

U16_T ParaNum: 指定參數代碼，請參考用法:

I32_T ParaData: 指定參數值，請參考用法:

I32_T *ParaData: 回傳參數值，請參考用法:

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

用於啟動 EC-Master 通訊之前，設定 EC-Master 通訊使用的參數，其參數列表如下。

參數代碼	說明	參數值
NEC_PARA_S_ECM_CYCLETIMEUS	EC-Master 通訊週期，單位 micro-second	250 ~ 1000000
NEC_PARA_S_NIC_INDEX	設定 NIC Port 號碼	0 ~ Max NIC port
NEC_PARA_ECM_RECV_TIMEOUT_US	定義 EtherCAT 網路封包回傳 timeout 時間，單位 micro-second。一般使用預設值即可。	250 ~ 2000000
NEC_PARA_ECM_LINK_ERR_MODE	網路斷線行為模式: LINKERR_AUTO: 當 EC-Slave(s) 斷線，EC-Master 自動偵測斷線的裝置。當裝置從新連線自動將 EC-Slaves 初始化並設定為“OP”狀態 LINKERR_MANUAL: 當某裝置斷線，其狀態會停留在 ERROR 狀態。當斷線裝置恢復連線(實體連線)，該	LINKERR_AUTO (0) LINKERR_MANUAL (1) LINKERR_STOP (2)

	<p>裝置將不會被重新初始化。</p> <p>LINKERR_STOP:</p> <p>只要有一裝置斷線，EC-Master 將網路中斷，並進入 ERROR 狀態。</p>	
<p>NEC_PARA_ECM_DC_CYC_TIME_MOD</p> <p>E</p>	<p>DC 時間設定模式:</p> <p>0: 根據 Master cycle time (預設)</p> <p>1: 根據 ENI 資料</p>	<p>0~1</p>

參閱::

NEC_GetRtMasterId();

7.5. 網路狀態存取相關函式

7.5.1. NEC_GetSlaveCount

讀取線上 EC-Slaves 裝置數量

C/C++語法:

```
RTN_ERR NEC_RtGetSlaveCount( U16_T MasterId, U16_T *Count );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 *NEC_GetRtMasterId()* 取得

U16_T *Count: 回傳 EC-Slaves 數量指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

當開啟 EC-Master 通訊後，使用此函數讀取 EC-Slave 的個數。此數量基本上由 ENI 檔案內容決定。

參閱:

NEC_GetRtMasterId()

7.5.2. NEC_GetNetworkState

讀取網路連線狀態

C/C++語法:

```
RTN_ERR NEC_GetNetworkState( U16_T MasterId, U16_T *State );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T * State: 回傳網路當前狀態指標

請參考下列定義:

STATE_STOPPED	(0) : Networking is stopped.
STATE_OPERATIONAL	(1) : Networking is in operation state.(EtherCAT in
STATE_ERROR	(2) : Networking / slaves errors, and stopped.
STATE_SLAVE_RETRY	(3) : Networking / slaves errors, and try to re-connect.

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，輪詢網路當前的狀態。

參閱:

```
NEC_GetRtMasterId();
```

7.5.3. NEC_GetSlaveState

讀取網路連線狀態

C/C++語法:

```
RTN_ERR NEC_GetSlaveState( U16_T MasterId, U16_T SlaveIndex, U8_T *StateArr,  
U16_T *ArrLen )
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveIndex: 指定起始的 EC-Slave 位置

U8_T StateArr: 回傳 Slave 狀態值陣列

U8_T ArrLen:

Input: 指定讀取 EC-Slaves 的數量，StateArr 陣列大小。

Output: 返回實際讀取 EC-Slaves 的數量

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，輪詢網路上所有 EC-Slaves 的當前狀態。

參閱:

NEC_GetRtMasterId();

7.5.4. NEC_GetStateError

讀取 EC-Master 狀態錯誤代碼

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetStateError( U16_T MasterId, I32_T *Code );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

I32_T *Code: 回傳狀態錯誤指標，錯誤代碼定義於 EcErrors.h 標頭檔中

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，EC-Master 會將狀態由“INIT”切換到“OP”狀態，若發生錯誤，則狀態會被設定為“ERROR”狀態，此時可使用此 API 讀取 EC-Master 狀態錯誤代碼。

參閱:

NEC_GetRtMasterId();NEC_GetErrorMsg()

7.5.5. NEC_GetErrorMsg

讀取 EC-Master 狀態錯誤字串

C/C++語法:

```
RTN_ERR NEC_GetErrorMsg( U16_T MasterId, char *ErrMsg_128_len );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

char *ErrMsg_128_len: 回傳狀態字串指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，EC-Master 會將狀態由“INIT”切換到“OP”狀態，若發生錯誤，則狀態會被設定為“ERROR”狀態，此時可使用此 API 讀取 EC-Master 狀態錯誤字串。

參閱:

NEC_GetRtMasterId();NEC_GetStateError()

7.6. DIO 控制相關函式

7.6.1. NEC_SetDo

設定 EC-Slave Digital output 輸出

C/C++語法:

```
RTN_ERR NEC_SetDo( U16_T MasterId, U16_T SlaveAddr, U16_T Offset, U16_T
SizeByte, const U8_T *DoData )
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T Offset: Slave ProcessData (Output)記憶體位移值

U16_T SizeByte: 設定 DO 資料長度，單位 Byte

const U8_T *DoData: 指定輸出資料常數指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，設定 EC-Slave Digital output 輸出；使用上需注意 SizeByte 與 DoData 長度。

參閱:

NEC_GetRtMasterId();

7.6.2. NEC_GetDo

讀取 EC-Slave Digital output 輸出

C/C++語法:

```
RTN_ERR NEC_GetDo( U16_T MasterId, U16_T SlaveAddr, U16_T Offset, U16_T
SizeByte, U8_T *DoData )
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T Offset: 記憶體位移值

U16_T SizeByte: 指定讀取長度

U8_T *DoData: 指定讀取資料常數指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，讀取 EC-Slave Digital output 輸出；使用上需注意 SizeByte 與 DoData 長度。

參閱:

NEC_GetRtMasterId();

7.6.3. NEC_GetDi

讀取 EC-Slave Digital input 輸入

C/C++語法:

```
RTN_ERR NEC_GetDi( U16_T MasterId, U16_T SlaveAddr, U16_T Offset, U16_T  
SizeByte, U8_T *DiData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T Offset: 記憶體位移值

U16_T SizeByte: 指定讀取長度

U8_T *DiData: 指定讀取資料常數指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，
錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

用於啟動 EC-Master 通訊之後，讀取 EC-Slave Digital input 輸出；使用上需注意
SizeByte 與 DiData 長度。

參閱:

NEC_GetRtMasterId();

7.7. CoE 通訊相關函式

7.7.1. NEC_SDODownloadEx

7.7.2. NEC_SDOUploadEx

7.7.3. NEC_SDODownload

7.7.4. NEC_SDOUpload

C/C++語法:

```
RTN_ERR NEC_SDODownloadEx( U16_T MasterId, U16_T SlaveAddr
    , U16_T Index, U8_T SubIndex , U8_T CtrlFlag
    , U32_T DataLenByte, U8_T *DataPtr, I32_T *AbortCode );
RTN_ERR NEC_SDOUploadEx( U16_T MasterId, U16_T SlaveAddr
    , U16_T Index, U8_T SubIndex , U8_T CtrlFlag
    , U32_T DataLenByte, U8_T *DataPtr, I32_T *AbortCode );
RTN_ERR NEC_SDODownload( U16_T MasterId, U16_T SlaveAddr, TSDO *HSdo );
RTN_ERR NEC_SDOUpload( U16_T MasterId, U16_T SlaveAddr, TSDO *HSdo );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TSDO *HSdo: SDO 結構資料指標。

TSDO 資料結構

```
typedef struct
{
    U16_T index;
    U8_T subIndex;
    U8_T ctrlFlag;
    U8_T *dataPtr;
    U16_T dataLenByte;
    U16_T status;
    I32_T abortCode;
} TSDO;
```

U16_T index: CANOpen Object index

U8_T subIndex: CANOpen Object sub-index

U8_T ctrlFlag: Reserved, 請設定為 0

U8_T *dataPtr: Download 或 Upload 的資料指標

U16_T dataLenByte: 資料長度

U16_T status: Reserved

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於完成一個 SDO Download 或 SDO Upload 的要求，當函數成功返回表示 SDO 傳輸已經完成。

參閱:

NEC_GetRtMasterId();

7.7.5. NEC_GetODListCount

讀取 EC-Slave 五個物件字典(Object Dictionary)種類個別數量

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetODListCount( U16_T MasterId, U16_T SlaveAddr,
TCoEODListCount *pCoeOdListCount );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoEODListCount * pCoeOdListCount: 結構資料指標

TCoEODListCount 資料結構

```
typedef struct
{
    U16_T index;
    U8_T subIndex;
    U8_T ctrlFlag;
    U8_T *dataPtr;
    U16_T dataLenByte;
    U16_T status;
    I32_T abortCode;
} TSDO;
```

U16_T numOfAllObj: 所有物件字典數量

U16_T numOfRxPdoObj: 可映射的 RxPDO 字典數量

U16_T numOfTxPdoObj: 可映射的 TxPDO 字典數量

U16_T numOfBackupObj: 可儲存的字典數量

U16_T numOfStartObj: 開機載入的字典數量

U16_T status: Reserved

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於送出一個 SDO information，取得 EC-Slave 五個物件字典種類個別數量，當函數成功返回表示 SDO information 傳輸已經完成。

参関:

NEC_GetODList();NEC_GetObjDesc();NEC_GetEntryDesc();

NEC_GetEmgDataCount();NEC_GetEmgData();NEC_GetRtMasterId();

7.7.6. NEC_GetODList

讀取 EC-Slaves 的各種類的物件字典(Object Dictionary)索引值(index)

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetODList(( U16_T MasterId, U16_T SlaveAddr, TCoEODList
*pCoeOdList );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoEODList * pCoeOdList: 結構資料指標

TCoEODList 資料結構

```
typedef struct
{
    U16_T listType;
    U16_T lenOfList;
    U16_T *plistData;
    U16_T status;
    I32_T abortCode;
} TCoEODList;
```

U16_T listType: 指定物件字典種類

U16_T lenOfList: 指定回傳資料指標陣列長度

U16_T *plistData: 指定回傳資料指標

U16_T status: Reserved

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件字典種類所有物件索引值，當函數成功返回表示 SDO information 傳輸已經完成。

參閱:

NEC_GetODListCount(); NEC_GetObjDesc(); NEC_GetEntryDesc();

NEC_EmgDataCount();NEC_EmgData()

7.7.7. NEC_GetObjDesc

讀取 EC-Slave 的單一物件 Object Description 資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetObjDesc( U16_T MasterId, U16_T SlaveAddr, TCoEObjDesc
*pCoeObjDesc );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoEODList * pCoeOdList: 結構資料指標

TCoEODList 資料結構

```
typedef struct
{
    U16_T index;
    U16_T dataType;
    U8_T maxNumOfSubIndex;
    U8_T objectCode;
    U16_T sizeOfName;
    U8_T *pName;
    U16_T reserved;
    U16_T status;
    I32_T abortCode;
} TCoEObjDesc;
```

U16_T index: 指定物件索引值

U16_T dataType: 回傳物件資料型別

U8_T maxNumOfSubIndex: 回傳物件最大子物件

U8_T objectCode: 回傳物件資料形式

U16_T sizeOfName: 指定回傳資料指標陣列長度

U8_T pName: 指定回傳資料指標

U16_T reserved: 保留

U16_T status: Reserved

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件 Object Description 資訊，當函數成功返回表示 SDO information 傳輸已經完成。

參閱:

NEC_GetODListCount(); NEC_GetODList(); NEC_GetEntryDesc();

NEC_GetEmgDataCount(); NEC_GetEmgData(); NEC_GetRtMasterId();

7.7.8. NEC_GetEntryDesc

讀取 EC-Slave 的單一物件進入 Entry Description 資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetEntryDesc( U16_T MasterId, U16_T SlaveAddr,  
TCoEEntryDesc *pCoeEntryDesc );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

TCoEODList * pCoeOdList: 結構資料指標

TCoEODList 資料結構

```
typedef struct  
{  
    U16_T index;  
    U8_T subIndex;  
    U8_T valueInfo;  
    U16_T dataType;  
    U16_T bitLength;  
    U16_T objectAccess;  
    U16_T sizeOfData  
    U8_T *pData;  
    U16_T status;  
    I32_T abortCode;  
} TCoEEntryDesc;
```

U16_T index: 指定物件索引值

U8_T subIndex: 指定物件子索引值

U8_T valueInfo: 指定回傳資訊(一般設 0)

U16_T dataType: 回傳物件資料型別

U16_T bitLength: 回傳物件資料長度

U8_T objectAccess: 回傳物存取屬性

U16_T sizeOfData: 指定回傳資料指標陣列長度

U8_T pData: 指定回傳資料指標

U16_T status: Reserved

I32_T abortCode: SDO abort code 或 EC-Master error code

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於送出一個 SDO information，取得指定物件 Entry Description 資訊，當函數成功返回表示 SDO information 傳輸已經完成。

參閱:

NEC_GetODListCount(); NEC_GetODList(); NEC_GetObjDesc();

NEC_GetEmgDataCount(); NEC_GetEmgData(); NEC_GetRtMasterId();

7.7.9. NEC_GetEmgDataCount

讀取 EC-Master 中 Emergency 訊息容器中訊息數量

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetEmgDataCount( U16_T MasterId, U16_T  
*pEmgDataCount );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T pEmgDataCount: 資料回傳指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用於取得網路拓譜上 EC-Slaves 產生的 Emergency 資料數量。

參閱:

```
NEC_GetODListCount();NEC_GetODList();NEC_GetObjDesc();  
NEC_GetEntryDesc();NEC_GetEmgData();NEC_GetRtMasterId();
```

7.7.10.NEC_GetEmgData

讀取 EC-Master 中 Emergency 訊息器中訊息

```
RTN_ERR FNTYPE NEC_RtGetEmgData( U16_T MasterId, TEmgData *pEmgData );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

TEmgData _T *pEmgData: 資料回傳結構指標

TEmgData 資料結構

```
typedef struct
{
    U16_T lenOfData;
    U16_T res;
    U16_T *pSlaveAddrDataArr;
    TCoEEmgMsg *pEmgMsgDataArr;
} TEmgData;
```

U16_T lenOfData: 指定資料回傳陣列指標長度.

U16_T res: 保留

U16_T *pSlaveAddrDataArr: 資料回傳結構指標

TCoEEmgMsg *pEmgMsgDataArr: 資料回傳結構指標

TCoEEmgMsg 資料結構

```
typedef struct
{
    U16_T errorCode;
    U8_T errorRegister;
    U8_T data[5];
} TCoEEmgMsg;
```

U16_T errorCode: 回傳錯誤碼

U8_T errorRegister: 回傳錯誤暫存器

U8_T data[5]: 回傳資料陣列

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS”(0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於EcErrors.h標頭檔中。

用法:

本函數用於讀取 EC-Master 中的 Emergency 資料。

參閱：

```
NEC_GetODListCount();NEC_GetODList();NEC_GetObjDesc();  
NEC_GetEntryDesc();NEC_GetEmgData();NEC_GetRtMasterId();
```

7.8. Process data 存取相關函式

7.8.1. NEC_RWProcessImage

存取 EC-Master 的 ProcessData 資料。

C/C++語法:

```
RTN_ERR NEC_ NEC_RWProcessImage( U16_T MasterId, U16_T RW, U16_T Offset,
U8_T *Data, U16_T Size );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T RW:寫入/讀取指定

 READ_PI (0) : Read ProcessImage (PDI)

 WRITE_PI (1) : Write ProcessImage (PDO)

 READ_PDO (2) : Read ProcessImage (PDO)

U16_T Offset: EC-Master ProcessData 記憶體偏移量，從 0 開始，單位 Byte

U8_T *Data: 資料指標

U16_T Size: 資料長度，單位 Byte

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函式可用來讀/寫 EC-Master (NexECMRtx) 內部的 Process Image (或稱 ProcessData)。ProcessData 原理及操作方式請參考 5.7 小節 Process Data 存取。

參閱:

NEC_GetRtMasterId();NEC_GetProcessImageSize();NEC_RWSlaveProcessImage ()

7.8.2. NEC_GetProcessImageSize

取得 EC-Master 的 ProcessData 記憶體長度。

C/C++語法:

```
RTN_ERR NEC_GetProcessImageSize( U16_T MasterId, U16_T *SizeOfInputByte,  
U16_T *SizeOfOutputByte );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T *SizeOfInputByte: 回傳 ProcessData Input 記憶體大小指標

U16_T *SizeOfOutputByte: 回傳 ProceData Output 記憶大小指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，
錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數可用來讀取 EC-Master 中 ProcessImage 記憶體的大小，ProcessData 原理
及操作方式請參考 5.7 小節 Process Data 存取。

參閱:

NEC_GetRtMasterId();NEC_RWProcessImage ();NEC_RWSlaveProcessImage ()

7.8.3. NEC_RWSlaveProcessImage

存取 EC-Slave 的 ProcessData 記憶體。

C/C++語法:

```
NEC_RWSlaveProcessImage( U16_T MasterId, U16_T SlaveAddr, U16_T RW, U16_T
Offset, U8_T *Data, U16_T Size );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號。ID 可由 NEC_GetRtMasterId()取得

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T RW:寫入/讀取指定

 READ_PI (0) : Read ProcessImage (PDI)

 WRITE_PI (1) : Write ProcessImage (PDO)

 READ_PDO (2) : Read ProcessImage (PDO)

U16_T Offset: EC-Slave ProcessData 記憶體偏移量，從 0 開始，單位 Byte

U8_T *Data: 資料指標

U16_T Size: 資料長度，單位 Byte

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

本函數用來讀取或寫入資料到 EC-Slave 所佔的 ProcessData 記憶體之中，ProcessData 原理及操作方式請參考 5.7 小節 Process Data 存取。

參閱:

NEC_GetRtMasterId();NEC_RWProcessImage ();NEC_GetProcessImageSize ()

7.9. 存取從站模組硬體資訊相關函式

7.9.1. NEC_GetConfiguredAddress

取得從站模組” Configured Station Address”

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetConfiguredAddress( U16_T MasterId, U16_T SlaveAddr,  
U16_T *pConfigAddr );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T *pConfigAddr: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

此函數用於取得指定從站模組的 Configured Station Address 資訊。例如，使用者可經由底下程式碼，得到從站模組代號 0 其 Configured Station Address

```
U16_T SlaveAddr = 0;    // The first slave  
U16_T ConfigAddr = 0;   // A variable for storing the configured address  
NEC_GetConfiguredAddress ( MasterId,  SlaveAddr,  &ConfigAddr );
```

參閱:

7.9.2. NEC_GetAliasAddress

取得從站模組” Configured Station Alias”

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetAliasAddress(U16_T MasterId, U16_T SlaveAddr, U16_T
*pAliasAddr);
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U16_T *pAliasAddr: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳 “ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

此函數用於取得指定從站模組的 Configured Station Alias 資訊。例如，使用者可經由底下程式碼，得到 Configured Station Alias 為 0x0021 其模組代號

```
U16_T i;
U16_T SlaveAddr, i;
U16_T RetAddr = 0;
U16_T SlaveCnt = 0;
U16_T const AliasAddr = 0x0021; // A const value for searching.
```

```
NEC_GetSlaveCount(MasterId, &SlaveCnt)
for( i = 0; i < SlaveCnt; ++i )
{
    NEC_GetAliasAddress(MasterId, i, &RetAddr);
    if( AliasAddr == RetAddr )
    {
        SlaveAddr = i; //find out!
        break;
    }
}
```

参関:

7.9.3. NEC_GetSlaveCoeProfileNum

取得從站模組“CoeProfileNum”資訊

C/C++語法:

```
RTN_ERR FNTYPE NEC_GetSlaveCoeProfileNum( U16_T MasterId, U16_T SlaveAddr,  
U32_T *pCoeProfileNum );
```

參數:

U16_T MasterId: 指定目標 EC-Master 代號，單一 EC-Master 請設為 0

U16_T SlaveAddr: 指定目標 EC-Slave 代號，從 0 開始依序增號

U32_T *pCoeProfileNum: 寫入資料指標

回傳值:

回傳錯誤代碼。

調用函數成功回傳“ECERR_SUCCESS” (0)，反之函數調用失敗回傳錯誤代碼，錯誤代碼定義於 EcErrors.h 標頭檔中。

用法:

此函數用於取得指定從站模組的 CoeProfileNum 資訊。例如，使用者可經由底下程式碼，得到網路上那些模組為驅動器(CoeProfileNum 等於 402)

```
U16_T SlaveAddr, i;
```

```
U16_T SlaveCnt = 0;
```

```
U32_T CoeProfileNum = 0;
```

```
NEC_GetSlaveCount(MasterId, &SlaveCnt)
```

```
for( i = 0; i < SlaveCnt; ++i )
```

```
{
```

```
    NEC_GetSlaveCoeProfileNum (MasterId, i, &CoeProfileNum);
```

```
    if( CoeProfileNum == 402)
```

```
    {
```

```
        Printf(“SlaveAddr:%d is a drive.\n”, i );
```

```
    }
```

```
}
```

參閱: